

# Introdução à Inteligência Artificial



Bem-vindo(a) ao curso Introdução à Inteligência Artificial!

A sociedade contemporânea tem vivenciado inúmeras mudanças tecnológicas, com muitas áreas deixando de estar presentes apenas em centros de pesquisas tecnológicas, empresas de ponta e até mesmo em filmes de ficção científica, e esse é o caso da Inteligência Artificial (IA).

A IA tem se tornado cada vez mais comum nos *Apps*, nos *games*, na localização geográfica, na medicina, na agricultura, nas redes sociais, nos eletrodomésticos, nos serviços de *streaming*, nos *marketplaces* e em praticamente tudo o que fazemos na Internet.

Embora em grande parte das vezes não nos apercebamos, a IA já faz parte do nosso cotidiano e já vem sendo chamada há algum tempo de o “Inglês” da atualidade, com isso empresas de todos os seguimentos tem requisitado profissionais que a domine em 2020 o então CEO do *SoftBank Group Corp.* <<https://group.softbank/en>> Marcelo Claure, afirmou: “*A revolução da IA chegou - seu impacto nos próximos 20 anos, será mais forte do o que foi visto nos últimos 300.*” - inúmeros outros grandes líderes, empresários e até mesmo governos têm investido na área, o que demonstra sua importância estratégica e o seu potencial econômico para sociedade em geral.

O potencial transformador da IA é imenso e vem mudando a forma como vivemos e trabalhamos a cada dia. Este curso certamente o(a) ajudará a se preparar para aproveitar as muitas oportunidades que a área oferece.

Durante nossa jornada você entenderá os principais fundamentos da IA, com ênfase em seus princípios básicos, áreas de aplicação e pesquisa. Trataremos do funcionamento de diferentes tipos de algoritmos, em especial os de Aprendizado de Máquina, Programação de Linguagem Natural, Visão Computacional, Redes Neurais e Aprendizado Profundo. Serão implementados ainda exemplos práticos de alguns dos algoritmos estudados.

Então vamos lá, aproveite não só o conteúdo trazido aqui, mas procure também acessar todas as nossas sugestões, bem como se empenhe em se aprofundar nas mesmas e a buscar por coisas novas.

## 1. INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

### a. Definições e Conceitos Básicos de IA

O termo Inteligência Artificial (IA), nasceu em 1956 no Encontro de Dartmouth (*The Dartmouth Summer Research Project on Artificial Intelligence*) e foi definido pelo cientista da computação estadunidense John McCarthy, como sendo: “*A capacidade de uma máquina realizar funções que, se realizadas pelo ser humano, seriam consideradas inteligentes.*” (Bittencourt, 1998). Desde então, inúmeras definições surgiram para o que seja IA, dentre as quais destacam-se algumas:

Araribóia (1988), definiram a IA como o ramo do conhecimento que trata entre outras coisas do projeto e da construção de computadores e robôs inteligentes.

Chorafas (1988), disse que a IA é um corpo científico preocupado com a criação de sistemas computadorizados que podem atingir níveis humanos de raciocínio. Mais precisamente, a IA é o ramo da informática que enfoca o desenvolvimento de programas de computadores capazes de desempenhar tarefas normalmente associadas ao comportamento humano inteligente.

A definição de (Keller, 1991), é que a IA é um campo de empreendimento que tem por objetivo conceber modelos computacionais de comportamento inteligente humano.

Rich & Knight (1993), definem que a IA é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor.

Rabuske (1995), disse que IA é o resultado da aplicação de técnicas e recursos, especialmente de natureza não numérica, viabilizando a solução de problemas que exigiriam do humano certo grau de raciocínio e de perícia.

A IA possui interesse multidisciplinar tão abrangente que pode ir da biologia à linguística, da engenharia à medicina, e assim por diante. Por atuar em áreas tão diferentes, (Ribeiro, 1999) afirmou que o aprendizado de IA é contínuo: “*Não existe o especialista em por que o assunto é muito vasto para admitir foco.*”. Sentimento partilhado também por (Russel & Norvig, 2004), que afirmaram que potencialmente relevante para qualquer esfera da atividade intelectual humana, sendo a mesma verdadeiramente um campo universal.

Já (Luger, 2004), disse que IA pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente.

Já foi possível perceber que cunhar uma definição “fechada” do que seja IA, não é uma tarefa fácil, afinal o próprio termo inteligênci em si, já é bastante abrangente e complexo. Vejamos então algumas definições mais recentes, começando pelos autores já visitados (Russel & Norvig), só que agora na 3ª edição de seu livro lançado em 2009, quando os mesmos definiam o termo como: “*Uma disciplina da ciência da computação, que se concentra na criação de sistemas capazes de realizar tarefas que exigem inteligência humana.*”.

A IA é empolgante - e um pouco assustadora. Está por parte. A IA ajuda a proteger você das fraudes, marca comédicas, auxilia no atendimento ao cliente e até ajudar a escolher o programa de TV e na limpeza da casa.

(Mueller & Massaron

### Você Sabia?

O Encontro de Dartmouth <<https://home.dartmouth.edu/about/artificial-intelligence-ai-coined-dartmouth>> gerou frutos incríveis para a área de IA, com a Dartmouth College <<https://home.dartmouth.edu/>> sendo considerada o berço histórico da IA. Recentemente a instituição organizou o congresso The Dartmouth AI Conference2023 <[https://digitalstrategies.tuck.dartmouth.edu/dartmouth\\_ai\\_conference/](https://digitalstrategies.tuck.dartmouth.edu/dartmouth_ai_conference/)>, o qual vale muito ser acessado e consultado pelos temas e discussões ocorridas.

## b. Organização da IA

Numa visão atual, Cozman et al. (2021) afirmam ser mais produtivo organizar a área de IA em torno de três eixos, quais sejam: Representação de Conhecimento; Tomada de Decisão; e Aprendizado de Máquina.

- **Representação de Conhecimento:** É domínio da epistemologia; o raciocínio é centrado na lógica. É a forma como se dá o processo de codificar o conhecimento do mundo real, de modo que o mesmo possa ser implementado computacionalmente. Existem vários paradigmas para a representação do conhecimento, cada um com suas características próprias e, portanto, com diferentes pontos fortes e fracos.
- **Tomada de Decisão:** De forma similar ao eixo anterior, é tópico basilar em campos como psicologia, economia, engenharia e ciências. É o processo de escolher uma ação a partir de um conjunto de opções. Essa ação deve ser a melhor, ou a mais adequada, para atingir um determinado objetivo. O processo envolve a análise de informações, avaliação de alternativas e a escolha da opção considerada mais vantajosa, eficiente ou eficaz. É uma tarefa fundamental para a IA, pois é o que permite que as aplicações executem tarefas complexas e autônomas.
- **Aprendizado de Máquina:** Trata de assuntos caros à aprendizagem (pedagogia), mas também de técnicas estatísticas para o processamento de dados. É o conjunto de processos que permitem que os computadores aprendam sem serem explicitamente programados. Em outras palavras, os sistemas de Aprendizado de Máquina ou Machine Learning (ML) são capazes de aprender com dados e melhorarem seu desempenho ao longo do tempo.

## Classificação da IA (Tipos de IA)

A IA basicamente é classificada em duas vertentes, as IAs fracas e as fortes. No caso das IAs fracas, em sua imensa maioria as mesmas não possuem características de aprendizagem e/ou adaptação a novas situações, contextos ou cenários. Já as IAs fortes, estão preparadas para aprenderem, pensarem e resolverem problemas de forma semelhante ao que ocorre com os seres humanos, e por isso mesmas possuem como características a adaptação e o aprendizado.

Ambas as classificações possuem vantagens e desvantagens. Sendo que as IAs fracas são mais comuns em aplicativos comerciais em soluções de automação, enquanto que as IAs fortes eram até pouco tempo atrás, mais aplicadas em pesquisas acadêmicas e práticas de vanguarda (adaptado de Russel & Norvig, 2009), todavia atualmente têm tido forte expansão nas mais diversas áreas.

- **Exemplos de aplicação de IAs fracas:** Assistentes virtuais como Alexa (Amazon Inc.), Siri (Apple Inc.) e Bixby (Samsung Electronics Co., Ltd.), chatbots, sistemas de recomendação, filtros avançados para serviços de e-mail, reconhecimento de bots sociais, reconhecimento de fala e imagem, identificação de padrões e tendências, etc.; e
- **Exemplos de aplicação de IAs fortes:** Em carros, drones e máquinas agrícolas autônomas, na compreensão de contextos culturais e nuances linguísticas de maneira profunda, permitindo entre outras possibilidades, traduções mais precisas e interpretações multilingüísticas contextuais em tempo real, inclusive em vídeo.

## Aprendizado de Máquina “ML - Machine Learning”

Quando falamos de ML - *Machine Learning* nos referimos a uma subcategoria da IA, sendo o mesmo o processo em que os computadores aproveitam propriedades de Redes Neurais Artificiais (RNAs) e Deep Learning ou Aprendizado Profundo para reconhecerem padrões e melhorarem a sua capacidade de identificar os mesmos. Com ajustes e dados suficientes, os algoritmos de aprendizado de máquina podem prever novos padrões e informações.

É muito comum no nosso dia-a-dia trabalharmos com dados (textos, números e informações em geral) brutos, e é também comum realizarmos com os mesmos um tratamento padronizado ou científico. A ciência de dados é a forma como podemos pegar esses dados e aplicarmos um pensamento científico e sistematizado sobre os mesmos, ou seja, os explorarmos através de métodos formais e disciplinados (matemáticos e/ou estatísticos/probabilísticos, etc.) para os tratarmos de modo que tenhamos maior certeza de que, e se estamos tratando e/ou tentando obter uma resposta de retorno mais adequada.

“A pesquisa em Aprendizado de Máquina faz parte da pesquisa em IA, que busca fornecer conhecimento aos computadores por meio de dados, observações e interações com o mundo. Esse conhecimento adquirido permite que os computadores generalizem corretamente para novas configurações.”

Yoshua Bengio (Cientista da Computação Canadense, ganhador do Prêmio Turing de 2018)

Os avanços da IA dependeram e dependem de evoluções computacionais “hardware” e também da eficácia de muitos de seus métodos “software”, dentre elas há um especial destaque do subcampo de ML e de algumas de suas subcategorias, tais como: NLP - *Natural Language Processing* ou Processamento de Linguagem Natural, FD - *Federated Learning* ou Aprendizado Federado, RL - *Reinforcement Learning* ou Aprendizado por Reforço e o DL - *Deep Learning*.

A implantação da IA nas empresas ainda é um desafio, sendo semelhante ao que ocorreu com as equipes de desenvolvimento de software há alguns anos frente aos problemas de versionamento, colaboração em grandes sistemas de informação e dimensionamento de recursos computacionais. Neste sentido tem-se observado um esforço para a adoção de MLOps - *Machine Learning Operations* ou Operações de Aprendizado de Máquina.

As MLOps visam atacar a natureza ainda desigual e isolada do desenvolvimento de grande parte das aplicações de IA, estabelecendo para tal, práticas de colaboração entre cientistas de dados, de modo a simplificar os processos de gerenciamento e automatizar a implantação de modelos de IA nos principais ambientes e/ou sistemas de software das organizações.

Em 2021 o relatório *Global AI Adoption Index* da IBM, apontava que 41% das empresas aceleraram a implementação da IA como resultado da pandemia (IBM, 2021), e ainda segundo Wiggers (2022) dados do *IDC's 2022 AI InfrastructureView*, mostravam que esses recém-chegados, juntaram-se aos 31% de empresas que já possuíam IA em produção ou estavam em testes ativos com tais tecnologias na época.

### Atenção

Neste curso não abordaremos as minúcias matemáticas, estatísticas, probabilísticas e/ou detalhes específicos da linguagem Python. Nosso intuito aqui é o de introduzir o tema da IA e demonstrar alguns casos práticos em que algoritmos específicos são bastante úteis e eficazes para solução de situações atualmente importantes.

Para o aprofundamento das áreas relacionadas ao nosso tema central e em especial às questões matemáticas e o seu inerente formalismo, sugerimos a pesquisa de referências especializadas. Pois os assuntos de entorno à IA, além de amplos, são passíveis de necessidade de aprofundamento para que possam ser compreendidos e implementados corretamente e eficazmente.

Dando prosseguimento a nossa jornada, como já enfatizamos em outros cursos que oferecemos, a escolha correta das ferramentas de desenvolvimento e sustentação de nossas aplicações é crucial para o sucesso de nossos projetos, e com IA isso não é diferente! Temos acompanhado diariamente o surgimento de soluções inteligentes, e para que isso seja possível é claro, há todo um ecossistema de

Como ocorreu em todos os nossos cursos, utilizamos a linguagem *Python* para realizarmos nossas codificações o que faremos também aqui, todavia como é objetivo de nosso programa a diversificação de recursos, exploraremos aqui diferentes alternativas de *IDE* - *Integrated Development Environment* ou Ambientes de Desenvolvimento Integrado, *APIs* - *Application Programming Interface* ou Interfaces de Programação de Aplicação, classes/pacotes, ambientes e/ou serviços de apoio.

No caso dos IDEs, neste curso optamos por utilizarmos duas soluções, quais sejam:  
o PyCharm<<https://www.jetbrains.com/pycharm/>> desktop o Google Colab "Colaboratory Notebook" <<https://colab.research.google.com/notebook>> online.

O PyCharm é uma solução desenvolvida pela empresa tcheca *JetBrains s.r.o.*, a qual possibilita que a programação em linguagem *Python* seja bastante facilitada e otimizada. A solução fornece de forma integrada ferramentas de depuração, análise de código, testagem de unidades, integração com sistemas de controle de versão e desenvolvimento *Web* com *Django*.

O ambiente PyCharm é distribuído em três versões: Uma paga "Professional Edition" (U\$ 249,00); e duas gratuitas, Uma também completa "Educational License" mas apenas para uso acadêmico individual de alunos e professores, para uso em salas de aula, em bootcamps e cursos <<https://www.jetbrains.com/community/education/#students/>>; e Uma mais restrita open source "Community Edition", que embora possua uma quantidade menor de recursos, permite que vários deles sejam incorporados manualmente para ampliação e otimização de seu desempenho.

A Figura 1 a seguir, ilustra a tela principal (inicial) do PyCharm.

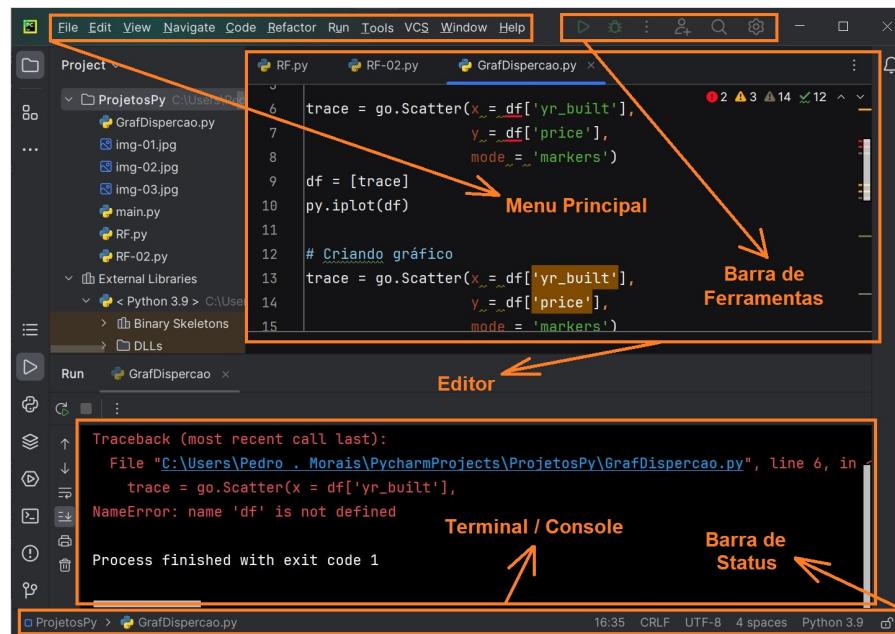


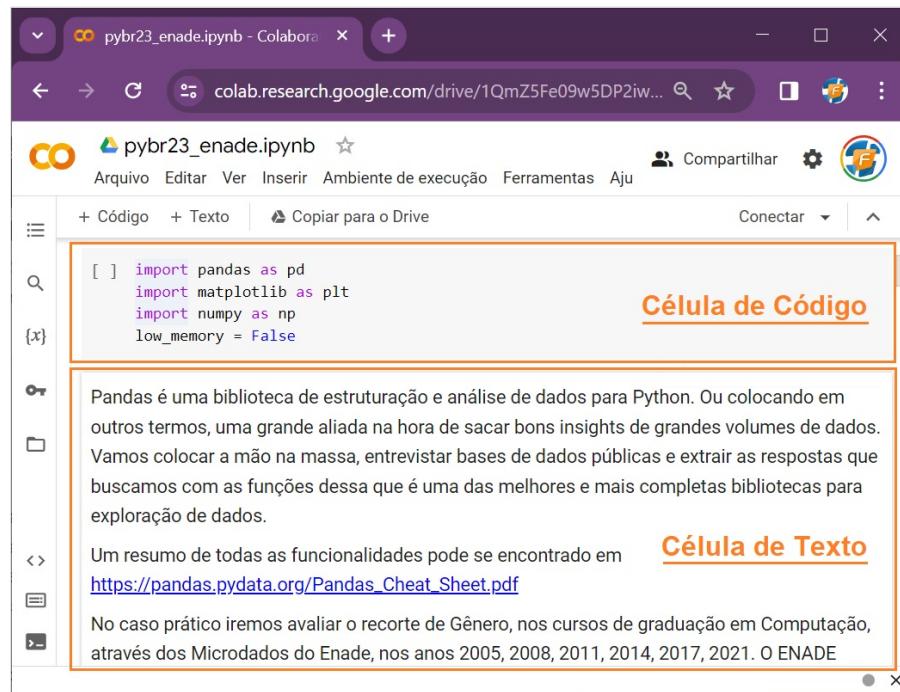
Figura 1: Tela inicial do PyCharm, destacando parte de suas principais funções e telas de trabalho.

O Google Colab é uma solução *online* disponibilizada pelo Google LLC, a qual propicia diretamente via *web browser*, o desenvolvimento e execução de códigos em *Python* sem a necessidade de instalação da linguagem e de grande parte de suas bibliotecas.

A solução disponibiliza alta performance e facilidade de compartilhamento de arquivos através do Google Drive, acesso a unidade disco, GPUs - *Graphics Processing Units* ou Unidades de Processamento Gráfico (placas de vídeo), vCPUs - *virtual Central Process Units* ou Unidades virtuais Centrais de Processamento e várias outras ferramentas/dispositivos com diferentes opções de utilização, desde a opção completamente gratuita até opções à la carte, sendo necessário para iniciar a sua utilização apenas que o usuário possui uma conta Google.

#### Nota:

A extensão dos arquivos criados no Google Colab é: **.ipynb** "Interactive Python Notebook" que é o formato original do Jupyter Notebook <<https://jupyter.org/>>, o qual permite criar e compartilhar arquivos interativos chamados de *notebooks* (cadernos). Os notebooks permitem que sejam combinados elementos de código, visualizações de dados, análises exploratórias e/ou relatórios com comentários que permitem inclusive, a inserção de imagens, áudios e vídeos, bem como, executar apenas trechos do código. Figura 2 a seguir, ilustra a distribuição básica de parte das ferramentas dispostas no Colab.



**Figura 2:** Tela principal do *Google Colab*, destacando células compondo o conjunto (*notebook*), com Código e Texto de Apoio

No *Colab* você poderá fazer o *download* de seus arquivos também em formato *Python “puro” .py*, acessando o MenuArquivo, o c também permite que arquivos e/ou projetos sejam salvos no *GDrive* <<https://drive.google.com/>> e/ou no *GitHub* <<https://github.com/>>. Bem como, poderá executar células de forma independente, o que para o caso de análise de dados por exemplo, pode ser bastante pois permite não só o ganho de tempo, mas também a redução da utilização de recursos de *hardware*.

## 2. TIPOS DE ML

O ML é uma área da ciência da computação diretamente relacionada à ciência de dados (*Data Science*), que se concentra no desenvolvimento de algoritmos que podem aprender com dados sem ser explicitamente programados. Existem três tipos principais de algoritmos de ML, quais sejam:

### Aprendizado Supervisionado

Esses algoritmos são treinados em um conjunto de dados que contém exemplos de entrada e saída correspondentes (os conjuntos de dados são rotulados). Os algoritmos então aprendem a associar as entradas às saídas correspondentes.

Por exemplo, algoritmos de aprendizado supervisionado podem ser usados para treinar um sistema de reconhecimento de imagens para identificar peças mecânicas perfeitas. O algoritmo seria treinado em um conjunto de dados de imagens de peças perfeitas e aprenderia a associar as características de tais imagens para reprová-las que não atendessem às especificações estabelecidas.

Tais algoritmos são usados em uma ampla variedade de aplicações, dentre as quais destacamos:

- **Reconhecimento de Padrões:** Onde são identificados padrões em dados, como padrões de voz, imagem e texto.
- **Classificação:** Classificando dados em categorias, como classificação de imagens, classificação de textos e classificação de dados médicos.
- **Régressão:** Prevendo valores contínuos, como previsão de vendas, previsão de preços e previsão de clima.

### Aprendizado Não Supervisionado

Os algoritmos são treinados em conjuntos de dados que não contêm exemplos de entrada e saída correspondentes (os conjuntos de dados não são rotulados). Os algoritmos então aprendem a identificar padrões nos dados sem saber o que os seus padrões representam.

Por exemplo, um algoritmo de aprendizado não supervisionado pode ser usado para agrupar dados de clientes semelhantes. O algoritmo aprenderia a identificar padrões nos dados dos clientes, como idade, renda e localização. O algoritmo poderia então agrupar clientes com base nesses padrões.

O algoritmos de aprendizado não supervisionado são usados em uma ampla variedade de aplicações, incluindo:

- **Redução de Dimensionalidade:** Reduzindo a dimensionalidade de dados (visualização de *Big Data*, elicitação de recursos, descoberta de estruturas e compreensão significativa), o que pode facilitar a análise dos mesmos.
- **Clusterização:** Agrupando dados em grupos, como agrupamento de clientes, agrupamento de dados de imagem e agrupamento de dados de texto (segmentação, recomendação e direcionamento).
- **Aprendizagem de Representações:** Aprendendo a representação de dados, que podem ser usados para melhorar o desempenho dos algoritmos de aprendizado supervisionado.

## Aprendizado por Reforço

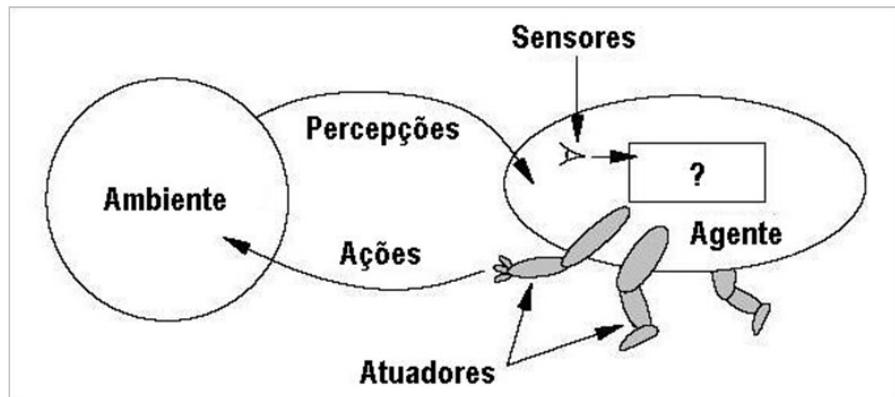
No aprendizado por reforço, um ou mais agente(s) inteligente(s) aprende(m) a tomar decisões em um ambiente de forma a maximizar uma recompensa. O(s) agente(s) recebe(m) *feedback* do ambiente, que é usado para ajustar sua política de tomada de decisão.

Por exemplo, um robô que aprende a vasculhar o terreno em um planeta, ao utilizar aprendizado por reforço recebe *feedbacks* dos sensores frente aos terrenos percorridos, como subida, descida, obstáculo ou buraco. O robô usa esses *feedbacks* para ajustar sua estratégia de exploração.

O aprendizado por reforço é usado em uma ampla variedade de aplicações, incluindo:

- **Robótica:** No treinamento de robôs para realizar tarefas complexas em ambientes hostis, andar, desviar, avaliar e pegar objeto
- **Controle de Sistemas:** No controle de sistemas complexos, como sistemas de tráfego, sistemas de energia e sistemas de projeto
- **Jogos:** Para treinar agentes para jogar jogos, como Xadrez, Go, Dota 2 e/ou melhorar a interação de NPCs - *Non-Players Character* ou Personagens-Não Jogáveis com jogadores humanos.

A Figura 3 a seguir, ilustra um agente em atuação com o ambiente.



**Figura 3:** Interação de um agente com o ambiente, adaptado de (Russel & Norvig, 2004)

Quando falamos em agentes inteligentes, estamos nos referindo tanto ao mundo físico com robôs por exemplo ou *softbots* (robôs implementados em software) - Resende (2003), definiu os agentes como personagens computacionais que atuam de acordo com um *script* definido, direta ou indiretamente, por um usuário. Eles podem atuar isoladamente ou em comunidade formando sistemas multiagentes. Já Russel & Norvig (2004), os definiram como sendo um sistema capaz de perceber por meio de sensores e agir em um dado ambiente por meio de atuadores.

As definições trazidas, podem ser enriquecidas pela descrição de um conjunto de propriedades usualmente aceitas e que os agentes devem possuir, quais sejam: Autonomia, Reatividade, Proatividade e Sociabilidade.

- **Autonomia:** Escolhe a ação a tomar baseado mais na própria experiência do que no conhecimento embutido pelo projetista. As ações do agente não requerem interferência humana direta;
- **Reatividade:** Pode perceber seu ambiente e responder às mudanças;
- **Proatividade:** Capaz de, além de responder a estímulos do ambiente, exibir um comportamento orientado a objetivos. Ou seja, é capaz de prever como atingir ou evitar um determinado estado ou objetivo. Pensar no futuro, antecipar, agindo em função de previsões

- **Sociabilidade:** Um agente pode interagir com outros agentes.

Nos métodos supervisionados informamos uma resposta automática, o que significa que fornecemos uma entrada (*input*) e também uma saída (*output*) associada à entrada, ou seja, informamos o que deve ser encontrado numa relação. Com eles podemos lidar com situações de classificação, que é a forma em que queremos classificar os dados em certas categorias ou em situações de regressão, obtendo um resultado que não tem classe. Esses métodos estão bastante relacionados aos métodos de aprendizado profundo ou *Learning*, o qual se baseia na aplicação de Redes Neurais ou *Neural Networks*, que veremos mais à frente.

Nos métodos não-supervisionados nós não informamos o que deve ser aprendido exatamente, e muito menos damos rótulos. Informamos apenas os dados e o método têm que aprender sozinho a dividir os grupos com base em suas características por conta própria.

Já a aprendizagem por reforço, opera entre os dois métodos anteriores: Supervisionado e Não-Supervisionado. Nela as máquinas (agentes) interagem com o ambiente por meio de ações em um esquema de tentativa e erro até aprenderem. O processo geral ocorre com o objetivo de o agente realizar uma ação no ambiente no qual o mesmo está inserido, onde se conseguir realizar o(s) objetivo(s) recebido (ganha) uma recompensa, e se não conseguir não recebe (não ganha) recompensa ou recebe uma punição. O processo se repete até que o agente aprenda a maximizar as recompensas e, portanto, aprenda a realizar o(s) seu(s) objetivo(s).

Cada tipo de aprendizado de máquina tem suas próprias vantagens e desvantagens. O tipo de aprendizado de máquina mais adequado para uma determinada tarefa depende dos dados disponíveis e dos objetivos da mesma.

## Algoritmos de Aprendizado de Máquina (ML)

### Árvores de Decisão

Árvores de decisão são um tipo de algoritmo de aprendizado supervisionado que pode ser usado para classificar ou prever valores numéricos. Elas são construídas a partir de um conjunto de dados de treinamento, no qual cada instância é representada por um conjunto de atributos e um rótulo. O algoritmo aprende a partir dos dados de treinamento, criando uma árvore de decisão que representa a relação entre os atributos e os rótulos.

Uma árvore de decisão é uma estrutura hierárquica em forma de grafo que consiste em nós e arestas. Cada nó representa uma pergunta sobre um atributo, e cada aresta representa uma resposta possível para essa pergunta (arestas conectam os nós e indicam o resultado de uma decisão). A árvore começa com um nó raiz, que representa a pergunta geral sobre os dados e dele, a árvore se ramifica em nós filhos, que representam subconjuntos dos dados. Cada nó filho representa uma pergunta mais específica sobre os dados, e assim por diante, até que um nó folha seja alcançado (as folhas representam os resultados ou as classes finais), sendo essa a previsão do algoritmo para a instância que está sendo classificada ou esperada.

As árvores de decisão precisam de um conjunto de dados de treinamento para aprender a relação entre os atributos e os rótulos. Para obter bons resultados os dados de treinamento devem ser o mais representativo possível sobre o que será usado para classificação e previsão.

A Figura 4 a seguir, ilustra o grafo de uma árvore de decisão.

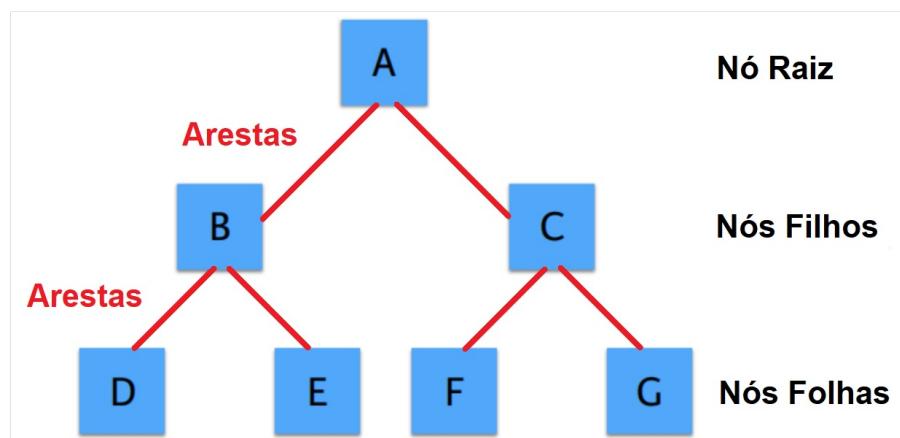


Figura 4: Exemplo de Árvore de Decisão.

As árvores de decisão são bastante versáteis, sendo usadas para uma variedade de tarefas, dentre as quais destacam-se: Na detecção de fraudes, na recomendação de produtos, em previsões financeiras, em previsões do tempo, em diagnósticos médicos, na segmentação e/ou direcionamento de campanhas de marketing, etc.

Alguns exemplos práticos de como as árvores de decisão são usadas no cotidiano das empresas e pessoas, são colocadas a seguir:

- **Amazon Inc.:** Recomenda produtos a seus clientes, para tal a empresa coleta dados sobre os históricos de compras, como quais produtos foram pesquisados, quais produtos foram visualizados e quais produtos foram adicionados ao carrinho de compras. A

empresa usa então esses dados para construir árvores de decisão que representem a relação entre os produtos que os cliente compraram e os produtos que eles podem estar interessados em comprar.

- **Netflix Inc.:** Indica filmes e séries de TV aos seus assinantes, coletando dados sobre os históricos de visualização, como quais filmes e séries de TV eles assistiram, quais eles avaliaram e quais foram adicionados às listas de observação e/ou de favoritos. A empresa utiliza essas informações para representar a relação de filmes e séries que os assinantes podem estar interessados e assistir e/ou para recomendar filmes com temáticas de interesse dos mesmos.
- **PayPal Holdings, Inc.:** Detecta fraudes sobre as transações feitas pelos usuários, para aprender sobre como as fraudes ocorrem. A empresa coleta dados como o valor da transação, endereço IP e qual o dispositivo usado pelo usuário para fazer a transação. A empresa então usa essas informações para construir árvores de decisão que representem a probabilidade de uma transação ser não fraudulenta.

## Regressão Linear

Os algoritmos de Regressão Linear, pertencem a classe dos algoritmos de aprendizado supervisionado, os mesmos são utilizados para modelar a relação entre uma variável dependente (a resposta ou o valor que queremos prever) e uma ou mais variáveis independentes (preditoras ou variáveis que são usadas para prever a variável dependente), sendo o objetivo focal das regressões o conhecimento sobre a relação existente entre variáveis quantitativas.

O objetivo focal das regressões é conhecer a relação existente entre variáveis quantitativas (valores numéricos), construídos a partir de um conjunto de dados de treinamento, no qual cada instância é representada por um conjunto de atributos e um valor real. Elas aprendem a partir desses dados de treinamento e criam uma equação linear que representa a relação entre os atributos e o valor resultante assumindo então a existência de uma relação linear entre as variáveis e procurando encontrar a linha que melhor se ajusta aos dados observados.

Quando aplicados corretamente os algoritmos de regressão linear fornecem *insights* valiosos, permitindo a tomada de decisões baseadas em informações com base nos dados avaliados. Um exemplo de tal eficácia ocorre com o Google LLC, que os utiliza para prever a relevância dos resultados de pesquisa realizadas no seu buscador, para tal coleta dados sobre as pesquisas realizadas, com base nas palavras-chave usadas, o histórico das pesquisas e a localização de quem está pesquisando. Para então construir uma equação linear que represente a relação entre essas informações e a relevância dos resultados das pesquisas.

Basicamente existem dois tipos de regressão linear. A regressão linear simples, com uma única variável independente (onde por exemplo x é uma variável dependente y) e a regressão linear múltipla, quando há mais de uma variável independente e o modelo é expandido para que sejam associados coeficientes para cada uma dessas variáveis.

Os algoritmos de regressão linear, são comumente utilizados para a previsão de vendas com base em dados históricos, previsão de preços de produtos ou serviços, previsão de desempenho e lesões de atletas, na previsão de desastres como terremotos e furacões, previsão do alcance e classificação de alertas sobre doenças, na previsão econômica entre outras.

## K-NN

O K-NN - K-Nearest Neighbors ou K-Vizinhos mais Próximos, é um algoritmo supervisionado que classifica amostras de conjunto de dados avaliando a distância em relação aos vizinhos mais próximos, e se os vizinhos mais próximos forem majoritariamente de uma determinada classe, a amostra em questão será classificada nesta categoria. Em suma o K-NN identifica os K dados de treinamento próximos do novo dado e o(s) atribui à classe mais comum desses K dados ao novo dado.

Para deixar mais claro a utilidade prática do algoritmo, suponha que temos um conjunto de dados de árvores de uma floresta, onde cada árvore é rotulada como uma das seguintes classes: Ipê, Jatobá ou Quaresmeira. O K-NN pode ser utilizado para classificar uma árvore desconhecida. Para fazer isso, identificaria os K dados de treinamento mais próximos de uma árvore desconhecida e atribuiria a classe mais comum desses K dados à mesma.

Fica claro portanto que o valor de K é um parâmetro importante do algoritmo, sendo que um valor de K alto torna o algoritmo mais conservador, atribuindo a classe mais comum a um novo dado apenas se houver muitos dados de treinamento próximos desse novo dado. Já um valor de K baixo torna o algoritmo mais agressivo, atribuindo a classe mais comum a um novo dado mesmo se houverem poucos dados de treinamento próximos desse novo dado.

É relativamente simples entendê-lo e implementá-lo. Ele é um algoritmo não paramétrico e, portanto, pouco sensível, o que significa que ele não faz suposições sobre a distribuição dos dados. Isso o torna uma boa escolha para problemas de classificação onde a distribuição dos dados é desconhecida ou não é possível fazer suposições sobre ela.

Os algoritmos K-NN, são comumente utilizados para a classificação de imagens, para o reconhecimento de fala, na classificação de mensagens de e-mail, na detecção de fraudes entre outras.

## Naive Bayes

São algoritmos de aprendizado supervisionado, comumente utilizados para classificação de conjuntos de dados. Baseiam-se no teorema de Bayes, que é uma fórmula matemática que relaciona a probabilidade de um evento ocorrer com a probabilidade de outros eventos terem ocorrido.

Essa classe de algoritmos assume que as variáveis independentes são independentesumas das outras, o que significa que a probabilidade de uma variável independente ocorrer não é afetada pela probabilidade de outra variável independente ocorrer. Essa suposição simplifica o cálculo da probabilidade posterior, que é a probabilidade de uma classe ocorrer dado um conjunto de dados.

Essa classe de algoritmos é comumente usada para diferentes formas de classificação em categorias, tais como: Categorias de notícias, ficção, tecnologia, marketing e etc.; Categorias de pessoas, animais, objetos e etc.; Categorias de e-mails - spams ou legítimos; Categorias de fraudes - transações fraudulentas ou legítimas. Também são utilizadas por assistentes digitais para o reconhecer a fala humana.

## Dica

Para saber como o teorema de Bayes funciona e pode ser utilizado, sugerimos que você assista aos vídeos intitulados "TEOREMA DE BAYES | Cálculo de Probabilidade | Responde Aí" <<https://www.youtube.com/watch?v=goaRH7a5hR4>> "Como fazer o cálculo de um projeto com Teorema de Bayes - Na prática!" <<https://www.youtube.com/watch?v=40S01DFX4kw>>, quais estão disponíveis no YouTube.

## SVM

Os algoritmos SVM - *Support Vector Machines* ou Máquinas de Vetores de Suporte, são bastante utilizados para classificação ou regressão. São baseados na ideia de encontrar um hiperplano (generalização do plano em diferentes números de dimensões "um lir decisão") que separe duas classes de dados com o maior distanciamento possível.

São comumente utilizados para analisar imagens de satélite, imagens agrícolas e de plantas, na previsão do tempo, na compreensão de imagens médicas, no entendimento do comportamento de consumidores, na previsão de valores numéricos, na análise de sentimento classificando textos como positivos, negativos ou neutros, para prever tendências e tomar decisões em investimentos financeiros, entre muitas outras possibilidades.

O código a seguir ilustra o uso do SVM para classificar as flores do dataset *Iris* <<https://archive.ics.uci.edu/dataset/53/iris>> em três espécies (classes) diferentes. O *kernel linear*<sup>1</sup> é utilizado aqui, mas dependendo da complexidade do problema, outros *kernels* (como RBF - *Radial Basis Function*) podem ser mais apropriados.

```
# Importação das bibliotecas

# Para o carregamento dos conjuntos de dados
from sklearn import datasets

# Para dividir os dados em conjuntos de treino e teste
from sklearn.model_selection import train_test_split

# Para utilizar o modelo SVM
from sklearn.svm import SVC

# Para avaliar a acurácia do modelo
from sklearn.metrics import accuracy_score

# Carregamento do dataset Iris
# X contém as features (atributos) e y as classes (rótulos) que queremos prever
iris = datasets.load_iris()

X = iris.data
y = iris.target

# Divisão do conjunto de dados em treino e teste
# Conjuntos de treino (80%) e teste (20%) usando a função train_test_split
# O argumento random_state controla a semente utilizada pelo gerador e garante a reprodutibilidade
# durante a divisão dos dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=50)

# Inicialização e treino do SVM
# O SVM é inicializado com um kernel linear e com um parâmetro de regularização C igual a 1
svm_classifier = SVC(kernel='linear', C=1)
svm_classifier.fit(X_train, y_train)

# Realiza previsões no conjunto de teste
y_pred = svm_classifier.predict(X_test)
```

```
# Avaliação da acurácia do modelo
# A função accuracy_score compara as previsões (y_pred) com os rótulos reais (y_test)
accuracy = accuracy_score(y_test, y_pred)
print("Saída do processamento . . .")
print("Acurácia do SVM:", accuracy)

Saída do processamento . . .
Acurácia do SVM: 0.9333333333333333
```

O resultado do processamento do código de exemplo retornando como saída do processamento a acurácia do SMV de 0.9333333333333333, significa que o sistema de ML acertou 93,33% das vezes. Em outras palavras, para cada 100 vezes que o sistema foi solicitado a fazer uma previsão, ele acertou 93 vezes.

A acurácia é uma medida de desempenho comumente usada para avaliar sistemas de ML, sendo a mesma calculada dividindo o número de previsões corretas pelo número total de previsões feitas. No nosso caso, com base no [dataset Iris](#)<sup>2</sup> utilizado, o algoritmo classificou 93 imagens corretamente e não conseguiu classificar ou classificou incorretamente 7 imagens.

Uma acurácia de 0,93 é um bom desempenho, mas não é perfeito. Um sistema com uma acurácia de 1,00 acertaria todas as previsões, mas isso raramente é obtido, pois na maioria dos casos, há alguma incerteza nos dados de treinamento, o que consequentemente leva inconclusões ou erros.

## Dica

Em ML os dados geralmente são divididos em três conjuntos: **Treino, Teste e Validação**.

O conjunto de treino é usado para treinar o modelo, os dados para esse conjunto devem ser representativos do conjunto de dados que o modelo será usado para prever;

O conjunto de teste é usado para avaliar o desempenho do modelo de ML, os dados para esse conjunto devem ser diferentes dados de treinamento para que o modelo não seja superestimado; e

O conjunto de validação é usado para ajustar os hiperparâmetros (parâmetros que controlam o comportamento) do modelo de ML.

A proporção dos dados para os conjuntos de treinamento, teste e validação pode variar dependendo do problema específico que está sendo resolvido. Em geral, é recomendado que o conjunto de dados de treinamento seja de pelo menos 80% do conjunto de dados total, o de teste de pelo menos 10% do conjunto de dados total e o de validação variando entre 10% e 20% do conjunto de dados total (neste caso, com os devidos ajustes sendo implementados nos dois conjuntos iniciais).

A razões pelas quais os conjuntos são divididos dessa forma, se dá entre outras:

Para que se evitem superajustes (quando o modelo se ajusta muito bem aos dados de treinamento, mas não se sai bem com dados novos). Separar os dados de treinamento e teste ajuda a evitar o superajuste, pois o modelo não vê os dados de teste até depois de ser treinado.

Para obter uma avaliação precisa do desempenho (se o modelo for avaliado apenas nos dados de treinamento, a avaliação se imprecisa, pois o modelo já conhece esses dados). Separar os dados de teste ajuda a obter uma avaliação precisa do desempenho do modelo em dados novos.

Para ajustar os hiperparâmetros com mais precisão (se o modelo for ajustado apenas nos dados de treinamento, os hiperparâmetros podem ser ajustados de forma incorreta, pois o modelo já conhece esses dados). Separar os dados de validação ajuda a ajustar os hiperparâmetros com mais precisão, pois o modelo é avaliado em dados novos.

Dividir os conjuntos de dados ajuda a garantir que o modelo não esteja simplesmente memorizando os dados de treino. Possibilita também avaliar o desempenho do modelo em dados novos e auxilia na identificação de possíveis problemas de *overfitting* (problemas que ocorrem quando um modelo de ML se ajusta muito bem aos dados de treino, mas não é capaz de generalizar para dados novos) - um sinal característico do *overfitting* é quando um modelo obtém desempenho muito alto no conjunto de treino, mas apresenta desempenho ruim no conjunto de teste.

<sup>1</sup> Um tipo de função linear simples e direta, que mapeia os dados de entrada em um espaço dimensional superior, sendo eficaz para problemas em que os dados são linearmente separáveis.

<sup>2</sup> Conjunto de dados multivariado com 150 amostras de flores do tipo Iris, cada uma com quatro Medidas: comprimento da sépala em cm, largura da sépala em cm, comprimento da pétala em cm e largura da pétala em cm; e três Classes: flores com sépalas curtas e estreitas e pétalas longas e estreitas "Iris Setosa", flores com sépalas longas e largas e pétalas curtas e largas "Iris Versicolor" e flores com sépalas longas e largas e pétalas curtas e largas "Iris Virginica".

## Redes Neurais

Algoritmos de Redes Neurais Artificiais (RNAs), podem e são utilizados para uma variedade extensa de tarefas, incluindo classificação, regressão, detecção de anomalias e aprendizado por reforço. Eles são inspirados no sistema neuronal humano, que é composto por neurônios que se comunicam por meio de sinapses. Essa classe de algoritmos pode ser utilizada em ambos os tipos de aprendizado de máquina: supervisionado e não-supervisionado.

As RNAs se baseiam em modelos matemáticos capazes de aprender e se adaptar de forma semelhante aos neurônios humanos compostos por um conjunto de nós, chamados de neurônios, que estão conectados entre si por meio de conexões, chamadas de sinapses.

O seu funcionamento básico se dá com cada neurônio recebendo um conjunto de entradas, que são processadas de acordo com a função de ativação. Sendo a saída do neurônio então transmitida para os neurônios conectados a ele.

O aprendizado nas RNAs ocorre por meio de um processo chamado treinamento, durante o qual a rede é apresentada a um conjunto de dados de treinamento, no qual cada instância é representada por um conjunto de entradas e um rótulo de saída. O objetivo do treinamento é ajustar os pesos das conexões entre os neurônios de forma que a rede seja capaz de prever o rótulo de saída correto para novas instâncias.

As RNAs são implementações poderosas, sendo capazes de aprender padrões complexos em dados e de gerar resultados muito precisos. Dentre os problemas mais comuns que podem ser atacados pelas RNAs, destacam-se: o reconhecimento de imagens (rosas, placas de carros, seleção/colheita de frutas, produtos dos mais diversos em linhas de produção); o reconhecimento de voz (implementado nos smartphones e imprescindíveis nos assistentes virtuais); o tratamento de linguagem natural (na tradução de idiomas para responder perguntas); e a aprendizagem de máquina preditiva (na previsão de eventos futuros, como o preço de ações ou a probabilidade de um cliente comprar ou não um produto).

Os tópicos que norteiam as RNAs são bastante amplos e via-de-regra, necessitam certo aprofundamento matemático-estatístico para serem implementadas eficientemente. Como esse não é foco em nosso curso, não nos aprofundaremos nos mesmos. Todavia, sugere-se fortemente que você procure conhecer mais sobre o assunto e como utilizá-lo em seus projetos.

Para dar um gostinho prático do que esses algoritmos são capazes, sugerimos que você acesse o excelente simulador de RNAs [Playground](https://playground.tensorflow.org/) <<https://playground.tensorflow.org/>>, o mesmo funciona de forma online, sendo parte integrante do ecossistema de produtos da empresa TensorFlow <<https://www.tensorflow.org/>> a qual oferece várias ferramentas de ML para auxílio na consolidação, limpeza e pré-processamento de dados em grande escala.

A Figura 5 a seguir, ilustra a simulação de uma RNA de 2 camadas, com 6 neurônios de entrada e 6 de saída, implementada e em execução no *Playground*.

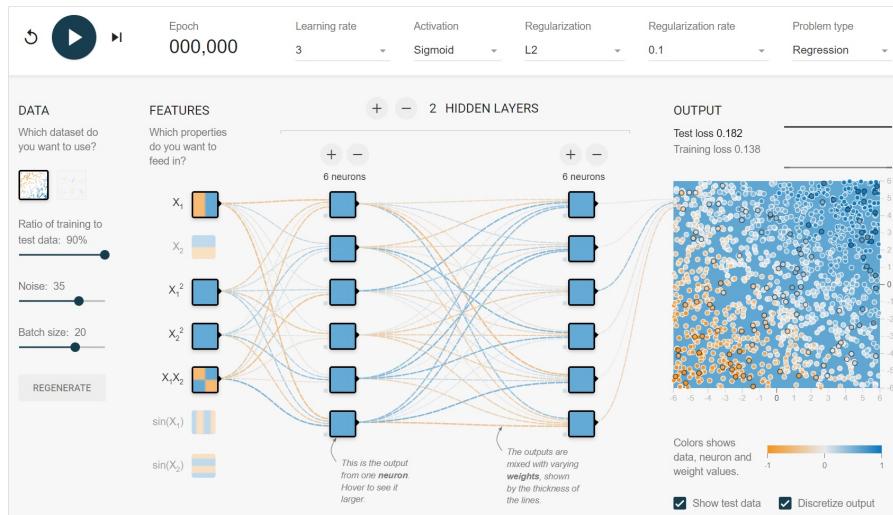


Figura 5: Simulação de uma RNA em execução no sistema *Playground*.

## Saiba Mais

Embora a ferramenta de *Playground* possa nos auxiliar a implementar, testar e analisar RNAs de forma prática e visual, para que de fato possamos usufruir de suas várias possibilidades, é necessário que conheçamos não só as principais características das RNAs, como também saibamos configurar, executar e interpretar corretamente os dados a serem inseridos, tratados e retornados pela ferramenta.

Neste sentido indicamos então, um artigo com noções básicas sobre redes neurais e como implementá-las, intitulado: “***Understanding neural networks with TensorFlow Playground***” disponível

no *Google Cloud* <<https://cloud.google.com/blog/products/ai-machine-learning/understanding-neural-networks-with-tensorflow-playground>> - e ainda três vídeos tutoriais: “*TensorFlow Playground*” <<https://www.youtube.com/watch?v=VQpIEPqxoCE>>, “*Neural Networks Made Simple with Tensorflow Playground*” <<https://www.youtube.com/watch?v=rto0Zfeqn8&t=600s>> e “*Tensor Flow - Redes Neurais - TensorFlow Playground*” <<https://www.youtube.com/watch?v=nmqLNOa32o0>> os quais estão disponíveis no YouTube e que certamente em muito contribuirão para o seu aperfeiçoamento.

#### Notas:

A *TensorFlow* também disponibiliza APIs e bibliotecas especializadas aplicáveis a uma ampla variedade de tarefas de ML e DL <<https://www.tensorflow.org/resources/libraries-extensions?hl=pt-br>>, que merecem ser pesquisadas e exploradas.

São ferramentas concorrentes do *Playground*: o *Keras* <<https://keras.io/>>, o *Theano* <<https://pypi.org/project/Theano/>> e o *Microsoft Cognitive Toolkit* (CNTK) <<https://learn.microsoft.com/pt-br/cognitive-toolkit/>>.

## Deep Learning

Aprendizado Profundo ou *Deep Learning* (DL) como o termo é mais conhecido em inglês, é uma subclasse do ML, a qual diz res algoritmos que possuem a capacidade de fazer com que máquinas aprendam a compreender representações complexas e hierárqui de dados. Tais características as tornam uma ferramenta poderosa em situações onde a modelagem tradicional pode ser desafiador entanto, é importante notar que o treinamento de modelos de DL muitas vezes requerem *hardware* especializado, como GPUs devid intensidade computacional envolvida.

A qualidade de performance de uma RNA no DL, varia muito de acordo com algoritmo utilizado, mas aliado a isso, também a qua do *dataset* (dados de treinamento da RNA) é essencial para que os dados obtidos sejam de fato assertivos.

## Pré-processamento de dados para ML

O pré-processamento de dados é uma etapa crucial no *pipeline*<sup>1</sup> de ML, pois é onde os dados brutos são transformados e organi de maneira a facilitar a sua análise e modelagem. Vale ressaltar a importância com o cuidado nesta etapa, pois a qualidade dos dad entrada afetará diretamente o desempenho e a eficácia dos modelos de ML.

As etapas mais comuns no pré-processamento envolvem:

- **Coleta dos Dados:** Envolve a reunião dos dados brutos de fontes relevantes para o problema em questão.
- **Limpeza dos Dados:** Tratamento de valores ausentes NaN - *Not a Number* (Não é um Número - valor especial ou “valor sentinel”, usado para representar resultados inválidos ou indefinidos em operações matemáticas), remoção de duplicidad correção de possíveis erros nos dados. Dados ausentes são comuns em muitas aplicações de análise de dados. O pacote *Pandas* por exemplo, deixa o trabalho com dados ausentes o menos problemático possível, pois por padrão exclui dados ausentes em todas as estatísticas descritivas de seus objetos (McKinney, 2018).
- **Padronização e Normalização dos Dados:** Padronização da escala dos dados para que todas as *features* (características “dimensão/escala das informações”) tenham a mesma importância. Normalização dos dados (torná-los mais adequados p algoritmos que são sensíveis à escala ou magnitude), como por exemplo entre 0 e 1.

Um exemplo prático disso é ilustrado nas Figuras 6 e 7, com o código implementado utilizando o pacote *scikit-learn*<<https://scikit-learn.org/stable/>> e a função de normalização *MinMax* (método de escalar os dados de um conjunto pa todos os valores fiquem entre 0 e 1) e o respectivo resultado da sua execução no *Google Colab*.

```

# Importa da classe MinMaxScaler o módulo preprocessing
# da biblioteca scikit-learn
from sklearn.preprocessing import MinMaxScaler

# Dados de exemplo - lista representando os dados
dados = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

# Inicializa o escalador "scaler"
scaler = MinMaxScaler()

# Ajusta o scaler aos dados "fit" e transforma-os usando
# a normalização Min-Max
dados_normalizados = scaler.fit_transform(dados)

print("Dados Originais:")
print(dados)
print("\nDados Normalizados:")
print(dados_normalizados)

```

**Figura 6:** Código de exemplo de Normalização de Dados.

 **Dados Originais:**  
`[[ -1, 2], [-0.5, 6], [0, 10], [1, 18]]`

**Dados Normalizados:**  
`[[ 0. 0.]
 [ 0.25 0.25]
 [ 0.5 0.5 ]
 [ 1. 1. ]]`

**Figura 7:** Resultado do código de exemplo de Normalização de Dados.

- **Conversão de Dados Categóricos:** Conversão das variáveis categóricas <sup>2</sup> por meio de uma codificação *one-hot* (técnica de representação de dados categóricos usada para converter variáveis categóricas em um formato que pode ser reconhecido por algoritmos de ML).
- **Engenharia de Recursos (Feature Engineering):** Criação de novas *features* relevantes ao problema. Extração de informações úteis dos dados existentes.
- **Divisão do Conjunto de Dados:** Separação dos dados em conjuntos de treinamento, validação e teste.

Um exemplo prático do que falamos sobre as etapas de pré-processamento, é descrito a seguir: Vamos considerar então um problema de previsão de preços de imóveis com base em dados como tamanho do imóvel, número de quartos, localização, etc.

- **Coleta de Dados:** Devemos obter dados sobre preços de imóveis, incluindo características relevantes sobre os mesmos.
- **Limpeza de Dados:** As entradas duplicadas devem ser removidas e os valores ausentes tratados, substituindo-os pela maioria dos valores existentes.
- **Padronização e Normalização:** O tamanho dos imóveis deve ser padronizado, para possuir média zero e desvio padrão 1; os preços devem ser normalizados para uma escala entre 0 e 1.
- **Conversão de Dados Categóricos:** Localização do imóvel deve ser convertida em variáveis *dummy* <sup>3</sup> (*one-hot*).
- **Engenharia de Recursos:** Uma nova *feature* deve ser criada para representar a relação entre o número de quartos e o tamanho do imóvel.
- **Divisão do Conjunto de Dados:** Os dados devem então ser separados em conjuntos de treinamento (80%), teste (10%) e validação (10%).

As etapas percorridas garantem que os dados estejam preparados para serem utilizados em algoritmos de ML, melhorando a precisão e a generalização dos modelos. Vale ressalva de que o processo em si pode variar dependendo do problema específico e do tipo de dados envolvidos, podendo então ter que ser adaptado.

<sup>1</sup> Sequência de etapas interconectadas que são organizadas para transformar dados brutos em um modelo de ML funcional. Tais etapas incluem via-de-regra: o pré-processamento de dados, o treinamento do modelo e a respectiva avaliação do seu desempenho.

<sup>2</sup> Variáveis que podem assumir um número limitado e geralmente fixo de valores, que representam diferentes categorias ou grupos. Essas variáveis são discretas e não contínuas, podendo ser divididas em duas categorias principais: nominal (baixo, médio, alto) e ordinal (1, 2, 3).

<sup>3</sup> Variáveis fictícias ou indicadoras, usadas para representar categorias em dados categóricos.

## Avaliação de modelos de ML

Trata-se de uma etapa crítica no desenvolvimento de qualquer sistema que envolva o desenvolvimento de uma solução na área de ciência de dados. Ela envolve a medição do desempenho do modelo em dados não vistos, para entender como ele é generalizado para novas instâncias.

A escolha de métricas de avaliação adequadas depende do tipo de problema (classificação, regressão, *clustering*, etc.) e dos requisitos específicos de cada projeto. Dentre as métricas mais utilizadas para avaliação de modelos de ML em diferentes contextos, têm-se:

### Classificação Binária:

- **Acurácia (Accuracy):** Contempla a proporção de previsões corretas em relação ao total de previsões.
- **Precisão (Precision):** Também conhecida como sensibilidade, abrange a proporção de verdadeiros positivos em relação ao total de positivos previstos.
- **Recall (Sensibilidade ou Revocação):** Diz respeito à proporção de verdadeiros positivos em relação ao total de positivos existentes.
- **F1-Score:** É a média harmônica entre precisão (*precision*) e revocação (*recall*). É uma métrica comum em problemas de classificação, a qual penaliza fortemente valores extremos, sendo, portanto, especialmente útil quando há um desequilíbrio significativo entre as classes.

### Classificação Multiclasse:

- **Acurácia Multiclasse:** É uma extensão da acurácia para problemas com mais de duas classes.
- **Matriz de Confusão (Confusion Matrix):** Tabelas que exibem as previsões corretas e incorretas, sendo úteis para entender onde os modelos estão errando.

### Regressão:

- **Erro Quadrático Médio (MSE - Mean Squared Error):** Média dos quadrados das diferenças entre as previsões e os valores reais.
- **Raiz do Erro Quadrático Médio (RMSE - Root Mean Squared Error):** Raiz quadrada do MSE, dando uma interpretação na mesma escala dos dados.
- **Coeficiente de Determinação ( $R^2$ ):** Mede a proporção da variância na variável dependente que é previsível a partir das variáveis independentes. Quando o  $R^2$  é próximo de 1, isso indica um bom ajuste.

### Clustering:

- **Índice de Silhueta (Silhouette Score):** Medida que indica o quanto semelhante um objeto é ao seu próprio cluster (coesão) em comparação com outros clusters (separação).
- **Inércia (Inertia):** Soma das distâncias quadráticas dos pontos para o centroide mais próximo, usado em algoritmos de clustering como K-Means.

### Processamento de Linguagem Natural (PLN):

- **Precisão (precision)** mede a capacidade do modelo de evitar classificar exemplos negativos como positivos, **Revocação (recall)** mede a capacidade do modelo de classificar exemplos positivos corretamente. **F1-Score** é a média harmônica entre precisão e revocação, a qual é muito utilizada para tarefas de classificação de texto.
- **BLEU Score - BiLingual Evaluation Understudy Score** avalia o desempenho de um modelo em tarefas de geração de traduções automáticas. É uma ótima medida de similaridade entre duas sequências de texto, sendo calculada comparando a sequência gerada pelo modelo com uma sequência de referência.
- **Perplexidade** medida de incerteza para modelos de linguagem, a qual é calculada como o inverso da probabilidade de uma sequência de texto gerada pelo modelo.
- **Validação Cruzada (Cross-Validation)** avalia o desempenho do modelo, o treinando e testando em diferentes subconjuntos de dados, o que contribui para reduzir a dependência de uma única divisão treino/teste.

Lembre-se de que a escolha de qualquer métrica depende do problema específico abordado e das implicações práticas da mesma frente ao seu contexto. Sendo que para qualquer situação por mais simples que seja e/ou com acesso a bons conjuntos de dados de treinamento e teste, ajustes finos ao modelo poderão ser necessários com base na métrica escolhida.

## Bibliotecas de Manipulação de Dados

Várias são as bibliotecas especializadas na manipulação de dados, em especial para grandes quantidades de informações sobre finanças, economia, estatística, *marketing*, *web analytics*, georreferenciamento, neurociência e muitas outras áreas. Python possui uma ótima variedade de pacotes para esse fim, entre eles destacam-se entre outros: o Pandas <<https://pandas.pydata.org/>>, o NumPy <<https://numpy.org/>>, o SciPy, <<https://scipy.org/>> e o Matplotlib <<https://matplotlib.org/stable/>>.

### Pandas

Quando falamos em Pandas - *Panel Data* ou Conjunto de Dados “dataset” para manipulação de estruturas multidimensionais no Python, estamos falando de manipulação de dados. O desenvolvimento da biblioteca, remonta o ano de 2008 e foi iniciado pela empresa americana de gestão de fundos de investimento AQR Capital Management <<https://www.aqr.com>>, tendo seu código aberto em 2009 sob licença BSD - Berkeley Software Distribution (licença de código aberto, a qual permite que o software distribuído sob ela seja incorporado a produtos proprietários).

### NumPy

O pacote NumPy - *Numerical Python* é essencial para quaisquer desenvolvedores que trabalhem com dados numéricos. Entre as situações mais comuns que podem ser tratadas por suas funções, destacam-se as associadas à ciência de dados (para análise de códigos, mineração de dados e visualização de dados); computação científica (para simulações, otimizações e cálculo numérico); e engenharia (para a análise de estruturas e processamento de sinais).

A biblioteca fornece estruturas de dados eficientes para o tratamento de arrays multidimensionais, operações matemáticas e estatísticas, regressões, classificação e análise de Fourier, além de vetorização de modo muito eficiente, o que pode melhorar significativamente o desempenho de aplicativos que trabalham com grandes quantidades de dados.

As Figuras 8 e 9 a seguir, ilustram um código simples para demonstrarmos a facilidade de uso da biblioteca NumPy e do pacote Matplotlib para a manipulação de objetos e exibição de seus resultados de forma gráfica.



```

# Importação de bibliotecas/pacotes
import numpy as np
import matplotlib.pyplot as plt

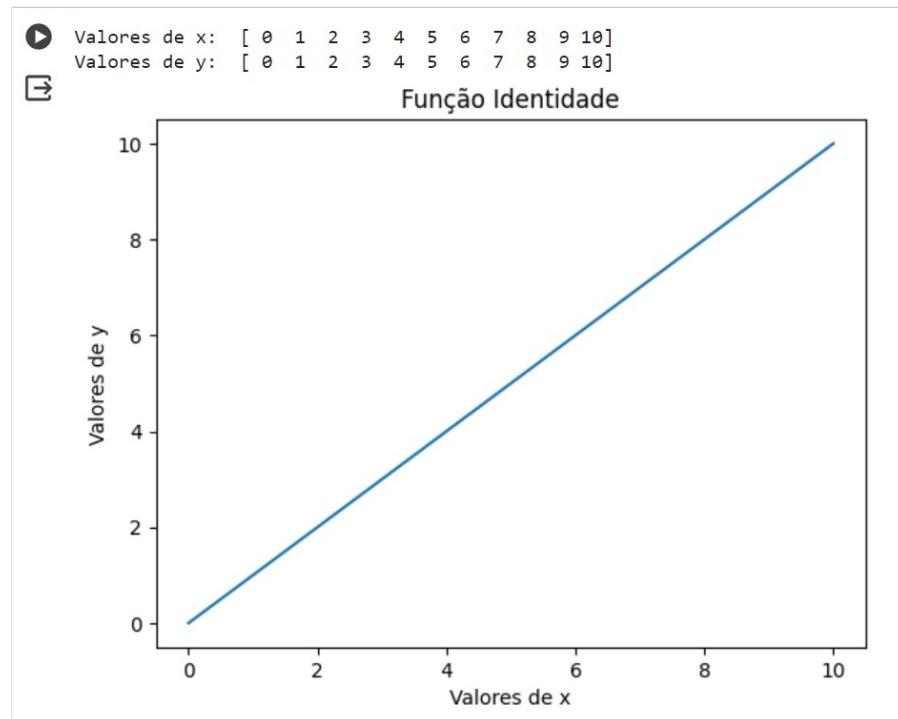
x = np.arange(0, 11, 1)
y = x

print('Valores de x: ', x)
print('Valores de y: ', y)

plt.plot(x, y)
plt.title("Função Identidade")
plt.xlabel("Valores de x")
plt.ylabel("Valores de y")
plt.show() # abre uma janela com o gráfico

```

**Figura 8:** Código de exemplo para importação de bibliotecas/pacotes e plotagem gráfica.



**Figura 9:** Resultado do código de exemplo com saída gráfica.

## SciPy

A biblioteca *SciPy - Scientific Python* é fundamental para computação científica, pois fornece algoritmos para otimização, integração, interpolação, problemas de autovalores (usados para a redução de parâmetros), equações algébricas, equações diferenciais e estatística amplamente aplicáveis em vários domínios. Ela estende o pacote *NumPy*, fornecendo ferramentas adicionais para computação de arrays e estruturas de dados especializadas, como matrizes esparsas e árvores.

## Matplotlib

A *Matplotlib - Mathematics Plotting Library*, é uma biblioteca abrangente para a criação de visualizações estáticas, animadas e interativas. Ela estende o pacote *NumPy*, fornecendo uma API orientada a objetos para incorporação de gráficos em aplicativos usando *kits* de ferramentas GUI de uso geral, como: *Tkinter* <<https://docs.python.org/3/library/tkinter.html>>, *wxPython* <<https://wxpython.org/index.html>>, *Qt* <<https://www.qt.io/qt-for-python>> / <[https://wiki.qt.io/Qt\\_for\\_Python](https://wiki.qt.io/Qt_for_Python)> ou *GTK* <<https://www.gtk.org/>>.

## Interfaces de Análise e Plotagem de Dados

As *interfaces* de análise e plotagem de dados, fornecem funções avançadas para a visualização e compreensão de conjuntos de dados complexos. Elas permitem estender funções de pacotes como os que vimos até aqui.

Existem várias opções de interfaces para a análise e plotagem (visualização) de dados, entre elas destacamos: *Seaborn* <<https://seaborn.pydata.org/>>, *scikit-learn*, *Basemap* <<https://pypi.org/project/basemap/>>, *Cartopy* <<https://pypi.org/project/Cartopy/>>; *Plotly* <<https://plotly.com/python/>>, *ggplot2* <<https://ggplot2.tidyverse.org/>> e a *PyGWalker* <<https://pypi.org/project/pygwalker/>>.

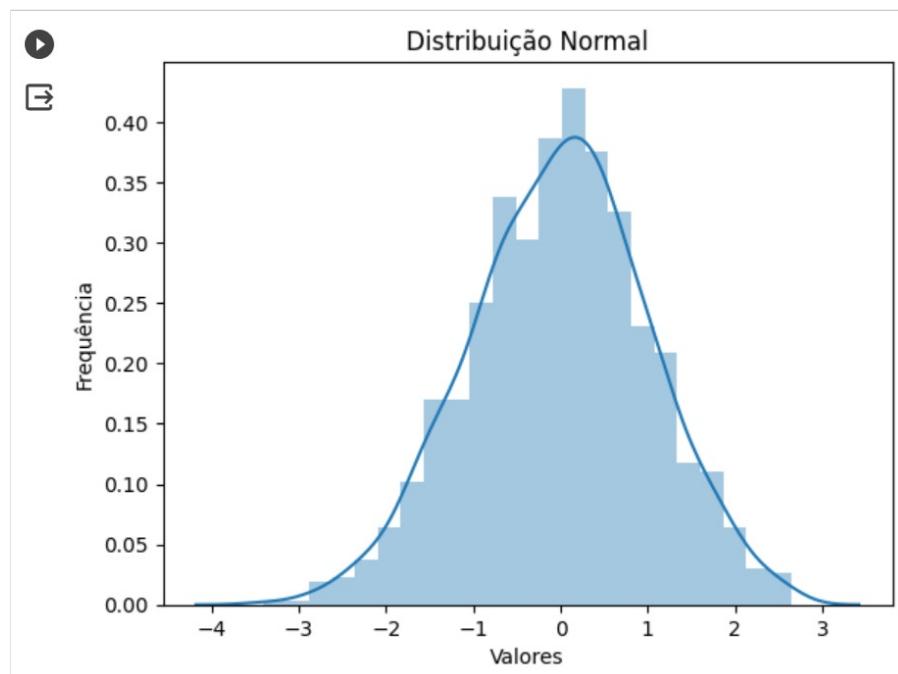
As Figuras 10 e 11 a seguir, ilustram um novo código para demonstrarmos a facilidade de uso da biblioteca *NumPy* e dos pacotes *Matplotlib* e *Seaborn* para a manipulação de objetos e exibição de seus resultados de forma gráfica.

```

1 # Importação das bibliotecas
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5
6 # Geração de x números randômicos distribuídos
7 # normalmente, com média 0 e desvio padrão 1
8 x = np.random.normal(0, 1, 1000)
9
10 # Plotagem do gráfico
11 # hist = Histograma, kde = Curva de Densidade
12 # rug = Não plotar as Marcas de Rugosidade
13 sns.distplot(x, hist=True, kde=True, rug=False)
14 plt.title("Distribuição Normal")
15 plt.xlabel("Valores")
16 plt.ylabel("Frequência")
17 plt.show()

```

**Figura 10:** Código de exemplo para importação de bibliotecas/pacotes e plotagem gráfica.



**Figura 11:** Resultado do código de exemplo com saída gráfica.

Vale ressaltar que as *interfaces* de extensão de funcionalidades para plotagem, análise e outras finalidades, possuem características particulares que podem ou não ser de seu interesse e/ou real necessidade.

O que queremos dizer com isso? Imagine por exemplo que você precise de uma funcionalidade para exibição de alguns gráficos uma página Web de modo estático, ou seja, sem que os gráficos se alterem em tempo de execução, mas você pode querer que o usuário final ao passar o ponteiro do mouse sobre diferentes partes do gráfico estático exibido, receba informações detalhadas sobre os dados da questão. Nesta situação uma *interface* específica pode ser melhor que outras.

Outras coisas que você pode levar em conta ao optar por uma *interface*, é que poderá priorizar a quantidade de código necessário para implementar o que necessita, a facilidade de entendimento do código, os tipos de gráficos disponíveis (2D, 3D, dinâmicos, estáticos etc.), o apelo estético, a forma de manipulação dos dados, a interatividade, ser de código aberto ou não e etc.

### Dica

Saber trabalhar com gráficos e realizar boas EDA - *Exploratory Data Analysis* ou Análises Exploratórias de Dados (AED), é fundamental para quem trabalha com IA e ciência de dados. Mas procure não se perder no mar de opções que existe para a plotagem de gráficos.

Há sim muitas ótimas opções, porém o mercado de trabalho exige cada vez mais pessoas que saibam resolver problemas de forma rápida e eficaz. Assim, procure conhecer bem e aplicar os recursos de boas bibliotecas, pois melhor que procurar aprender uma nova biblioteca a cada dia, e muitas vezes por algo que será pouquíssimo utilizado, é saber de fato utilizar e aplicar o que é realmente necessário.

### 3. INTRODUÇÃO AO PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

O NLP - *Natural Language Processing* ou Processamento de Linguagem Natural (PLN) é a área da ciência da computação (subárea da IA) que estuda como os computadores podem entender e processar a linguagem humana.

Os algoritmos de PLN procuram compreender a composição e a interpretação da comunicação humana, seja em textos escritos falados, de modo que a partir do instante em que um sistema for abastecido com um conjunto de entrada, o mesmo possa ser convertido de forma a conseguir-se uma interação fluída e o mais humanizada possível, geralmente na forma de um bate-papo.

A linguagem humana possui várias nuances, e para comprehendê-las de forma profunda as pesquisas em PLN as organizam em processamento. Os quais listamos os principais a seguir:

- **Nível Fonológico:** É responsável por analisar os significados dos sons da linguagem. Ele identifica os fonemas (unidades mínimas da fala) e como os mesmos são combinados para formar palavras.
- **Nível Fonético:** Realiza a análise física dos sons produzidos pelo aparelho fonador humano, debruçando-se sobre como os são produzidos, transmitidos e percebidos. Junto ao nível anterior, ambos são essenciais para entender a produção e a percepção da linguagem falada.
- **Nível Morfológico:** Analisa a estrutura das palavras, identificando os morfemas (unidades menores da palavra) e as palavras em acordo com suas classes gramaticais.
- **Nível Sintático:** Analisa a estrutura das frases, identificando as relações entre as palavras em uma frase e a estrutura da frase.
- **Nível Semântico:** Busca entender o significado da linguagem (sentido dos vocábulos de uma língua), identificando o significado das palavras e frases, e as relações entre as ideias expressas na linguagem.
- **Nível Pragmático:** Diz respeito ao contexto da linguagem (as entrelinhas). Reconhece o propósito da linguagem e identifica como a mesma é usada em um contexto específico.
- **Nível de Discurso:** Envolve o contexto da comunicação, revelando: quem, para quem e sobre o que se fala. Busca compreender a organização dos parágrafos, seções e a relação entre diferentes partes do texto.
- **Nível de Reconhecimento de Entidades Nomeadas:** O (REN) ou NER - *Named Entity Recognition*, busca identificar e classificar entidades como nomes de pessoas, locais, organizações, datas, etc. para extrair informações específicas e relevantes dos textos.
- **Nível de Análise de Sentimento:** Procura determinar a polaridade emocional dos textos, classificando se os mesmos expressam sentimentos positivos, negativos ou neutros.
- **Nível de Geração de Linguagem:** Objetiva a criação de textos *humanos-like*<sup>1</sup>, ou seja, a produção de respostas ou geração de textos automaticamente (como em chatbots ou sistemas de tradução).

Os níveis trazidos não são independentes e frequentemente estão interconectados. Para tratá-los o PLN utiliza técnicas e algoritmos que buscam alcançar uma compreensão abrangente e precisa da linguagem natural em seu significado e contexto. O uso desses níveis e/ou de algum outro que eventualmente seja necessário, irá variar dependendo da aplicação específica, desde assistentes virtuais a sistemas de análise de dados, tradução automática e geração e/ou sumarização de conteúdos.

<sup>1</sup> Refere-se a textos gerados por sistemas de PLN que imitam, em grande medida, a linguagem utilizada por seres humanos.

#### Representação de Textos

A representação de textos em PLN é o processo de converter um texto em uma tradução matemática que possa ser usada por um sistema e/ou aplicação. Essa representação é necessária para que se possa entender o texto e realizar tarefas como tradução, resumo, análise de sentimento, escrita criativa, etc. Existem diferentes abordagens para a representação de textos em PLN, uma das mais comuns é usar vetores de palavras.

Os vetores de palavras, são vetores de números que representam as palavras de um idioma. A representação de uma palavra em vetor de palavras é baseada em seu significado, contexto e frequência de uso.

Outra abordagem para a representação de textos é através do uso de RNAs. As quais podem aprender padrões complexos em c e serem usadas para representar textos de forma eficiente e eficaz, mesmo que estes sejam complexos ou ambíguos.

A escolha da abordagem para a representação de textos em PLN depende da tarefa específica que o sistema terá que realizar. F exemplo, as RNAs são geralmente mais eficientes para tarefas que requerem um entendimento profundo do texto, como a tradução automática. Já os vetores de palavras são geralmente mais eficientes para tarefas que requerem um entendimento menos profundo texto, como a classificação textual.

A representação de textos em PLN é uma área de pesquisa bastante ativa e em constante desenvolvimento. Pesquisadores estã sempre à procura de novas abordagens para representar textos e áudios de forma mais completa e compreensível.

Para que que um algoritmo de PLN consiga compreender a forma como se dá a comunicação entre os humanos, é preciso que s apliquem técnicas de [extração de características](#) (*features extraction*). Desse modo é possível traduzir [dados não estruturados](#) (independente da linguagem, textos e áudios) em [dados estruturados](#). Veremos a seguir como alguns modelos permite realizemos diferentes operações de representação para o entendimento de textos.

## Bag-of-Words (BoW)

O BoW é um método de representação de texto que converte cada documento num vetor de valores booleanos (0 e 1), indicando presença ou ausência de cada palavra em um vocabulário pré-definido. Apesar de simples, o método é bastante eficiente em repres texto para fins de classificação, sumarização e outras tarefas de PLN, todavia não leva em consideração o contexto ou a ordem das palavras, o que pode limitar sua precisão.

Um exemplo prático da implementação do BoW é ilustrado no código a seguir, no qual utilizamos a biblioteca *NLTK* e sua função tokenização de palavras ***word\_tokenize***.

```
# Importação da(s) biblioteca(s)
import nltk

# Representação de um texto como um vetor BoW
def bag_of_words(text, vocabulary):
    # Tokenize o texto.
    tokens = nltk.word_tokenize(text)

    # Cria um vetor Bow.
    bow = [0] * len(vocabulary)

    # Para cada palavra no texto,
    for token in tokens:
        # Verifique se a palavra está no vocabulário
        if token in vocabulary:
            # Se sim, marque a posição correspondente do vetor BoW como 1
            bow[vocabulary[token]] = 1

    return bow # Retorna o vetor BoW que representa o texto

# Vocabulário a ser usado
vocabulary = {
    "gato": 0, # Posição 1 do vetor
    "cachorro": 1, # Posição 2 do vetor
    "bola": 2, # Posição 3 do vetor
    "bicicleta": 3, # Posição 4 do vetor
    "Azul": 4, # Posição 5 do vetor
    "Bandeira": 5, # Posição 6 do vetor
    "carro": 6, # Posição 7 do vetor
    "viagem": 7, # Posição 8 do vetor
}

# Representação do texto
text = "O gato e o cachorro são bons mamíferos para levar em viagens de carro."
bow = bag_of_words(text, vocabulary)

print('\nO resultado encontrado foi . . .')
print(bow)
```

Em nosso exemplo, o vocabulário contém 8 palavras: "gato", "cachorro", "bola", "bicicleta", "Azul", "Bandeira", "carro" e "viagem". E o texto "O gato e o cachorro são b mamíferos para levar em viagens de carro." o qual é tokenizado em 15 palavras: "O", "gato", "e", "o", "cachorro", "são", "bons", "mamíferos", "para", "levar", "em", "viagens", "de", "carro" e "."

Quanto à implementação primeiro tokenizamos o texto, dividindo-o em palavras individuais. Em seguida um vetor **bow** foi criado mesmo tamanho do vocabulário. Para cada palavra no texto, o código verifica se a palavra está no vocabulário. Se sim, o código ma posição correspondente do vetor **bow** como 1 e ao final o vetor **bow** é retornado, e então o seu conteúdo é exibido em tela através saída **print()**.

No nosso caso o resultado da saída foi o que se segue:

### O resultado encontrado foi . . .

[1, 1, 0, 0, 0, 1, 0]

O resultado indica que as palavras "**gato**", "**cachorro**" e "**carro**" estão presentes no texto, enquanto as 5 demais palavras constam do vocabulário, não.

O método *BoW*, possui como vantagens a simplicidade e "eficiência", podendo ser usado em varias situações. Em contra partida não leva em consideração o contexto das palavras.

### TF-IDF

A TF-IDF - *Term Frequency-Inverse Document Frequency* ou a Frequência do Termo-Inverso da Frequência do Termo do Documento é uma medida estatística que avalia a relevância de uma palavra para um documento em uma coleção de documentos. Isso é multiplicando duas métricas: 1<sup>a</sup> - Quantas vezes uma palavra aparece em um documento; e 2<sup>a</sup> - A frequência inversa da palavra no documento em um conjunto de documentos.

Os algoritmos TD-IDF equilibram a importância das ocorrências das palavras, ou seja, eles dão menos importância para as palavras que aparecem mais e privilegiam as que ocorrem um número menor de vezes. Pode até parecer estranho, mas ao passarmos uma lista assim para um classificador, obteremos um modelo bem mais realista e menos enviesado.

### Word Embeddings

São representações vetoriais de palavras que capturam seu significado semântico. Tais representações são frequentemente usadas em tarefas de PLN, como classificação de texto, sumarização e tradução automática. Existem vários métodos para gerar *Word Embeddings* "Incorporação de Palavras", uma forma comum é usar RNAs para aprender as representações vetoriais. Essas redes são então treinadas em um conjunto de dados de texto, onde cada palavra é representada por um índice, com a rede aprendendo a mapear esses índices para vetores de números reais.

### Técnicas de PLN

São o conjunto de métodos e algoritmos usados para analisar e entender a linguagem humana. Essas técnicas são usadas em uma ampla gama de aplicações, como tradução automática, análise de sentimentos, processamento de perguntas e respostas e extração de informações em geral. A seguir citamos e descrevemos o funcionamento de algumas das principais técnicas de PLN e suas aplicações.

### Tokenização

É o processo de dividir um texto em *tokens* (unidades significativas de texto: como palavras, frases ou símbolos). É uma etapa fundamental em tarefas como na análise de sentimentos e na tradução automática.

### Stemming

É o processo de reduzir uma palavra a sua forma básica, independentemente de sua forma flexionada. Por exemplo, o stemming das palavras "correr", "correu" e "correrá" resultará na palavra "correr". Tal processo é útil para tarefas como indexação de texto e recuperação de informações.

### POS Tagging

*Part-of-Speech Tagging* é o processo de atribuir uma categoria gramatical a cada *token* em um texto. As categorias gramaticais mais comuns são os substantivos, os verbos, os adjetivos, os advérbios, os pronomes, as conjunções e as preposições. O *POS tagging* é útil para tarefas como análise sintática e tradução automática.

### Named Entity Recognition (NER)

É o processo de identificar entidades nomeadas em um texto. As entidades nomeadas são nomes de pessoas, lugares, organizações, produtos ou eventos. O *NER* é útil para tarefas como extração de informações e análise de sentimento.

### Chunking

É o processo de agrupar *tokens* em unidades sintáticas maiores, como frases ou cláusulas. O *chunking* é útil para tarefas como análise sintática e a tradução automática.

### Parsing

É o processo de analisar a estrutura sintática de um texto. O *parsing* é útil para tarefas como tradução automática e análise de sentimentos.

**Tabela 1:** Resumo dos processos utilizados nas técnicas de PLN descritas.

Técnica	Processo
Tokenização	Divide um texto em <i>tokens</i>
Stemming	Reduz uma palavra a sua forma básica
POS Tagging	Atribui uma categoria gramatical a cada <i>token</i>
NER	Identifica entidades nomeadas em um texto
Chunking	Agrupa <i>tokens</i> em unidades sintáticas maiores
Parsing	Analisa a estrutura sintática de um texto

A seguir implementamos um código em *Python*, o qual exemplifica como um texto base pode ser analisado e manipulado utilizando parte das técnicas de PLN vistas até aqui. Para tal utilizamos algumas funções disponíveis na biblioteca NLTK - *Natural Language Toolkit* <<https://www.nltk.org/>>.

```
# Importação de biblioteca(s)
import nltk
from nltk.corpus import treebank

# Pode ser necessário baixar pacotes adicionais da biblioteca NLTK
# nltk.download()

# Texto a ser trabalhado
texto = 'Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow.'
print('\nTexto bruto:', texto)

# Fazendo a separação do texto por pontos
print('\nTexto separado:', texto.split('.'))

# Dividindo o texto em frases. Aqui o algoritmo já identifica o texto em duas frases
frase = nltk.tokenize.sent_tokenize(texto)
print('\nIdentificação do número de frases:', frase)

# Dividindo o texto em "tokens" ou palavras - tokenização
token = nltk.word_tokenize(texto)
print('\nImpressão dos tokens:', token)

# Divisão dos tokens em classes com identificação sintática e semântica
classes_token = nltk.pos_tag(token)
print('\nImpressão dos tokens (Pos-Tagging - Part-of-Speech Taggin no formato Penn TreeBank):', classes_token)

# Detecção de entidades - agrupamento de tokens em unidades sintáticas
entidades = nltk.chunk.ne_chunk(classes_token)
print('\nImpressão das entidades (Chunking):', entidades)
```

As respectivas saídas **print()** do código são exibidas a seguir, e então comentadas:

#### Texto bruto:

Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a very nice fellow.

#### Texto separado:

['Mr', ' Green killed Colonel Mustard in the study with the candlestick', ' Mr', ' Green is not a very nice fellow', '']

#### Identificação do número de frases:

['Mr. Green killed Colonel Mustard in the study with the candlestick.', 'Mr. Green is not a very nice fellow.']}

#### Impressão dos tokens:

['Mr.', 'Green', 'killed', 'Colonel', 'Mustard', 'in', 'the', 'study', 'with', 'the', 'candlestick', '!', 'Mr.', 'Green', 'is', 'not', 'a', 'very', 'nice', 'fello

**Impressão dos tokens (Pos-Tagging - Part-of-Speech Taggin no formato Penn TreeBank):**

[('Mr.', 'NNP'), ('Green', 'NNP'), ('killed', 'VBD'), ('Colonel', 'NNP'), ('Mustard', 'NNP'), ('in', 'IN'), ('the', 'DT'), ('study', 'NN'), ('with', 'IN'), ('the', 'DT'), ('candlestick', 'NN'), ('.', '.'), ('Mr.', 'NNP'), ('Green', 'NNP'), ('is', 'VBZ'), ('not', 'RB'), ('a', 'DT'), ('very', 'RB'), ('nice', 'JJ'), ('fell', 'NN'), ('.', '.')]

**Impressão das entidades (Chunking):**

(S  
 (PERSON Mr./NNP)  
 (PERSON Green/NNP)  
 killed/VBD  
 (ORGANIZATION Colonel/NNP Mustard/NNP)  
 in/IN  
 the/DT  
 study/NN  
 with/IN  
 the/DT  
 candlestick/NN  
 ./.  
 (PERSON Mr./NNP Green/NNP)  
 is/VBZ  
 not/RB  
 a/DT  
 very/RB  
 nice/JJ  
 fellow/NN  
 ./.)

O código que implementamos apesar de simples, mostra algumas análises bastante importantes e elucidativas ao nosso tema, então ao que queremos demonstrar.

O programa recebe e exibe o texto: “**Mr. Green killed Colonel Mustard in the study with the candlestick. Mr. Green is not a nice fellow**”.

O texto é então separado por pontos por meio da função **split**.

A seguir o texto é analisado e separado por meio da função **tokenize.sent\_tokenize**, a qual identifica que o mesmo é composto duas frases e então as separa.

Exibe-se por meio da função **word\_tokenize**, todos os *tokens* encontrados no texto de forma separada, com o detalhe de que os elementos constantes da frase (inclusive pontuação) são classificados como *tokens* pela função.

Realizamos então um *POS Tagging* por meio da função **pos\_tag**, a qual se apoia no *Penn Treebank* <<https://cs.nyu.edu/~grishman/jet/guide/PennPOS.html>> que é o formato utilizado como base para o parser de Michael Cohen do Depto. de Computação e Ciência da Informação da Universidade da Pensylvania (algoritmo de *parsing* desenvolvido em 1997, baseado num modelo probabilístico que usa uma combinação de regras gramaticais e informações de contexto para atribuir *tags* de discurso a palavras em uma sentença). O mesmo é um *corpus* anotado na língua inglesa com informações sintáticas e semânticas num formato que consiste em *tags* de *part-of-speech* e informações sintáticas, apresentadas no formato de texto entre parênteses.

Nessa etapa do nosso exemplo, o algoritmo identificou semanticamente que ('Green', 'NNP') não é uma cor, mas sim que tal token respeito ao Senhor Green, ou seja, uma pessoa; ('killed', 'VBD') foi identificado como uma ação de tempo no passado; e ('very', 'RB') advérbio, o que está correto sintática e semanticamente.

**NNP** = Proper noun, singular ou Substantivo próprio, no singular

**VBD** = Verb, past tense ou Verbo, pretérito (no passado)

**RB** =Adverb ou Advérbio

Ao final aplicamos ao texto a técnica de *Chunking*, a qual foi executada por meio da função **chunk.ne\_chunk**, que o organizou em unidades sintáticas maiores. Em nosso caso: PERSON ou Pessoa e ORGANIZATION ou Organização, incluindo ainda as

informações semânticas levantadas também no *POS Tagging* anteriormente.

Vale ressaltar que o nosso exemplo foi desenvolvido com um texto em inglês e utilizando um *parser* para o mesmo idioma. Mas se preocupe, você poderá desenvolver seus testes e posteriormente aplicações mais completas, com base na nossa língua portuguesa qual no caso da biblioteca *NLTK*, você pode consultar como poderá utilizar, bem como quais os seus parâmetros mais adequados, acessando a página oficial da biblioteca em: <[https://www.nltk.org/howto/portuguese\\_en.html](https://www.nltk.org/howto/portuguese_en.html)>.

## Exemplos de aplicações de PLN

### Classificação de Texto

A classificação de texto é uma atividade muito importante em PLN, e na prática a utilizamos por várias vezes cotidianamente sem mesmo nos darmos conta. Um exemplo de classificação bastante comum, ocorre quando recebemos nossos e-mails “os quais contêm textos”, e a eles são aplicados um modelo de ML, o qual os classifica como “legítimos” ou “spams”. Esse tipo de estratégia é uma aplicação clássica já utilizada há bastante tempo por todos os provedores de mensageria.

Esse tipo de estratégia também pode ser aplicada para a classificação de documentos jurídicos e/ou comerciais, em categorias competições, documentos confidenciais, relatórios gerenciais, mensagens referentes a assuntos de interesse, como quais termos estão alta nas redes sociais, como determinados produtos estão sendo avaliados através dos *reviews* dos clientes que os adquiriram e outras opções.

A classificação também ocorre nos assistentes virtuais, os quais ao receberem instruções por voz, precisam classificá-las. Isso pedir para tocar uma música, é diferente de definir um alarme, que é diferente de pedir um resumo das notícias do dia relacionadas a esportes a motor.

São excelentes opções algorítmicas para classificação de textos: o SVM em especial para classificação binária multiclasse; o *Na Bayes* frequentemente usado para tarefas com dados de alta dimensionalidade; a Regressão Logística para tarefas de classificação binária e para estimativa probabilística de uma instância pertencer a uma determinada classe; as Árvores de Decisão para a divisão dados em subconjuntos com base em características, sendo as mesmas frequentemente usadas para tarefas de classificação e regressão; e as RNAs usadas para tarefas de classificação, regressão e PLN.

Quanto as bibliotecas mais indicadas para esse tipo de tarefa, destacam-se: a *scikit-learn* para classificações mais “simples”, o *TensorFlow* e a *PyTorch*<<https://pytorch.org/>> para classificações complexas.

### Sumarização

O processo de sumarização diz respeito a análise de um documento “texto” de origem, e dele extrair informações consideradas importantes e/ou principais, sendo ao final do processo gerado um novo documento “texto” sumarizado. Esse tipo de estratégia é útil em cenários onde possuímos uma quantidade grande dados “palavras” e um período de tempo restrito para consumi-las (lê-las).

Nesses casos uma ótima estratégia é que apliquemos ao documento base, algoritmos de sumarização. Os quais reduzirão o conteúdo inicial, selecionando e retirando pontos de interesse e os destacando no próprio documento ou os inserindo num documento alvo, o qual possuirá as partes selecionadas do documento original.

Dentro desse contexto podemos imaginar situações que envolvam a sumarização de notícias, de artigos científicos ou livros, de documentos jurídicos e etc. Para tal poderíamos então aplicar técnicas de extração e/ou abstração e por meio da aplicação de algoritmos que mensuram a frequência de palavras, a distância entre palavras, o algoritmo de Luhn, o algoritmo de Similaridade do Cosseno e outros, efetuar diferentes sumarizações textuais.

Quanto as bibliotecas para esse tipo de tarefa, excelentes opções são: a *sumy* <<https://pypi.org/project/sumy/>>, a *pysummarization* <<https://pypi.org/project/pysummarization/>>, a *gensim* <<https://pypi.org/project/gensim/>>, a *BERT summarizer* <<https://pypi.org/project/bert-summarizer/>>, a *spaCy* <<https://spacy.io/>> e a *NLTK*.

### Geração de Texto

A geração automática de texto tem se tornado uma atividade cada vez mais comum em muitas áreas. Isso porque vem sendo largamente implementada em aplicações de correção automática, em sistemas autônomos de diálogos como o que são realizados por *chatbots*, na sugestão de próximas palavras e até mesmo frases, na sugestão de pautas para temas e/ou campanhas de *marketing* em traduções automáticas, na criação de conteúdos como artigos, histórias e poemas, entre muitas outras possibilidades.

Os algoritmos mais populares e eficazes para a geração de texto, usualmente operam de duas formas, quais sejam:

- **Geração de texto baseada em regras:** Usam um conjunto de regras para gerar textos. As regras podem ser baseadas em gramática, sintaxe ou semântica. Apesar de relativamente simples, os algoritmos baseados em regras podem gerar texto em diferentes estilos específicos; e
- **Geração de texto baseada em ML:** Forma pela qual a geração de texto usa um modelo de linguagem para gerar textos. Esse modelo é treinado em grandes conjuntos de dados de texto e pode aprender a gerar texto semelhante ao texto do conjunto de treinamento. Essa forma de abordagem pode gerar textos mais criativos e informativos do que a anterior, todavia tendem a ser complexos de implementar, bem como a requererem recursos computacionais mais robustos e custosos.

Em relação as bibliotecas para esse tipo de tarefa, opções bastante robustas incluem: a *transformers* <<https://pypi.org/project/transformers/>> que realiza a geração de texto baseados em ML, incluindo o *GPT-3* (atualmente os algoritmos mais avançados para tal propósito), a *NLTK* e a *gensim*.

Algoritmos de PLN e em especial os para reconhecimento de fala NLU - *Natural Language Understanding*, como os utilizados em tradutores *online* e assistentes pessoais, vem sendo cada vez mais utilizados para diferentes propósitos.

Por necessitarem lidar com ambiguidades, situações probabilísticas e de raciocínio para interpretarem corretamente o mundo real que é bastante complexo. Essas classes de algoritmos são constantemente pesquisadas e aprimoradas, e isso tem se intensificado, com a ampliação e a consequente popularização de aplicações que suportam tais recursos, surgem também usuários cada vez mais exigentes em termos de precisão, eficiência e qualidade das aplicações de PLN.

## 4. INTRODUÇÃO À VISÃO COMPUTACIONAL

Visão Computacional é o campo que estuda o desenvolvimento de algoritmos e técnicas que permitem que computadores possa visualizar “enxergar” e interpretar “compreender” o mundo ao seu redor. Isso é feito por meio da análise de imagens e vídeos, com o objetivo de extrair informações relevantes, como objetos, pessoas, padrões e relacionamentos.

Como as RNAs, é também um campo complexo e desafiador, pois requer o desenvolvimento de algoritmos que sejam capazes com a imensa variabilidade do mundo real. No entanto, os avanços recentes em ML, especialmente em DL, estão possibilitando o desenvolvimento de sistemas de visão computacional cada vez mais poderosos e precisos.

Entre os principais componentes da visão computacional, destacam-se os que se seguem:

- **Aquisição de Dados:** Adquirir dados visuais é o primeiro passo na visão computacional. O mesmo envolve geralmente o uso de câmeras para capturar imagens ou vídeos.
- **Pré-processamento:** Antes de aplicar algoritmos de ML, é necessário pré-processar as imagens para melhorar a qualidade e a extração de suas características. Tal componente pode envolver a normalização, o redimensionamento e a filtragem das imagens.
- **Extração de Características:** É um componente crucial, o qual envolve identificar padrões, formas, cores e outras características relevantes nas imagens. Existem técnicas tradicionais, como o uso de filtros de imagem, e abordagens mais avançadas, como CNNs - *Convolutional Neural Networks* ou [Redes Neurais Convolucionais](#)<sup>1</sup>.
- **ML:** Após a extração de características, essas informações são alimentadas em algoritmos de ML. Modelos como SVM - *Support Vector Machines*, árvores de decisão e, mais comumente, CNNs são aplicados para aprender padrões e realizar tarefas específicas.
- **Classificação e Detecção:** Os modelos treinados são usados para classificar objetos em categorias específicas ou para detectar a presença de objetos em uma imagem. A classificação atribui uma categoria específica a uma imagem, enquanto a detecção indica onde localiza a presença de objetos.
- **Pós-processamento:** Com os resultados do modelo obtidos, podem ser necessárias etapas adicionais para refinar e melhorar os resultados, como a remoção de falsos positivos ou a melhoria da precisão.

### Saiba Mais

Sempre que você tiver um tempinho livre, que tal aprender um pouco mais sobre temas atuais relacionados a IA? Então, para isso uma ótima opção são os podcasts!

Nesse sentido, indicamos então um dos episódios do “**Podcast: Papo de IA**”, o qual temos certeza que além de útil será bastante motivador.

**Podcast Papo de IA - Episódio Nº 5:** Nova IA do TwitterX, alunos criam nudes com IA, Novidades do Google, Biden assina decreto para regulamentar IA e mais...

<sup>1</sup>Classe de RNAs do tipo *feed-forward* (RNAs que recebem dados na entrada e geram resultados na saída, sem nenhum *loop* de *feedback*, ou seja, os dados fluem de entrada para a saída em uma única direção) as quais são extensamente aplicadas no processamento e análise de imagens.

## Representação de Imagens

É o processo que envolve a conversão de uma imagem em um formato no qual a mesma possa ser interpretada por um computador. Este processo é necessário para que o computador possa realizar tarefas como classificação de imagens, detecção de objetos e

reconhecimento de faces. Existem vários métodos diferentes para representar imagens na visão computacional. Sendo os mais com os que se seguem:

- **Representação vetorial:** Método onde cada *pixel* da imagem é representado por um vetor de números. Os valores dos números do vetor representam a intensidade da cor do *pixel*.
- **Representação matricial:** Método em que a imagem é representada como uma matriz de números. Cada linha da matriz representa uma linha da imagem, e cada coluna representa um *pixel* da linha.
- **Representação por histograma:** Método pelo qual a imagem é representada estatisticamente por um histograma, o qual ilustra a distribuição de intensidades de cor da imagem. Esse método pode fornecer *insights* sobre o contraste e a dominância de cores.

Como em outros algoritmos, a escolha do método de representação mais adequado dependerá do problema específico que estiver sendo resolvido. Por exemplo:

A representação vetorial é geralmente usada para tarefas de classificação de imagens, pois permite que o computador compare imagens com base em seus valores de *pixels*; Já a representação matricial, é comumente usada para tarefas de detecção de objeto, pois permite que o computador localize objetos na imagem; Enquanto que a representação por histograma é mais usada para tarefas de reconhecimento de faces, pois permite que o computador compare-as com base em sua distribuição de intensidades de cor.

Estes são apenas alguns exemplos de como representar imagens por meio da visão computacional. Existem muitos outros métodos que podem ser usados, e como já citado outras vezes, a escolha mais acertada sempre dependerá do problema específico ao qual se necessita resolver.

## Técnicas de Detecção de Objetos

Em visão computacional as técnicas de detecção de objetos são divididas em duas categorias principais. Quais sejam:

- **Baseadas em características:** Identificam objetos na imagem procurando por características que são específicas (coloração, textura, pontos, formas geométricas “linhas e curvas” ou outras propriedades) do objeto a ser detectado. São as técnicas mais tradicionais de detecção de objetos, sendo relativamente simples de implementar e podendo ser usadas para detectar uma ampla variedade de objetos. Entre as suas estratégias mais comuns estão:
  - **Detecção de bordas:** Identifica bordas na imagem que podem ser usadas para identificar objetos. As bordas são regiões de imagem onde há uma mudança abrupta na intensidade de cor ou textura.
  - **Detecção de regiões:** Identifica regiões na imagem que podem ser usadas para identificar objetos. As regiões são grupos de *pixels* adjacentes que têm características semelhantes.
- **Baseadas em ML:** Usam ML para aprender a identificar objetos na imagem. Nela os modelos de ML são treinados em um conjunto de dados de imagens que já foram rotulados com as classes de objetos presentes nas imagens. Entre as suas estratégias mais comuns estão:
  - **Redes Neurais Convolucionais:** São altamente eficazes para tarefas de detecção de objetos. Onde por meio de modelos treinados, como o YOLO - You Only Look Once e SSD - Single Shot Multibox Detector, resultados excelentes podem ser obtidos.
  - **Filtros:** São utilizados para redução ou eliminação de ruídos. Por meio por exemplo da aplicação de filtros de gradiente, como o filtro Sobel, para destacar bordas na imagem e da detecção de contornos com o algoritmo do filtro Canny aplicado para encontrar limites nos objetos.

## Técnicas de Segmentação de Imagens

A segmentação de imagens é uma etapa crucial na visão computacional, nela o objetivo é dividir uma imagem em regiões significativas ou objetos. Existem várias técnicas de segmentação, cada uma com suas características e aplicações. A seguir listamos três das principais técnicas de segmentação.

- **Segmentação por Limiar (*Thresholding*):** Envolve a aplicação de um limite para converter uma imagem em escala de cinza para uma imagem binária (preto e branco), onde os *pixels* abaixo de um determinado limite são considerados como uma classe e os *pixels* acima do mesmo, considerados como outra.
- **Segmentação por Contornos:** Detectam contornos na imagem para delinear as regiões de interesse. Pode ser usada em conjunto com a detecção de contornos após técnicas como Canny. A seguir ilustramos o código de uma aplicação Python implementada no PyCharm, a qual operacionaliza a segmentação por contornos e logo a seguir na Figura 12, é ilustrada a imagem correspondente à execução do código.

```
# Importação das bibliotecas
import cv2
import matplotlib.pyplot as plt
```

```

# Leitura de uma imagem em escala de cinza
imagem = cv2.imread('Imagen_em_escala_de_cinza.jpg', cv2.IMREAD_GRAYSCALE)

# Aplicação do algoritmo Canny para detecção de contornos
bordas = cv2.Canny(imagem, 30, 100)

# Encontrar contornos na imagem
contornos, _ =
cv2.findContours(bordas, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

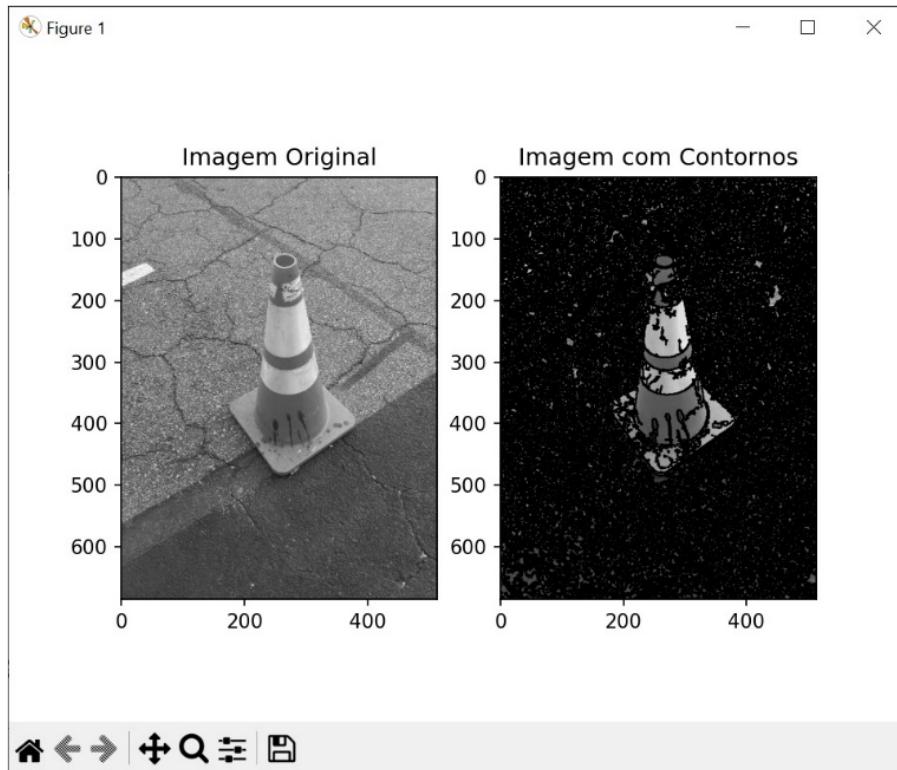
# Desenhar contornos na imagem original
imagem_contornos = cv2.drawContours(imagem.copy(), contornos, -1, (0, 255, 0), 2)

# Exibir imagem original e imagem com contornos
plt.subplot(1, 2, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagen Original')

plt.subplot(1, 2, 2)
plt.imshow(imagem_contornos, cmap='gray')
plt.title('Imagen com Contornos')

plt.show()

```



**Figura 12:** Resultado da utilização da biblioteca OpenCv com a aplicação da função Canny

- **Segmentação por Região (Watershed):** São algoritmos que segmentam imagens em regiões, usando para tal uma abordagem topológica para dividi-las por meio de gradientes. São especialmente úteis em casos onde há sobreposição de objetos.

A segmentação de imagens possui muitas aplicações, como por exemplo na agricultura para a detecção de pragas, na indústria de detecção de defeitos, em biometria para autorização de acesso, na micrografia para detecção de falhas estruturais, na análise de exames diversos para detecção de doenças, na restauração de documentos e em muitas outras situações.

### Dica

Bibliotecas como a OpenCV <<https://opencv.org/>> e a scikit-image <<https://scikit-image.org/>>, possuem várias funções robustas prontas para o tratamento de problemas de segmentação de imagens e merecem serem exploradas.

## Aplicações de Visão Computacional

Como vimos o campo da visão computacional é bastante amplo, e o mesmo traz inovações a todo tempo. Entre as aplicações as estamos expostos e que em grande parte das vezes se quer nos apercebemos que estamos sendo submetidos ou mesmo fazendo i estão os algoritmos de reconhecimento de rostos.

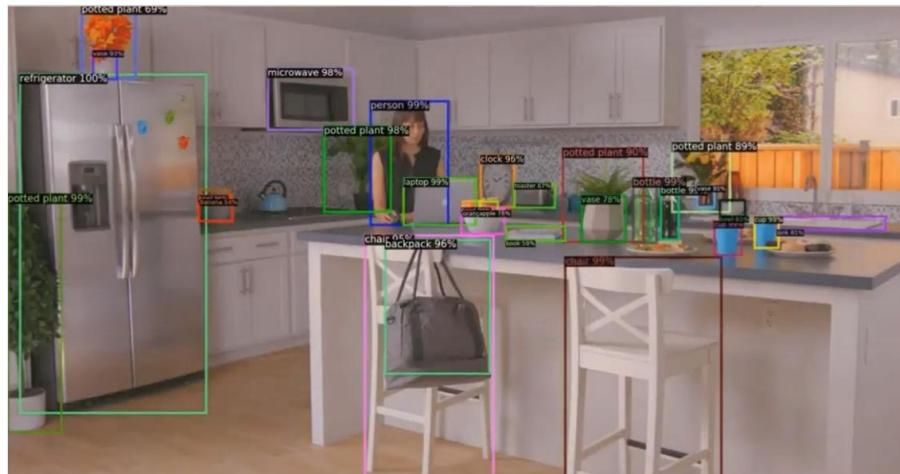
Um exemplo bastante conhecido e utilizado já há bastante tempo pelo *Facebook*, são os algoritmos que reconhecem usuários da social por meios das fotos e vídeos postados, os quais permitem a marcação automática de amigos, conhecidos e/ou mesmo por temas/campos de interesse.

Muitas são as pesquisas e experiências inovadoras nesta área, entre os bons esforços atuais, destacamos aqui a coleção de algoritmos de visão computacional *Facebook Detectron 2* <<https://ai.meta.com/tools/detectron2/>> e <<https://github.com/facebookresearch/detectron2/blob/main/README.md>>, os quais são desenvolvidos pela FAIR - *Facebook AI Research* da empresa *Meta Platforms, Inc.* para detecção e segmentação de objetos em tempo real, e que podem ser instanciados através da biblioteca *Python PyTorch* especializada em visão computacional e PLN, a qual foi originalmente desenvolvida pela *Meta AI* <<https://ai.meta.com/>> e que atualmente é parte do guarda-chuvas da *Linux Foundation* <<https://www.linuxfoundation.org/>>.

As Figuras 13, 14 e 15 a seguir, ilustram a capacidade dos algoritmos *Facebook Detectron 2* em reconhecer e classificar em tempo real, diferentes “objetos”.



**Figura 13:** Algoritmos *Facebook Detectron 2*, reconhecendo pessoas e seus corpos (através da percepção de membros: verifica se estão em sentadas, deitadas, se movendo, paradas, etc.)



**Figura 14:** Algoritmos *Facebook Detectron 2*, reconhecendo objetos por delimitação de região e com aplicação de algoritmos de probabilidade “grau de certeza” de reconhecimento dos objetos capturados



**Figura 15:** Algoritmos Facebook Detectron 2, reconhecendo objetos por delimitação de contorno e com aplicação de algoritmos de probabilidade “certeza” de reconhecimento dos objetos capturados

## 5. APLICAÇÕES DE INTELIGÊNCIA ARTIFICIAL

Sistemas ou produtos que usam IA para realizar tarefas ou fornecer serviços, são considerados aplicações de IA. Essas aplicações, como já discutimos, estão cada vez mais comuns e atualmente são encontradas nas mais diversas áreas, com destaque especial para saúde, as finanças, o marketing, nas indústrias, nos transportes, nos jogos e na programação (inclusive para criar novas aplicações). Seguir, elencamos alguns pontos em estabelecimento e/ou já estabelecidos na área.

### Na Saúde

- **Reconhecimento de Imagens Médicas:** Muitas aplicações identificam padrões em imagens médicas, como tumores, lesões e anomalias, o que contribui para que os médicos diagnostiquem e tratem doenças com mais precisão e rapidez.
- **IA Conversacional:** Vem sendo usada para desenvolver chatbots que fornecem atendimento a pacientes, o que pode contribuir para a melhoria da experiência e liberar tempo para os médicos.
- **Pesquisa Farmacêutica:** Auxilia na identificação de novos compostos farmacológicos e a avaliar a eficácia dos mesmos, o que acelera o processo de desenvolvimento de novos medicamentos.

### Nas Finanças

- **Previsão de Risco de Crédito:** Estão entre as aplicações mais antigas e estabelecidas da IA, vêm sendo usadas para prever riscos de inadimplência de clientes, com base em seus históricos de crédito e outros dados. Isso ajuda instituições financeiras a tomar decisões mais informadas sobre empréstimos e financiamentos, bem como a diminuir a probabilidade de não recebimento, o que consequentemente propicia a redução de taxas conexas.
- **Fraudes:** Amplamente usada para detectar e reconhecer fraudes financeiras, como fraudes em cartões de crédito (analizando padrões de compras, compras em locais ou com valores incomuns), na lavagem de dinheiro, em empréstimos, na solicitação de empréstimos referente a pagamentos de seguros, etc. o que leva tais instituições e seus clientes a minimizarem perdas financeiras.
- **Gestão de Carteiras:** Usadas para otimizar a gestão de carteiras de investimentos, com base em fatores como risco, de retorno, perfil do investidor. Isso pode ajudar os investidores a alcançar seus objetivos financeiros com mais eficiência e sem a necessidade de um especialista humano.

### No Marketing

- **Recomendação de Produtos e Serviços:** O ML pode ser usado para recomendar produtos e serviços aos usuários, com base em seus históricos de compras e navegação. Isso pode ajudar as empresas a aumentar as vendas e a satisfação de seus clientes.
- **Personalização da Experiência do Cliente:** Usado para personalizar a experiência dos clientes, com base em suas preferências históricas de interações, criando assim relacionamentos mais fortes e engajadores.
- **Análise de Dados:** Usado para analisar dados de marketing, como dados de vendas, dados de mídia social e dados de pesquisa, auxiliando as empresas a entender melhor seus clientes e a tomar decisões mais informadas sobre suas estratégias comerciais.

### Na Indústria

- **Controle de Qualidade:** Usado para identificar defeitos em produtos manufaturados, com base em dados de sensores e imagens, não só melhorando a qualidade dos produtos, mas também reduzindo o desperdício de materiais.
- **Otimização de Processos:** Usado para otimizar processos de manufatura, como a sequência de produção e a alocação de recursos. Isso pode ajudar as empresas a aumentar a sua eficiência e a produtividade.

- **Segurança:** Usado para detectar falhas em equipamentos e prevenir acidentes. Isso pode ajudar as empresas a proteger seus funcionários e instalações.

### Nos Transportes

- **Autonomia:** O ML é essencial para o desenvolvimento de veículos autônomos, sendo usado para controlá-los, identificar obstáculos e tomar decisões.
- **Gestão de tráfego:** Usado para melhorar a gestão de tráfego, com base em dados de sensores, câmeras e outros sistemas, contribuindo com a redução de congestionamentos e melhorando a segurança no trânsito.
- **Logística:** Amplamente usado para melhorar a logística, com base em dados de pedidos, estoques e entregas, reduzindo custos ao otimizando a entrega de produtos perecíveis e até organizando o acomodamento e a retirada de cargas.

### Nos Jogos

- **Criação de Conteúdo:** Auxilia na criação de conteúdos criativos, como imagens, cenários e objetos, textos/narrativas, músicas, animações, vídeos, etc. Por exemplo, pode auxiliar na criação de novas histórias e personagens, gerar músicas/trilhas sonoras ou aumentar a inteligência de NPCs - *Non-player Character* ou Personagens Não-jogáveis.
- **Melhoria da Criatividade:** Pode ajudar artistas a serem mais criativos. Por exemplo, na geração de ideias para novos projetos ou fornecer feedbacks sobre o trabalho criativo e oferecer sugestões para a superação de bloqueios criativos.
- **Melhoria da Produtividade:** Pode ser usada para automatizar tarefas repetitivas e aumentar a produtividade. Por exemplo, pode produzir relatórios, diálogos e textos para outros idiomas e até mesmo escrever códigos.

### Na Programação

- **Melhoria da Produtividade:** Pode ser usada para automatizar tarefas repetitivas e aumentar a produtividade. Por exemplo, pode produzir relatórios, traduzir diálogos e textos para outros idiomas e até mesmo escrever códigos.

Essas são apenas algumas aplicações práticas da IA em diferentes setores. A área continua a se desenvolver e amadurecer, e certamente teremos muitos mais exemplos de suas aplicações num futuro próximo. Todavia, antes de encerrarmos nosso curso, não poderíamos deixar de trazer as aplicações mais

afamadas atualmente na área da IA, ou seja, o ramo dos Geradores Transformadores Pré-treinados ou GPTs.

Os GPTs - *Generative Pre-trained Transformer* são ferramentas conversacionais de IA, as quais são treinadas em grandes conjuntos de dados de texto e código, sendo capazes de gerar texto, traduzir idiomas, escrever diferentes tipos de conteúdo criativo e responder perguntas de forma informativa, via-de-regra, por meio de *prompts*.

## ChatGPT vs. Bard

O ChatGPT <<https://chat.openai.com/>> da empresa de pesquisa e implantação de IA - OpenAI <<https://openai.com/>>, como o próprio nome indica, trata-se de um *Chat* referindo-se à sua funcionalidade de interação textual *Chatbot* e GPT refere-se a *Generative Pre-trained Transformer* ou Gerador Transformador Pré-treinado. Seu funcionamento se dá por meio de RNAs projetadas para processar e gerar texto natural por meio de um modelo de linguagem gerativo. A operação da ferramenta é baseada em ML e PLN, o que proporciona respostas assertivas e complexas devido à precisão lógica de seus algoritmos.

Como principal concorrente ao ChatGPT, existe o Bard <<https://bard.google.com/>>, o qual é um modelo de linguagem factual da Google AI <<https://ai.google/>>, laboratório pertencente ao leque de empresas do Google LLC.. O Bard também foi treinado em um enorme conjunto de dados de texto e código, e baseado nesse conhecimento, ele consegue gerar texto semelhante ao humano em resposta a uma ampla gama de *prompts* e perguntas, e apesar de ainda estar em desenvolvimento, já apresenta ótimo desempenho em diversas áreas.

Embora utilizem modelos de aprendizagem diferentes, ambas as ferramentas se baseiam em *prompts* de entrada por texto (no formato de *chatbots*) e se alimentam de informações coletadas da Internet. As quais são a base de dados de seus algoritmos, os quais buscam padrões e realizam o cruzamento das informações disponíveis.

Pela forma simples e facilitada como permitem que os usuários interajam, os *chatbots* tem se tornado cada vez mais populares. Isto é, elas utilizam de um modelo de linguagem "mais humana" apoiada na IA, denominado LaMDA - *Language Model for Dialogue Applications* ou Modelo de Linguagem para Aplicações de Diálogo.

Ambas as ferramentas respondem perguntas diversas e realizam tarefas como escrever uma carta de recomendação ou até mesmo códigos de programação. O Bard, por exemplo, se sobrepõe ao ChatGPT nesta função pois consegue escrever códigos em 20 linguagens de programação e ainda explicar como eles funcionam. A Tabela 2 a seguir, compara as duas ferramentas em termos de suas principais características.

**Tabela 2:** Principais características do ChatGPT e Bard

Características	<i>ChatGPT</i>	<i>Bard</i>
Tipo de modelo	<ul style="list-style-type: none"> <li>Generativo</li> </ul>	<ul style="list-style-type: none"> <li>Factual</li> </ul>
Forças	<ul style="list-style-type: none"> <li>Fornece informações precisas e relevantes</li> <li>Mais adequado para tarefas que exigem precisão e confiabilidade, como responder a perguntas ou traduzir idiomas</li> </ul>	<ul style="list-style-type: none"> <li>Gera formatos de texto criativos</li> <li>Mais adequados para tarefas que exigem criatividade e originalidade, como criar conteúdo criativo ou gerar formatos de texto personalizados</li> </ul>
Fraquezas	<ul style="list-style-type: none"> <li>Pode ser limitado em formatos de texto criativos</li> <li>Utiliza uma base de dados "fixa" - fixada atualmente até setembro de 2021</li> <li>Sua versão gratuita possui restrições de uso</li> </ul>	<ul style="list-style-type: none"> <li>Pode ser menos preciso em fornecer informações</li> <li>Utiliza o motor de busca do Google e, portanto, pode não estar totalmente atualizado quase que em tempo de execução</li> <li>Por vezes oferece opções (alternativas) de respostas</li> <li>Permite a leitura de respostas em voz alta</li> </ul>
Exemplo prático de uso	<ul style="list-style-type: none"> <li>Para responder a uma pergunta sobre a história Egípcia seria mais adequado, pois forneceria informações precisas e relevantes sobre o assunto</li> </ul>	<ul style="list-style-type: none"> <li>Para escrever um poema sobre o sentido da vida seria mais adequado, pois seria capaz de gerar algo criativo e original sobre o tema</li> </ul>

Ainda na linha das modernas aplicações de IA operadas por *prompts* e que tem seu conhecimento baseado na *Web*, temos os geradores específicos de conteúdo. Entre eles destacam-se os que geram imagens.

Nessa linha, destacamos o *DALL-E 3*<<https://openai.com/dall-e-3>>, o qual é uma aplicação que cria imagens a partir de descrições textuais. Para tal ele usa sua versão inicial de 12 bilhões de parâmetros do modelo *GPT-3 Transformer*, para interpretar entradas de linguagem natural e gerar imagens.

*DALL-E 3* pode criar imagens de objetos realistas, bem como objetos que não existem na realidade. Seu nome é uma junção de *WALL-E* (personagem "Robô" protagonista do filme de animação homônimo da *The Walt Disney Company* de 2008) e *Salvador Dalí* (pintor espanhol, 1904-1989).

A Figura 16 a seguir, ilustra uma imagem que geramos com a ferramenta em questão a partir de um *prompt* detalhado.



**Figura 16:** Imagem gerada com o DALL·E 3.

A imagem em questão foi gerada ao introduzirmos o *prompt* abaixo, o qual criou um conjunto de 4 imagens, das quais escolhemos a imagem em questão. A escolha dos *prompts* dependerá do que você deseja enfatizar nas suas imagens. Por exemplo, você pode escrever um *prompt* que pretende mostrar como a IA está sendo usada para tornar nossas vidas mais convenientes, mais seguras e/ou mais produtivas. Um possível *prompt* para tal poderia ser algo como:

#### **Prompt DALL·E 3**

[Image of a busy city street at night, with a variety of people and vehicles. In the foreground, a group of people are walking down the street, talking and laughing. One of them is wearing a pair of smart glasses that are displaying directions to their destination. In the background, a self-driving car is driving down the street, with no one inside. The car is using its sensors and cameras to navigate the city safely. Create the faces of people in the background in ultra realistic definition.]

Quanto à imagem que escolhemos valem as seguintes ressalvas, nosso *prompt* apesar de bem descritivo, ficou um pouco longo e que as ferramentas de IA GPTs tem procurado entender através de informações mais curtas. Outro ponto, é que o DALL·E 3 ainda tem restrições para diferentes dimensões e formatos de saída de arquivos.

Duas excelentes alternativas ao DALL·E 3, são: o Midjourney <<https://www.midjourney.com/>> e o Leonardo.Ai <<https://leonardo.ai>>

#### Desafio

Como desafio final, propomos que você escolha um dos tópicos modernos que mencionamos a pouco, e pesquise-o mais a fundo. Certamente você encontrará artigos, reportagens, livros, etc. sobre o mesmo. Depois de sua pesquisa, escreva um breve resumo do que você aprendeu e tente implementar de forma prática (preferencialmente com Python), um exemplo mesmo que seja simples.

Essa atividade será uma ótima maneira de aprimorar seus conhecimentos e se preparar para o futuro da IA.

Como vimos ao longo de todo o nosso curso, a IA é uma área em constante evolução, com novos desenvolvimentos acontecendo todo momento. Embora não seja fácil determinar onde as aplicações de IA irão acontecer, antes de finalizarmos nosso curso, elencaremos algumas áreas que têm sido cogitadas pelas comunidades de pesquisa, como tendências de crescimento.

- **IA quântica:** Área emergente que usa os princípios da mecânica quântica para criar algoritmos de IA mais poderosos.
- **IA explicável:** Campo de pesquisa que busca desenvolver algoritmos de IA que possam explicar suas decisões.

- **IA responsável:** Linha de pesquisa que busca desenvolver algoritmos de IA que sejam seguros e éticos.

A IA é provavelmente a área tecnológica que mais irá revolucionar o mundo, e além de atuar em novas áreas, certamente se aprimorará em muitas outras. A esta altura entendemos que isso se dará com mais ênfase em aplicações ligadas ao meio ambiente, segurança pública e privada, as novas formas de arte e entretenimento e na educação.

Todavia para que isso aconteça, certamente ela terá que continuar a se tornar mais poderosa e acessível, o que trará impactos (positivos e negativos) bastante significativos na sociedade.

Se a IA irá tirar empregos, sim! Mas também irá gerá-los!

Frases como: “*A inteligência artificial não vai tirar seu emprego agora. Uma pessoa utilizando inteligência artificial sim!*” e “*Você perde o emprego para a inteligência artificial, mas para quem usa a inteligência artificial melhor.*” já são bastante difundidas, e esse: (alertas) populares não devem ser subestimados.

## ATÉ BREVE!

Caro(a) aprendiz, chegamos ao fim da nossa jornada pelo Curso de Introdução à Inteligência Artificial. Cada tópico, cada linha de código e os exemplos trazidos, foram pensados com muito carinho para que você percorresse o que de fato lhe trouxesse significado. Esperamos sinceramente ter alcançado tal intento!

Como pôde perceber, a área da IA é fascinante e oferece muitas possibilidades. Mas é preciso ter sempre em mente que a mesma é multidisciplinar, o que exige não apenas esforço, mas também repertório e proficiência em diversas áreas. Então para ter sucesso como um profissional da área, sugerimos que sempre que possível, você busque aprofundar seus conhecimentos nas principais áreas que suportam, quais sejam:

- **Matemática, Estatística e Probabilidade:** A IA usa uma variedade de técnicas matemáticas, portanto, é importante ter uma boa base em suas áreas afins;
- **Programação:** A IA é uma área da computação, assim é essencial possuir um bom conhecimento de programação, pois isso permitirá o desenvolvimento de algoritmos e aplicações mais refinadas;
- **Redes:** Frequentemente é necessário que o uso de aplicações de IA ocorra em ambientes de rede confinados e restritos, então entender claramente como as redes funcionam é fundamental;
- **Internet:** Como ocorre em diversas atividades modernas, a IA é usada em uma ampla gama de aplicações na Internet, portanto é importante entender como a grande rede, suas ferramentas e as suas tecnologias operam; e
- **Plataformas IaaS, PaaS, SaaS e Simuladores:** Em grande parte de suas aplicações, a IA necessita de infraestrutura e ser especializados, assim é importante compreender como essas plataformas/serviços funcionam para implementar, testar (principalmente manter) as aplicações de IA em execução.

Quanta coisa, não é? Mas não se preocupe, pois os seus primeiros passos na área foram muito bem dados! Você já saiu na frenagem agora é a hora de fazer escolhas e continuar a se aprimorar na área.

Parabéns por chegar até aqui, desejamos que nosso curso lhe tenha trazido real significado e desejamos muito em breve, encontrar(a) aplicando IA na sua profissão!

Abraços e MUITO obrigado!

## Tags do conteúdo

Inteligência Artificial, IA

Representação de Conhecimento, Tomada de Decisão, Aprendizado de Máquina, Machine Learning, ML

Aprendizado Supervisionado, Aprendizado Não-Supervisionado, Aprendizado por Reforço

Redes Neurais Artificiais, Rede Neural, RNA, Aprendizado Profundo, Deep Learning

Árvores de Decisão, Regressão Linear, K-NN, K-Nearest Neighbors, Naive Baye

Natural Language Processing, NLP, Processamento de Linguagem Natural, PLN, Natural Language Understanding, NLU

Bag-of-Words, BoW, Term Frequency-Inverse Document Frequency, TF-IDF, Frequência do Termo-Inverso da Frequência do Termo do Documento, Word Embeddings

## Referências

- ARARIBÓIA, G. *Inteligência Artificial Um Curso Prático*. Livros Técnicos e Científicos Editora Ltda., 1988.
- BITTENCOURT, Guilherme. *Inteligência Artificial: Ferramentas e Teorias*. Editora da UFSC, Florianópolis, 1998.
- CHORAFAS, Dimitris N. *Sistemas Especialistas: Aplicações Comerciais*. McGraw-Hill, São Paulo, 1988.
- COZMAN, Fabio G. et al.. *Inteligência artificial: avanços e tendências*. Universidade de São Paulo. Instituto de Estudos Avançados, 2021.
- DEITEL, P. J. & DEITEL, H. M. *Python for Programmers: with Big Data and Artificial Intelligence Case Studies*. Pearson Illustration Edition, 2019.
- IBM. *Global AI Adoption Index 2021: New research commissioned by IBM in partnership with Morning Consult*. International Business Machines Corporation, April 2021. Disponível em: <[https://filecache.mediaroom.com/mr5mr\\_ibmnewsroom/191468/IBM's%20Global%20AI%20Adoption%20Index%202021\\_Executive\\_Summary.pdf](https://filecache.mediaroom.com/mr5mr_ibmnewsroom/191468/IBM's%20Global%20AI%20Adoption%20Index%202021_Executive_Summary.pdf)>. Acesso em 19 de setembro de 2023.
- KELLER, Robert. *Tecnologia de Sistemas Especialistas: Desenvolvimento e Aplicação*. Makron; McGraw-Hill, São Paulo, 1991.
- LUGER, George F. *Inteligência Artificial - Estruturas e Estratégias para a Solução de Problemas Complexos*. 4ª ed. Bookman, Alegre, 2004.
- MCKINNEY, Wes. *Python para Análise de Dados: Tratamento de dados com Pandas, Numpy e Ipython*. O'Reilly - Novatec, 201
- MUELLER, John Paul & MASSARON, Luca. *Inteligência Artificial Para Leigos*. Alta Books, 2019.
- NUÑEZ, Michael. *Meta AI unveils 'Seamless' translator for real-time communication across languages*. VentureBeat: The AI Impact Tour, December 01, 2023. Disponível em: <<https://venturebeat.com/ai/meta-ai-unveils-seamless-translator-for-real-time-communication-across-languages/>>. Acesso em 01 de dezembro de 2023.
- RABUSKE, Renato Antônio. *Inteligência Artificial*. Editora da UFSC, Florianópolis, 1995.
- REZENDE, Solange Oliveira. *Sistemas Inteligentes: Fundamentos e Aplicações*. Manole, São Paulo; Barueri, 2003.
- RICH, Elaine & KNIGHT, Kevin. *Inteligência Artificial*. 2ª ed. Makron Books, São Paulo, 1993.
- RIBEIRO, Carlos Henrique Costa. *Notas de Aula da Disciplina Inteligência Artificial*. Não Publicado, ITA, São José dos Campos, agosto de 1999.
- RUSSEL, Stuart J. & NORVIG, Peter. *Inteligência Artificial*. Editora Campus, Rio de Janeiro, 2004.
- WIGGERS, Kyle. *Executives discuss top challenges in deploying AI - and how to solve them*. VentureBeat: The AI Impact Tour, March 12, 2022. Disponível em: <<https://venturebeat.com/ai/executives-discuss-top-challenges-in-deploying-ai-and-how-to-solve-the-challenges/>>. Acesso em 14 de setembro de 2023.

Nota:

Design by Fábrica de Conteúdos Educação ©