



# Técnicas de Machine Learning

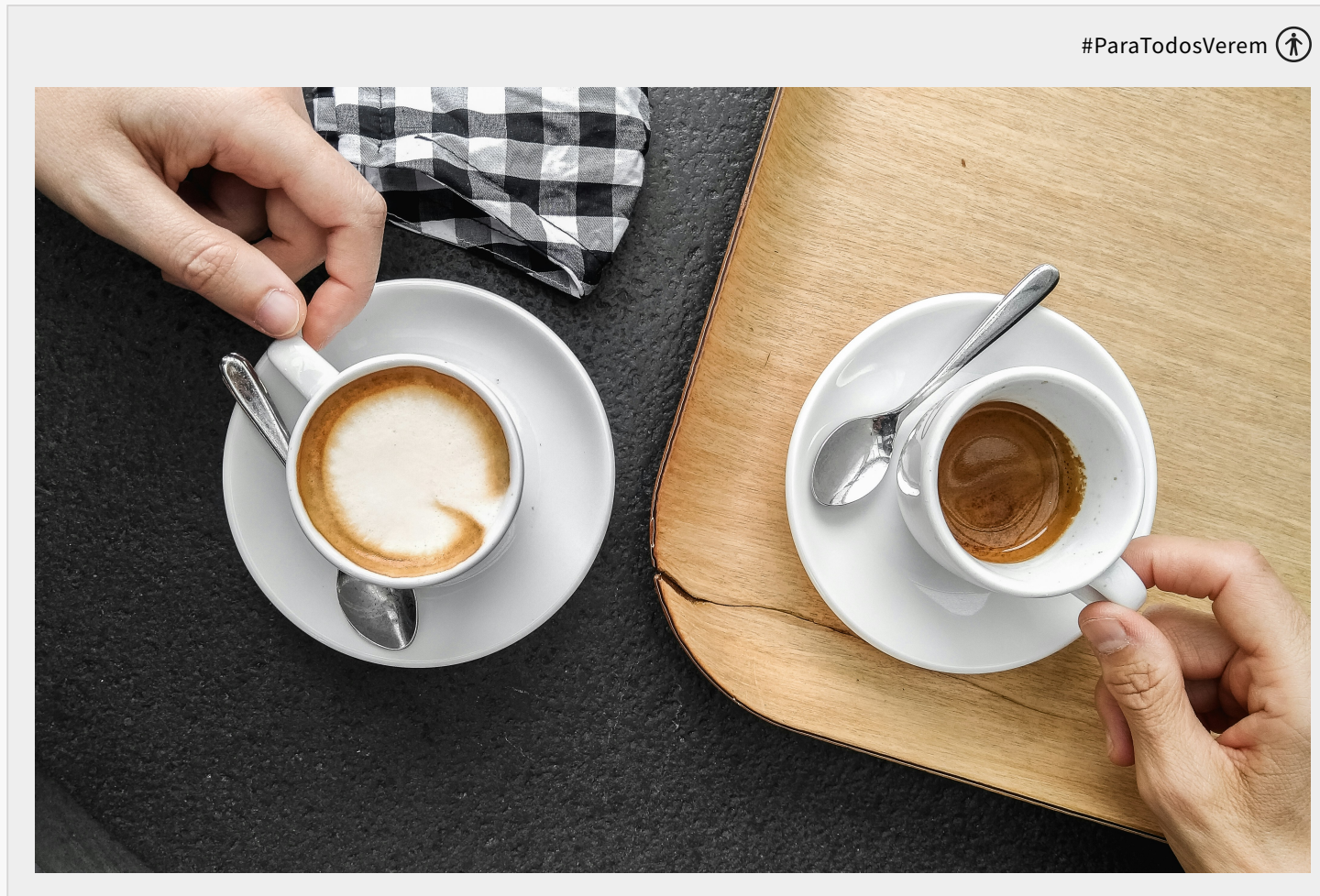
UNIDADE 04


Criando a sua própria IA: parte I

| TOMANDO UM CAFÉ

O famoso cafezinho é aquele evento esperado pelo trabalhador de TI. É o momento em que pausamos um pouco as nossas atividades, relaxamos por um momento, conversamos um pouco e nos preparamos para voltar à mão na massa (ou ao teclado). Trata-se de respiro e, talvez, de reflexão. Vamos usar o tempinho do nosso café para lembrarmos com calma sobre o que falamos até o momento e sobre o que deveremos fazer nesta semana, beleza?

Figura 1 – “Hmmm, café.”



#ParaTodosVerem 

Fonte: Giulia Bertelli / Unsplash

O que acha de juntarmos tudo o que vimos até agora?



Olha só: lá no início da disciplina tivemos como intenção explicar o que é *machine learning* (ML). Comentamos sobre a relação entre ML e IA e, ainda, falamos sobre alguns conceitos-chave. Esses conceitos incluem aprendizagem supervisionada, aprendizagem não supervisionada, classificação, regressão, *datasets*, atributos, *labels*, classes, instâncias, entre outros. Vimos como implementamos diferentes algoritmos em python utilizando o Jupyter, uma poderosa aplicação que usamos dentro de um navegador para criarmos algoritmos com *notebooks* e células.

Também comentamos que o desenvolvimento de algoritmos de ML em python pressupõe o uso de **bibliotecas**. Essas bibliotecas são algoritmos em python desenvolvidos por outras pessoas/universidades/empresas/organizações com funcionalidades que vão além do que é oferecido por padrão no python. Elas incluem, por exemplo, o scikit-learn, o pandas, o NumPy, o Prophet, o Tensorflow, o Keras e o Seaborn. Para ser honesto, o python é praticamente a linguagem de programação para trabalhos de ML justamente pela grande quantidade de bibliotecas com qualidade que estão prontas para uso.

Além dos conteúdos das unidades anteriores, também disponibilizamos alguns vídeos mostrando como trabalhamos com o Jupyter e com as bibliotecas de forma geral. Também disponibilizamos alguns *notebooks* exemplificando como essas bibliotecas funcionam na prática: principalmente o scikit-learn, pandas, Prophet, XGBoost e LightGBM. Caso não tenha consultado esses *notebooks*, sugiro que volte atrás e os consulte. Tente reproduzi-los, recriando o código do zero. Busque entender o que significa cada comando e lembre-se de que a **documentação** dessas bibliotecas abre todas as funcionalidades que elas possuem – geralmente, há exemplos que nos ajudam bastante a entender o seu funcionamento.

Note que o seu diferencial como profissional pode estar na sua **autonomia** em ler e compreender diferentes documentações – ainda que nem todas as documentações serão perfeitas: algumas serão desatualizadas e outras incompletas.

Vale lembrar mais uma vez que toda a caminhada que tivemos até o momento com os textos, vídeos e códigos foram para chegarmos aqui. A atividade da última unidade também serviu para que você pudesse entender como diferentes algoritmos são testados em diferentes *datasets*. Logo, agora você já deve ter todos os insumos necessários para criar do zero um bom algoritmo preditivo – em outras palavras, agora você criará a sua própria IA utilizando python.

## | O QUE SE ESPERA

Nesta semana, faremos a primeira atividade somativa da disciplina. Nela, os *datasets* foram retirados do Kaggle. O Kaggle é uma rede social/comunidade do Google voltada a cientistas de dados. Você pode encontrar lá outros exemplos de *notebooks* criados por profissionais experientes bases de dados para praticar seus conhecimentos e competições criadas por outras empresas para tentar resolver problemas reais – algumas dessas competições fornecem inclusive prêmios em dinheiro aos melhores colocados e podem ser uma boa oportunidade para que você aprenda formas diferentes de trabalhar com dados.

No momento, espera-se que você demonstre seus conhecimentos em três etapas do processo de ciência de dados: **preparação dos dados, seleção do modelo e treinamento do modelo**.

Figura 2 - Processo de ML



Fonte: Lenz, 2020 (adaptado).

## | COLETA DE DADOS

O passo de **coleta dos dados** não é coberto com profundidade nesta disciplina, uma vez que já receberá bases de dados prontas (isto é, os *datasets* em CSV ou em planilhas do Excel). Fora da disciplina e em ambientes profissionais, é comum prepararmos uma base sobre planilhas disponibilizadas para nós, em bancos de dados, utilizando SQL, em dados externos da internet ou uma mistura de todos esses cenários.

Observe que em todos os exemplos que vimos até o momento, a **carga de dados** é realizada utilizando a biblioteca pandas e utilizando principalmente o *read\_csv* e o *read\_excel*. Note que o pandas também disponibiliza outros tipos de leitura incluindo, por exemplo, uma leitura direto de um banco de dados em SQL (pelo *read\_sql*) e de tabelas em páginas HTML (pelo *read\_html*). Também é legal você saber que tabelas muito grandes – ou seja, em aplicações de *big data* – geralmente são processadas com o *PySpark* e o *Koalas*. Por outro lado, nesta disciplina acabamos nos concentrando no Pandas mesmo. Se a leitura der certo, teremos um **dataframe** pronto para manipulação. As funções *head*, *tail* e *describe* ajudam bastante para ver se a “cara” do *dataset* faz sentido com o que estávamos imaginando. Também existe outra biblioteca chamada *pandas-profiling* que pode ser útil nas análises de uma forma opcional.

- Bibliotecas recomendadas: *pandas*
- Funções recomendadas: *read\_csv* e *read\_excel* para leitura; *head*, *tail* e *describe* para analisar o *dataframe*.

## | PREPARAÇÃO DOS DADOS

Por **preparação dos dados** consideramos o **tratamento** e a **limpeza** dos dados para que o *dataframe* seja o mais útil (e utilizável) para um algoritmo. Aqui usamos as técnicas de preparação e análise exploratória de dados, bem como as técnicas de **aprendizagem não supervisionada**. Por essa razão, podem existir abaixo algumas menções a técnicas que não foram abordadas ainda na disciplina: não se preocupe, ao executar a atividade, somente trabalharemos com as técnicas que você já viu durante o curso, beleza?

Existem alguns passos que realizaremos nessa fase. Não entenda isso como um fluxograma ou uma sequência de passos: nem todas as ações são realizadas para **todos** os *datasets* e muito menos não seguimos sempre essa mesma ordem. O bom senso acaba apoiando bastante aqui: se coloque no lugar de um algoritmo preditivo: o que o **ajudaria** a aprender? O que **atrapalharia** o seu aprendizado sabendo que ele busca, antes de mais nada, a **reconhecer padrões**? Dito isso, alguns dos passos seriam os seguintes:

1. Existe algum identificador como nome, CPF, código de matrícula ou quaisquer outros identificadores únicos – ou seja, que só servem para **identificar** cada instância? Se existe, pode ser que valha a pena removê-lo, já que o algoritmo não teria muito o que aprender com vários códigos que

não ajudam a identificar nenhum padrão.

- Existem colunas com grande parte dos **dados nulos**? Se temos poucas amostras para uma coluna em específico, pode ser que valha a pena removê-la. Se somente temos **algumas** instâncias com dados nulos para um certo atributo, pode ser que seja melhor somente **imputar** dados ausentes – seja com um valor constante, uma média ou, ainda, usando outras técnicas para inferir o valor faltante.
- Existem colunas categóricas? Se estiverem como um tipo texto, pode ser necessário convertê-las antes para números (ex.: “Paraná” para “1”). Técnicas como o LabelEncoder fazem isso. Se já estiverem como números pode ser interessante convertê-las em colunas binárias via *one-hot encoding* (OneHotEncoder e o get\_dummies fazem isso). Note, por outro lado, que é importante **somente** fazer isso nas colunas categóricas que precisam ser convertidas.
- Existem **outliers/anomalias** (isto é, valores que estão muito abaixo ou muito acima do normal)? Analisou, com o uso de histogramas, o comportamento das colunas de dados numéricos? Esses valores muito acima ou muito abaixo do normal estão corretos, ou erros de dados são possíveis? Caso sejam erros, valeria a pena remover essas instâncias contendo os dados ou somente ajustar os valores?
- Os atributos numéricos estão normalizados? Técnicas como o StandardScaler e RobustScaler podem ser úteis.
- Pode ser que após todo esse processo tenhamos até mesmo mais atributos do que instâncias. Seria válida a aplicação de técnicas de aprendizagem não supervisionada como **seleção de atributos** ou de **extração de atributos**?

A ideia aqui é a de termos um *dataset* que esteja o mais “limpo” possível para que um algoritmo aprenda: lembra-se da máxima do *garbage in, garbage out*? É justamente esse *garbage* que tentamos resolver nessa fase.

- 
- Bibliotecas recomendadas: scikit-learn e pandas.
  - Funções recomendadas: LabelEncoder, OneHotEncoder e get\_dummies para conversão de colunas categóricas; StandardScaler e RobustScaler para normalização; SelectKBest, VarianceThreshold e SelectPercentile para a seleção de atributos; PCA para a extração de atributos.

## | SELEÇÃO DO MODELO

Tão logo temos o nosso dataset “saneado” (isto é, limpo) podemos começar a preparar o terreno para a **aprendizagem supervisionada**. Qual algoritmo escolheremos?

- Identifique a coluna que deseja *prever* (também conhecida em alguns casos como *classe/label*).

2. Olhando o comportamento dos dados, você entende que se trata de um problema de **classificação**, **regressão** ou **previsão de séries temporais**?
3. Após entender o tipo de problema, você escolheria **qual técnica** para resolvê-lo? Em base do seu histórico prévio e analisando as **cinco tribos** que comentamos na unidade anterior, qual algoritmo poderia ser mais rápido e ter resultados mais próximos da realidade?



### IMPORTANTE

**Atenção:** aqui dificilmente haverá uma técnica que sempre funcionará melhor para todos os casos. Não há nada de errado em você ser “fã” de alguma técnica como o LightGBM, XGBoost, SVM, RandomForest e outros, mas **jamaiz** assuma que alguma dessas técnicas seria a melhor. Teste outras técnicas – busque conhecer outras técnicas. Leia as documentações e teste também outras configurações para a mesma técnica. Note que elas possuem diferentes parâmetros que também podem resultar em um aprendizado completamente diferente.

- Bibliotecas recomendadas: [scikit-learn](#), [XGBoost](#), [LightGBM](#) e [Prophet](#).
- Funções recomendadas: *múltiplas* (abordadas na unidade anterior).

## | TREINAMENTO DO MODELO

Após preparar o seu *dataset*, entender o tipo de algoritmo que usáramos e qual técnica empregáramos poderemos, finalmente, treinar o modelo.

O processo de treinamento é essencialmente dividido da seguinte forma:

1. Primeiro, separamos o nosso *dataset* entre uma base de **treino** e de **teste**. Isso é importante para que possamos entender como o algoritmo funciona na prática, evitando, assim, riscos de *overfit*. Essa divisão entre treino e teste é feita geralmente com a função **train\_test\_split**. Existem também em alguns casos o treinamento por validação cruzada, ou cross-validation (CV). O CV poderá ser abordado em maior profundidade em oportunidades futuras. A divisão é geralmente 60 a 80% para treino e 40 a 20% para teste.



a. Principalmente em casos de classificação, poderemos ter problemas de classes desbalanceadas em que essencialmente temos muitas instâncias de uma classe e poucas de outra. É interessante você saber que existem técnicas de *bootstrapping* para resolver isso como o SMOTE, por exemplo. Por outro lado, esse tipo de problema específico não será trabalhado nesta disciplina.

2. Realizamos então o **treinamento** do modelo com o *dataset* de treino. Note que na maior parte dos exemplos da unidade anterior o treinamento ocorria com a função *fit*.

3. Após o treinamento estar finalizado, podemos já prever outros casos. A base de **teste** pode ser utilizada para entender e comparar o que o algoritmo **previu** (com a função *predict*) com a realidade. A utilidade disso está no fato de conseguirmos entender e comparar o que o algoritmo previu com o que ele deveria ter previsto de fato – em novos casos não conseguiríamos fazer essa comparação. Sendo assim, o momento para verificar se o algoritmo está ou não funcionando como deveria é agora.

---

Sendo assim, nessa etapa temos:

- Bibliotecas recomendadas: [scikit-learn](#), [XGBoost](#), [LightGBM](#) e [Prophet](#).
- Funções recomendadas: *train\_test\_split* (para dividir o *dataset* entre treino e teste); *fit* (para efetivamente treinar o algoritmo); e *predict* (para gerar previsões).

## | AVALIAÇÃO DO MODELO E PASSOS SEGUINTE

No momento, nos contentamos com a geração de novas previsões. A partir das próximas unidades, aprenderemos a verificar a *performance* do modelo – por *performance* entendemos basicamente a resposta para a seguinte pergunta: “O modelo é bom?”. Para isto, utilizamos um conjunto de métricas estatísticas e visuais que nos apoiam na tomada de decisão.

Vimos anteriormente alguns aspectos da aprendizagem não supervisionada e, depois, da aprendizagem supervisionada. Por outro lado, vimos ambos os passos de forma separada. Agora é a hora de vermos como ambos os conceitos se juntam para criarmos a nossa própria IA. Vamos juntos?



Indo até a metade do caminho



## | CONCLUSÃO

Nesta unidade, atendemos a dois pontos principais: o primeiro foi explicar o funcionamento do processo de treinamento de algoritmos de ML até o passo de *treinamento do modelo*. O segundo foi explicar como ocorre cada um dos pontos desse processo a partir do que vimos nas unidades anteriores.

A leitura dos textos mais os vídeos e códigos que já disponibilizamos anteriormente nos capacitam a realizar a atividade avaliativa da disciplina. Logo, nesta semana, passaremos mais tempo desenvolvendo as atividades do que qualquer outra coisa. Vamos lá?



Fonte: Jefferson Santos / Unsplash

## | REFERÊNCIAS BIBLIOGRÁFICAS

HUYEN, C. **Projetando sistemas de *machine learning***: processo interativo para aplicações prontas para produção. Rio de Janeiro: Editora Alta Books, 2024.

LENZ, M. L. *et al.* **Fundamentos de aprendizagem de máquina**. Porto Alegre: Sagah, 2020.

RUSSELL, S.; NORVIG, P. **Inteligência artificial**. 4 ed. Rio de Janeiro: LTC, 2024.



© PUCPR - Todos os direitos reservados.