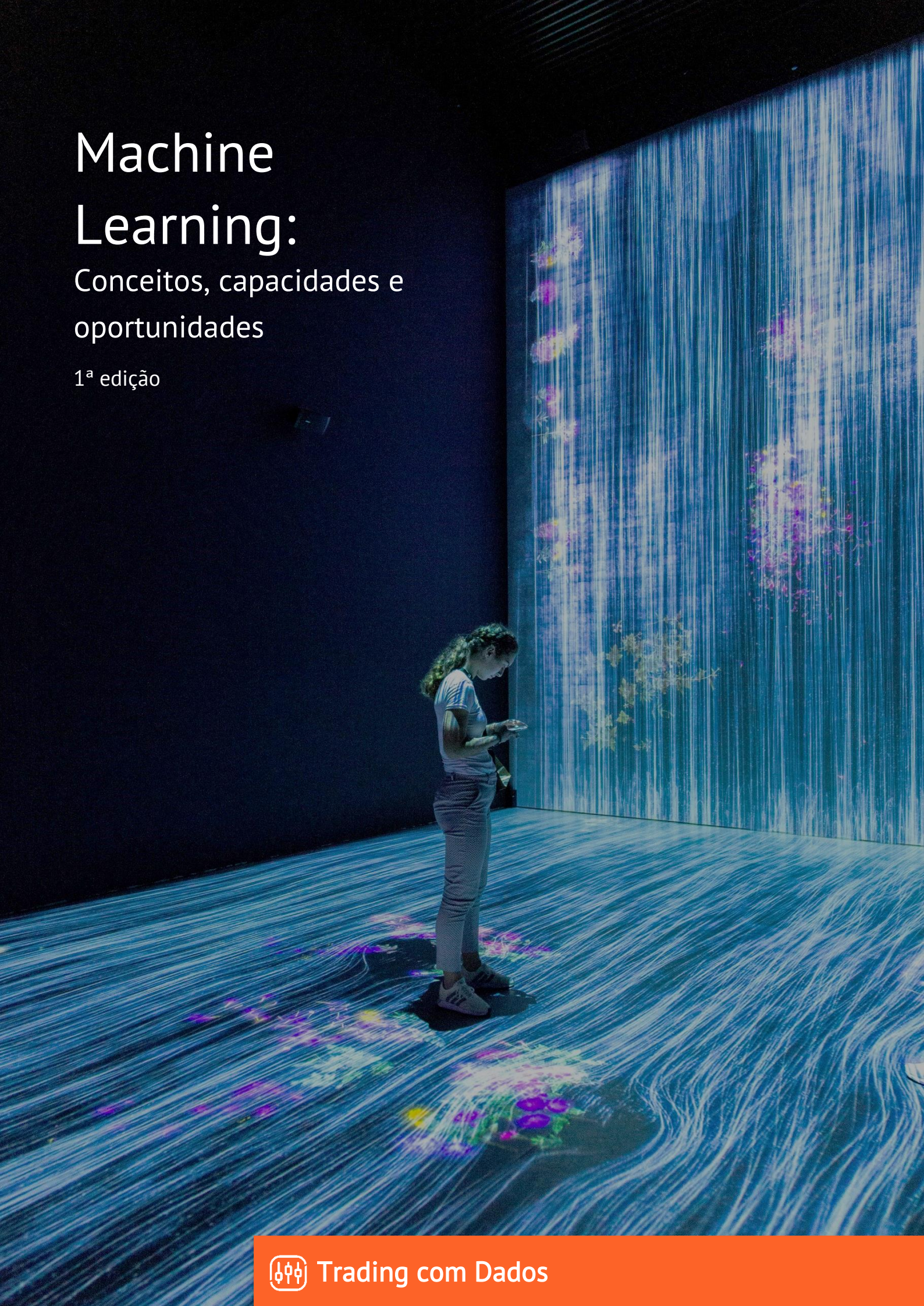
Machine Learning:

Conceitos, capacidades e oportunidades

1ª edição



Trading com Dados

Sobre o autor

Jarbas Carriconde é Cientista de Dados na XP Inc, Engenheiro de Automação formado pela Universidade Federal de Rio Grande - FURG. Entusiasta de Ciência de Dados aplicada ao mercado Financeiro e a negócios. Possui experiência em Ciência de Dados para o setor Varejista, obteve certificação ANCORD e foi sócio de Escritório de Agentes Autônomos. Em 2014 participou do quadro Jovens Inventores do Caldeirão do Huck com mais dois colegas através do projeto "Bafômetro de Caminhão". Apaixonado por tecnologia, economia, ciência de dados, esportes e bons livros.

Machine Learning: Conceitos, capacidades e oportunidades foi escrito por Jarbas Carriconde e editado e divulgado pela Trading com Dados.

Para enviar comentários e dúvidas, entre em contato pelo: contato@tradingcomdados.com

Fotos de
Mahdis Mousavi
Frankie Chamaki
Dennis Kummer
Christopher Hautier
Margareth Weir
Jude Back
Niko
Joshua Sortino
Solen Feyissa
Koleen Gladden
Cade Prior

Sumário

Sumário
(continuação)

Introdução

" Sem dados você é apenas mais uma pessoa com uma opinião.

W. E. Deming

O círculo do Zorro

Se você chegou até aqui provavelmente está na busca pela iniciação na área de ciência de dados, e mais do que isso, através de uma linguagem simples e acessível. Esse é provavelmente um dos *ebooks* mais didáticos que você encontrará escrito em língua portuguesa sobre o tema. Aqui serão explicitados os principais conceitos necessários para que você seja introduzido ao mundo da ciência de dados, e não, não são necessárias resoluções de provas matemáticas complexas. Aqui será exposto estritamente o necessário, e a "real" sobre o que é mais importante saber no início da sua trajetória nessa área fascinante, além de algumas "verdades insofismáveis" que ninguém te conta quando você compra aquele curso online de R$ 20,00.

Trago aqui conceitos traduzidos da maneira mais didática possível. Igual muitos de vocês, não sou um gênio, e a consolidação desses conceitos fora feita em cima de muito esforço. Aqui ilustro nada mais do que a representação da minha metodologia de estudo, pego conceitos de diversos lugares acerca de um assunto e os escrevo com minhas palavras para fins didáticos, já pensando em facilitar o caminho quando for revisar o assunto para solidificar esses conceitos, prática essa que sempre foi efetiva em meus estudos.

Uso dessa etapa introdutória para citar "o círculo do zorro", teoria explorada no livro "O Jeito *Harvard* de Ser Feliz" de Shawn Achor (cuja leitura recomendo fortemente), e que cabe perfeitamente para quem deseja explorar a ciência de dados. Isso pode ser feito por qualquer um, seja lá qual for a sua formação, existem cientistas de dados advindos de diversas áreas, inclusive aquelas sem ênfase em ciências exatas.

É normal estar perdido em um mar de informação, principalmente quando se lida com tecnologia, e tenham a certeza que "trabalhar na coisa certa provavelmente é mais importante do que se empenhar". Seguindo esse *ebook* tenha a certeza de que poderá se empenhar na coisa certa nesse novo início de ciclo do aprendizado. E quando nos empenhamos descoordenadamente corremos o risco de não suportar a carga de conhecimento para o qual não estamos ainda preparados, o estresse se acumula rapidamente e a euforia dá lugar ao desânimo. O círculo do Zorro é trazido do famoso personagem, que ao ser treinado por Don Diego, é colocado dentro de um círculo menor que está dentro de um círculo maior, e ouve de seu mestre: "Este círculo será o seu mundo. Sua vida toda. Não existe nada fora disso até eu dizer." E só depois de dominar o primeiro círculo, pouco a pouco, é que o discípulo começou a se transformar no lendário Zorro, assumindo o lugar de Don Diego. Ou seja, você deve restringir o seu foco antes de tudo! Ainda mais se tratando de tecnologia, esse *ebook* é o seu círculo pequeno em ciência de dados. Depois de ver esses esforços produzirem resultados, você deterá a confiança e o conhecimento para expandir o seu círculo, conquistando cada vez mais uma área maior, de forma gradativa e sustentável.

Verdades insofismáveis

Ainda introdutoriamente, insiro aqui algumas observações que gostaria que tivessem me dito quando comecei a estudar a área! Esperto é aquele que aprende com o erro dos outros, mas o mais importante eu creio que seja andar sempre com os pés no chão. Aqui trago informações que podem aumentar a sua visão sistêmica do assunto, ao invés de simplesmente escalar uma montanha no escuro, sem saber exatamente onde isso vai dar. Assim sendo, seguem as principais informações:

- Nossa busca é por Ciência de Dados, e não matemática avançada. Embora matemática seja importante, por vezes vemos pessoas se atendo a detalhes que não são da nossa alçada. Por exemplo, quando falarmos de árvores de decisão, veremos que não precisamos saber a fórmula da entropia, mas apenas que o algoritmo procura o recurso que lhe dará o maior ganho de informação. Evidentemente que a quem interessar o aprofundamento, seja por conhecimento ou para fins mais acadêmicos, sabedoria sempre será bem-vinda.
- Quando você estudar algum material sobre Ciência de Dados, implicitamente se dará uma importância, e até uma glamourização, em cima dos modelos. Quem aqui nunca ouviu falar do famoso "*Deep Learning*"? Quando na verdade poderia ser chamado de *perceptron* multicamadas. O que eu quero dizer é que essa é sim uma parte técnica importante e deve se ter um bom entendimento de como os modelos funcionam e em quais casos eles tendem a funcionar melhor. Ainda que não exista regra, sempre é aconselhável fazer todos os testes e verificar de fato qual performa melhor. Mas o que ninguém te fala é que a parte da modelagem não despende nem 20% do seu tempo, e que a parte mais importante está no tratamento de dados, Análise Exploratória de Dados e *Feature Engineering*. Esses últimos dois aspectos dizem respeito ao intangível, ao conhecimento dos dados e do fenômeno que se deseja modelar. Por exemplo, nas vendas de um site de *e-commerce*, o cientista deve entender as nuances do negócio para entender quais recursos (*features*) são mais importantes de se implantar no modelo.
- Essa é a parte "artística" da profissão e isso diferencia aqueles que se destacam de verdade. Já que a parte técnica, de conhecimento de modelos e outros recursos bastante usados, tornam-se apenas um pré-requisito e não um diferencial, quando se fala em Ciência de Dados se supõe que um profissional da área obrigatoriamente já tenha todos os conhecimentos técnicos que você será submetido aqui, e o diferencial sejam os "*insights*" e como serão estruturados e implantados recursos com grande poder preditivo para o alvo que se deseja predizer. De nada adianta modelos altamente complexos em cima de recursos (dados) sem poder preditivo, e às vezes um modelo simples pode ser uma boa solução em dados sem ruídos e com grande poder preditivo, ou seja, que sejam capazes de representar bem aquele fenômeno a ser previsto.

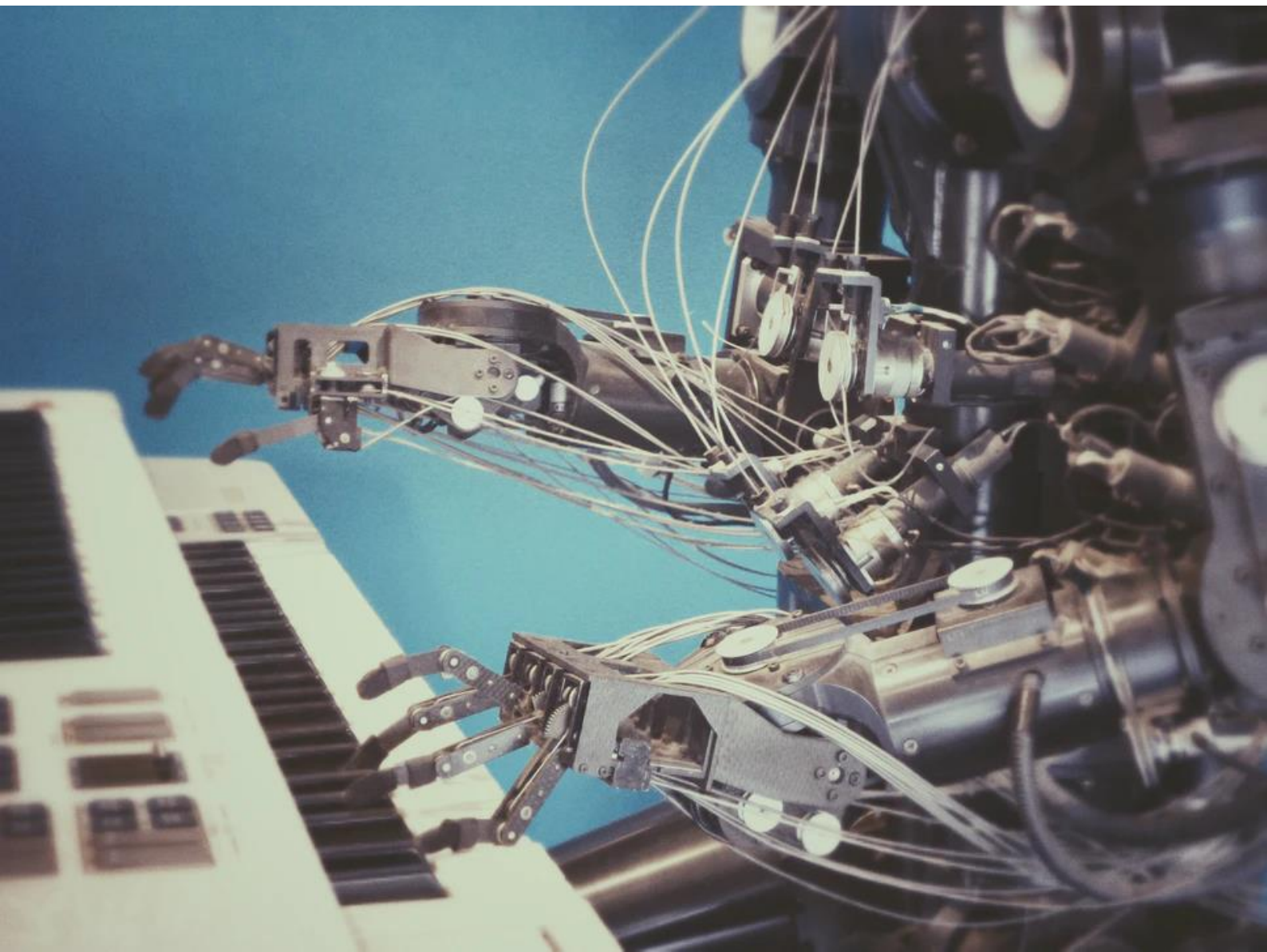
Fim do início

Entenda: o "processo" deve ser respeitado. Esse é o ponta pé inicial, que deve ser lido, relido, voltar aqui quando se deparar com dúvidas, mas o principal é que essa seja uma porta escancarada para um caminho sem volta. O mundo caminha a passos largos para uma cultura cada vez mais analítica, e se você que não quer ficar de fora disso, enxergue nessa leitura o início amigável de um caminho que mostra que todo conhecimento pode ser acessível na realidade atual, e nesse caso a disciplina vale muito mais que qualquer talento natural. A pessoa disciplinada pode chegar onde quiser.

Jarbas Carriconde

O que é aprendizado de máquina?

A evolução do reconhecimento de padrões e do aprendizado das máquinas.



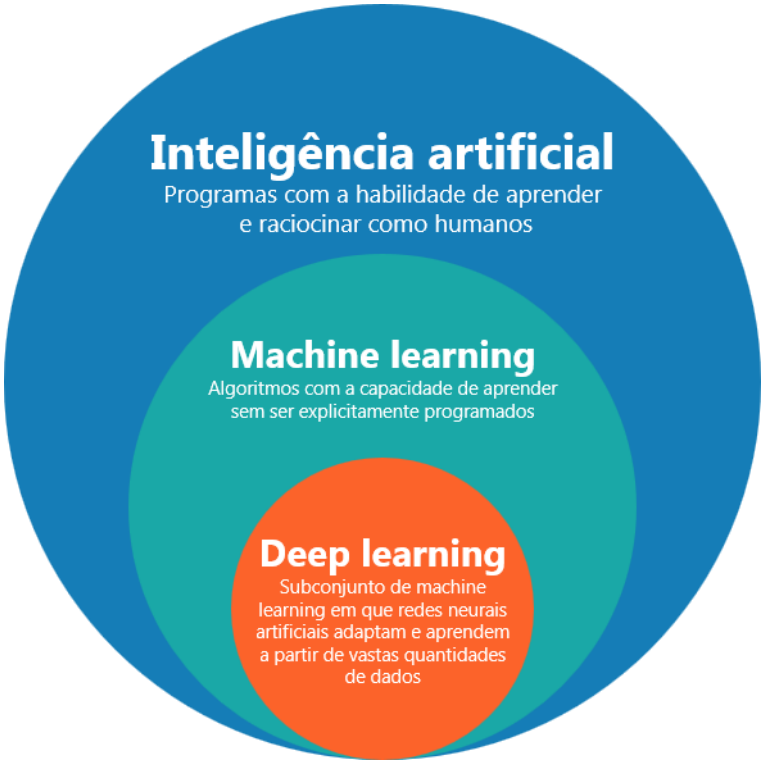
Quando começamos a adentrar nesse assunto, sempre escutamos alguns termos que parecem de certa forma misturados, mas nunca sabemos ao certo como situá-los, sejam eles: Inteligência Artificial, *Machine Learning* (Aprendizado de Máquina), *Neural Networks* (Redes Neurais) ou *Deep Learning.* Usarei a maioria dos termos em inglês que é como se está mais habituado na área.

Como primeiro passo, antes de entrar somente no que é o *Machine Learning*, vamos esclarecer como diferenciar a sopa de letrinhas que se torna a primeira confusão quando estamos aprendendo sobre o assunto. A imagem abaixo elucida as categorias.

Inteligência Artificial: o ponto de vista mais macro sobre o assunto, dentro desse conceito amplo está todo aquele ramo da tecnologia que tenta simular a habilidade humana de raciocínio e aprendizagem.

Machine Learning: finalmente, como subcategoria da inteligência artificial se encontra o assunto central desse *ebook*. O aprendizado de máquina, como intuitivamente mostra o nome, nos possibilita programar computadores(máquinas), de forma que eles possam obter aprendizado, identificar padrões, de um conjunto de dados. Analogamente, um ser humano, na sua respectiva área de atuação, obtém dados desde quando era estagiário até obter um notório aprendizado ao longo dos anos na sua respectiva área de atuação, que o possibilita identificar certos padrões e antecipar algumas ocorrências típicas.

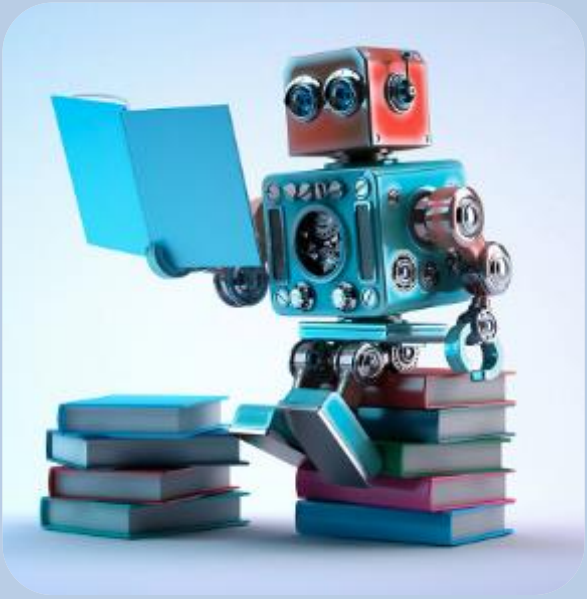
Deep Learning: subcategoria do aprendizado de máquina, esse é o assunto que se tornou moda na área há algum tempo. Inclusive, o próprio nome é julgado por muitos como uma jogada de *marketing*. Se quiséssemos ser mais técnicos ou ir mais direto ao assunto, poderíamos na verdade chamar apenas de redes neurais ou *perceptron* multicamadas. Esse é um assunto um pouco mais complexo que será tratado nesse *ebook* com o devido cuidado, tentando ao máximo elucidar aquilo que nos parece de outro mundo quando escutamos pela primeira vez. Por fora, definimos o aprendizado profundo (na tradução) como um tipo de aprendizado de máquina mais complexo e robusto capaz de atuar e identificar padrões em bases de dados muito grandes. Seu funcionamento é uma analogia ao aprendizado neural do cérebro humano, e tenta replicar o aprendizado dos neurônios que acontece através das sinapses, por exemplo. No capítulo que tratará do assunto, serão mostradas as diversas especificidades do assunto, vantagens e desvantagens além do racional do seu funcionamento.



É através do deep learning que a inteligência artificial tem conseguido resultados impressionantes em áreas e atividades normalmente destinadas a seres humanos, tais como carros autônomos, reconhecimento facial, detecção de doenças usando imagens, dentre outros.

Ainda sobre o aprendizado de máquina

O tamanho da base de dados é fundamental, uma vez que quanto maior, mais experiência poderá ser computada, o que afeta diretamente o desempenho do algoritmo ao tentar reconhecer padrões. Os mais variados métodos de *Machine Learning* têm as mesmas finalidades: utilizar um modelo para obter capacidade de predição e de forma automática conduzir o processo de treinamento do algoritmo em cima da base de dados.



► Quanto mais dados o algoritmo tem acesso, maior será o potencial de aprendizado

Por mais trivial que pareça, sempre achei difícil explicar para leigos o que são esses "dados", como transmitir o que são de uma forma mais visual. Sendo assim, podemos pensar numa base de dados como uma tabela de *Excel*, onde as linhas são as amostras, e as colunas os parâmetros do fenômeno a ser modelado. Essa estrutura é o que chamamos de "dados estruturados", justamente por causa da característica ordenada e organizada que esses dados apresentam. Qualquer coisa que foge a esse padrão recebe o nome de "dados desestruturados", e nessa categoria podemos incluir fotos, vídeos, áudios, textos diversos, entre outros.

No exemplo a seguir temos um dos *datasets* (conjunto de dados) mais clássicos quando estamos aprendendo sobre ciência de dados e de *machine learning*, o *dataset* de classificação de flores "íris". Nessa base de dados (ou tabela, se preferir assim chamar) cada linha é uma amostra referente a uma flor, e as colunas são parâmetros que remetem às características dessas flores. Ou seja, são medidas que determinam as suas pétalas, e ao fim, na última coluna, a classificação sobre qual tipo de iris aquela amostra pertence. Que no caso configura o fenômeno que desejamos predizer através do nosso algoritmo de *Machine Learning*.

sepal_length	sepal_width	petal_length	petal_width	class
6.4	2.8	5.6	2.1	virginica
5.7	3.8	1.7	0.3	setosa
7.4	2.8	6.1	1.9	virginica
7.6	3.0	6.6	2.1	virginica
7.3	2.9	6.3	1.8	virginica
6.0	2.9	4.5	1.5	versicolor
6.0	2.7	5.1	1.6	versicolor
5.8	4.0	1.2	0.2	setosa
5.4	3.9	1.7	0.4	setosa
6.3	2.8	5.1	1.5	virginica

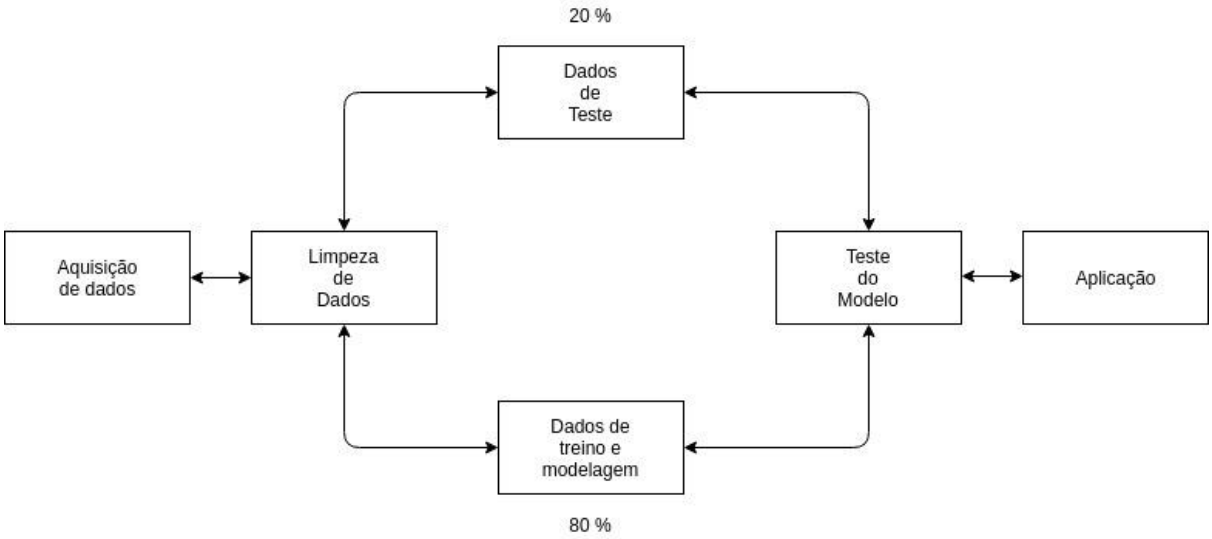
Uma vez elucidado ao máximo o que é uma base de dados, com um exemplo bem simples como o *dataset* de flores íris, vamos entender como se dá o processo de modelagem por meio de um algoritmo de *machine learning*.

O *machine learning* é direcionado para a modelagem do que chamamos de "fora-da-amostra". Se é um termo novo, não se preocupe, você já vai entender. Mesmo que os modelos sejam construídos em cima de uma determinada base de dados, o objetivo é que ele obtenha um bom desempenho ao realizar predições em uma base de dados que não foi usada no momento da modelagem.

Entendendo o processo: o *machine learning* consiste em quatro processos básicos:

1) **Aquisição de dados/limpeza de dados**: tomar posse dos dados, seja do banco de dados de uma empresa que deseja uma modelagem em específico ou de alguma outra fonte. Eventualmente será necessário realizar a chamada limpeza de dados, pois no mundo real os dados não chegam até nós perfeitos, por muitas vezes há dados faltantes, ou letras onde devia haver números e vice-versa, entre outros.
2) **Divisão entre dados de teste/treino e teste do modelo**: Imagine que o seu *dataset* tenha 1500 linhas(amostras) com os parâmetros e sua respectiva classificação do tipo de flor de Iris que cada amostra representa. Deve-se haver uma divisão entre dados que o algoritmo usará para treinar o modelo, e dados "escondidos" que o modelo desconhece, para que seja feito uma validação e ele possa desempenhar seu poder preditivo em cima de amostras desconhecidas. Assim, poderemos comparar os seus resultados com os resultados reais, adquirindo uma confiabilidade maior de como o modelo performa em dados desconhecidos comparados aos que foram usados para modelagem. Assim na figura abaixo pegaríamos 80% das amostras e guardaríamos 20% para realizar o teste.
3) **Aplicação**: após as etapas anteriores bem sucedidas, utiliza-se uma nova base de dados que não foi utilizada antes para que sejam executadas predições.

Acompanhe a descrição dos processos com o esquemático abaixo e ficará mais claro.



Analogia

Abrindo espaço para informalidade (ainda mais), separo espaço para trazer um *insight* de analogia que me fora concebido em dado momento, sempre na tentativa de fazer associações para tentar desmistificar conceitos. Não sei se você que lê esse *ebook* gosta de esportes e futebol, caso não, foque apenas no recado. Vale alertar que essa analogia é desprovida de clubismo e tem o foco estritamente no conteúdo a ser passado.

Estava eu vendo uma entrevista do célebre goleiro Rogério Ceni e sem querer ele me forneceu informações valiosas sobre como explicar aprendizado de máquina ao mencionar algumas fases da sua carreira.

Na ocasião, Rogério fora questionado sobre como lidar com a idade avançada para atuar em alto nível, com a perda da explosão e da condição atlética em decorrência do tempo, ciclo natural de qualquer atleta. Sua resposta foi taxativa: a experiência!

E foi mais além: com a perda de condição física inerente a idade, Rogério salientou que ele jogava com as probabilidades, avaliava de onde a bola vinha e como ela vinha, para intuitivamente indicar de forma predeterminada onde a bola iria. Isso possibilitava a ele antecipar os movimentos, podendo ser mais assertivo compensando a falta de explosão física de um atleta com mais de 30 anos idade.

Com tudo que leu aqui, consegue fazer a associação? Rogério passou a vida treinando, jogando e se dedicando, isso é o seu banco de dados, o que o permitiu modelar comportamentos sobre os arremates vindos em direção a sua baliza, obtendo como entradas, o lugar de onde o chute viria, como o jogador adversário chutava a bola, e tudo isso ao longo do tempo permitiu que ele mentalmente

fizesse "predições" ao receber as entradas, e assim tomar decisões assertivas, que mesmo com a potência física decrescendo, se manteve atuando em alto nível até aos 42 anos. Todos sabem como esporte de alto rendimento tende a prejudicar fisicamente os atletas com o passar do tempo, por sorte os algoritmos na mente de Rogério e o seu empenho em "treinar" o permitiram uma longeva e bem sucedida carreira até o último dia como atleta profissional.

Overfitting e Underfitting

Esse aqui é um tópico **fundamental** para entender a performance de um modelo. Você deve estar se perguntando: como sei se o meu modelo será considerado bom? Muitas das respostas passam pelo entendimento do que é ***Overfitting*** e ***Underfitting***. Além de que posteriormente, ao final desse *ebook* abordaremos o capítulo de **Métricas de Desempenho** onde você saberá como são feitas as métricas para cada tipo de problema que um determinado algoritmo está tentando resolver, e a partir disso ter um indicativo sobre a qualidade daquele modelo.

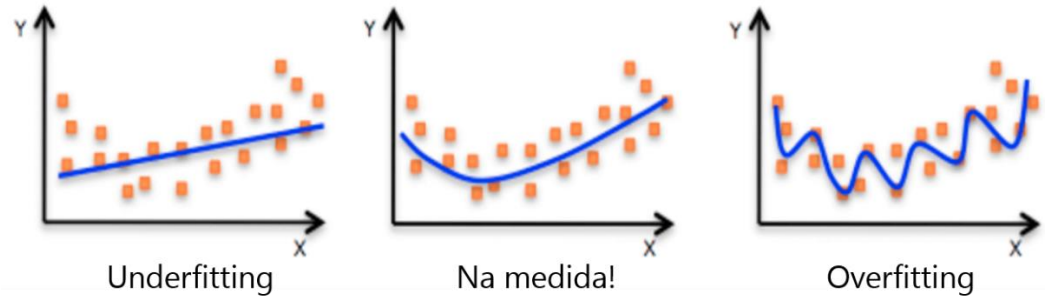
Quando estamos modelando algo, é muito tentador achar que é só colocar o algoritmo mais "porrada", que irá resolver tudo e nos fornecer o melhor modelo. Mas há vários aspectos as serem considerados. É importante frisarmos que em se tratando de *Machine Learning* sempre teremos as premissas de que "tal algoritmo funciona melhor em determinada situação", mas sempre deve-se realizar testes de fato e observar comportamentos. Onde eu quero chegar: muitas vezes o nosso modelo pode ser pouco complexo para conseguir identificar padrões em alguma base de dados robusta (com dados mais complexos), que envolvem muitas variáveis e muitas amostras, e isso causa o **Underfitting**, conforme a primeira imagem abaixo(da esquerda para direita). Podemos imaginar que a linha em azul é o nosso modelo e os pontos são as amostras, assim vemos que o modelo nesse caso não é capaz de captar com precisão o movimento dos dados, o que ocasiona uma má performance e um modelo de precisão pobre, pois o modelo não tem poder para entender o padrão dos dados.

Por outro lado, temos um cenário ideal, que é quando o modelo entende o padrão dos dados da melhor forma possível sem que tenha "decorado" os dados de treinamento, como vemos na figura do meio. Quando falamos de modelo nesse contexto de *Overfitting* e *Underfitting*, uma palavra se torna muito importante: **generalização**.

É importante que o modelo saiba generalizar para ter uma boa performance, que ele compreenda os padrões de uma forma que consiga entender onde estará uma amostra que ele não conheceu no momento do treinamento. Trazendo para a vida real: imagine que você estuda com afinco para uma prova. Você entendeu o conteúdo ou apenas decorou?

Se você entendeu de fato, na hora da prova saberá lidar com alguma questão que não estava igual na sua lista de exercício usada para estudo, pois você entendeu o racional por trás daquilo. Se

você apenas decorou, e na hora da prova caiu algo um pouco diferente do que estava na sua lista de exercícios, você não sabe resolver, e é provável que você tenha caído em um *overfitting*.



Com a analogia do estudo creio que tenha ficado claro o que é o *overfitting*, conforme mostra a terceira figura mais à direita, o modelo decora os dados e portanto ficará limitado ao universo restrito da quantidade de dados que ele teve acesso, não sabendo reconhecer com precisão nada fora daquilo. Outra maneira divertida de entender o *overfitting* é com esse *meme* clássico abaixo.

► Por que você dorme em determinada posição o resto da cama se torna inútil?

Dados Reais: Tratamento dos Dados

Esse tópico merece uma reflexão a parte, justamente por ser mais difícil de visualizar quando estamos no processo de entendimento inicial do que é o **tratamento de dados**. Quero dizer que inicialmente, quando você ver os problemas mais clássicos para quem está começando a aprender, os dados irão vir todos limpos e sem nenhum ajuste a ser feito, justamente por aquilo de que batemos na tecla: começamos pequenos e vamos entendendo as coisas pouco a pouco. No entanto fica o alerta: é impossível na realidade, ao resolver um problema de verdade receber dados sem que se tenha que fazer uma **limpeza** e **tratamento**, pelo menos no ano de 2020. Mas o que eu quero dizer com tratamento? Vamos falar ações utilizadas geralmente:

- **Remoção de Outliers**: pensemos em um exemplo interessante para vermos por que a remoção de *outliers* é essencial: vamos pensar em um auditório que queremos calcular a média patrimonial das pessoas na plateia, imagine se uma dessas pessoas for o Bill Gates. Ele irá distorcer muito a realidade da média patrimonial ali justamente por ser um **outlier**, sendo assim, a simples remoção dele se torna muito plausível para não termos ruído nos dados que iriam distorcer a realidade.
- **Preenchimento de dados faltantes**: por problema na origem dos dados, pode haver colunas onde há linhas em que se tem espaços vazios, logo há algumas maneiras de amenizar isso. Um dos exemplos mais básicos é preencher com a média geral. Imagine novamente o exemplo da plateia e a média patrimonial, só que há um lugar vazio, portanto não há um valor do patrimônio que representa aquele lugar vazio no cálculo. Assim, o preenchemos com a média do restante geral. No entanto há outros métodos mais sofisticados de preenchimento de dados faltantes que não a média, como a mediana (valor central entre todos) ou até mesmo algoritmos de aprendizado supervisionado (que veremos na sequência) simplesmente para preencher dados faltantes.
- **Remoção de variáveis (colunas) sem poder preditivo**: Muitas vezes o nosso conjunto de dados pode trazer variáveis que não nos ajudam em nada, pelo contrário, só trazem ruído que prejudicam a compreensão do algoritmo. Esse é um aspecto de entendimento do que se está tentando modelar, ou seja, menos técnica e mais de entendimento da atividade fim. Se estamos tentando modelar o preço de um carro, a data de nascimento do seu antigo dono provavelmente não terá um poder preditivo significativo, por exemplo.
- **Padronização ou normalização dos dados**: Esse é um aspecto bem interessante, pois ele evita que o algoritmo tenha um viés devido a variáveis que tenham grandezas maiores. Por exemplo: imagine **altura** e **renda**. obviamente o valor numérico da renda será muito maior que o que mede altura, mas isso não quer dizer que a renda, por ser maior, seja mais importante que a altura em algum modelo de *machine learning*. Sendo assim, para que tenhamos uma

equivalência de pesos, aplicamos uma normalização ou padronização. A diferença é que a padronização converte todos os valores para que tenham média 0 e desvio padrão 1, enquanto a normalização converte todos os valores proporcionalmente para que equivalham a valores entre 0 e 1, ou -1 e 1 se houverem valores negativos. Esse processo é especialmente útil em algoritmos que não sejam com base em árvores de decisão (veremos na parte sobre modelos supervisionado as diferenças).

Vários Nomes Para as Mesmas Coisas

Vale deixar claro que ao longo do *ebook* usamos diferentes termos, mas na verdade queremos dizer a mesma coisa muitas vezes:

Colunas/*features*/recursos/atributos/variáveis/parâmetros: normalmente se trata das características dos dados, tal qual fora explicado no exemplo dos dados de íris. Ou seja, como aqui trataremos dos ditos dados estruturados, a coluna sempre irá se referir às características sobre algo que estará sendo modelado. Mas também poderá ser chamado de recursos ou de *features*(tradução em inglês).

Linhas/amostras/registros: as amostras são representadas nas linhas quando falamos de dados estruturados. No exemplo da íris, uma amostra quer dizer que temos o registro de uma flor, suas medidas (de acordo com as colunas) e a sua classificação. Então as linhas correspondem a cada uma das unidades que são objetos de caracterização por meio do que as medidas das colunas descrevem.

***Dataset*/Conjunto de Dados/Tabela/Base de Dados:** aqui nos referimos a toda a composição de linhas e colunas que temos acesso para modelar um determinado fenômeno. Como explicamos anteriormente, fica mais fácil imaginarmos como se fosse uma tabela do *Excel*. Ou melhor exemplificado anteriormente, temos o *dataset* de íris que possui dados que nos permitem modelar os tipos das flores registradas.

Algoritmo/modelo: invariavelmente iremos nos referir a um modelo de *machine learning*, ou a um algoritmo de *machine learning*. E na verdade queremos dizer a mesma coisa: "o modelo de regressão linear", ou "o algoritmo de regressão linear".

Classes/rótulos/labels: termo que você irá escutar muito atrelado aos algoritmos de aprendizado supervisionado para resolver problema de classificação. Justamente por isso, classes/rótulos/labels dizem respeito a qual classe determinada amostra pertence. Como um câncer, que poderá ter duas classificações: benigno ou maligno.

Tipos de aprendizado

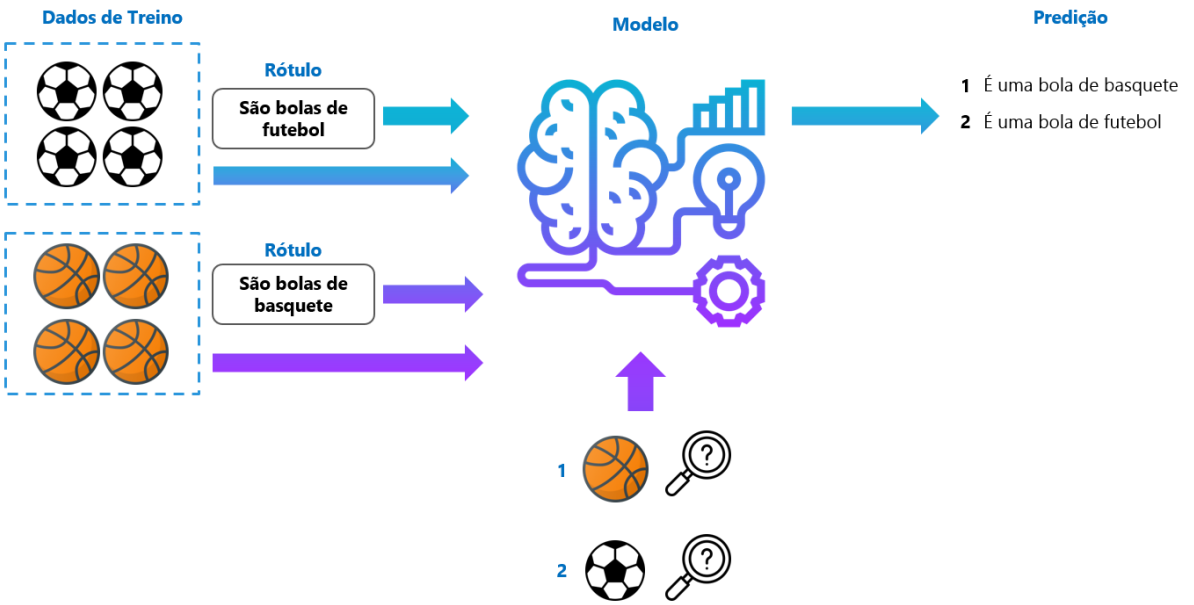
Temos alguns tipos de aprendizado em Machine Learning que devem ser salientados. São quatro tipos ao todo, mas aqui nos ateremos aos dois mais utilizados na prática que são: **Aprendizado Supervisionado** e **Aprendizado Não-Supervisionado**.

Aprendizado Supervisionado

O mais usado, normalmente o que mais agrega em *Machine Learning* aplicado a negócios. Nesse caso, os dados de treino contêm os rótulos estabelecidos, como no caso da flor de íris no capítulo anterior, no momento do treinamento cada amostra já continha sua respectiva classificação. Ou seja, é um tipo onde se tem os direcionamentos que se quer alcançar. Um problema clássico de aprendizagem supervisionada é um classificador, onde pode-se dar o exemplo de um filtro de caixa de *spam* para correio eletrônico. É feita a modelagem com uma base de treinamento que obtém se cada e-mail é *spam* ou não, e assim o modelo deve aprender como classificar novos e-mails de uma base de dados desconhecida.

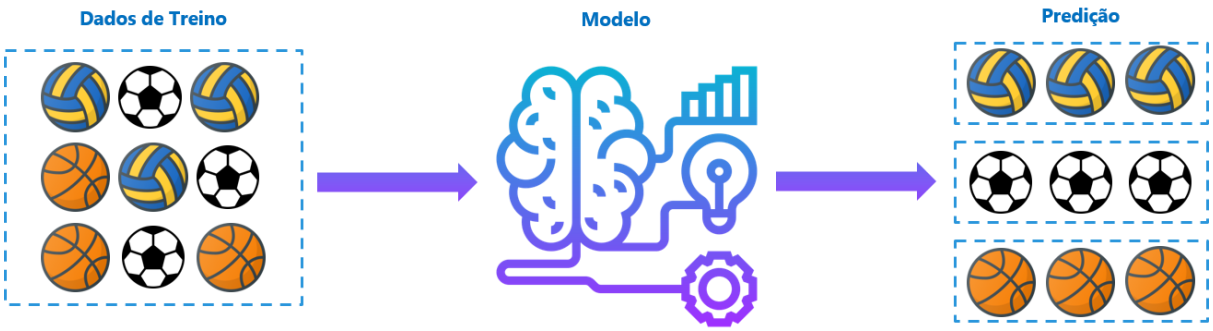
Outro problema clássico é o de regressão, que trata da predição de um *target* (alvo) que agora é um valor numérico. Por exemplo, o preço de um carro, onde tem-se um conjunto de dados com parâmetros preditores como: quilometragem, ano, marca e outros. Para treinar um modelo, se faz necessário um conjunto de dados robusto com muitos registros de carros, incluindo os parâmetros preditores e os seus preços, que no caso são as variáveis a serem preditas, fazendo com que se caracterize um problema de regressão em aprendizado supervisionado.

Na imagem, temos uma ilustração que define bem o aprendizado supervisionado, onde temos dados de treino que contém bolas de futebol e basquete, e após treinamento o modelo deve classificá-los conforme rótulos aprendidos nos treinamentos, qual a classificação daquela amostra que o modelo desconhece, determinando na saída a classificação correta com base na experiência adquirida no momento de treinamento.



Aprendizado Não Supervisionado

Nesse tipo de aprendizagem os dados para treino não possuem rótulos como no exemplo anterior. Nesse caso o algoritmo busca agrupar amostras semelhantes. No caso do e-mail com *spam* que falamos anteriormente, nesse caso, o algoritmo buscaria agrupá-los por títulos e remetentes similares por exemplo. Aqui trago também a ilustração nos moldes do anterior para realçar as diferenças. Nesse caso apresentamos uma base de dados de treinamento como registros sem nenhuma classificação, e o algoritmos busca por si só separá-los por suas características e similaridades.



Tipos de Aprendizado Menos Usados

Só para citarmos brevemente, temos os tipos que são ainda menos usados como o Aprendizado Semi-supervisionado e o Aprendizado por Reforço.

Aprendizado Semi-Supervisionado: este tipo de aprendizagem utiliza na sua base treinamento, tanto dados com rótulos de classificação, como dados sem rótulos de classificação.

Aprendizado por Reforço: ainda mais distante da aplicação em negócios, esse é o tipo de aprendizagem que mais se assemelha na forma como o ser humano aprende. Imagine que temos um agente em interação com um ambiente, assim ele deve aprender quais ações serão executadas de acordo com as especificidades da situação. Tão logo, quando executa ações corretas recebe uma recompensa, ou uma penalidade por tomar ação errada. Assim, com base na tentativa e erro (interação com o ambiente) busca as maiores recompensas possíveis ao longo do tempo em cada ação tomada no ambiente. Se alguma vez você ouvir falar sobre uma inteligência artificial que venceu um jogo de xadrez, provavelmente se trata de um tipo de aprendizagem por reforço.

A origem

"Qual é o parasita mais resistente? Uma bactéria? Um vírus? Não. Uma ideia! Resistente e altamente contagiosa. Uma vez que uma ideia se apodera da mente, é quase impossível erradicá-la. Uma ideia que é totalmente formada e compreendida, permanece para sempre.

D. Cobb

Evolução histórica

Por mais que o assunto esteja na moda e seja considerado o futuro no qual se integrará cada vez mais com diversas áreas do conhecimento, o aprendizado de máquina já vem sendo tratado pelos estudiosos há um bom tempo.

O princípio de tudo vem com nada mais nada menos que Alan Turing, criador da Máquina de Turing, tido como pai da computação e da inteligência artificial. Tendo como feito notório os serviços prestados à inteligência britânica, ajudando-os a vencer a segunda guerra mundial, através de uma máquina conseguiu decodificar o sistema de comunicação da inteligência alemã dando vantagem estratégica aos britânicos. Se esse assunto for do seu interesse recomendo o ótimo filme chamado "O Jogo da Imitação", do ano de 2014.

Em 1952, um rapaz chamado Arthur Samuel desenvolveu um programa de computador para jogar damas, e cunhou o fatídico termo "*Machine Learning*".

Então é importante situar, que por mais que tenhamos a impressão de que esses temas são totalmente novos oriundos apenas da velocidade exponencial das mudanças tecnológicas, há muito estes assuntos já vêm sendo tratados. Temos, por exemplo, as famosas redes neurais artificiais sendo amplamente abordadas nos anos 60.

Mas talvez a pergunta que você deve estar se fazendo é por que somente agora houve um foco intenso no desenvolvimento de aplicações envolvendo conceitos tão antigos?

A resposta se chama "custo computacional". Lembra da época do disquete, depois do CD, DVD, *Blu-ray*...até os dias atuais? As habilidades dos computadores de ver, entender, e interagir com o mundo crescem abruptamente, logo isso permite que sejam criadas quantidades muito maiores de dados em decorrência de diversos fenômenos, tão logo a capacidade desses mesmos computadores de processar, analisar e aprender com esses dados também crescem na mesma medida. Basta você pensar no quanto de dados você gera todos os dias, desde pesquisas no *Google*, lugares onde você esteve, sites que você acessou, produtos que você pesquisou (ou até mesmo falou), e por uma casualidade da vida milhares de anúncios daqueles produtos sorrateiramente lhe são apresentados na internet, ou através de alguma rede social.

Logicamente, essa pequena síntese temporal que fizemos aqui se torna importante para compreender por que o aprendizado de máquina e a ciência de dados se tornam cada vez mais parte de uma cultura analítica em diversas áreas como saúde (HealthTec), Direito (LawTech), ou as mais conhecidas da área financeira (Fintech), só para citar alguns casos, tornando necessário o conhecimento básico mostrado aqui para que você saia do zero.

O fato é: os dados são ativos, que geram valor inigualável se forem bem utilizados, economizando energia e reduzindo achismos. Se pensarmos que as capacidades computacionais seguirão crescendo, teremos um universo de possibilidades sem volta pela frente. Conceitos muito mais sofisticados que

são abordados ainda em um campo mais teórico nos dias de hoje, podem ser aplicados tais quais o *machine learning* nos anos 50 e as redes neurais artificiais nos anos 60 e 70.

Machine Learning x Estatistica

Sempre chega aquele momento em que passamos a nos questionar qual seria a diferença objetiva entre o *machine learning* e a estatística, especialmente para aqueles que não vieram originalmente de uma formação de exatas. Pois bem, vamos elucidar esse ponto brevemente, de uma forma simples, antes mesmo que a dúvida venha à tona.

Em última análise, recapitulando o que já foi dito nas seções anteriores, o *machine learning* obtém aprendizado com base em um conjunto de dados sem depender de programação baseada em regras, em outras palavras, ao aplicar o modelo certo nos dados certos, ele lhe retornará predições e automaticamente aprende aquilo de forma efetiva. Por outro lado, modelos estatísticos estão muito mais ligados em como as variáveis se relacionam nos dados e como colocar isso em um formato de equação matemática.

A diferença fica mais clara quando constatamos que o *machine learning* é uma subárea da ciência da computação e da inteligência artificial, que lida com a construção de sistemas que podem aprender com os dados, enquanto a estatística é uma subárea da matemática.

Realçando pontos já abordados, o maior acesso ao poder computacional e a disponibilidade cada vez maior de grandes conjuntos de dados permitem que possamos "treinar computadores" para aprender com os dados, já a modelagem estatística existe muito antes dos computadores terem sido inventados.

Se pudéssemos diferenciar rapidamente, diríamos que o *machine learning* tem ênfase na otimização e performance e a estatística trata sobre "inferência". Por exemplo, assim cada um deles descreveriam o resultado do mesmo modelo:

Machine Learner: o modelo tem uma precisão de 90% para predizer a variável Y, com base nas variáveis A, B e C que temos nos nossos dados.

Estatístico: o modelo tem uma precisão de 90% para predizer a variável Y, dadas as variáveis A, B e C, no entanto, estou 80% certo que você irá obter o mesmo resultado ao aplicar esse modelo.

O *machine learning* não requer suposições anteriores. Tudo que precisamos é jogar os dados que temos e o algoritmo "faz tudo", descobre os padrões e faz previsões. Assim, logicamente ele é aplicado a conjuntos de dados grandes, sendo o poder preditivo altamente correlacionado com tamanho dos dados, ou seja, quanto mais experiência for coletada pelo algoritmo, mais ele aprenderá. A estatística procura entender como os dados foram coletados, o que esperar se a coleta dos dados fosse feita várias vezes, etc. Dessa forma, é necessário ter um conhecimento mais específico do que as variáveis tratam e de onde vem. Por isso, e por ter sido desenvolvida em circunstâncias onde não havia

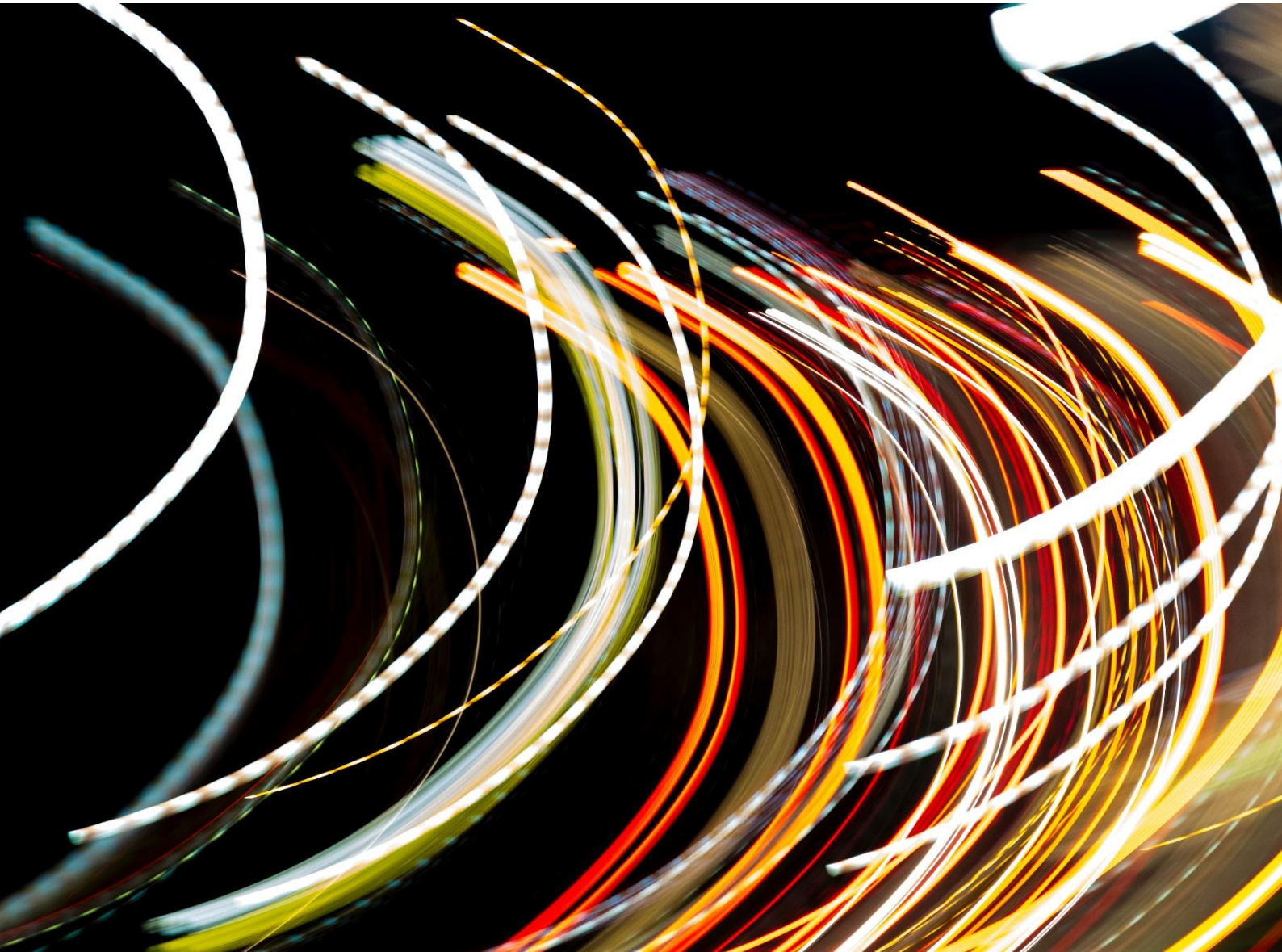
poder computacional, a estatística é aplicada num contexto onde não há processamento de grande volume de dados.

Ou seja, podemos concluir que as duas áreas se complementam e podem ser colaborativas. Podemos usar a estatística especialmente para entender os dados e fazer análises que nos permitem encontrar variáveis mais significativas para o que queremos predizer, imputando a um modelo de *machine learning* variáveis com poder preditivo maior. Em outras palavras, variáveis que "querem dizer algo" para a finalidade que estamos tentando modelar.

Fique tranquilo, não nos aprofundaremos em estatística aqui, apenas situar as diferenças nesse início de trajetória já é de grande valia para separar os diferentes conceitos na cabeça. Vamos nos atentar aos dados e aos modelos de *machine learning* que são o nosso foco de maior importância agora.

Modelos

Aqui iremos entender o funcionamento dos algoritmos
e suas aplicações segundo as respectivas especificidades.



Finalmente vamos entrar na parte dos **modelos**. Qual é a lógica por trás de cada um e quais as suas particularidades. Neste capítulo você terá o entendimento da forma mais didática possível sobre os modelos mais clássicos em aprendizado supervisionado de *Machine Learning*, os primeiros que todos que iniciam na área obrigatoriamente devem saber, ou ao menos, ter uma breve noção. Esses modelos criaram o *Machine Learning*, e não ao contrário.

Vale reiterar que aqui os modelos serão expostos conceitualmente, já que essencialmente, para os fins que almejamos, é necessário entender o comportamento do modelo, quais seus vieses e limitações, e não propriamente obter um aprofundamento em questões matemáticas. É tudo uma questão de prioridades, e a nossa nesse momento é utilizar modelos para predições, o que nos liberta de entrar em detalhes internos de cada modelo, mas torna absolutamente fundamental o entendimento do racional por trás do comportamento de cada um.

Visualizando os Exemplos

Os algoritmos de *machine learning* que serão explicados na sequência serão ilustrados em um formato simplificado. É fundamental que no processo de entendimento de como opera um algoritmo, ter o apoio de um recurso visual. E justamente por isso, quando falamos em "formato simplificado", quero dizer que sempre trabalharemos com duas dimensões.

Quando falamos em dimensões, queremos dizer que traremos exemplos supondo que estejamos modelando uma base de dados que tenha apenas dois parâmetros(colunas). Como no exemplo de prever o preço de uma casa, onde os parâmetros poderiam ser: número de quartos e preço do metro quadrado. Ou seja, como trabalhamos apenas com esses dois, significa que teremos duas dimensões.

Concordamos que é muito mais fácil de visualizar algo em formato bidimensional, que se movimenta apenas nos planos horizontal e vertical. Por outro lado, torna-se impossível de enxergar algo que tenha 10 dimensões.

O fato é que na maioria das vezes as bases de dados irão vir com dezenas de colunas, ou seja, dezenas de dimensões. Mas é importante exemplificarmos e visualizarmos os modelos no espaço em formato bidimensional. Só assim é possível entender a lógica por trás de cada algoritmo, e assim ter a sua compreensão de forma mais profunda, bem como suas vantagens e limitações.

Problemas de Regressão e Classificação

Como fora explicado no capítulo que traz os conceitos de tipos de aprendizado, vale destacar novamente os dois tipos de problema que os modelos de aprendizado supervisionado resolvem. Vamos separar bem para ficar bem fixado na memória:

- **Regressão:** diz respeito a um valor numérico a ser atingido na previsão, como por exemplo: preço, tamanho, quantidade. Toda variável que puder ser traduzida para um número quantificável.
- **Classificação:** são rótulos pré-definidos com os quais queremos predizer amostras desconhecidas. Como se num conjunto de dados tivéssemos registros de carros pertencentes a três marcas, e com base nas variáveis que dizem respeito às características do carro, o nosso modelo consiga predizer de qual marca é um carro para uma amostra que o modelo desconhece, somente com base nos padrões reconhecidos no momento do treinamento. Quando falamos em modelos de classificação estamos preocupados em prever ***classes***, e não números.

Para reforçar esses conceitos, vale a pena retornar ao capítulo de **Tipos de Aprendizado** onde essa questão é abordada também com exemplos para reiterar a distinção entre os dois problemas.

Vale ressaltar que as métricas que dizem se o modelo está performando bem ou não, são distintas para problemas de regressão e classificação, e serão abordados mais adiante, de forma clara.

Regressão Linear

O primeiro algoritmo que vamos falar não poderia ser outro. O mais clássico, simples, e o modelo de aprendizado supervisionado mais usado. Nada mais é do que um método para predizer uma variável alvo de valor numérico através da determinação da melhor relação linear entre a variável que é explicativa e a variável dependente.

Não entendeu nada? Vem comigo que com o exemplo ficará claro como céu de brigadeiro.

Imaginemos logo de cara o que seria uma aplicação na vida real através de um raciocínio básico para colocarmos em prática o que seria uma Regressão Simples: queremos estimar o preço de uma casa e para isso buscamos variáveis explicativas que tenham forte relação com a variável alvo, que seria a variável dependente. Ao se tratar de uma regressão simples buscamos uma variável que faça sentido no contexto que estamos tentando empregar.

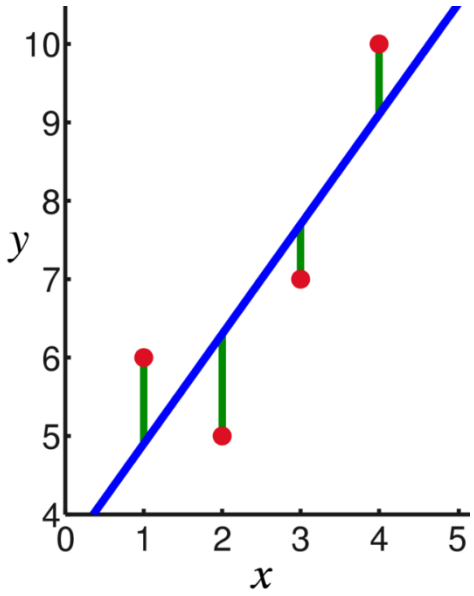
Antes de destacar as variáveis do nosso exemplo dos preços das casas, vamos diferenciar as regressões neste entendimento inicial:

- **Regressão Simples:** Usando apenas uma variável.
- **Regressão Multivariada:** Usando mais de uma variável.

Em uma regressão simples, poderemos ter:

- **Variável Explicativa:** Tamanho da Casa
- **Variável Alvo:** Preço

Todos concordamos que faria sentido, via de regra, usar o tamanho da casa para estimar o seu preço. Ao olhar a imagem abaixo, vamos colocar o eixo X como sendo os valores da casa e o eixo Y como sendo seu preço, assim ao olhar imagem vemos a aplicação de uma regressão linear. Ao tomar conhecimento de um conjunto de dados que tenham tamanhos de casas e seus respectivos preços, o modelo aprende para delimitar uma **linha de tendência** que seria essa linha azul traçada, que é uma linha que busca "abraçar" todos os pontos, de forma que a linha minimize o erro ao máximo em relação aos pontos vermelhos(que são os pontos de encontro entre preço e tamanho reais).



Não é difícil de imaginar que uma regressão simples omitirá alguns pontos importantes que vão ocasionar em erros. Para fins didáticos é importante visualizarmos que a regressão traz um modelo que identifica uma linha de tendência com base no que foi aprendido e busca seguir essa tendência para prever valores futuros. Através desse exemplo básico, o racional por trás deste algoritmo fica claro.

Aqui podemos destacar outro ponto importante, nem todas regressões terão algum tipo de performance útil, se as variáveis explicativas não fizerem sentido na vida real em acordo com o nosso alvo. Por exemplo, será que a idade dos proprietários teria mais relação com o preço da casa do que o tamanho da casa conforme relatamos no exemplo? Provavelmente não. Talvez esse seja o exemplo mais simples para mostrar como não adianta ter um modelo complexo se as variáveis não tiverem poder

preditivo sobre o alvo desejado. Conhecer variáveis preditoras demanda conhecimento além de algoritmos e programação, mas sim do caso em si que estamos analisando.

Agora, imaginemos a Regressão Multivariada, que ao invés de considerar uma variável como na regressão simples, considera múltiplas variáveis para fornecer uma previsão mais próxima da realidade. Sempre ressaltando a importância de usar as variáveis certas para evitar resultados enganadores.

O princípio é exatamente o mesmo que relatamos na Regressão Simples. Estima a melhor linha de tendência de forma que a linha tenha o menor erro possível em relação ao ponto de encontro entre as variáveis explicativas (pontos vermelhos na imagem da regressão simples).

O exemplo da imagem é perfeito para fins didáticos pois fica extremamente difícil representar a linha de tendência quando estamos lidando com muitas variáveis. Basta ver que, se tivermos mais de duas variáveis explicativas fica difícil representar graficamente, como na simples imagem de duas dimensões acima. Se tiver 3 variáveis explicativas, ainda assim precisaríamos de algum *software* sofisticado para nos mostrar a representação gráfica de uma regressão linear tridimensional. Mais que isso? Você já viu algum gráfico de 4, 5 dimensões? Eu também não.

Mas, deixando claro que a análise gráfica é para fins de entendimento, ao aplicar a regressão, não temos que necessariamente ver sua representação gráfica e sim monitorar métricas de erro que trataremos em capítulos posteriores, que é realmente o que nos mostra se o nosso modelo tem boa capacidade preditiva, que via de regra será o que nos interessa.

A Regressão Linear, por ser considerado o método mais clássico, é talvez o menos complexo quando comparado aos outros métodos, assim tendo várias limitações que não permitem atuações sofisticadas em dados muitos ruidosos e complexos, embora ainda seja usada em vários contextos.

Explicabilidade: Um bom debate gerado pela Regressão

Uma das grandes virtudes da Regressão Linear é algo muito discutido no âmbito do *Machine Learning* aplicado a negócios: o modelo me retorna explicações ou devo simplesmente confiar cegamente em algo que para mim é desconhecido?

Normalmente esse é um *trade-off* que vai da particularidade de cada ambiente de negócios. Muitas vezes é feita a opção de aplicar algum modelo menos robusto, mas que seja possível extrair alguma explicação como: "a variável X influenciou essa quantidade no resultado, então devemos focar nossos esforços aqui".

Esse é um tema de grande debate quando se fala de algoritmos bem complexos como redes neurais, que falaremos mais tarde, mas de difícil extração de alguma explicação. Ainda assim há uma ferramenta sofisticada que vem ganhando notoriedade chamada SHAP, baseada em teoria de jogos, oriunda dos jogos de RPG que busca justificar tomadas de decisão. Essa ferramenta vem tomando

espaço por justamente complementar algoritmos complexos que trazem grandes resultados, aliado a explicações de como cada variável influenciou no alvo que se tenta prever.

Por fim, não se assuste com a equação matemática abaixo, tudo que se precisa entender é o porquê a regressão linear apresenta como ponte forte a explicabilidade. Ela consegue quantificar a influência da variável no alvo a ser predito com base na relação matemática abaixo.

Esqueça os símbolos que lhe parecerem estranhos na equação abaixo, e vamos focar no entendimento aplicado do que estamos trazendo aqui. Notamos que o **Tamanho da Casa** possui 1,32 do chamado coeficiente de multiplicação, o número de quartos é 1,25. Lembra quando falamos de **dados normalizados** lá no capítulo 2? Pois então, vamos considerar esse cenário aqui neste exemplo. Ou seja, utilizaremos esse recurso para darmos a mesma importância para os valores de Tamanho da Casa e Números de Quartos.

Concordamos que uma casa pode ter 5 quartos e uma área de 300 metros quadrados. O número 300 é bem maior que 5, assim o modelo entenderá as importâncias distorcidas pela diferença considerável entre esses valores, descaracterizando suas reais importâncias no momento de prever o que desejamos. Sendo assim, ao considerar que normalizamos esses dados torna-se fundamental, damos a mesma importância para variáveis distintas independente da magnitude de seus valores.

Portanto, podemos observar que o Tamanho da Casa tem um peso maior na determinação do preço final da casa! O Tamanho da Casa possui um peso 5,6% maior na determinação do preço final se comparado ao Número de Quartos.

Uma outra forma de observar as relações seria: para cada unidade no aumento do Tamanho da Casa, o preço da casa subiria R$ 1,32, desde que o Número de Quartos permanecesse o mesmo.

Imagine isso em algum problema de negócios, os responsáveis por dados sempre precisaram justificar suas análises para os tomadores de decisão. Esse é só mais uma dentre outras séries de diretrizes que envolvem aplicação de algoritmos de *Machine Learning* na vida real.

$$Preco = \beta + 1.32 TamanhodaCasa + 1.25 NumerodeQuartos + \sum$$

Regressão Linear: Prós e Contras

A grande vantagem da regressão linear, como salientamos antes, é que devido a sua baixa complexidade. Quando comparada a outros, conseguimos sempre encontrar uma explicação do porquê temos determinado resultado, o que em muitos contextos de negócio é extremamente importante.

Porém, há muitos problemas em aplicações reais, simplesmente porque em problemas reais, raramente as variáveis preditoras tem relação **linear** com a variável que se deseja predizer, o que ocasiona uma limitação significativa desse modelo. Outra limitação é que se no nosso conjunto de

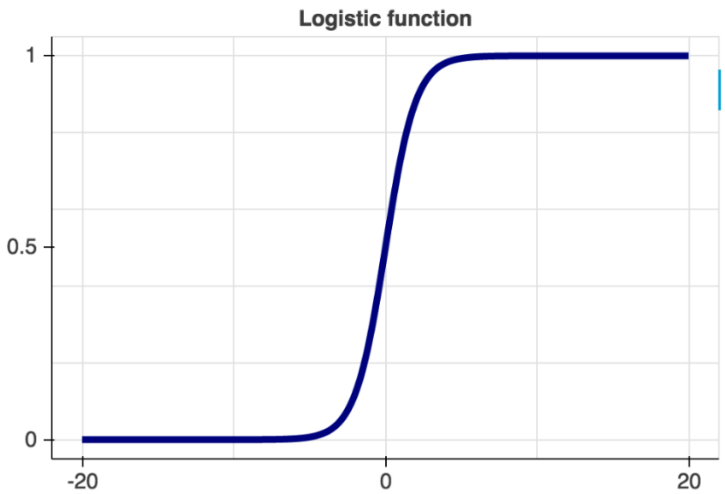
dados tivermos o número de amostras(linhas) menor que o número de variáveis(colunas), a regressão linear não deve ser usada, pois apresentará ***overfitting***, conceito que explicamos em capítulos anteriores. Também, é muito sensível aos chamados ***outliers***, aqueles valores anômalos que destoam da média geral, logo devem ser removidos do conjunto de dados ao aplicar uma regressão linear, pois podem prejudicar muito seu desempenho.

Regressão Logística

Apesar de se chamar regressão, não se confunda. A Regressão Logística é um algoritmo de **classificação**, naturalmente um dos mais clássicos, e é usado quando desejamos atribuir rótulos nas nossas predições. Esse algoritmo busca, com base nos dados e suas variáveis, encontrar probabilidades de a predição pertencer a determinado rótulo.

Por motivos sugestivos, não é possível utilizar Regressão Linear nos mesmos problemas que a Regressão Logística, justamente pelo fato da Regressão Linear permitir valor menores que 0 e maiores que 100, já a Regressão Logística atribui a probabilidade de a predição pertencer a determinada classe. Normalmente, atribui-se que em problemas binários, se a probabilidade retornada for maior que 50%, ela é atribuída diretamente como pertencente a uma classe (classe 1, por exemplo), e abaixo disso de outra (classe 0).

Vamos ao exemplo didático, analisando o gráfico amigável a seguir.



Imagine que temos um conjunto de dados, onde cada linha(amostra) representa uma pessoa física, e as colunas são suas características financeiras, e a variável classificável que se deseja predizer é se aquela pessoa é inadimplente ou não. Vamos analisar isso da perspectiva de uma empresa que opera fornecendo crédito.

O *dataset* é treinado por um algoritmo de regressão logística que começa a identificar padrões entre as pessoas que se tornaram inadimplentes, assim o modelo se torna capaz de detectar se a aquela pessoa provavelmente se tornará inadimplente no momento em que o modelo conhece seus dados financeiros.

Após treinamento, o modelo recebe dados financeiros sobre uma pessoa que não estava nos dados quando fora treinado, assim ele está apto a dar um veredicto e retorna uma probabilidade de 0,8(observando o eixo vertical na imagem abaixo). Como é um valor acima de 0.5, pode-se atribuir que aquela pessoa, segundo o modelo, se tornará inadimplente, cabendo aos tomadores de decisão da empresa decidirem como proceder, se irão conceder crédito aquela pessoa ou não, ou até mesmo cobrar taxas mais altas.

Regressão Logística: Prós e Contras

Se trata de uma técnica amplamente utilizada por ser eficiente e não requer muito poder de processamento computacional, além de que, é fácil de implementar e de treinar. A regressão logística é muitas vezes usada como uma primeira abordagem, quando queremos resolver um problema e não queremos começar com um modelo mais complexo, então aplicamos um algoritmo de fácil implementação para ver como o nosso conjunto de dados se comporta. Assim, conseguimos ter um comparativo do quão melhor seria quando aplicamos um algoritmo de alta complexidade.

Temos algumas desvantagens, como também o fato de não ser de alta complexidade, como falado anteriormente, e é facilmente superado por outros algoritmos de fácil acesso. Para se obter um bom desempenho, há uma alta dependência da qualidade do conjunto de dados, ou seja, deverá ser identificado somente variáveis que sejam pertinentes a variável que desejamos predizer, sendo assim a regressão logística se limita a esse cenário. Por fim, ela apenas soluciona problema de classificação pela natureza de como ele infere seus resultados, além de ser vulnerável ao ***overfitting***.

KNN

O algoritmo KNN quer dizer *K Nearest Neighbors*, ou **K Vizinhos Mais Próximos**. Novamente, usaremos uma imagem auxiliar abaixo, acompanhado das explicações, para esclarecer como esse algoritmo opera, que aliás, é de uma forma bem simples.

Para facilitar o entendimento, olhamos a imagem disposta em formato bidimensional (eixo vertical(X1) e horizontal(X2)), onde cada eixo será a representação de uma variável no nosso banco de dados usado pelo algoritmo para treinamento. Tal como fizemos na explicação anterior, imaginemos a questão do risco de crédito, onde só temos duas variáveis, que seriam duas colunas no nosso conjunto de dados. Uma variável poderia ser o salário daquela pessoa física, e a outra o número de pessoas que residem na mesma casa que ela.

Os eixos se cruzam nos pontos, e os dados são fornecidos ao KNN, de forma que ele possa realizar o treinamento em cima daqueles dados para reconhecer o padrão em dados não conhecidos.

Vamos pensar que a imagem abaixo representa o modelo. Imagine que ele computou os dados para fins de treinamento, e após treinado, consegue distinguir bem, com base nas duas variáveis, a que classe aquela pessoa pertence: inadimplente ou não.

Agora pensemos que depois que treinamos o modelo e ele já está "pronto" para prever novos casos, passamos uma nova observação para que ele realize uma predição. Observe na imagem o ponto vermelho. Esse é um novo ponto, uma nova observação, um novo caso de uma pessoa que o modelo não observou no treinamento.

Observe também que para esse novo ponto, nós já temos os valores que precisamos para que o modelo realize a predição, ou seja, as duas variáveis: salário e número de pessoas que residem na mesma casa daquela pessoa. Essas são as nossas variáveis dependentes, ou explicativas, que serão passadas ao modelo para que ele realize a análise de crédito.

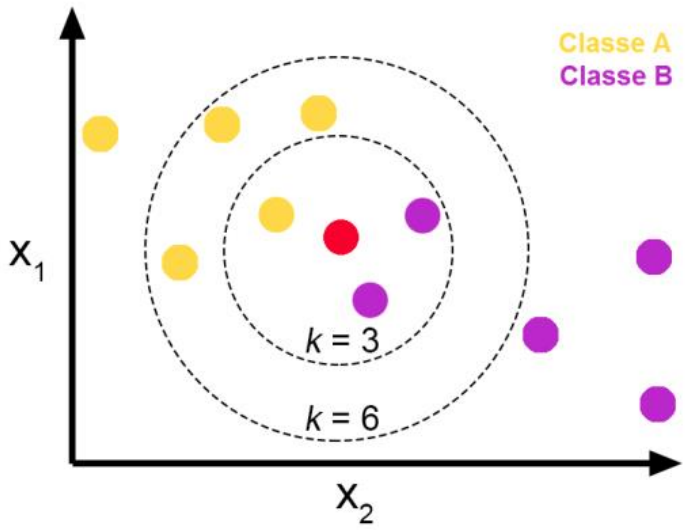
A partir daí que acontece a mágica! O modelo mapeou a distinção entre as classes, e o modelo pede que você diga qual o K dele. O que isso quer dizer? Quantos vizinhos ele deve olhar para classificar aquela amostra que ele desconhece. Ou seja, se meu K=3, ele vai olhar os três pontos mais próximos e computar uma "votação" dos pontos, se dois dos três pontos pertencem a classe B, o modelo aponta que aquela amostra pertence a classe B, simples assim.

Mas talvez você deva estar se perguntando: como eu posso determinar o número de vizinhos, o tal valor de K? Novamente caímos no assunto das métricas: se o K for muito alto ou muito baixo, pode ser que o modelo se confunda e não consiga fazer distinção por haver muitos vizinhos congestionados naquele raio de distância. Portanto, uma das maneiras de especular o K é usar as métricas de desempenho apropriadas ao algoritmo (que veremos mais adiante) para monitorar como o modelo performa frente a determinado número de K.

Se visualizarmos que a métrica de avaliação do modelo frente a dados que ele desconhece (dados que não foram utilizados no momento do treinamento) for boa, sinal que encontramos um bom valor para determinar K.

Vale ressaltar que um modelo de KNN pode conter várias entradas e não somente duas, como exemplificado aqui, tal qual a regressão linear. Usamos duas variáveis apenas para fins didáticos de explicar o racional do algoritmo com apoio visual da imagem. Como faríamos para explicar um modelo com 10 variáveis e um gráfico de dez dimensões? Meio difícil não acha?

Independentemente da quantidade de variáveis o *modus operandi* é o mesmo.



KNN: Prós e Contras

O KNN tem um apelido curioso, chamado de **lazy learner** (algo como aprendizado preguiçoso), pois ele se comporta diferente dos outros algoritmos. Ele não faz um processo de treinamento propriamente dito, mas armazena os dados e obtém o aprendizado apenas no momento da previsão, em tempo real, na hora de indicar de qual dos vizinhos mais próximos a previsão poderá ser categorizada. Assim, pela sua própria natureza, é de muito fácil implementação. Temos que nos atentar somente para a quantidade de **K**, e qual metodologia será usada para calcular as **distâncias** que irão determinar a proximidade entre os vizinhos.

Por outro lado, o KNN apresenta limitações com conjunto de dados muito grandes, pois isso aumenta o custo computacional de calcular a distância de um novo ponto de dados em relação ao

restante dos pontos existentes. Isso acaba degradando o desempenho do algoritmo. Também por dificuldade de calcular distâncias, ele não funciona bem quando o conjunto de dados apresenta muitas colunas, justamente por ter que considerar cada coluna no cálculo da distância.

No exemplo acima teríamos duas colunas apenas (X1 e X2) e fica fácil de visualizar. Agora imagine em múltiplas colunas tendo que realizar cálculo de distância em um número alto de dimensões.

Outro fator importante é o da necessidade da normalização dos dados. Como vimos no tópico sobre tratamento de dados que citado anteriormente, essa é uma metodologia usada para dar a mesma importância para cada coluna de valores numéricos, e é um recurso absolutamente necessário para evitar que o KNN forneça previsões erradas. A normalização se torna necessária justamente para que diferentes magnitudes de valores não influenciem erroneamente no cálculo da distância entre os pontos para achar os vizinhos mais próximos.

Por fim, ainda sobre tratamento de dados, o KNN é altamente sensível a ***outliers***, dados faltantes e ruído. Portanto, aplicar este algoritmo requer um trabalho minucioso de limpeza e tratamento de dados, removendo *outliers* e preenchendo dados faltantes. Lembrando que esses conceitos são explicados ao final do capítulo **O que é aprendizado de máquina?**, onde é falando sobre como são os dados na vida real.

Naive Bayes

Este não é algoritmo mais complexo, mas é talvez um dos mais difíceis de explicar para quem não tem familiaridade com exatas. Mas não se assuste! Se este *ebook* não o decepcionou até aqui, não será agora!

O *Naive Bayes* é um modelo de classificação baseado no **Teorema de Bayes**. A primeira coisa que temos que ter em mente é que esse modelo não captura as relações entre variáveis explicativas (parâmetros, as colunas de um conjunto de dados). Ou seja, para ele, essas variáveis não se relacionam entre si, mas contribuem individualmente para realizar uma predição, para chegar no resultado da variável resposta. Mas como assim? Imagine que um modelo classificador de frutas que veja uma fruta que possui formato redondo, 8 centímetros, cor laranja e seja uma laranja.

Mesmo que essas variáveis dependam umas das outras no mundo real, afinal estão relacionadas a um mesmo objeto, o modelo apenas enxerga que elas contribuem individualmente e de forma independente umas das outras para que seja constatado que a fruta é uma laranja. Ao contrário, por exemplo, do algoritmo anterior de KNN, onde no exemplo que observamos, há uma relação entre as duas variáveis. Calma que vai ficando mais claro.

Novamente, vamos apelar ao melhor método de entendimento, o exemplo. E tudo que você vai precisar é olhar atentamente para pequenas tabelas e entender de operações bem básicas que aprendemos na escola. Ainda assim, será tudo explicado para que fique bem claro. Imaginemos um exemplo extremamente simples, onde temos um conjunto de dados que tem como variável preditora o Tempo no sentido de condições climáticas, e o nosso algoritmo terá que descobrir se a pessoa sairá para a rua com base nos dados fornecidos a ele para treinamento. Lembrando que é um exemplo bem simples, com apenas uma variável preditora (explicativa ou independente) e um alvo de classificação (variável resposta) que responde: sairá para a rua? Temos como possíveis classificações "Sim" ou "Não". Vejamos a tabela que deixa isso mais claro abaixo.

Tempo	Sair para Rua
Sol	Não
Nublado	Sim
Nublado	Sim
Sol	Sim
Sol	Sim
Nublado	Sim
Chuvoso	Não
Chuvoso	Não
Sol	Sim
Chuvoso	Sim
Sol	Não
Nublado	Sim
Nublado	Sim
Chuvoso	Não

Uma vez que temos os registros de todas as situações possíveis de acordo com a condição climática, podemos prosseguir com a próxima etapa, que é a criação de uma **Tabela de Frequência**. A intenção dela é facilitar nosso entendimento do problema para nos auxiliar nos cálculos.

O que estamos fazendo na tabela abaixo é simplesmente realçar quantas vezes há a ocorrência de "Sim" e "Não" na sua totalidade. E ainda fica mais intuitivo para ver quantas vezes "Sim" e "Não" foram registrados para cada condição climática.

Tabela de Frequência		
Tempo	Não	Sim
Nublado		4
Chuvoso	3	2
Sol	2	3
Total	*5*	*9*

E finalmente, vamos para a última tabela, que é a continuação da anterior, a chamada **Tabela de Probabilidade**.

Vamos por partes. A primeira tabela que mostra os registros, tem um total de 14 amostras, ou seja, se a quantidade "Sim" fosse de 50%, eu teria que ter 7 amostras registradas com "Sim", pois 14/7 é igual a 0,5, o que nos dá 50%. A tabela abaixo faz precisamente isso, calcula as probabilidades, tanto de "Sim" como de "Não", mas também a probabilidade das condições climáticas.

Olhando a coluna "Sim", vemos que "Sim" ocorre um total de 9 vezes, 9 dividido por 14 que é o total, resulta em 0.64, logo temos 64% dos casos como "Sim". Mesma coisa se analisarmos a outra coluna de classificação.

Por outro lado, é exatamente assim com as linhas, pegando por exemplo a condição climática "Chuvoso", somando todas as vezes que a condição aparece na nossa amostra da primeira tabela, temos um total de 5 aparições, divididos por 14 temos 0.36, que nos retorna 36% de frequência dessa condição climática nos nossos dados.

Estamos firmes até aqui?

Tabela de Probabilidade				
Tempo	Não	Sim		
Nublado		4	4/14	**0.29**
Chuvoso	3	2	5/14	**0.36**
Sol	2	3	5/14	**0.36**
Total	5	9		
	5/14	9/14		
	0.36	**0.64**		

Agora tudo fará sentido. Munidos do raciocínio desenvolvido através das nossas tabelas até a **Tabela de Frequência**, finalmente aplicaremos a equação do **Teorema de Bayes** e poderemos ver exatamente como o algoritmo opera. Fique tranquilo, que é uma equação simples, tendo como base apenas operações básicas que aprendemos na escola.

Se você não tem familiaridade, não se assuste e continue lendo que tudo irá fazer sentido aos poucos. Vemos a equação abaixo onde temos as explicações do que significa cada variável da equação e como ela se aplica no nosso simples exemplo apenas substituindo valores e fazendo contas básicas.

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Seguem as variáveis:

- **P(A|B)** - A equação busca a probabilidade de ocorrência de uma classe ou rótulo, dado um preditor. Como por exemplo, a probabilidade de ser "Sim" dada a condição climática de "Sol".
- **P(B|A)** - Probabilidade registrada na tabela de frequência em que houve "Sim" e foi registrada condição climática "Sol". Por exemplo, em 9 dias houve "Sim", a pessoa foi para a rua, e desses 9 dias em 3 houve registro de condição climática "Sol".
- **P(A)** - Probabilidade registrada na tabela de frequência em que houve ocorrência da classe, como por exemplo, probabilidade registrada segundo a tabela de frequência, em todos os casos que foi registrado "Sim".
- **P(B)** - Probabilidade registrada na tabela de frequência em que houve "Sol" na condição climática, retornando à probabilidade em cima do total de ocorrência de "Sol" nos nossos dados.

Agora substituindo os valores, conforme a nossa fórmula, as explicações do que é cada coisa na equação, e substituindo os valores que já nos são fornecidos na Tabela de Frequência, podemos ver como opera um algoritmo de **Naive Bayes**.

Devemos fazer a seguinte pergunta: a pessoa sairá para a rua (**Sim** ou **Não**) se a condição climática apresenta **Sol**? O teorema de *Naive Bayes*, com base em toda linha de raciocínio montada aqui, nos responderá essa pergunta, como mostraremos a seguir.

Passo por passo, substituindo os valores da equação segundo a tabela de frequência.

Se der **Sol**, qual a probabilidade da pessoa sair para a rua?

P(Sim | Sol) = ?

$$P(Sim \mid Sol) = \frac{P(Sol \mid Sim) * P(Sim)}{P(Sol)}$$

Agora, trazendo da tabela de frequência:

Probabilidade, segundo nossos dados, em que houve **Sim,** e também os dias que houve Sol e a pessoa foi para a rua, foi registrado que a condição climática era de **Sol.** Em 9 dias houve Sim, segundo os nossos dados, e entre esses 9, em 3 havia ocorrência de **Sol:**

P(Sol | Sim) = 3/9 = ***0.33***

Entre todas as ocorrências, em quantas delas foi registrado **Sim**: dos 14 registros dos nossos dados, em 9 a pessoa de fato foi para a rua:

P(Sim) = 9/14 = ***0.64***

Entre todas as ocorrências, em quantas delas houve Sol: dos 14 registros dos nossos dados, em 5 a condição climática apresentou Sol:

P(Sol) = 5/14 = ***0.36***

Agora é só correr para o abraço!

Uma simples calculadora vai resolver nosso problema. Multiplique as duas primeiras variáveis e faça a divisão pela terceira:

*P(Sim | Sol) = 0.33 * 0.64 / 0.36*

P(Sim | Sol) = 0.2112 / 0.36

P(Sim | Sol) = 0.587

Mas podemos arredondar para **0.6**, o que nos diz que, segundo nossos dados, há cerca de 60% de chance de haver **Sol** e em uma amostra desconhecida pelo nosso modelo que foi treinado, a pessoa optar por **Sim**, sair para a rua.

Talvez fizesse mais sentido que num cenário de Sol, na realidade, essa probabilidade fosse bem mais alta, mas lembre-se que estamos aqui apenas tratando de um exemplo simples para fins didáticos.

Antes que você pense: não, não é necessário você calcular todos esses passos. Normalmente um *software*, ou um código de programação vai fazer isso para você. O importante é ter o entendimento do racional por trás do modelo quando for necessário utilizá-lo. Isso é fundamental para identificar pontos fortes e fracos do modelo.

O algoritmo *Naive Bayes* vai usar um método semelhante para prever probabilidades em um grande conjunto de dados com diferentes variáveis preditivas, e não só uma como no nosso exemplo, ou ainda, múltiplas classes a serem preditas, e não somente duas como mostramos aqui (Sim ou Não). Tal qual na explicação dos modelos anteriores, faz todo sentido começar pequeno e se fazer de exemplos simples para absorver a lógica de como cada modelo funciona, que não necessariamente é algo trivial.

Naive Bayes: Prós e Contras

Fácil e rápido de implementar, funciona bem quando temos um problema de classificação para predizer múltiplas classes. Especialmente quando as variáveis são independentes entre si de fato (não há nenhuma relação uma com a outra), o *Naive Bayes* performa melhor quando comparado a outros modelos clássicos como a Regressão Logística, além de precisar de menos dados de treinamento para ter um bom desempenho.

Uma desvantagem é justamente que para uma boa performance é necessária essa independência entre as variáveis preditoras. No entanto, na maioria dos casos na realidade as variáveis preditoras são dependentes entre si, o que afeta a performance do algoritmo nesse caso.

Árvore de Decisão

Um dos algoritmos mais interessantes de se explicar com exemplos, e um dos mais importantes e fundamentais. Com base nas árvores de decisão são constituídos vários outros modelos mais avançados usados na prática e em competições de desempenho de **Machine Learning**. Vamos tocar nesse ponto crucial mais adiante**.**

Modelos de árvore de decisão utilizam nomenclatura que nos fazem lembrar uma árvore de fato

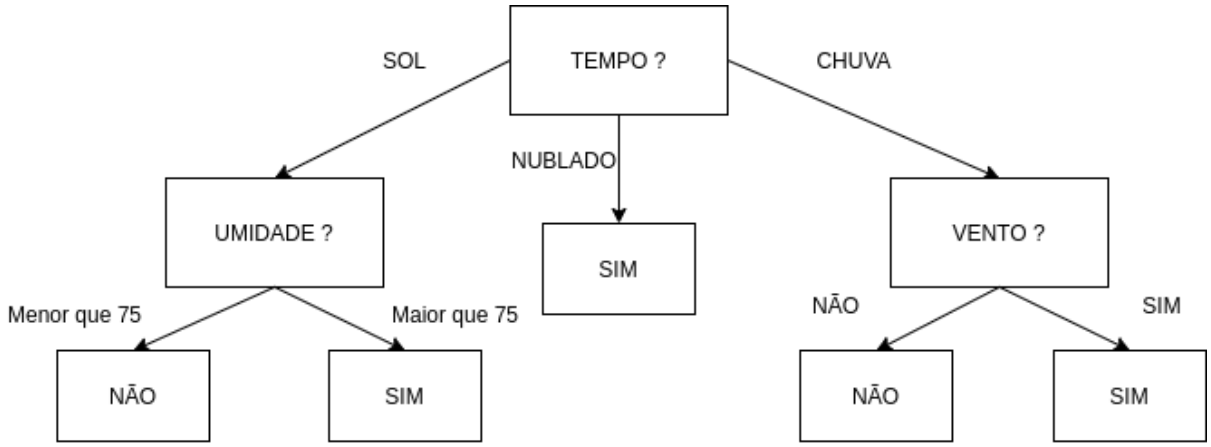
A árvore de decisão tem uma abordagem "dividir-para-conquistar", onde para resolver um problema mais complexo, o algoritmo subdivide o problema em problemas menores.

De cara, vamos ao exemplo. Imagine que uma pessoa deseja tomar uma decisão se aquele é um dia propício para uma corrida ao ar livre, e para isso ele vai avaliar três variáveis que irão afetar a decisão dele: umidade, vento e tempo (no sentido de condição climática, tal qual utilizado na explicação do modelo *Naive Bayes*).

Transferindo para a nossa realidade, supondo que temos dados onde cada linha é uma amostra se a pessoa decidiu ou não correr ao livre, e temos três colunas representando as variáveis.

Mas como a árvore de decisão opera?

A imagem abaixo nos dá noção exata de como percorrer "problemas menores" para resolver um problema maior.



Uma árvore de decisão tem uma nomenclatura estranha, mas com o nosso exemplo ficará fácil. A primeira variável a tomar decisão é o **tempo**, portanto é chamada de **raiz**. Tal qual no exemplo anterior Tempo tem três condições climáticas, e de acordo com o clima a árvore avança para a próxima variável. Todas as "instâncias" onde as decisões são tomadas usando as variáveis Tempo, Umidade e Vento são chamados de **nós**, portanto a variável Tempo nesse caso é um nó e a raiz da árvore. Observando a figura abaixo vai ficar mais fácil de entender.

Se o tempo estiver ensolarado, ele verifica a umidade. Se essa for menor que 75, a pessoa não irá correr ao livre, caso ao contrário e for maior que 75 ela decide que há boas condições para correr ao ar livre. Toda saída dos nós é chamada de **ramos**. Podemos então dizer que as variáveis Tempo, Umidade ou Vento, ao tomar uma decisão percorrem um ramo até a próxima etapa da árvore de decisão.

Por fim, nós temos a parte final da árvore, chamada de **folha**, aquela que aponta "Sim" ou "Não" em relação ao fato de pessoa ir correr ao ar livre ou não.

Mas você pode estar se perguntando: qual o motivo da variável **tempo** ser a **raiz**, ou seja, a primeira variável a determinar o rumo da árvore de decisão?

Isso tem uma resposta chamada **ganho de informação**. A árvore sempre divide primeiro aquela variável que ela considera que pode lhe ser mais determinante para o resultado. Vamos novamente ao nosso tradicional exemplo.

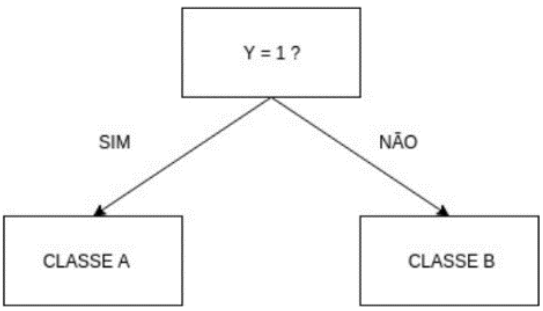
Eu poderia lhe mostrar uma equação de entropia que calcula o ganho de informação, mas não farei isso. Mostrarei com um exemplo prático como funciona esse processo de identificar o maior **ganho de informação**, que é essencialmente, o que você tem que saber aqui.

Vamos a tabela abaixo, que apresenta um conjunto de dados genérico, onde as variáveis explicativas seriam **X**, **Y** e **Z**. Para essas variáveis, temos classificações (**A** e **B**) na nossa variável resposta. A combinação dessas variáveis explicativas resulta em alguma das duas classes na variável resposta. Mas como uma árvore de decisão agiria nesse caso para identificar qual teria maior ganho de informação?

Vamos observar a tabela.

X	Y	Z	Classe
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

Olhando a tabela é possível constatar que a variável **Y** apresenta o maior ganho de informação, pois olhando apenas ela, é possível chegar a uma conclusão segundo as classes com os dados que temos, as outra variáveis não apresentam essa distinção tão clara. Sempre que a variável **Y** é 1, a classe é A, e sempre que for 0, a classe é B, o que nos permite inferir que essa variável nos abastece com mais conclusões sobre esses dados. Assim, podemos ver como ficaria a nossa árvore.



Essa lógica por trás da árvore de decisão também nos fornece o componente da explicabilidade, no qual entendemos qual a variável mais importante na decisão que o algoritmo está tomando.

No entanto, o algoritmo é pouco complexo e tem dificuldades em resolver problemas mais sofisticados, o que nos lembra um pouco a regressão linear lá do início deste capítulo, por ser explicável, mas pouco complexo para resolver problemas mais difíceis.

A árvore de decisão é capaz de resolver tanto problemas de classificação como de regressão, seguindo a mesma lógica mostrada aqui, além de ser o algoritmo "base" para uma série de algoritmos bem conhecidos e eficientes que utilizam "conjuntos de árvores de decisão" com diferentes particularidades cada, para ter um algoritmo mais robusto.

Árvore de Decisão: Prós e Contras

Uma grande vantagem da árvore de decisão é que ela necessita de muito menos cuidado no tratamento dos dados em comparação a outros métodos. Não é necessário nenhum tipo de normalização dos dados, *outliers,* ou ajuste de dados faltantes que afetem a construção do modelo.

Tudo aquilo que batemos na tecla ao falar de tratamento de dados não afeta a árvore de decisão devido a natureza do algoritmo, que ao identificar o **ganho de informação**, conforme explicamos, isola essas anomalias dos dados justamente porque não fornecem informações para que o modelo possa inferir a predição.

Por outro lado, o modelo é bem instável. Se os dados sofrerem uma pequena mudança que seja, a árvore já sofrerá uma mudança muito grande na sua estrutura (daí vem a Floresta Aleatória na sequência, como exemplo de maior estabilidade) quando necessário retreinar um conjunto de dados por exemplo. Além de que, é um pouco mais custoso computacionalmente quando comparado a outros algoritmos clássicos.

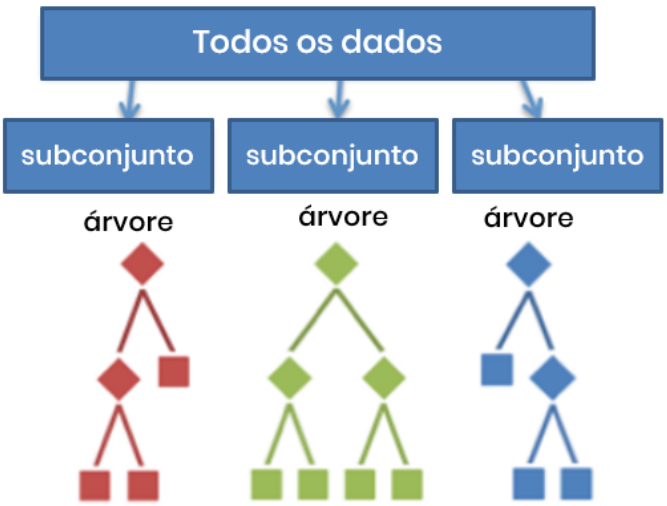
Floresta Aleatória (Random Forest)

Como falamos anteriormente, as árvores de decisão servem como "base" para muitos outros algoritmos, os chamados algoritmos de *ensemble learning* (aprendizado em conjunto), que recebem esse nome por se apropriarem de um conjunto de algoritmos para formar outro. Esse é o caso do modelo que veremos agora, com um nome bem sugestivo: **Floresta Aleatória**. Esse algoritmo pega várias árvores de decisão para constituir um algoritmo único e fazer uma modelagem mais robusta do que uma árvore de decisão faria sozinha.

É um dos algoritmos mais populares no mundo do *Machine Learning*. Muitas pessoas o usam como uma "primeira tentativa", quando estão conhecendo os dados. Como falei antes, devemos

sempre começar "pequenos" em se tratando de *machine learning*, quero dizer, sem coisas mirabolantes. Assim, antes de partir para tentar algum modelo mais complexo, a Florestas Aleatória é uma boa primeira opção para ver qual o desempenho dos dados frente ao alvo que se deseja predizer. Daí então poderemos ter uma noção melhor se devemos adicionar ou remover variáveis (colunas) que possam ter mais poder preditivo na entrada, ou até mesmo tentar um modelo mais complexo.

A floresta aleatória funciona através da construção de várias árvores de decisão, como podemos ver na imagem abaixo.

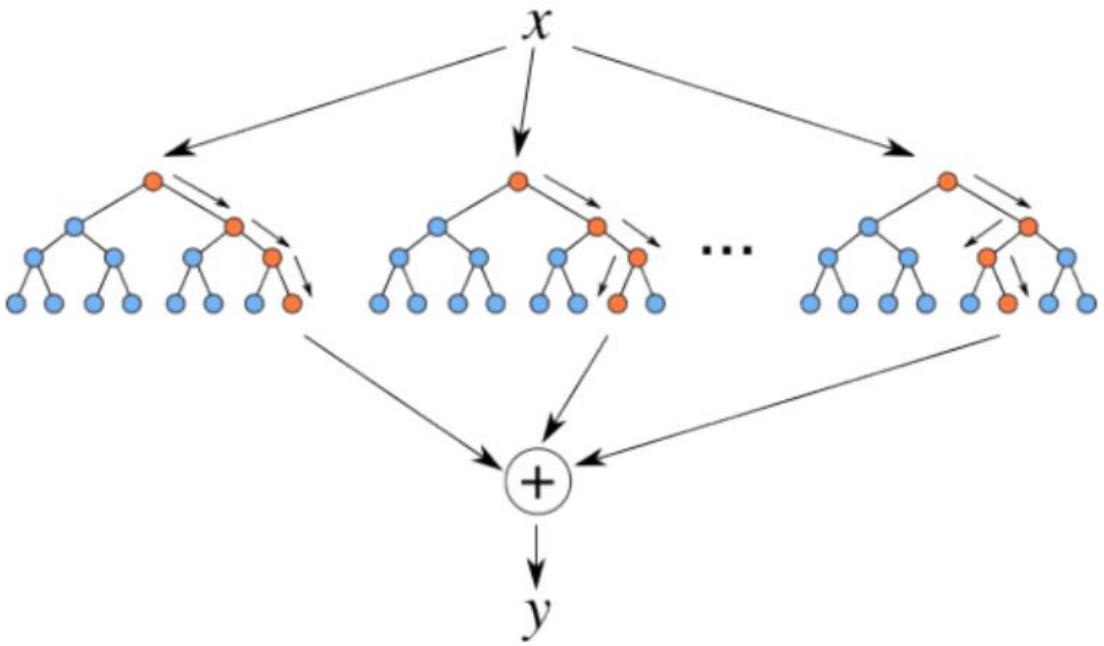


Durante o treinamento, o algoritmo constrói essas árvores. Tal qual a sua metodologia base, ele é capaz de resolver tanto problemas de regressão como de classificação. Para os casos de regressão, o algoritmo toma decisões calculando a média dos valores de cada árvore individual. Para os casos de classificação, utiliza o processo chamado de votação, onde simplesmente cada árvore que compõe a floresta "vota" a classe resultante na sua saída, sendo de determinada a classe da floresta, a classe com mais votos pelas árvores.

A **aleatoriedade** do modelo vem da sua metodologia. Diferente da árvore de decisão que procura construir uma estrutura única com base no conjunto de dados, a floresta aleatória visa criar várias árvores de decisão utilizando variáveis (colunas) e amostras (linhas) de forma aleatória, separando em grupos para cada árvore que o compõe. Essa metodologia torna o algoritmo mais robusto e estável em relação a árvore de decisão, ou seja, tem mais poder de explorar a capacidade preditiva dos dados. Se o algoritmo for treinando várias vezes para um mesmo conjunto de dados ele alcançará resultados de desempenho mais semelhantes, configurando maior estabilidade.

A imagem abaixo ilustra uma disposição genérica de uma Floresta Aleatória. Temos que **X** representa um conjunto de dados para treinamento que acabou por gerar várias árvores de decisão.

Essas árvores irão em conjunto, por meio de votação ou média, para problemas de classificação e regressão respectivamente, dar cada uma a sua contribuição individual para compor a decisão final do valor ou da classe inferida representada em **Y.**



Importante notarmos que as árvores de decisão construídas pelo algoritmo de Floresta Aleatória são criadas paralelamente, e por isso independem umas das outras. Isso porque o algoritmo "retira" amostras aleatórias de linhas e colunas da tabela original, e faz uma árvore para cada uma dessas amostras. O resultado final é simplesmente o resultado ponderado de cada uma dessas árvores separadamente, como falamos anteriormente, por média(regressão) ou votação(classificação). Estatisticamente falando, damos a esse processo o nome de "*Bagging*", que vem de "*Bootstrap aggregating*". Para entender melhor o conceito de "*bagging*", veja o esquema abaixo:

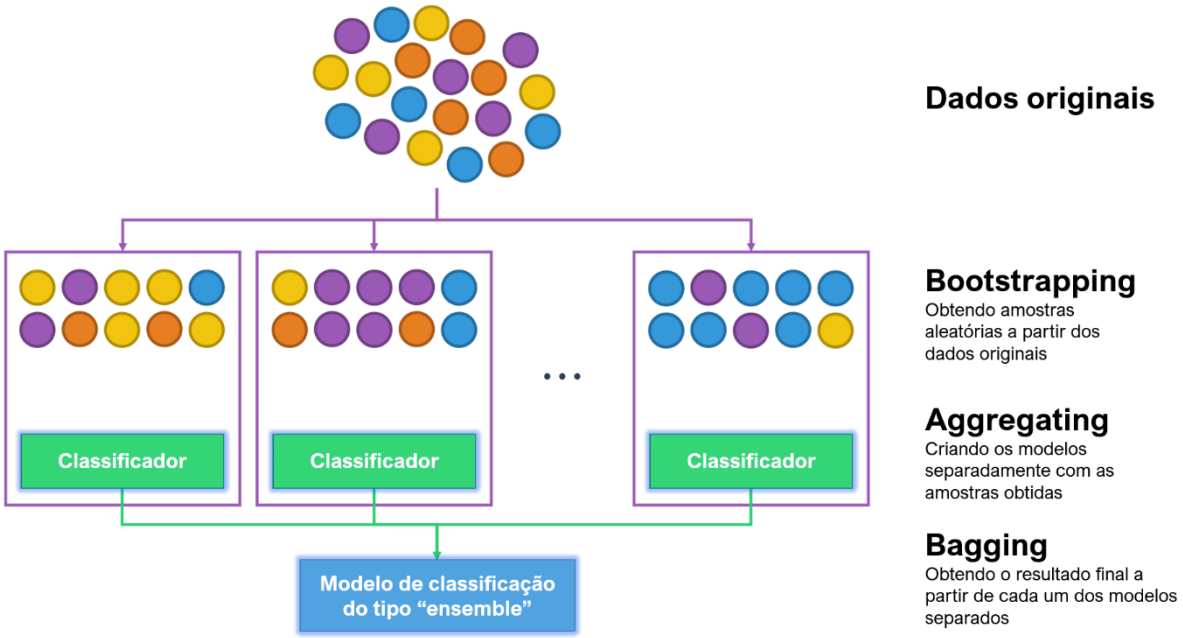


Imagem alterada da original (fonte: wikipedia)

Quando falamos "Classificador" na imagem acima, entenda isso como qualquer modelo de classificação simples, como uma árvore de decisão fazendo classificações sobre um conjunto de dados, por exemplo. Em "Modelo de classificação do tipo *ensemble*", resulta na votação dos componentes "Classificadores".

Floresta Aleatória: Prós e Contras

A floresta aleatória tem a capacidade de reduzir significativamente a possibilidade de ***overfitting***, justamente por reunir várias árvores e pegar uma média delas, além de que, por consequência disso, tem uma maior **estabilidade** nos resultados (ao contrário da instabilidade que relatamos ao falar de árvores de decisão). Também possui capacidade de **explicabilidade**, a árvore de decisão pode nos mostrar quais variáveis são mais importantes na predição realizada pelo modelo. Por ser um algoritmo baseado em árvores, também não precisa de muito tratamento de dados, já que a natureza das árvores de decisão isola variáveis que fornecem pouca **informação** no sentido explicado em árvore de decisão com a figura e tabela. O algoritmo é fácil de usar, além de que pode ser usado para resolver problemas de **regressão** e **classificação**. É considerada por muitos participantes de competições de desempenho de *machine learning* um excelente algoritmo para servir como o primeiro modelo a ser usado antes de avançar para um mais complexo, justamente por apresentar uma excelente relação de custo computacional e desempenho.

Como algumas desvantagens, temos que é mais complexo e custoso computacionalmente que a árvore de decisão, o que torna o algoritmo mais lento e mais problemático se necessário usá-lo em

problemas que necessite de algum treino em tempo real (já que uma predição pode necessitar de várias árvores).

Conjunto de Árvores: Boosting

Um tema que também vale a pena ser mencionado diz a respeito dos vários algoritmos de *Ensemble Learning*, ou nesse caso, *Ensemble Tree*s, que necessitam um entendimento mais complexo, e são amplamente usados em competições de *Machine Learning*, além de *cases* de negócios também. Embora não nos aprofundemos no tema neste *ebook*, vale a pena mencioná-los, e ressaltar que para entendê-los é bom ter uma base solidificada de como funcionam Árvores de Decisão.

Até o momento, vimos um algoritmo de *Ensemble Trees* (*Random Forests*, ou Florestas Aleatórias) onde o princípio básico de funcionamento era a criação de várias árvores de decisão de forma paralela para no final obter um resultado ponderado de todas elas. Na seção anterior, vimos que essas árvores são treinadas de forma independente, isto é, cada árvore usava uma amostra de várias linhas selecionadas aleatoriamente para serem treinadas, independente dos dados originais.

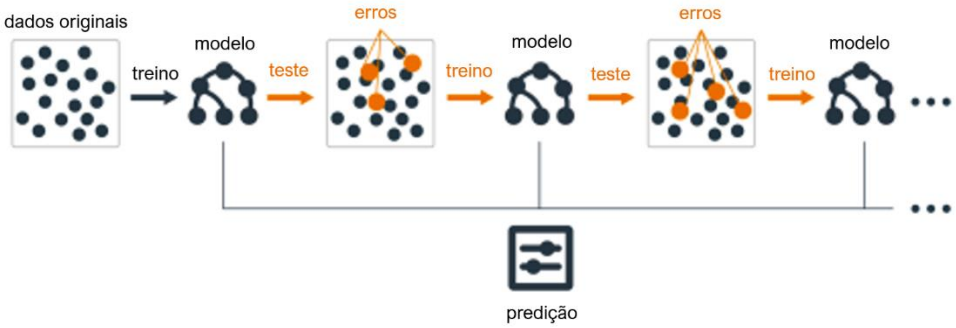
E se ao invés de treinar essas árvores de forma paralela e independentes umas das outras, nós criássemos um jeito de melhorar essas árvores ao longo do tempo? Se conseguíssemos pensar num jeito de fazer com que as árvores posteriores aprendessem com o resultado das arvores anteriores? Teríamos um ganho de capacidade preditiva, não acha?

Foi justamente com esse intuito que o algoritmo de "Boosting" foi criado. Nele, a intenção é fazer com que a árvore que acabou de ser treinada identifique quais foram as observações onde errou, e "repasse" essa informação para a segunda árvore, que vai se "esforçar" mais para evitar que erre essas mesmas observações. A segunda árvore fará a mesma coisa, e passará para a terceira árvore quais foram os seus erros. A terceira árvore vai colocar um peso maior nessas observações para reduzir a chance de errá-las. O processo continua indefinidamente até atingir o número máximo de árvores treinadas que foi definido na implementação do algoritmo.

Boosting é um termo "guarda-chuva" que se refere a uma família de algoritmos que tem como característica usar um algoritmo chamado de *weak learner* (aprendizado fraco), de forma sequencial, um após o outro, onde a cada sequência, o erro de um algoritmo de aprendizado fraco é capaz de melhorar o próximo. Nesse caso, o algoritmo *weak learner* é a árvore de decisão, pois como falamos anteriormente é um algortimo pouco sofisiticado quando atuando sozinho.

No entanto, a metodologia *boosting* permite que usemos várias árvores de decisão, na qual treinamos cada modelo de árvore um após o outro e a cada vez que um é treinado ele é capaz de identificar erros e ir melhorando o próximo modelo de árvore gradativamente a cada vez.

Veja a figura abaixo para entender melhor o funcionamento de um algoritmo de *Boosting* genérico:



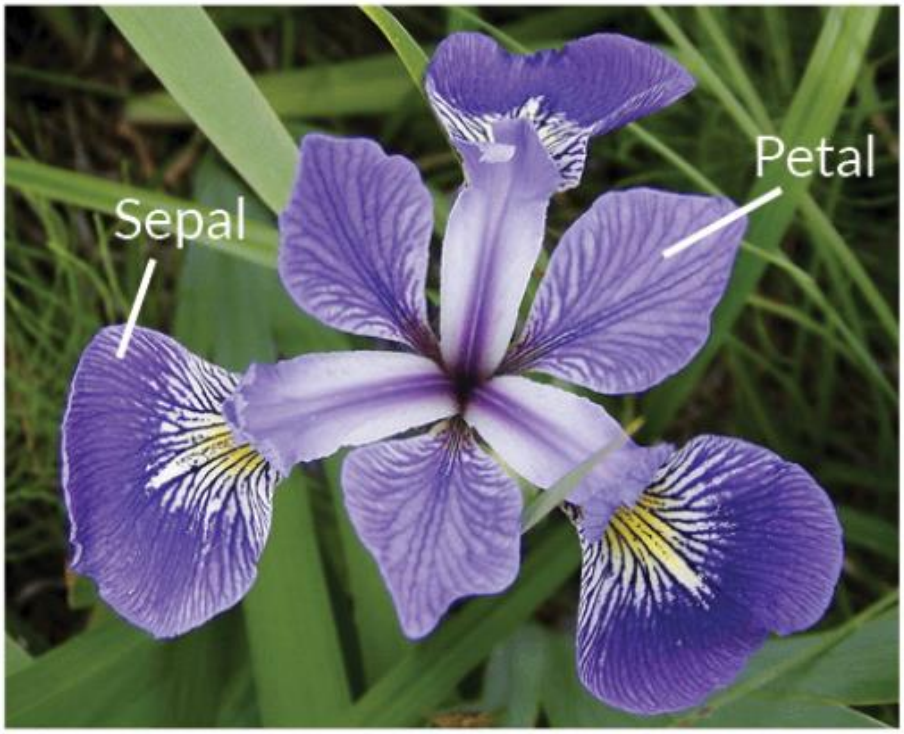
Dentro disso, cada modelo **Boosting** tem suas peculiaridades de como realiza essa metodologia de melhoramento dos algoritmos sequencialmente, que fogem do nosso escopo aqui. Mas aqui vão alguns dos algoritmos **Boosting** mais populares:

- ***Gradiente Boosting***
- ***AdaBoost***
- ***XGBoost***

SVM

O nome SVM vem de *Support Vector Machine*, ou em tradução: Máquina de Vetor de Suporte. Este é um modelo que a explicação acompanhada de figura é fundamental para se ter um bom entendimento. O SVM é na maioria da vezes usado em problemas de classificação, tal qual a regressão logística, mas também pode ser usada em problemas de regressão.

Primeiro, na imagem abaixo, vamos entender o problema. Esse é um exemplo clássico (que já citamos anteriormente) quando se está aprendendo *Machine Learning*, onde temos uma base de dados em que os modelos buscam classificar flores do tipo Íris. As variáveis de entrada (colunas) são medidas das pétalas das flores. Essa base de dados fornece as medidas das pétalas com classificações para que o modelo treine e aprenda a identificar a qual classe pertence uma flor somente conhecendo suas medidas de pétalas. No caso as classificações são de *Iris versicolor* e *Iris setosa*.



Iris Versicolor **Iris Setosa**

E nota-se que temos dois espaços amostrais, onde o eixo horizontal é a medida do comprimento da pétala (*Petal length*) e o eixo vertical a largura da pétala (*Petal width*). Assim cada ponto vai estar disposto onde essas características se encontrarem no espaço.

Vamos recordar do exemplo de classificação das flores Iris nas classes versicolor e setosa

A lógica do SVM se baseia em mapear um espaço amostral, como os dois mostrados abaixo, e traçar uma linha que possa distinguir as classes da melhor forma possível. O exemplo da figura à direita é uma representação de sucesso do modelo, mas vamos por partes, entender o contexto do espaço amostral da esquerda para finalmente chegar num exemplo de ideal.

	Petal length	Petal Width	Species
0	5.1	3.5	setosa
1	4.9	3.0	setosa
2	4.7	3.2	setosa
3	4.6	3.1	setosa
4	5.0	3.6	setosa

Lembre-se que a tabela tem mais ou menos essa estrutura

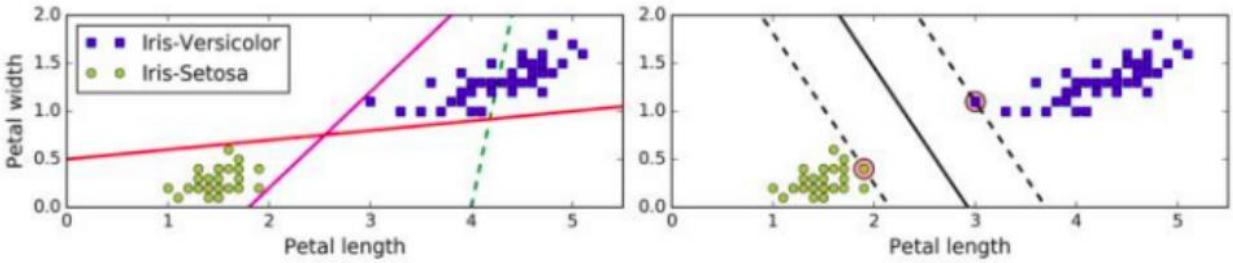
Dito isso, vamos analisar a imagem da sua esquerda, onde temos três linhas traçadas em meio aos pontos, uma tracejada e duas linhas normais. Cada linha é como se fosse um modelo que tem como missão classificar os dados. Lembrando que cada ponto é uma amostra dos dados, como se fosse uma linha na nossa tabela de dados, ao passo que os eixos vertical e horizontal são a representação das colunas no gráfico. Por fim, observe que os pontos também estão rotulados de acordo com a sua classe: *versicolor* e *setosa*.

O modelo representado através da linha tracejada se mostra totalmente ineficiente pois não possui capacidade nenhuma de distinguir as classes, enquanto os outros dois modelos distinguem perfeitamente. Mas há um problema. lembra dos conceitos de ***Overfiting*** apresentado no início do livro? Pois então, o modelo (linha vermelha, por exemplo) aprendeu tão milimetricamente os dados de treino, que na verdade ele não aprendeu, mas sim decorou. Assim, ele perde capacidade de generalizar! Logo, se ele observar uma amostra que esteja um pouco fora do que estava nos dados de treinamento, acabará por classificar errado. Quando formos metrificar o desempenho desse modelo com vários dados que ele desconhece, irá ter uma performance pífia.

E finalmente chegamos na figura da direita, o SVM sendo bem aplicado. Nesse caso vemos que a linha não apenas separa as classes de forma bem definida, mas também mantém a maior distância possível entre as classes, ou seja, o modelo tem o poder de **generalização** que falamos antes. Dessa forma, o modelo irá reconhecer dados que não sejam exatamente iguais aos conhecidos quando o modelo foi treinado e gerado. As métricas de performance do modelo tenderão a um excelente desempenho quando dados que ele nunca viu lhe forem imputados. Aqui, ele não decorou os dados de treinamento, mas sim aprendeu a deixar uma margem para identificar corretamente amostras desconhecidas que não sejam exatamente iguais às que o modelo conheceu na etapa de treinamento.

E por fim, a sacada final. Por que **vetores de suporte**? Bom, os vetores de suportes são exatamente os **pontos** que ajudam o modelo a definir onde colocar a reta que distingue da melhor forma possível as classes. Na imagem da direita temos circuladas em destaque os dois pontos de cada classe que são responsáveis por nortear onde a reta do modelo fará a divisão. A reta então é colocada no meio, entre os dois limites estabelecidos por esses pontos em destaque, os vetores de suporte.

Mais uma vez, nos valemos de um exemplo simples para obter o entendimento. Em dados com muitas variáveis que não temos somente as duas medidas das pétalas como no exemplo, o pressuposto é o mesmo, só que de difícil visualização.



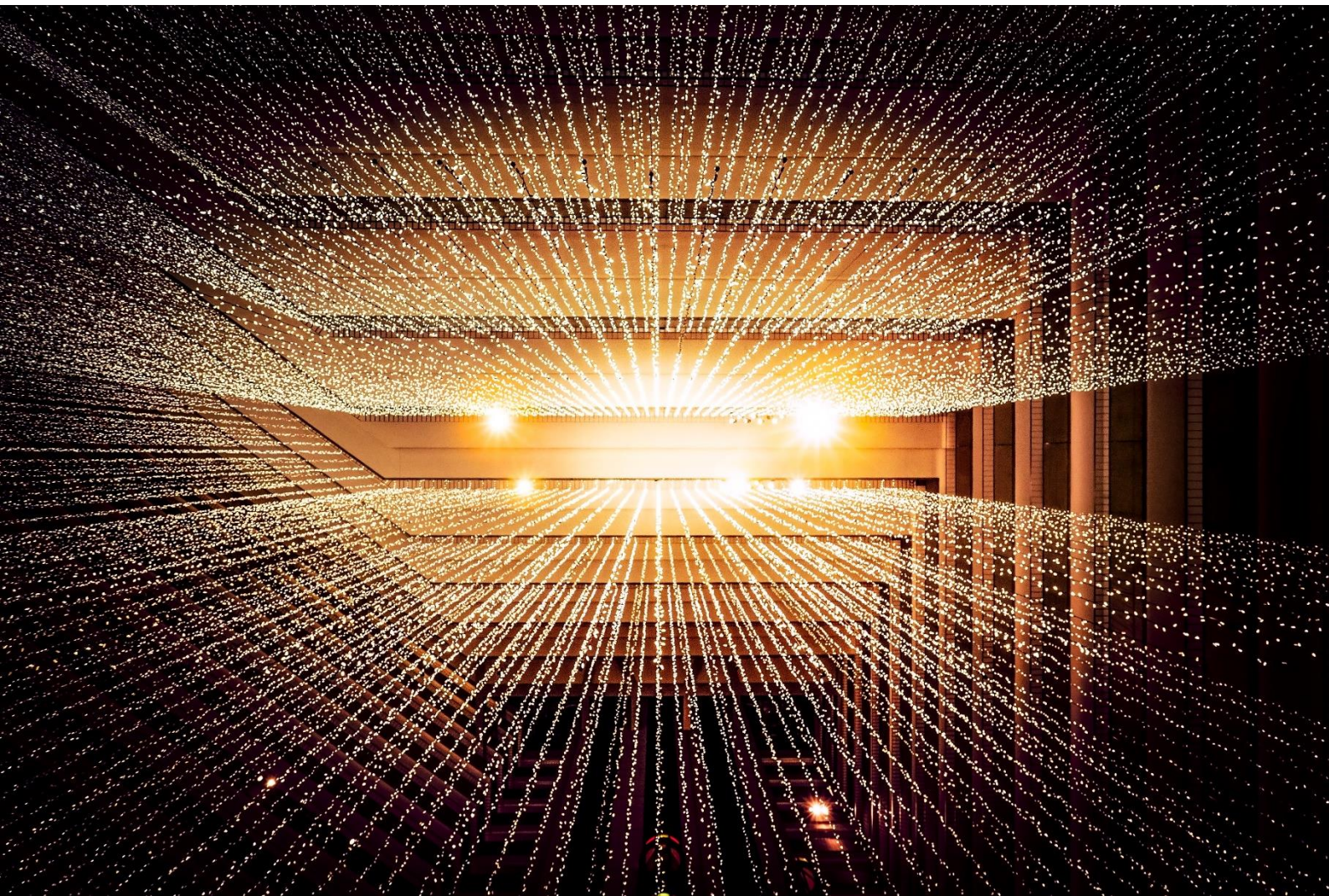
SVM: Prós e Contras

O SVM apresenta uma vantagem importante que possibilita uma maior chance de **evitar** o ***overfitting***. Funciona muito bem quando há uma distinção clara entre classes (para problemas de **classificação**), além de ser mais efetivo quando o número de variáveis (colunas) é maior que o número de amostras(linhas).

É um algoritmo custoso computacionalmente, sensível aos **outliers** e a dados faltantes e com muitos ruídos, dados que possuem variáveis (colunas) que nada contribuem, além de não dar uma explicação probabilística ao determinar uma classificação (como por exemplo: essa amostra tem chance de pertencer a classe X com 80% de certeza segundo o modelo). Pelo seu custo computacional, ele se torna muito demorado na etapa de treinamento, e o algoritmo não se ajusta bem em conjuntos de dados muito grandes.

Aprendizado não-supervisionado

Uma breve explicação sobre *K-Means clustering* e como podemos utilizar esse tipo de aprendizado.



Embora o aprendizado não supervisionado tenha papel importante e atuação chave dentro da tomada de decisão no mundo do *machine learning*, é bem menos comum que os diversos tipos de modelos de aprendizado supervisionado. Portanto, aqui será mostrado o modelo mais clássico deste tipo de aprendizado, que sintetiza bem a forma como é feito um processo de agrupamento, ou **clusterização**, para utilizar um termo mais usual no meio do *Machine Learning*. Falamos sobre o que trata o aprendizado não supervisionado no **capítulo sobre tipos de aprendizado**, e aqui resguardado a aquele que deve ser o primeiro modelo que você deve saber quando ouvir falar sobre este tipo de aprendizado.

K-Means Clustering

O objetivo central do **K-Means** é delimitar grupos com base em suas semelhanças, e tais semelhanças devem ser identificadas com base nas características dos dados, ou seja, suas variáveis (colunas). Detalhe: sem qualquer pista da existência de categorias por trás daqueles dados. Não há um alvo como tínhamos em problemas de regressão (valor numérico) ou classificação (rótulos), como no aprendizado supervisionado.

O K-Means tenta atribuir cada ponto (linha/amostra) dos dados em um conjunto de **K** grupos, daí vem o nome. K é a quantidade de grupos que definimos para que o algoritmo faça a divisão dos dados nesses grupos de acordo com características similares.

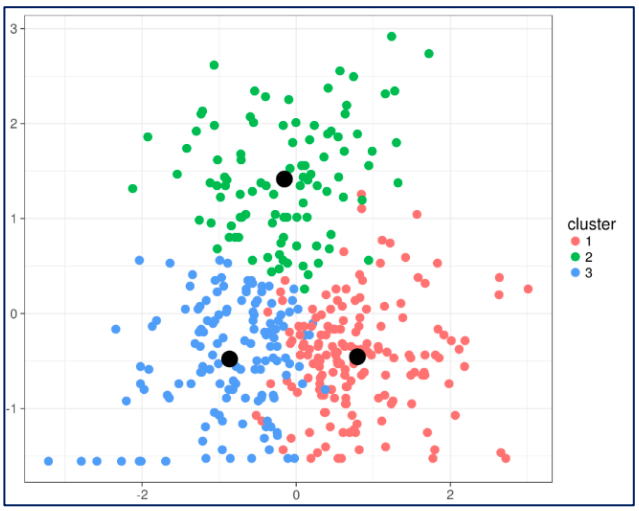
Vamos entender o processo: primeiro definimos o número de K, de forma meramente especulativa para tentar visualizar se há uma separação clara desses grupos. Na vida real nem tudo é bonitinho como na imagem abaixo. O K, por consequência, se refere a quantidade de **centróides**, que para quem não é de exatas, trata de um termo que vem da geometria, mas podemos imaginar que o centróide é como se fosse o **centro** de cada um dos grupos que estamos tentando definir.

Imaginemos mais uma vez, tal qual nos exemplos de aprendizado supervisionado, uma imagem como temos abaixo, pontos de dados espalhados em um espaço bidimensional. Ou seja, nos nossos dados temos apenas duas variáveis (colunas) para tirarmos dali grupos separados por características semelhantes.

Com isso definido, podemos visualizar os dados dispostos no espaço bidimensional. O eixo horizontal representa os valores de uma das variáveis dos nossos dados e eixo vertical de outro. Inicialmente, só temos os dados representados pelos seus pontos no gráfico que correspondem às amostras (cada linha da nossa base de dados), e ainda não temos o conhecimento se ali há a possibilidade de distinção de grupos bem definidos.

Então, primeiro de tudo, o algoritmo seleciona amostras aleatoriamente conforme o número K, que será a quantidade de grupos procurados. Logo, supondo o K = 3, o modelo seleciona primeiro três amostras aleatórias para dar início ao processo de agrupamento. A partir de então, cada amostra que

usamos para inicializar o processo "vai ganhando território". O algoritmo calcula a "distância" de todos os outros pontos para cada um dos três pontos inicializados aleatoriamente, e cada ponto de dados irá pertencer ao grupo daquele ponto inicial, no qual a sua distância é a menor. Só que ele faz esse processo repetidas vezes, vê qual ponto está mais próximo do grupo, e recalcula o centróide, determinando uma versão mais atualizada do centro daquele grupo. O algoritmo encontrará uma convergência, ou seja, terminará a sua execução, quando os cálculos dos centróides se estabilizarem e não sofrerem mais alterações. Isso significa que cada grupo já "abocanhou" todas as amostras possíveis dentro daquele grupo de dados. Terminada essa etapa, não há como um grupo (agrupamento, ou "cluster") trazer amostras de mais nenhum outro.



Se observarmos a imagem ao lado, veremos que os pontos pretos representam cada um dos centróides exatamente no centro de cada grupo. Ao final, ele está estável já que não consegue "recrutar" novas amostras para então recalcular o seu centro. Esse processo é feito repetidamente desde o momento onde a primeira amostra é inicializada aleatoriamente, e se torna um "centróide" com base apenas nessa primeira amostra. O algoritmo segue então calculando novos centros a cada vez que mapeia a distância dos pontos subsequentes em relação a aquele centro. Dessa forma, determina a qual grupo pertence aquele ponto, respondendo a seguinte pergunta: de qual dos centros aquele ponto está mais perto?

Por fim, um alerta ligado aos casos reais: Nem sempre o algoritmo consegue retornar grupos separados e bem definidos como na imagem. Nem sempre os dados serão capazes de fornecer informações para que haja algum tipo de distinção de forma que consigamos agrupar dados com base em características!

Nem sempre o K-Means será um sucesso em problemas reais. Outra questão interessante é: como saber o número que tenho que definir para K? Pode ser empírico, ou em outras palavras, tentativa e erro, chuta um K e observa como o gráfico fará a distinção para aquele número de grupos determinados. Ainda assim, há um método mais sofisticado de fácil implementação chamado "Método do Cotovelo" (*elbow method*). Se algum dia você se aprofundar e for implementar o K-Means de fato, uma rápida pesquisa dará a você esse método que lhe permite determinar o melhor K possível para aqueles dados que você está tentando agrupar.

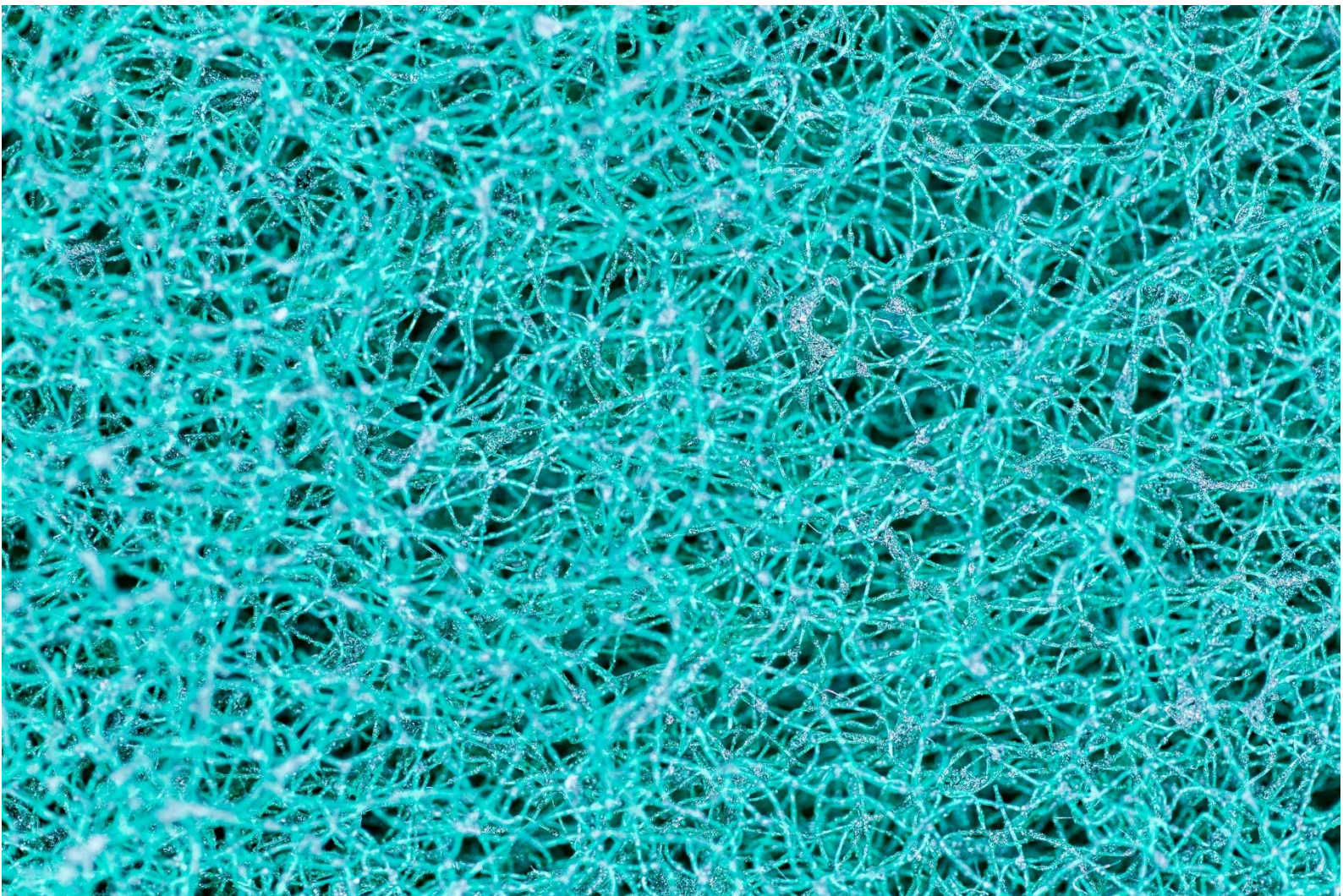
K Means Clustering: Prós e Contras

O K-Means é relativamente simples de implementar, o que se torna uma opção interessante em um problema de aprendizado não supervisionado, principalmente quando queremos começar com um algoritmo rápido e simples. Por isso, normalmente esse é o primeiro modelo a ser implementado para observar como o conjunto de dados se comporta antes de avançar para algo mais complexo.

Como desvantagem, temos que a escolha do **K** é manual e um tanto especulativa, sendo necessário observar graficamente (como na imagem acima) se os agrupamentos fazem sentido e estão minimamente divididos. Lembra na explicação que falamos que os agrupamentos são inicializados aleatoriamente, e a partir daí que os agrupamentos vão ganhando território? Pois então, a dependência dessa aleatoriedade passa ser uma desvantagem, necessitando executar o K-Means várias vezes para observar no gráfico qual possui o melhor resultado. E por fim, os ***outliers*** acabam influenciando muito, e negativamente, em uma distorção do agrupamento. Esses *outliers* podem ter o "seu próprio grupo" ao invés de serem ignorados, logo recomenda-se a remoção deles antes de aplicar o K-Means.

Redes neurais artificiais

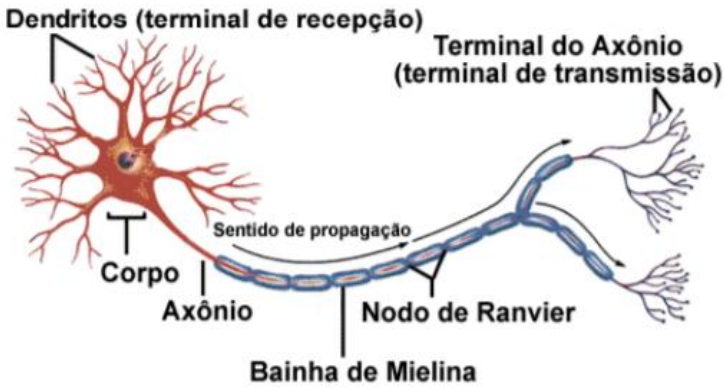
Como funcionam as redes neurais e para que servem?



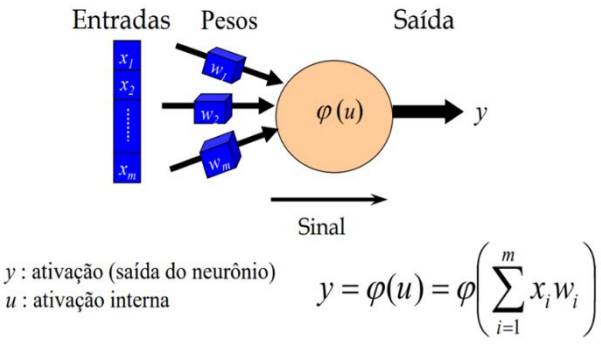
Se você recordar, citamos previamente o significado de Deep *Learning* lá no **capítulo 2**, o situando como uma subárea do *Machine Learning*. Mas aqui, mais uma vez, tal qual alertamos anteriormente, pode haver confusão entre as várias *buzzwords*.

O capítulo se chama Redes Neurais Artificiais, comumente mais citada pelo seu nome em inglês *Artificial Neural Networks*, pode-se dizer que esse tópico compõem as premissas básicas para que entendamos na sequência o ***Deep Learning***, uma subárea de Redes Neurais Artificiais.

Até aqui você deve ter notado que usamos e abusamos de analogias para facilitar o entendimento e os conceitos. No entanto, a analogia aqui se torna quase que uma regra, pois as Redes Neurais Artificiais têm em seu funcionamento uma espécie de réplica de como opera a Rede Neural do cérebro humano. Nos referimos ao processo feito na forma como o conhecimento é adquirido através da comunicação entre os neurônios, o chamado processo de **sinapse**. Veja abaixo a estrutura de um neurônio (fonte: **USP**):



Obviamente, que esse não é um *ebook* de anatomia, então não precisamos nos preocupar minuciosamente com cada parte no que tange o funcionamento dos neurônios do cérebro. Porém, se torna muito interessantes termos um entendimento generalista de comos esses neurônios operam e qual o seu propósito.



y : ativação (saída do neurônio)
u : ativação interna

$$y = \varphi(u) = \varphi\left(\sum_{i=1}^{m} x_i w_i\right)$$

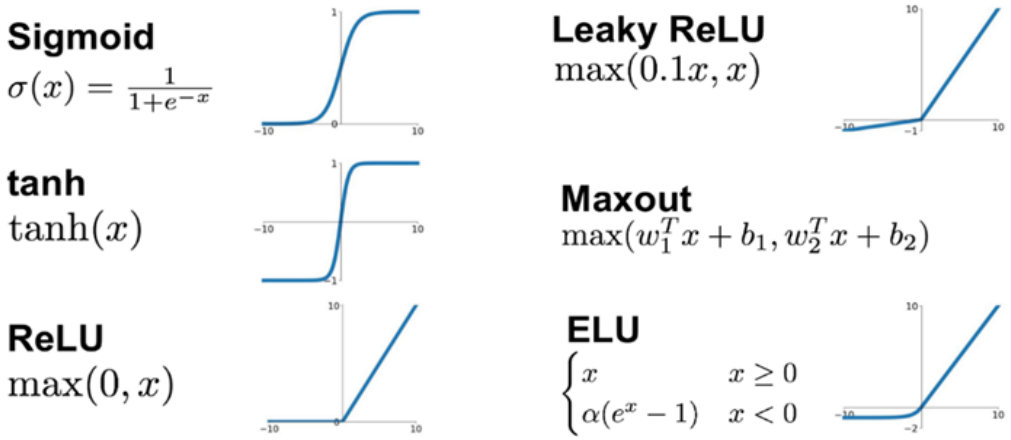
Fonte: Prof. Renato Tinós, Introdução às Redes Neurais Artificiais, Depto. de Computação e Matemática (FFCLRP/USP).

Não se assuste com as fórmulas matemáticas da imagem acima. A intenção dela é basicamente mostrar como o neurônio pode ser representado matematicamente. No neurônio biológico o sinal absorvido é interpretado, transformado e enviado para algum outro lugar. No neurônio artificial esse processo também deve ocorrer, só que nesse caso, precisamos encontrar uma representação matemática para ele.

A representação matemática funciona da seguinte forma: um sinal pode ter 'n' entradas, tal como mostra a imagem: **x1**, **x2**, ou até **xn**, onde esse 'n' muda de acordo com a aplicação. Essas entradas podem ter maior ou menor relevância para o processo que o neurônio estiver tratando. Por isso, cada uma dessas entradas recebe um peso **w**. Quanto mais importante for aquela entrada para o neurônio, maior será o seu peso **w**.

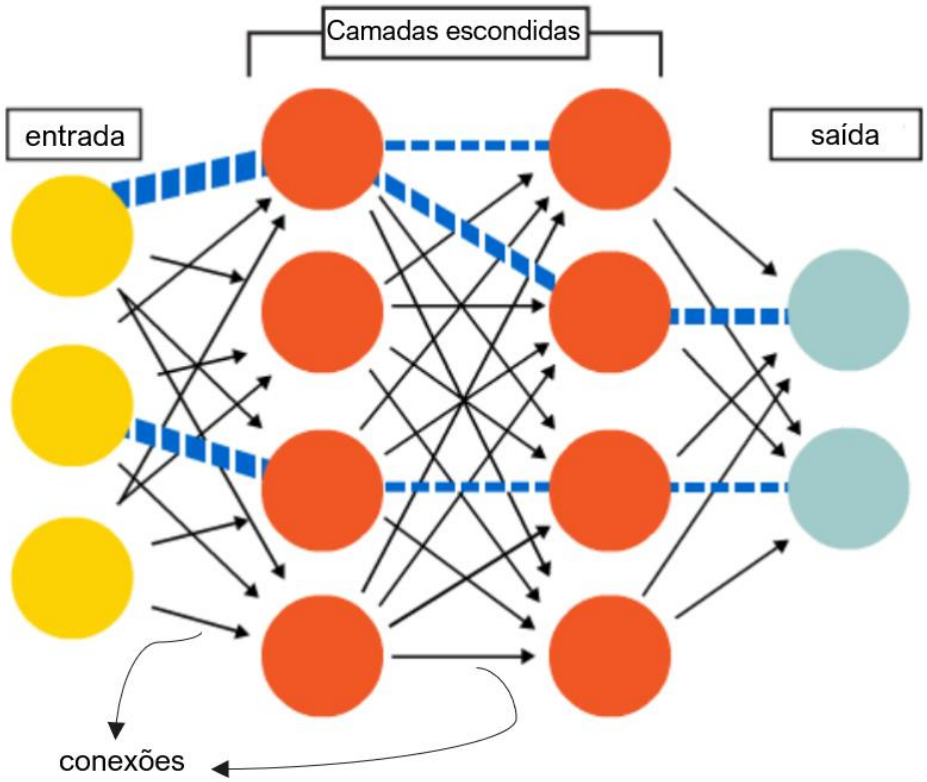
Uma vez que o sinal entrou no neurônio é já recebeu o seu "peso" adequado, passará agora pela **função de ativação** (círculo bege na imagem acima). Essa parte do neurônio recebe esse nome por ter o papel de "ativar" o neurônio quando o sinal entra nele. A função dela é transformar o sinal de acordo com alguma configuração pré-definida. Uma vez que essa função "liberta" o resultado, o neurônio vai passar esse sinal transformado para algum outro lugar, que pode ser um outro neurônio na rede neural ou o resultado da rede neural propriamente dita na sua saída.

Você não precisa se preocupar nesse momento qual é o formato da função de ativação ou que cálculos elas realizam exatamente. Apenas tenha em mente que a função de ativação serve como o componente central de um neurônio que atua para modificar o sinal que ele recebeu, ou seja, uma espécie de "filtro". A título de curiosidade, apresentamos a você algumas das funções de ativação mais utilizadas dentro dos neurônios.



Uma vez que já entendemos bem a estrutura dos neurônios artificiais, podemos prosseguir com o aprendizado das redes neurais. Elas são simplesmente um modelo que utiliza vários neurônios na sua

arquitetura, ou literalmente uma rede de neurônios. Vamos observar a imagem abaixo para entender melhor a estrutura de uma rede neural.



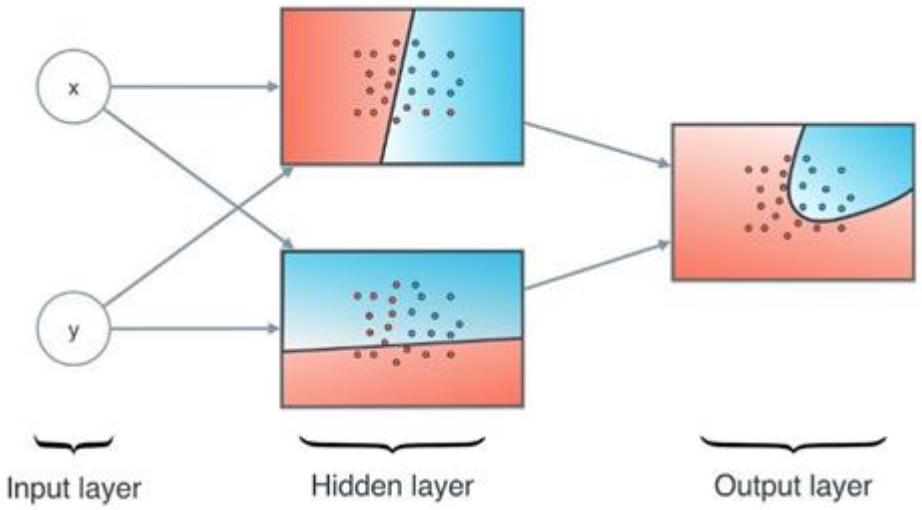
Observe que a rede neural é organizada em **camadas**. Há a camada inicial à esquerda, chamada de camada de entrada, ou do inglês, ***input layer***. O sinal é passado para as camadas seguintes, que preenchem o meio da rede neural. Essas camadas recebem o nome de "camadas escondidas", ou ***hidden layers***, devido ao fato de que quando a rede neural cresce muito, se torna praticamente impossível acompanhar toda a estrutura dessas camadas centrais. Por fim, temos a camada de saída, ou ***output layer***, onde o resultado é disponibilizado. Vale lembrar que cada círculo que compõe as camadas corresponde aos **neurônios** da rede neural.

Quer simular uma rede neural? Clique **aqui**.

Após você visualizar a estrutura de uma rede neural acima, aqui é a ocasião perfeita para você entender a relação **Rede Neural Artificial** e ***Deep Learning***. Simplesmente, o *Deep Learning* se caracteriza por ter inúmeras camadas ocultas, as *hiddens layers*, e justamente essas várias camadas denotam um aprendizado mais **profundo**, uma vez que a informação terá que percorrer mais etapas de aprendizado até a camada de saída. Como já mencionamos, muitos dizem que *Deep Learning* é um nome midiático, quando na verdade poderíamos chamar de *Perceptron* Multicamadas. Mas, para deixar bem claro, o *Deep Learning* nada mais é que uma Rede Neural Artificial com mais de uma camada

oculta (as camadas em laranja na imagem acima). Portanto trata de uma ferramenta mais robusta, poderosa e custosa computacionalmente.

Qual a vantagem de se utilizar várias camadas e vários neurônios? A figura abaixo dá uma pista:



Observe que um neurônio sozinho não é muito habilidoso na hora de separar duas classes dentro de um conjunto de dados. Entretanto, se unimos os resultados desses dois neurônios, conseguimos um modelo de classificação que é bem melhor que os dois separados. Observe que os neurônios isoladamente tendem a fazer a separação apenas de forma linear. Quando juntamos os resultados dos dois neurônios, somos capazes de obter uma separação não-linear mais complexa, e consequentemente, mais precisa.

Uma vez que entendemos isso, as redes neurais começam a ficar cada vez mais complexas. Isso porque o número de camadas ocultas, e consequentemente o número de neurônios, começa a aumentar consideravelmente, até que se torne inviável representar essas redes neurais para analisá-las visualmente. É a partir daí que entramos no ponto do *Deep Learning*, o **Aprendizado Profundo**, que recebe esse nome justamente pela profundidade de camadas da rede neural, tal qual mencionamos anteriormente.

Mais uma vez, nesse capítulo trouxemos esse exemplo simplificado com o intuito de trazer uma visualização do que está acontecendo, além de identificar o racional do processo realizado pelo algoritmo.

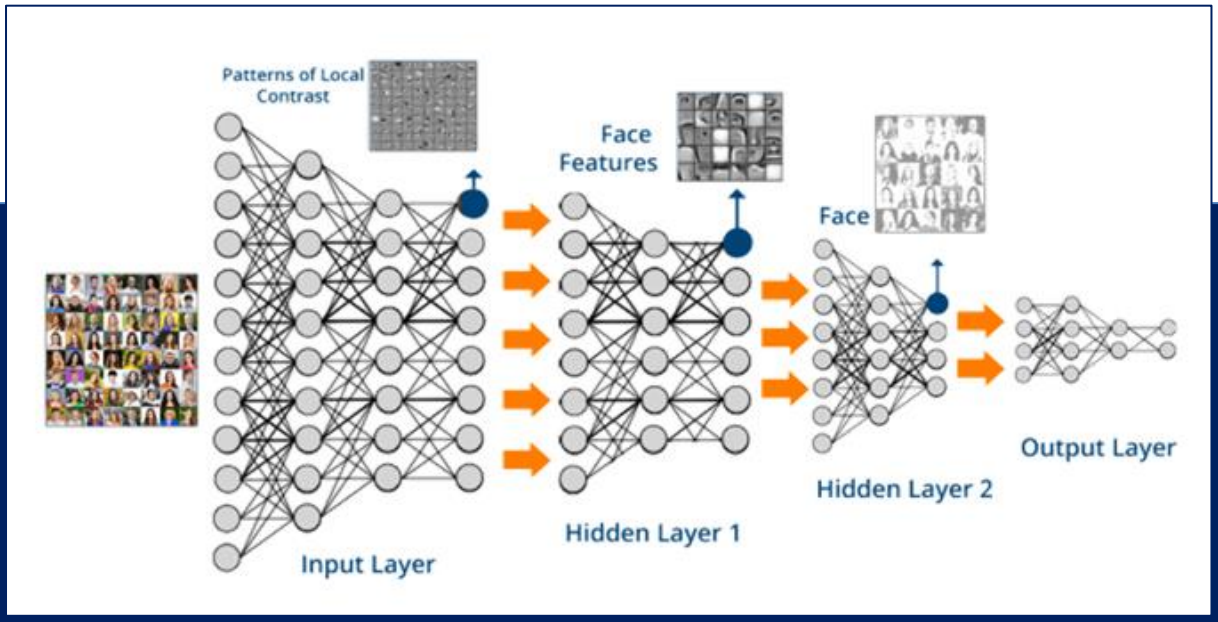
Aplicações de Redes Neurais Artificiais

Aqui trago alguns dos principais exemplos de aplicações de Redes Neurais Artificiais em forma de **Deep Learning**. Muitas aplicações estão mais perto do seu cotidiano do que você imagina, e essas

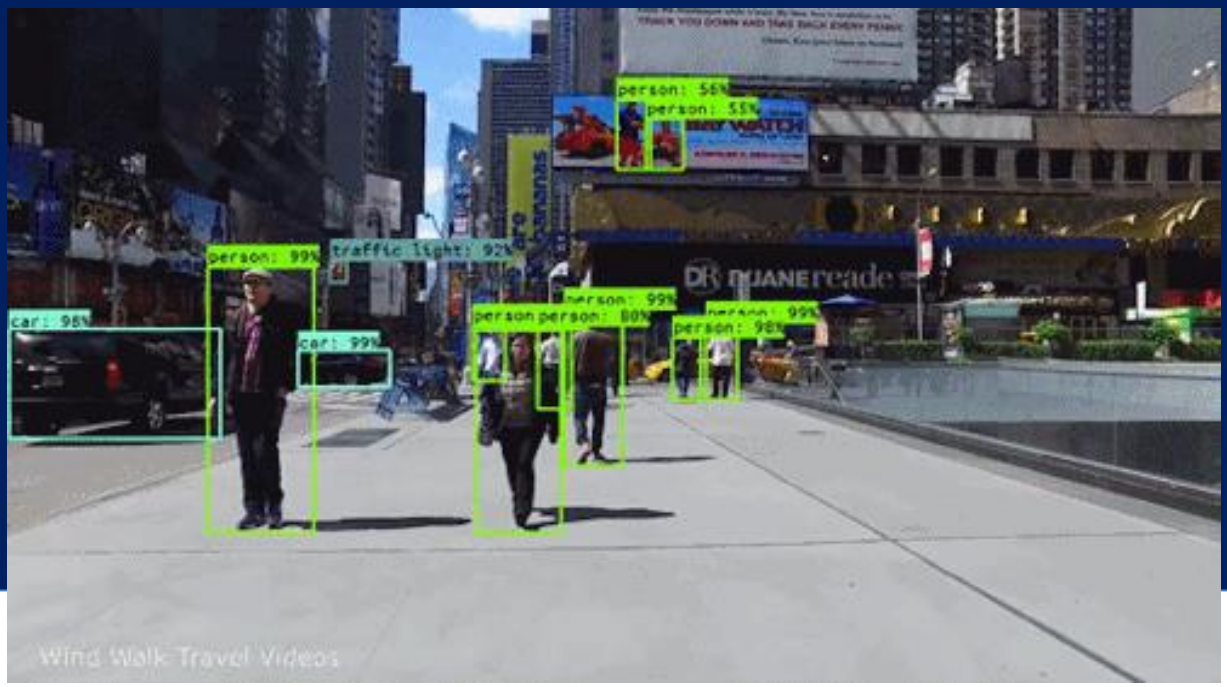
práticas se tornaram cada vez mais comuns e partes de nossas vidas, é com certeza um caminho sem volta.

Carros Autônomos: você talvez já deve ter ouvido falar dos carros da *Tesla*, o carro autônomo, e através das Redes Neurais, ele consegue detectar os outros carros e objetos na pista para tomar as decisões enquanto conduz o veículo.

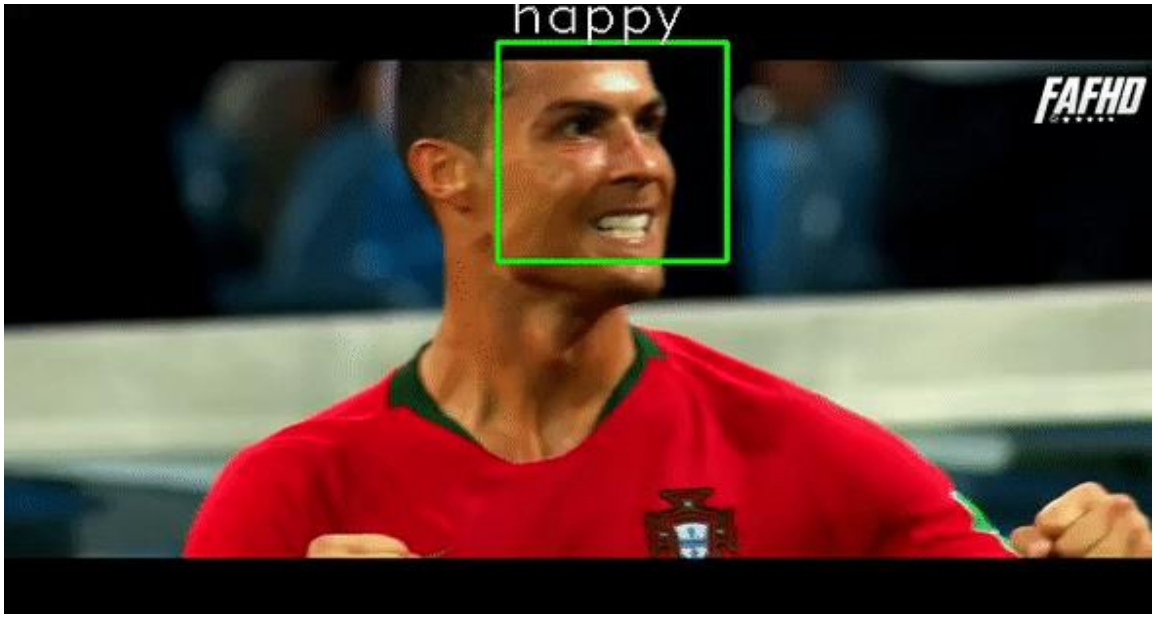
Reconhecimento Facial: esse você já deve ter visto na prática. Quando em uma foto no *Facebook*, a própria rede social seleciona e sugere que você marque uma pessoa da sua rede de contatos no momento que a foto é adicionada.



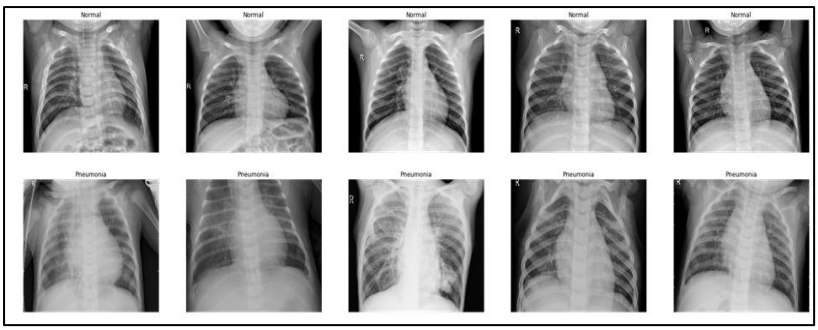
Abaixo temos um exemplo especialmente utilizado na China, na qual o governo usa técnicas de visão computacional com Redes Neurais para monitorar o que as pessoas estão fazendo em espaço público. Inclusive podendo ser usada juntamente com a técnica de reconhecimento facial.



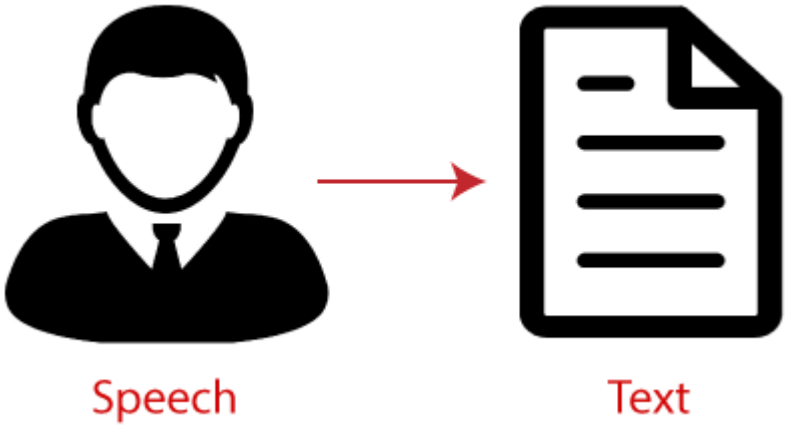
Reconhecimento de Emoções: A Rede Neural pode treinar através de um banco de dados com várias expressões de pessoas, e assim para capturar o padrão de como as micro expressões do rosto se comportam para manifestar emoções. Um exemplo que teve apelo midiático foi quando uma matéria do **Estadão** utilizou técnicas de Redes Neurais para detectar as emoções dos candidatos ao longo de um dos debates para a eleição presidencial do Brasil no ano de 2018.



Aplicações na área da saúde: Essa tem sido uma das áreas mais exploradas no uso de Redes Neurais. Muitas vezes através das *HealthTechs*, onde há uma gama de aplicações diversas. Como por exemplo: um algoritmo que identifica se a pessoa está com pneumonia ou não, ou que identifica se a pessoa tem um câncer benigno ou maligno, entre outras. Muitas vezes o algoritmo necessita de um banco de dados com diagnósticos já conhecidos, para que ele possa aprender esses padrões e replicar em pacientes onde ainda ele não conhece o diagnóstico final, mas tem acesso a seus exames. Com o aprendizado adquirido será possível chegar a um resultado de forma mais rápida que através do próprio médico.



Geração Automática de Legendas: Exato, as redes neurais são capazes de gerar legendas através da voz emitida. Faz a tradução da linguagem escrita para uma linguagem falada.



Essas são apenas algumas das principais aplicações de Redes Neurais Artificiais e *Deep Learning*. Mas poderíamos citar inúmeras outras, como tradução automática de idiomas, ou um algoritmo que consegue jogar poker, geração automática de melodias musicais, aplicações no setor agro conseguindo identificar qualidade de insumos através de visão computacional, entre muitas outras.

Esses exemplos é para que você tenha em mente quão poderosa é a ferramenta, e o quanto espaço há para que ela seja explorada ao longo dos próximos anos. É muito do que fora falado no **capítulo 4**, muitas das coisas relacionadas a essas tecnologias ainda, atualmente, são mais estudadas no âmbito acadêmico do que propriamente aplicados na prática. Mas da mesma forma que todas as tecnologias que vemos consolidadas agora, foram exploradas e estudadas desde os anos 60 e 70. Provavelmente teremos um futuro de infinitas possibilidades com o aumento da capacidade computacional e de capacidade de coleta de dados. Lembra do exemplo do disquete CD, DVD e por aí vai? Vimos a capacidade de armazenamento computacional crescendo exponencialmente ao longo dos anos e a tendência é que permaneça em um ritmo acelerado.

Não nos aprofundamos tanto quanto gostaríamos no presente tópico aqui neste *ebook*. Por se tratar de um tema mais avançado, e esse ser um *ebook* introdutório, considerei que agora o mais importante seria uma apresentação ao tema, um primeiro contato, e se você prosseguir na área, será inevitável o aprofundamento no mesmo. No entanto, o objetivo é uma quebra de gelo e que você possa vislumbrar de forma mais clara seus funcionamentos, aplicações e o racional de como este tipo de modelagem opera.

Redes Neurais Artificiais: Prós e Contras

Como já citamos, e implicitamente podemos notar, as Redes Neurais Artificias são ferramentas muito poderosas, que poderá funcionar bem em dados lineares e não lineares, estruturados e não estruturados. Só que devida a sua capacidade, as Redes Neurais demandam muitos dados no seu processo de modelagem para que possam obter um bom aprendizado. Em outras palavras, precisa enxergar em seus dados de treino, uma representatividade robusta de casos para que possa performar todo o seu potencial quando aplicado a uma solução real.

Talvez o principal problema dessa tecnologia seja a necessidade de uma estrutura parruda. E com isso quero dizer, que pela sua natureza, demanda quantidade de banco de dados muito grandes, além de que a complexidade do seu processamento também exige muito do custo computacional. Então não é simplesmente ter uma ferramenta poderosa e aplicá-la, requer uma infraestrutura cara, além de que recai naquilo que falamos ao longo desse *ebook*: de que adianta um algoritmo complexo, se os nossos dados não têm poder preditivo sobre o que queremos modelar? Há muitas outras implicações nas decisões de negócios na hora de escolher um algoritmo mais complexo como esse ou não, por exemplo: a aplicação que estou fazendo necessita uma resposta rápida? Tudo bem ter uma performance um pouco pior, mas um custo de tempo e computacional menor? Tudo isso irá depender justamente do contexto que você aplicará o algoritmo, a finalidade.

Tem uma história do *kaggle* que exemplifica bem o que quero trazer aqui. Para os desavisados, o *kaggle* é uma plataforma de competição de *Machine Learning* onde várias empresas, incluindo as maiores do mundo, propõe desafios e recompensas para aqueles que elaborarem as melhores soluções de *Machine Learning* aplicadas aos dados e métricas propostos por eles. Uma empresa de *streaming*, certa vez propôs um desafio, visando que o algoritmo fosse assertivo ao recomendar novos filmes e séries para seus usuários com base no que este usuário já havia consumido. Acontece que a solução vencedora era de um algoritmo onde seu treinamento demoraria quase 30 dias para realizar a modelagem, tamanha a complexidade. Ou seja, um usuário que quer ver um filme em seu horário de lazer não vai esperar isso tudo para obter uma recomendação, certamente. Sendo assim a empresa se antecipa, obtendo uma solução complexa que obviamente tem muito potencial para ser útil no futuro, mas como decisão de negócio atualmente, a performance do modelo fica em segundo plano, tendo que buscar um equilíbrio entre performance e tempo de resposta.

Por último, uma desvantagem amplamente discutida sobre as Redes Neurais é que para muitos ela é uma *black-box*, caixa preta. Justamente como mencionamos antes, o seu funcionamento é complexo, e um executivo de alta patente que precisa tomar uma decisão não tem tempo (e nem valeria a pena) para se ater a detalhes matemáticos e entender como uma tecnologia de alta complexidade

funciona. Então, para muitos o seu funcionamento é um mistério, o que causa uma certa desconfiança, além de uma resistência inicial sobre a sua confiabilidade e seu desempenho.

Ao contrário de outros algoritmos que citamos anteriormente como: árvores de decisão e seus algoritmos baseados, ou até mesmo a regressão linear, não é possível atribuir qual o impacto das variáveis do conjunto de dados no que o modelo está prevendo em sua saída. Felizmente, como já mencionamos, recentemente uma ferramenta chamada SHAP tem auxiliado em dar explicabilidade em modelos com enorme potencial e complexidade. Uma vez que a explicação se torna fundamental quando estamos em um contexto de negócios, principalmente para quebrar objeções de outros setores envolvidos em tomadas de decisões, que não necessariamente tem expertise em tecnologia e modelos de *machine learning*. Afinal uma empresa sempre é composta de times e pessoas multidisciplinares, e se faz necessário um denominador comum entre todas as partes.

Métricas de desempenho dos modelos

Aqui aprendemos a medir a performance e acurácia dos modelos que desenvolvemos, sendo um tópico muito importante para determinarmos a usabilidade do nosso algoritmo.

Ao longo do *ebook*, você deve ter visto muitas palavras como **desempenho**, **performance** e **métricas**. Aqui explicaremos os tipos de métricas que nos indicam se um modelo está se comportando bem, e obviamente temos tipos de métricas adequadas a cada tipo de modelo. Então, um modelo que procura solucionar um problema de regressão (valor numérico), não pode ter a mesma métrica que um modelo que procura solucionar um problema de classificação (rótulos).

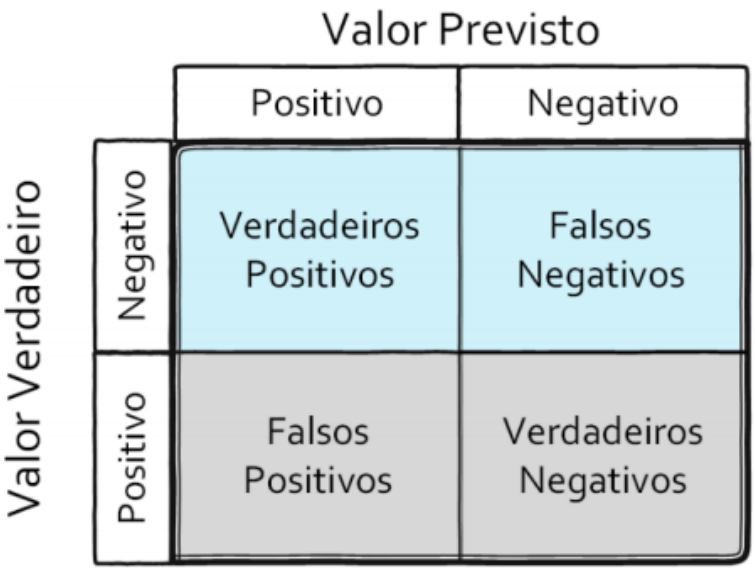
Lembre-se lá do início do *ebook*, no **capítulo 2**, quando falamos como funcionava o processo de aprendizado de um modelo de *machine learning*. No diagrama, colocamos que ao termos uma base de dados, dividimos esses dados em **treino** e **teste**. Logo, o modelo usa a base de treino para o seu processo de modelagem (treinamento) e identificação dos padrões existentes, e para validar se o modelo performa bem usamos os dados de teste.

Qual a lógica? Os dados de teste ficaram separados, "escondidos" do processo de modelagem, então o modelo jamais viu aqueles dados. Só que nós temos aqueles dados de teste e sabemos como de fato eles se comportaram! Assim, mandamos para o modelo os dados de teste para identificar se ele conseguirá ter poder de predição. Dessa forma, poderemos ver o que o modelo concluí e comparamos com o resultado real que aconteceu de fato. Isso nos permitirá ter uma diretriz e extrair métricas para averiguar o desempenho do modelo. Tudo para ter a melhor estimativa de como o modelo se comportará num cenário real, "em produção" como dizemos no mundo do *machine learning*.

As métricas envolvem alguns conceitos básicos de matemática, mas nada de outro mundo. Explicaremos detalhadamente, dividindo em **métricas para classificação** e **métricas para regressão**. Lembrando que abordaremos as métricas mais clássicas, tendo muitas outras disponíveis, mas aqui serão mostradas aquelas que devem ser as primeiras que você precisa saber.

Métricas para Classificação

Para entendermos as principais métricas de classificação devemos observar a chamada **Matriz de Confusão** (nome sugestivo, não?) na imagem abaixo (fonte: **Medium – Model Validation Data Science**):



Lembre-se do que falamos anteriormente a respeito dos rótulos ou das "classes" dos dados. A intenção dessa tabela é justamente determinar se o nosso modelo acertou ao dizer que uma amostra pertence a uma determinada classe. Por exemplo: temos pacientes com e sem pneumonia, e criamos um modelo para identificar os pacientes que de fato têm a doença. Uma forma de avaliar o desempenho do modelo é comparar a classificação que o modelo deu para esses pacientes com a situação real deles. A matriz de confusão é intuitiva, mas vamos esclarecer o que é cada um dos quatro quadrantes.

Não sei em que época você está lendo este *ebook*, mas escrevi no auge da crise do **covid-19**. Vou usar o exemplo dos testes para detecção da doença para deixar claro do que estamos falando aqui. Em época de covid, muito tem se falado dos "testes rápidos", no entanto eles não se demonstram absolutamente confiáveis. Muitas vezes, são enviadas amostras para análise em laboratórios, que demoram mais, mas são totalmente confiáveis. Imaginemos essa relação entre os **testes rápidos** e os **testes em laboratório**, para entendermos abaixo:

- **Verdadeiro Positivo (VP)**: teste rápido deu que a pessoa possui covid-19, o que foi confirmado em laboratório, portanto configura Verdadeiro Positivo.
- **Verdadeiro Negativo (VN):** teste rápido deu que a pessoa não possui covid-19, o que foi confirmado em laboratório, portanto configura um Verdadeiro Negativo.
- **Falso Positivo (FP)**: esse é o que mais escutamos falar na época do covid: O teste rápido deu positivo, ou seja, acredita-se que a pessoa possui covid-19. Isso acaba sendo

desmentido pelo laboratório, que confirma: a pessoa não está com covid-19. Vemos uma falha do teste rápido, configurando Falso Positivo.

- **Falso Negativo (FN)**: por fim, imaginemos que que o teste rápido deu negativo apontando que a pessoa não está com covid-19. Ele acaba sendo desmentido pelo teste em laboratório. Na verdade, a pessoa está sim com covid-19, configurando uma falha no teste rápido, portanto um Falso Negativo.

Tudo bem, compreendemos essas 4 bases fundamentais para entender as métricas de classificação. A diferença é que no contexto de *machine learning*, analogamente o **teste rápido** seria a classificação apontada pelo **modelo** e o **teste em laboratório** o que aconteceu **de fato**, podendo assim comparar a assertividade através de alguns cálculos para entender o desempenho. Assim temos as 4 métricas principais abordadas: **Accuracy**, **Precision**, **Recall** e **F1-Score**.

- **Accuracy**: avalia a performance do modelo no geral. Mede a quantidade de vezes que o modelo classificou corretamente entre todas as classificações feitas, conforme mostra a equação. Repare que na parte de cima da divisão temos todas as vezes que modelo acertou e embaixo todas as classificações:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

- **Precision**: entre todas as classificações que o modelo apresentou como positivo, busca quantificar quantas eram positivas de fato, como mostramos na equação. Repare que em cima temos todas as vezes que o modelo acertou um positivo que era de fato, e embaixo todas as vezes que ele apontou que era positivo, independentemente de ser de fato ou não.

$$Precision = \frac{VP}{VP + FP}$$

- **Recall**: dentre todas as vezes que a classificação é positiva na realidade, quantas o modelo conseguiu acertar, conforme equação. Repare que em cima temos todas as vezes que o modelo apontou que era positivo e era de fato, e embaixo todas as vezes que tínhamos casos que eram

positivos na realidade, tanto VP quando o modelo acertou os positivos ou FN, quando o modelo apontou como negativo, mas na verdade era positivo.

$$Recall = \frac{VP}{VP + FN}$$

- **F1-Score**: essa métrica nos dá uma combinação de *Precision* e *Recall*, usando ambos para trazer um número único, conforme mostra equação. Ou seja, para calcular essa métrica precisamos dos valores de *Precision* e *Recall* calculados nas equações explicadas anteriormente.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Conclusão geral sobre métricas de Classificação

Com base nas métricas, não há melhor nem pior, irá depender do contexto do problema que se deseja resolver. É bom olhar todas para se ter diferentes perspectivas do problema. Normalmente, procuramos que o modelo tenha bons resultados em todas as métricas. A intenção é que se possa obter uma conclusão do resultado dessas várias métricas. Como em uma pesquisa eleitoral, várias pesquisas terão resultados um pouco diferentes, mas todas apontando na mesma direção.

Trazendo um exemplo real de como definir quando utilizar cada métrica. Imagine um grande banco que tenta identificar fraudadores para cancelar o cartão de crédito desse cliente evitando prejuízo. A primeira coisa que saberemos é que esse conjunto de dados será desbalanceado, ou seja, como fraude é um evento raro, teremos muito menos ocorrências de fraude no nosso *dataset*. Se passarmos isso para o modelo, ele pode ter uma *accuracy* alta, mas o modelo estará enviesado pois ele acessou muito mais casos onde não havia fraude do que onde havia. Sendo assim ao olharmos apenas *accuracy*, saberemos as vezes que o modelo acertou no geral, mas como no caso da fraude temos mais acessos a casos que não houve fraude, a métrica terá seus resultados baseados nos acessos em casos que ele acertou justamente nos casos que não houve fraude. Sendo assim, nesse caso, deve-se tomar cuidado por um contexto de negócio: se for cancelado o cartão de crédito de um cliente que não é fraudador, a instituição pode criar uma indisposição com um bom cliente.

Portanto, nesse exemplo, em que devemos ter cuidado para não atingir clientes com práticas corretas, devemos olhar para outras métricas como ***Recall*** e ***Precision***, justamente pois avaliam a performance do modelo em cima de quantos verdadeiros positivos o modelo acertou. Assim, concordamos que aqui faz mais sentido olhar do ponto de vista dessas métricas para reduzir chances de incorrer em erros de **Falso Positivos** como citado anteriormente.

As circunstâncias de negócios são diversas, se a decisão em cima do modelo fosse enviar um email para evitar que um cliente cancelasse uma assinatura, teria bem menos impacto do que cancelar o cartão de crédito dele. Logo, quero trazer aqui as várias nuances que compõe a tomada de decisão na hora de escolher a melhor métrica de avaliação do modelo.

Métricas para Regressão

Vamos às métricas de problemas de Regressão. Lembre-se que aqui o resultado do modelo se trata de valores numéricos e contínuos. Essas métricas envolvem simples cálculos matemáticos que esmiuçaremos aqui. Abordaremos três métricas de regressão principais: **MSE (Mean Squared Error)**, **RMSE (Root Mean Squared Error)** e **MAE (Mean Absolute Error)**.

A diferença entre o **MSE** e **RMSE** é apenas uma, então podemos pegar o mesmo exemplo e salientar essa diferença para que fique claro.

- **MSE:** é a média do quadrado dos erros, uma das métricas de regressão mais utilizadas. A fórmula abaixo nos mostra, que para cada linha, obtém-se o valor real e o valor predito, subtrai-se o valor real do predito e eleva-se o resultado ao quadrado, e após faz-se uma média desse resultado para todas as linhas. Ficará mais claro no exemplo com a tabela mostrando o passo-a-passo.

$$MSE = \frac{1}{n}\Sigma\left(y - \hat{y}\right)^2$$

- **RMSE:** raiz da média do quadrado dos erros, é a métrica mais usada amplamente. Ela penaliza erros grandes, já que são elevados ao quadrado, e serve quando queremos modelos onde erros grandes são muito indesejados. Muito parecido com o MSE, com uma única diferença. Faz-se o mesmo processo, pega-se o valor real e subtrai o predito, o resultado elevamos ao quadrado,

isso para cada linha. E faz-se a média desse valor calculado para todas as linhas. Até aqui igual ao MSE, certo? Então, aplica-se a raiz quadrada em cima desse valor único, como se o RMSE só aplicasse uma raiz quadrada em cima do resultado de MSE, e isso é tudo que os diferencia em termos matemáticos. Usaremos o mesmo exemplo na tabela para ambos e ficará mais claro.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Segue o exemplo segundo a tabela abaixo. Para fins didáticos, temos uma pequena base de dados com duas variáveis (X e Y), que são a respeito das características do que se deseja modelar. Logo em seguida, temos o resultado de fato do que se deseja modelar (Valor Real), e o resultado que o modelo teria predito (Valor Predito). Em seguida, temos uma coluna que elucida o cálculo feito, onde traz a subtração do resultado real menos o que foi predito elevado ao quadrado. E por fim o resultado do erro.

Por mais que você olhe e possa achar que o Valor Real nos retorna uma classificação, isso pouco importa, por estarmos tentando predizer aqui um valor contínuo e não simplesmente um rótulo, e para fins didáticos o exemplo abaixo torna-se bastante claro.

X	Y	Valor Real	Valor Predito	Cálculo	Erro
0	0	0	0,406	(0 - 0,406)²	0,164
0	1	1	0,432	(1 - 0,432)²	0,322
1	0	1	0,437	(1 - 0,437)²	0,316
1	1	0	0,458	(0 - 0,458)²	0,209

Até esse ponto, tudo igual para **MSE** e **RMSE**, agora cada um segue um caminho distinto:

- **MSE**: Soma dos Erros, dividindo pela quantidade de amostras.
  - Soma dos erros: 0,164 + 0,322 + 0,316 + 0,209 = 1,011
  - Quantidade de amostras (linhas): 4
  - **MSE** = 1.011 / 4 = **0.252**

- **RMSE**: Soma dos Erros, dividindo pela quantidade de amostras. Depois disso, aplicamos uma raíz quadrada. Basta só aplicar uma raiz quadrada sobre o MSE:

- Soma dos erros: 0,164 + 0,322 + 0,316 + 0,209 = 1,011
- Quantidade de amostras(linhas): 4
- MSE = 1.011 / 4 = 0.252
- **RMSE** = $\sqrt{0.252}$ = **0.501**

Finalmente, chegamos no **MAE (Mean Absolute Error)**, que não chega a ser tão diferente das métricas anteriores. É até mais simples! Novamente, apelamos a um exemplo com tabela para ficar mais claro.

- **MAE:** É a média absoluta do erro, ou seja, faz a média da diferença entre o valor real e o que foi predito para cada uma das linhas. Essa métrica é mais robusta para identificar valores que estão muito errados, e não penaliza tanto erros menores como os dois que vimos anteriormente. Não é indicado para onde desejamos prestar atenção a valores mais extremos. Ficará mais claro no exemplo acompanhando a tabela a seguir.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

Segue o exemplo, semelhante ao anterior, só que agora, por ser uma **média absoluta**, a mudança aqui é que não elevamos a diferença entre o valor real e o valor predito ao quadrado. Além disso, por ser **absoluto**, significa que o erro não terá diferenciação entre valores positivos e negativos. Por exemplo, o cálculo na primeira linha teria que ser -0,406, mas por se tratar de um número absoluto, não teremos diferenciação entre sinais. Será 0,406, não havendo valores negativos, apenas sua diferença absoluta.

X	Y	Valor Real	Valor Predito	Cálculo	Erro
0	0	0	0,406	(0 - 0,406)	0,406
0	1	1	0,432	(1 - 0,432)	0,568
1	0	1	0,437	(1 - 0,437)	0,563
1	1	0	0,458	(0 - 0,458)	0,458

- **MAE**: Soma dos Erros, dividindo pela quantidade de amostras. Após calcularmos a diferença **absoluta**, dividimos pela quantidade total de amostras (linhas).
  - Soma dos erros: 0,406 + 0,568 + 0,563 + 0,458 = 1,995
  - Quantidade de amostras(linhas): 4
  - **MAE** = 1,995 / 4 = **0,499**

Conclusão geral sobre métricas de Regressão

Você deve estar se perguntando: qual métrica usar e em qual circunstância? Pois bem, falamos aqui apenas de três principais, mas existem tantas outras usadas. Para saber qual métrica usar, você deve conhecer relativamente o problema de regressão que está sendo solucionado através de um modelo supervisionado de *machine learning*.

Primeiros vamos tratar do **MSE** e do **RMSE**, que como vimos anteriormente, eles elevam o erro ao quadrado, ou seja, **punem outliers** com mais rigor. Vamos ao exemplo: imagine que nos dados de teste temos que o valor real de fato é 2, mas o modelo previu 8, o MSE e o RMSE elevaram esse erro que é de valor 4, ao quadrado, ou seja, 16. Então podemos concluir que dados onde há muitos *outliers*, é interessante usar essas métricas de erro justamente por elas serem sensíveis na identificação dessas ocorrências. Assim fica mais fácil detectá-los e removê-los.

Mas ainda assim, qual a diferença entre **MSE** e **RMSE** em termos práticos? Bem, o **RMSE** é mais fácil de entender. Imaginemos que a nossa variável a ser predita está em unidade de dinheiro, tipo o dólar. Ao final de tudo, supondo um erro médio do quadrado (MSE) de 16. Só que a unidade não é dólar, mas sim dólar ao quadrado. Logo, para fins de entendimento a raiz quadrada do erro médio (RMSE) será 4, pois o RMSE é a raiz do MSE, e a raiz de quadrada de 16 é 4 dólares. No fim temos em RMSE uma métrica que pune ***outliers*** e é mais fácil de interpretar por extrair a raiz e trazer uma unidade que está elevada ao quadrado de volta a sua unidade original, facilitando entendimento do erro.

Mas e o **MAE**? Novamente, requer entendimento dos dados. O MAE deverá ser utilizado quando ***outliers*** não são preocupação, pois eles são mais raros e queremos concentrar nossos esforços apenas detectar um erro mais harmônico no modelo. Como exemplo, em dados de mercado financeiro, que são extremamente ruidosos, temos muita utilização do MAE, justamente pelo fato de ***outliers*** serem uma exceção extrema, assim nos preocupamos em entender o erro médio do modelo apenas.

Por fim, é interessante observar todos os erros para tomada de decisão, realizar melhorias e experimentos e ver como os erros são afetados, especialmente se você tem muitas dúvidas sobre os dados com que está lidando. Mas é sempre interessante observar o erro de diferentes pontos de vista, tanto do erro que identifica e se torna muito alto por causa de ***outliers***, como um erro médio mais geral.

Considerações finais

Chegamos ao fim do livro, mas ainda temos muito a aprender sobre dados e muitos desafios pela frente. Este é apenas o início da escalada. A seguir conclui-se a obra e são trazidos alguns pensamentos de **William Edward Deming**.

Como último ato deste *ebook*, gostaria de trazer uma charmosa frase que simboliza os adeptos da cultura analítica. Pense em quantos anos, um cara nascido em 1900 e que veio a falecer em 1993, esteve à frente de seu tempo. William Edward Deming, foi um dos mais célebres propagadores da importância dos dados. Mais do que um estatístico, sua carreira teve várias faces como: acadêmico, pesquisador, engenheiro eletricista de formação e consultor de gestão.

Considerado uma lenda sobre gestão de processos e qualidade. Ficou marcado, principalmente pelo milagre econômico japonês, pós segunda guerra, quando o país saiu das cinzas para se tornar uma das economias mais pujantes do mundo. Muito dessa virada de mesa atribuída aos princípios pregados por Deming, inclusive sendo reconhecido em território japonês muitos antes do que nos EUA, sua terra natal.

Em Deus nós confiamos. Todos os outros devem trazer dados.

Agora imagine, um dos patriarcas das decisões orientadas a dados foi responsável por grandes mudanças na gestão e na medição da eficiência de processos, especialmente em termos de manufatura. Tudo isso em uma época onde mal tínhamos acesso a computadores, quanto mais capacidade para armazenar grandes quantidades de dados.

Dr. W. Edwatd Deming, trouxe a tona muitas reflexões sobre cultura analítica, ainda mais quando soltou outra excelente frase sobre a importância de dados.

Sem dados você é apenas mais uma pessoa com uma opinião

Provavelmente você já entendeu onde eu quero chegar. Nós como seres humanos sempre estamos sujeitos a vieses emocionais e cognitivos nas nossas impressões sobre qualquer assunto, e muitas vezes essas impressões poderão estar distantes da realidade e do que vai nos trazer resultados na realidade, especialmente no âmbito dos negócios. Se um homem é capaz de trazer essas conclusões nos anos 60, 70 e 80, imagine o que podemos alcançar hoje e nas décadas subsequentes, o poder que temos nas mãos, e que ainda há uma margem gigantesca para ser explorada.

Não se esqueça: aquele que chega primeiro, bebe a fonte limpa. Que esse seja o seu primeiro despertar para o poder dos dados, e como essa capacidade de armazenar, processar, e aprender através dos algoritmos de *machine learning* é um caminho sem volta que apenas tende a evoluir. Não somente em termos de tecnologia, mas especialmente na maneira como as pessoas vão entendendo e se adaptando a esse tipo de tecnologia, entendendo que isso não se trata de mágica e nem de bola de cristal, mas sim sobre "redução de achismos" e decisões mais assertivas ao longo do tempo. Em outras palavras: sempre acertar bem mais do que errar em suas decisões. Esse processo percorrerá um caminho natural, como já vem sendo, e continuará, principalmente a partir do momento em que este tipo de recurso se torna um diferencial competitivo cada vez mais evidente em empresas de sucesso.

Tenha em mente que se trata de um estudo contínuo e este *ebook*, como já afirmado, é a porta de entrada. Aqui buscamos trazer o primeiro contato sobre as diversas áreas do conhecimento que se entrelaçam além de, sempre através de exemplos, mostrar suas implicações no mundo real.

Aproveite o oceano azul, continue estudando e direcionando o poder dos dados e do aprendizado de máquina para as suas áreas de interesse, todos os lugares onde dados são gerados, são fontes de aplicações de algoritmos de *machine learning*.