

HinA-ia

POR VITOR ALMEIDA E LUCAS MACEDO



Lucas Macedo da Silva

Estudante de Engenharia da Computação

Vitor de Almeida Silva

Estudante de Engenharia da Computação



1^a Edição – 2019 Goiânia

“Dedico este pequeno trabalho a minha família e a meus amigos que tem me apoiado. Mesmo que não haja esperança, vença o seu destino e crie seu próprio caminho!” – Lucas Macedo da Silva

“Dedico este trabalho a Deus a minha Família, amigos e professores, que me incentivaram e apoiaram para a realização deste projeto de trazer aos acadêmicos informações sobre Inteligência Artificial de uma forma inusitada.” – Vitor de Almeida Silva

Sumário

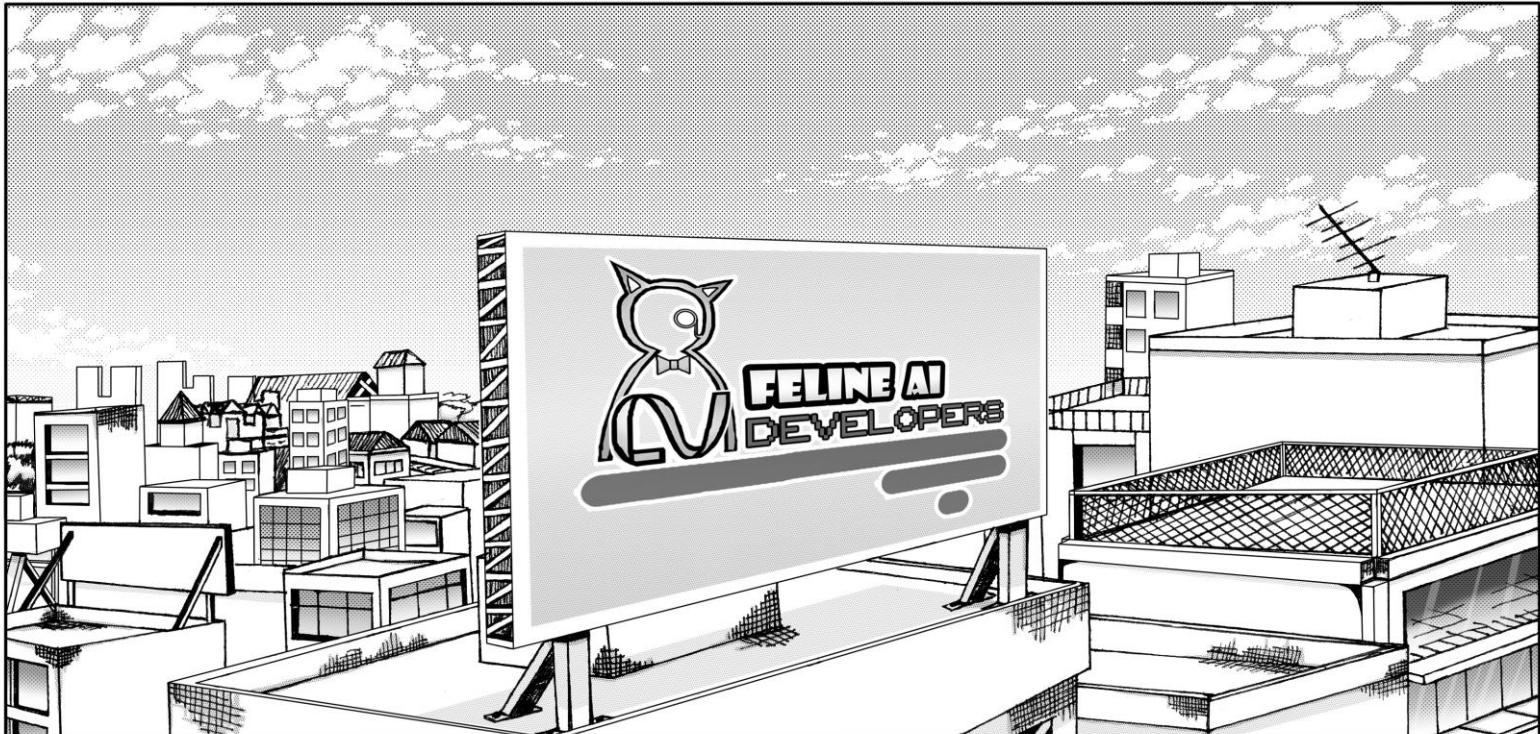
1 INTELIGÊNCIA ARTIFICIAL.....	11
1.1 O que é inteligência artificial?	11
1.2 É possível criar uma IA que dominará o mundo?	11
1.3 O que eu preciso para criar uma inteligência artificial?.....	13
2 MACHINE LEARNING.....	14
2.1 O que é <i>Machine Learning</i> ?	14
2.2 Por que eu tenho que treinar minha IA?	15
2.3 O que eu preciso para aprender <i>machine learning</i>	16
2.4 Meu computador saberá a resposta de tudo depois de treinado?	17
2.5 Como saber se os resultados obtidos são satisfatórios?	17
3 ANALISE DE SENTIMENTOS.....	18
3.1 O que é análise de sentimentos?	18
3.2 O que compõe a análise de sentimentos?.....	18
3.3 Quais métodos utilizar para analisar sentimentos em uma sentença?	19
3.4 Por que aprender sobre análise de sentimentos?	19
4 REDES NEURAIS ARTIFICIAIS	20
4.1 O que é uma rede neural artificial?.....	20
4.2 Quais são os tipos de redes neurais artificiais?	20
4.3 Como funciona uma rede neural artificial?	22
5 PROJETO	23
5.1 Instalando as bibliotecas necessárias	24
5.2 Parte 1 - Análise dos dados	26
5.3 Parte 2 - Escolha do modelo	31
5.4 Parte 3 - Divisão dos dados	32
5.5 Parte 4 - Treino do modelo.....	33
5.6 Parte 5 - Avaliação do modelo	34
39	
Curiosidades	40
Agradecimentos.....	42
Referências bibliográficas.....	43

ATENÇÃO: LEIA DA ESQUERDA PARA A DIREITA

HINA.ia

POR VITOR ALMEIDA E LUCAS MACEDO





AFINAL,
PORQUE PER-
DEU TANTO TEM-
PO FAZENDO
ESSE CORPO
PARA ELA!?

ESTOL-
RAMOS MUITO
O PRAZO



HUNF,
NÃO FALE
ASSIM, QUAL
O PRINCÍPIO
DE UMA
IA?

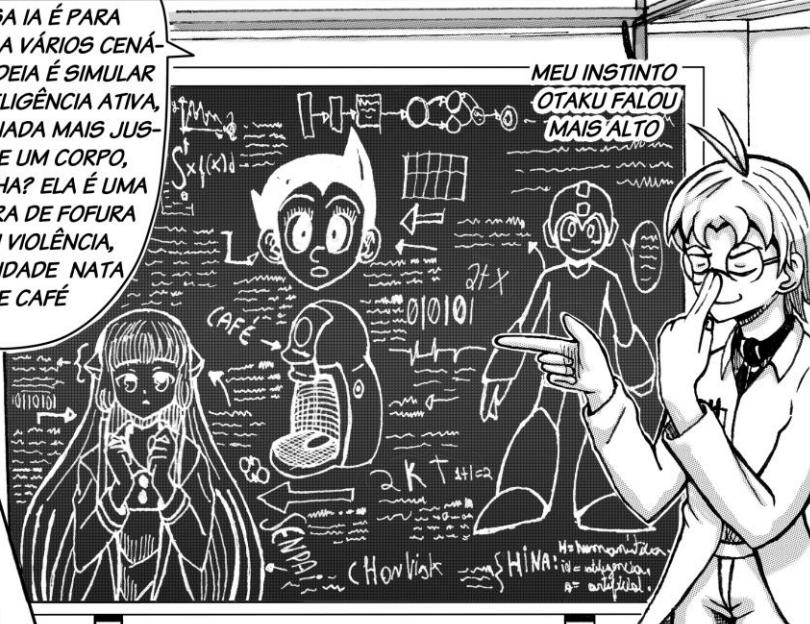
NÃO É SE
ASSEMELHAR A FOR-
MA COMO OS HUMANOS
AGEM EM VISTA DE
ALGUNS PROBLEMAS
DOS MAIS DIVER-
SOS?

BIRIM

BRIM

NOSSA IA É PARA
ATENDER A VÁRIOS CENÁ-
RIOS, A IDEIA É SIMULAR
UMA INTELIGÊNCIA ATIVA,
ENTÃO, NADA MAIS JUS-
TO QUE UM CORPO,
NÃO ACHA? ELA É UMA
MISTURA DE FOFURA
COM VIOLÊNCIA, HABILIDADE NATA
E CAFÉ

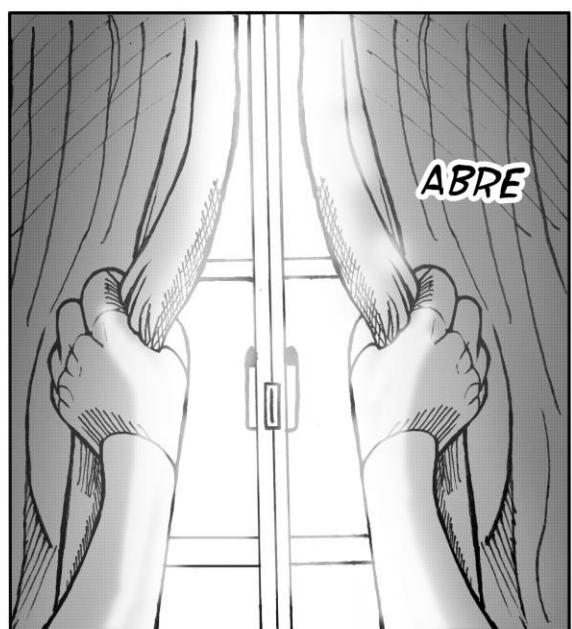
MEU INSTINTO
OTAKU FALOU
MAIS ALTO



VOCÊ REAL-
MENTE GOSTA
DE PERDER
TEMPO, MAS
TANTO FAZ,
JÁ QUE A EM-
PRESA É TÃO
LIBERAL

GOSTEI
DA PARTE
DO CAFÉ ♥



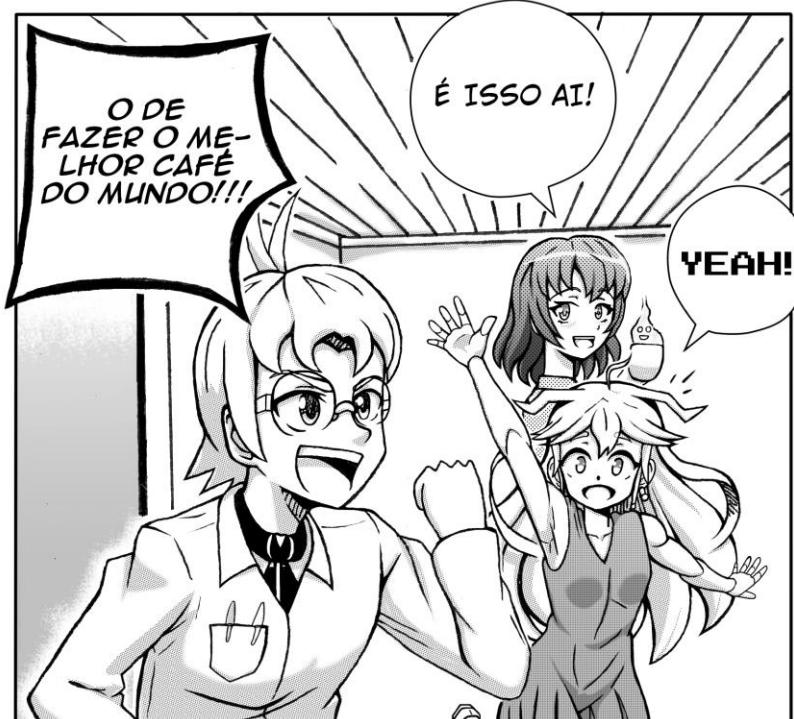
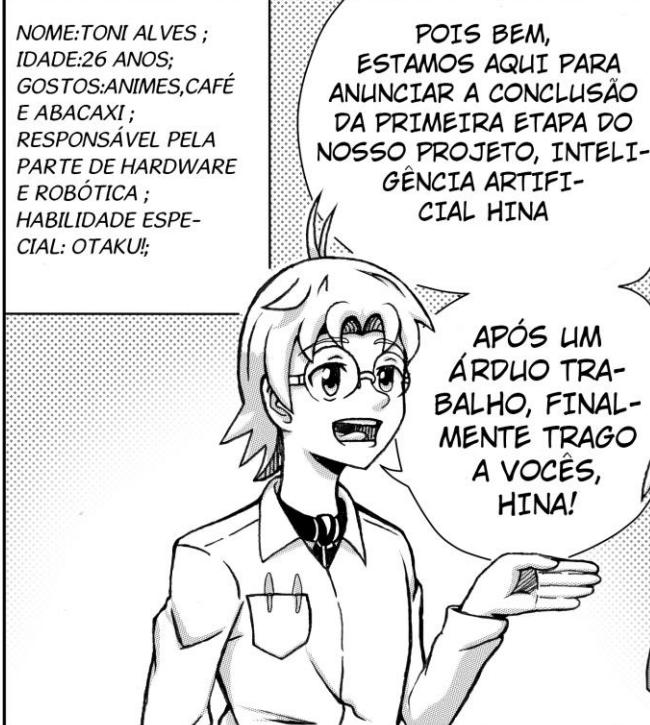
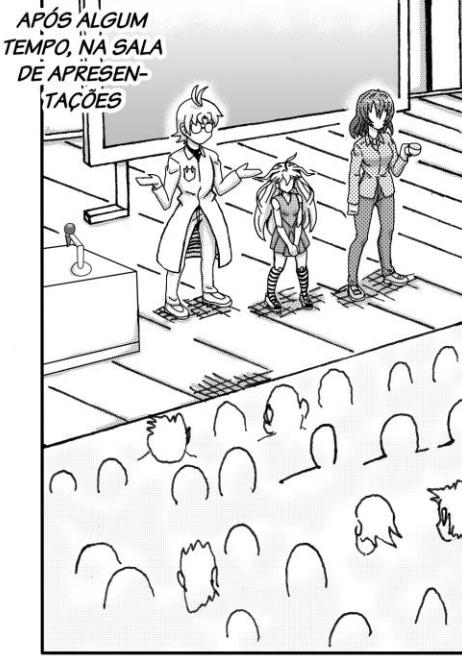


HELLO WORLD!!! ❤



HINA.ia

POR VITOR ALMEIDA E LUCAS MACEDO



1 INTELIGÊNCIA ARTIFICIAL



1.1 O que é inteligência artificial?

A inteligência artificial (IA) é um dos ramos do conhecimento humano, que tem como objetivo a construção de algoritmos, técnicas e métodos que possibilitem as máquinas (Computadores) sejam capazes de se adaptar a diferentes contextos, gerando a ideia de pensamento.

Atualmente, trata-se de uma das áreas com um alto índice de crescimento, já que suas aplicações são as mais diversas, desde reconhecimento facial em aplicativos de *smartphone* e até mesmo, na ajuda ao combate ao câncer. Hoje seu grande potencial pode ser

visto com maior eficiência, dado que com a grande capacidade de armazenamento e processamento dos computadores atuais, o seu desenvolvimento se tornou possível. Dessa forma, somando o grande conjunto de dados disponíveis, a evolução se tornou mais rápida.

1.2 É possível criar uma IA que dominará o mundo?

Pode até ser possível, não se pode duvidar da capacidade humana. Se algo assim ocorrer... os seres humanos deixaram de ser os únicos seres racionais e se verão dominados por



seres superiores que nunca erram. O pensamento de tal possibilidade é aterrorizante e um tanto quanto ficcionista. Porém, cabe a cada cientista e desenvolvedor pensar no bem da humanidade e não em seu próprio.

Mesmo assim, antes de criar um escudo impenetrável o cientista criará a única arma capaz de perfurá-lo. Afinal, se o mundo fosse perfeito não haveria sentido na vida. A ciência trata-se da compreensão da imperfeição e sua tentativa de aperfeiçoá-la. Criar algo perfeito, não significará o fim da busca pela perfeição?

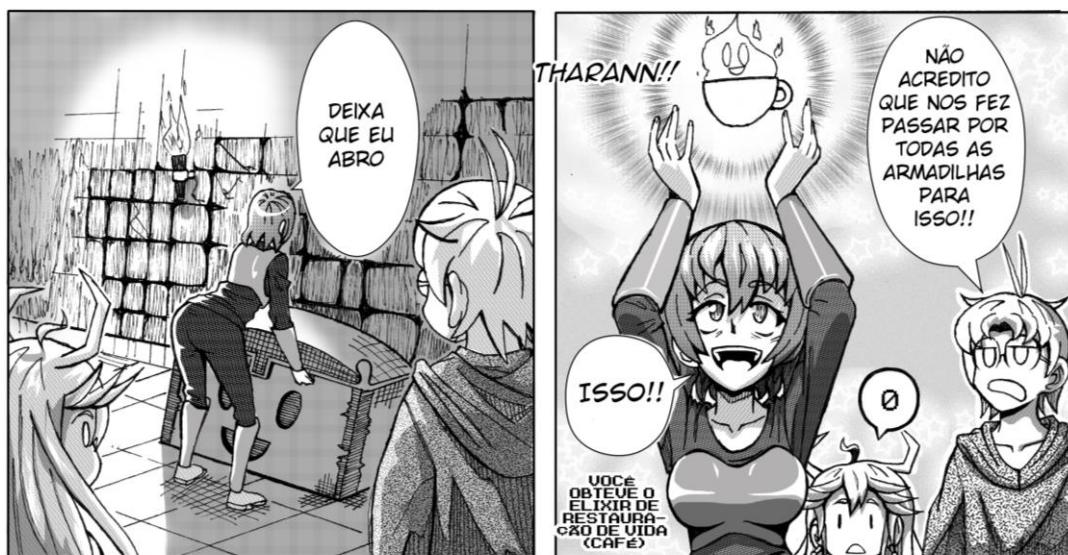
Por que ao invés de utilizar todo esse conhecimento para o mal? Utilize-o para desenvolver uma IA para uma coisa boa, como criar o melhor café do mundo, por exemplo.



1.3 O que eu preciso para criar uma inteligência artificial?

A IA se baseia nos seres humanos e na sua capacidade de adaptação, portanto, para a criação de uma IA, deve-se compreender bem o comportamento humano, estudar *Machine Learning*, saber programação e claro, saber matemática. Os modelos que definem o pensamento humano são em suma modelos matemáticos.

Então não desista aspirante a cientista! Você é capaz, só dedique algumas horas de estudo e poderá criar coisas extraordinárias. Só não se esqueça de acompanhar a jornada da Inteligência Artificial Hina, em sua busca pelo elixir de restauração de vida (também conhecido como café).



2 MACHINE LEARNING



2.1 O que é *Machine Learning*?

A *Machine Learning* (aprendizado de máquina) é uma das vertentes da inteligência artificial, que estuda algoritmos que permitam o computador “pensar” e executar atividades conforme um ser humano.

Ela tem algumas divisões como aprendizagem supervisionada, que ocorre quando a máquina é treinada com exemplos e rótulos. Nesse cenário o ser humano faz o papel de supervisor e verifica se os resultados estão corretos, podendo então modificar algum parâmetro, fazendo assim, com que se obtenha o resultado esperado. Existe também a aprendizagem não supervisionada, que ocorre quando não há intervenção humana no aprendizado.

Além destes, existe o aprendizado semi supervisionado, o intermediário entre o supervisionado e o não supervisionado, que ocorre quando a máquina é treinada com exemplos que possuem respostas, e exemplos que não possuem respostas. Também existe o treinamento por reforço, aqui o computador aprende com erros até se tornar bom naquilo que está fazendo, a Hina é um exemplo deste tipo de inteligência artificial.

A *Deep Learning* (aprendizagem profunda) trata-se de uma derivação da *Machine Learning*, com o intuito de obter um conhecimento mais profundo inerente aos

dados de entrada. Dito de outra forma, a *Deep Learning* seria como imaginar que o computador é capaz de recriar o funcionamento do cérebro humano.

2.2 Por que eu tenho que treinar minha IA?

É importante treinar adequadamente sua IA, já que se o treinamento for realizado de forma incorreta, pode ocorrer situações desastrosas como, por exemplo, a destruição do planeta. O aprendizado está relacionado a assimilação de conceitos, a maioria deles são dados com exemplos a partir de um supervisor, como ocorre nas salas de aula.



Os professores nos passam seu conhecimento a partir de palavras e exemplos, nós então os assimilamos, em analogia com a *Machine Learning* seria o aprendizado supervisionado. Já quando estamos resolvendo exercícios sem o auxílio do professor, apenas com nossa experiência vivenciada em sala de aula, estamos realizando uma aprendizagem não supervisionada. O aprendizado semi supervisionado seria quando estamos resolvendo exercícios e temos apenas respostas de uma parte deles, como ocorre em grande parte dos livros de Cálculo Diferencial e Integral I, que nos dispõe respostas apenas dos exercícios ímpares.

Com isso, se a aprendizagem realizada não for precisa, os resultados obtidos quando a IA for trabalhar sozinha serão errados e dependendo da aplicação podem gerar inconvenientes. Portanto, ao realizar as etapas de treinamento é importante realiza-lo de forma bastante precisa.



2.3 O que eu preciso para aprender *machine learning*?

Os modelos desenvolvidos para simularem o aprendizado são baseados em noções matemáticas, é de vital importância seu conhecimento, entre as matérias destacam-se: probabilidade e estatística, álgebra linear e cálculo diferencial e integral.

Também é necessário saber alguma linguagem de programação, a mais recomendada é o Python, já que sua aprendizagem é mais fácil. Python foi criada para tal propósito, possibilitar o uso de métodos de *Machine Learning*. A linguagem conta com algumas bibliotecas implementadas para facilitar o desenvolvimento de modelos de *Machine Learning* como a *scikit learn*.

Além do mais, é importante saber analisar resultados (ser um bom professor), para treinar corretamente a máquina, e aprender com os erros. Errar é humano, uma máquina não deve errar.



2.4 Meu computador saberá a resposta de tudo depois de treinado?

Tudo que seu computador saberá após a aplicação das técnicas de *Machine Learning*, e do treinamento, serão baseados no conjunto de dados utilizados para treiná-lo e de como foi realizado o treinamento. Quanto maior a quantidade de dados e de treinamentos realizados melhores serão seus resultados obtidos.

Porém, a relação dos dados também é importante, seu computador será capaz de compreender apenas coisas lógicas como, por exemplo, se vai chover hoje, mas não saberá dizer quantos peixes subirão em uma árvore hoje. Treine seu computador com moderação.



2.5 Como saber se os resultados obtidos são satisfatórios?

Existem diversas formas de mensurar os resultados obtidos em um treinamento, como, por exemplo, pode se realizar uma diferença entre o resultado obtido e o esperado, e quanto mais próximo de zero, melhor o computador está conseguindo aprender. O universo das medidas de qualidade de um resultado é imenso, existem além dessa, várias outras métricas como: média, desvio padrão, erro quadrático e etc, etc, etc

Portanto é importante possuir o conhecimento de tais medidas, para não se enganar com os resultados, e achar que o computador já irá predizer os números sorteados na Mega Sena. Se durante suas análises você obter como resposta o número 42, você acaba de encontrar o sentido da vida, ou quase.

3 ANALISE DE SENTIMENTOS

3.1 O que é análise de sentimentos?

A análise de sentimentos é uma técnica que nos permite dizer se uma frase é positiva ou negativa. Não no sentido matemático da coisa, mas sim, no sentido sentimental. Em outras palavras, busca definir se a opinião foi expressa de forma positiva ou negativa.

Imagine que uma empresa quer avaliar seu produto, baseado na opinião de seus usuários, frases como “Esse produto é ótimo!” ou “Atendeu todas minhas expectativas” são exemplos de frases positivas; já frases como “Prometeu muito e não realizou nada” ou “Pior produto que já comprei” são exemplo de frases negativas. A empresa pode então utilizar essas opiniões para avaliar seu produto.

A técnica entra em cena automatizando a análise das opiniões dos clientes, e classificando-as em positiva, quando o produto satisfez o cliente, ou negativa, quando o produto não satisfez o cliente. Com isso a mesma pode verificar se dá continuidade no produto, ou definir algo novo que os clientes querem ver em versões futuras, além disso, podem definir o que pode ser melhorado para abranger um maior número de clientes.

3.2 O que compõe a análise de sentimentos?

Trata-se de uma técnica que utiliza *Machine Learning* para treinar o modelo a ser utilizado e analisar os resultados. Além do mais existem vários métodos para análise das palavras, grande parte deles, baseados em análise probabilística e da frequência em que uma palavra aparece no texto.

Portanto, é composta da aplicação de conceitos de *Machine Learning* e de análises probabilísticas.

3.3 Quais métodos utilizar para analisar sentimentos em uma sentença?



Muita matemática
para pouco café.

Existem vários métodos, que permitem analisar o sentimento de uma sentença, entre eles, destaca-se o Método *Naive Bayes*, sendo um dos mais simples, busca categorizar textos baseando-se na frequência em que determinada palavra aparece no mesmo. Além deste, existem a análise a partir de *emoticons*, que é simples e direta na definição do sentimento que a frase quer transmitir, também fazem parte desse escopo as árvores de decisão, entre outras.

3.4 Por que aprender sobre análise de sentimentos?

As aplicações da análise de sentimentos são diversas, seja no mercado de trabalho avaliando produtos de uma empresa, seja em redes sociais analisando o quanto bem vai a campanha de um candidato ou até mesmo analisando as mensagens do(a) *crush* para verificar os sentimentos nelas presentes.



4 REDES NEURAIS ARTIFICIAIS

4.1 O que é uma rede neural artificial?

Uma rede neural artificial pode ser vista como uma forma de imitar o funcionamento do cérebro humano, o cérebro humano é um dos “computadores” mais potentes do mundo. Um exemplo é o sistema visual, os seres humanos são capazes de processar quando um objeto vem em sua direção e tentar desviar dele em apenas alguns milésimos de segundos. O cérebro humano é incrível!

O cérebro biológico é composto por milhares de neurônios que comunicam entre si e realizam as mais diversas atividades, além de permitirem a aquisição de conhecimento. As redes neurais artificiais são uma imitação de como essa rede de neurônios assimila informações e obtêm respostas para alguma pergunta. Uma dessas perguntas poderia ser, por exemplo, se a partir de uma foto é possível definir se existe ou não um gato nela.

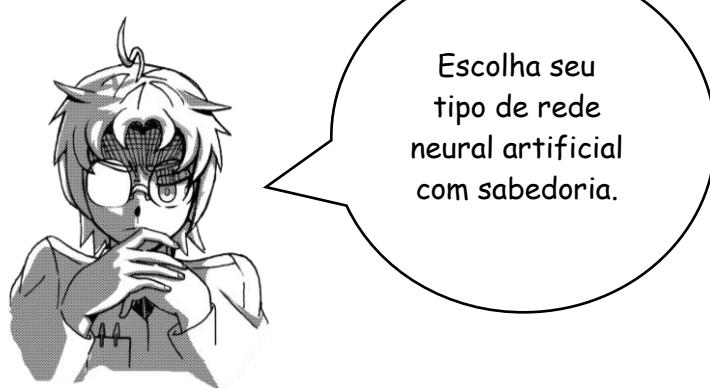
Vale ressaltar que este tópico será explorado de forma superficial, dado que, o mesmo é complexo, e pode vir a exigir conhecimentos mais avançados. Entretanto, ao ler ele você terá uma visão geral do conceito. Então, para dever de lar - já que você pode não morar em uma casa - pesquise bastante sobre redes neurais artificiais, ela é uma área que vem crescendo muito, além de ser promissora. Com isso, depois de ler os próximos parágrafos, não se esqueça de se tornar um ancião em redes neurais artificiais.

4.2 Quais são os tipos de redes neurais artificiais?

Existem muitas arquiteturas de redes neurais artificiais, todas elas com suas peculiaridades e aplicações, existem diversas delas. Além disso, elas também podem ser combinadas permitindo a criação de novos tipos de rede. Somado a isso, é uma das áreas

que está em constante desenvolvimento, pesquisadores do mundo inteiro estão devotos em descobrir novas peculiaridades sobre as mesmas para resolução dos mais diversos problemas.

Porém, em meio a tanta coisa, podemos destacar dois tipos de redes neurais artificiais:



- **Redes recorrentes:** São redes que se auto alimentam, contém loops e consequentemente uma memória que a permite obter experiência e melhorar sua performance. Seu uso implica em sua atualização. Ou seja, por exemplo, a rede recorrente pode ser vista como a mãe que promete comprar na volta, o filho persistente, fica voltando com sua mãe até que a compra seja realizada.

Dentre suas aplicações destacam-se: o processamento de dados sequenciais como o som, e a criação de vídeo games autônomos.

- **Redes convolucionais:** São redes com vasta aplicação no processamento digital de imagens, além de serem bastante complexas, já que trabalham em sua maior parte com imagens. Elas recebem as imagens como entrada, depois separam seus pixels, os distribuem aos neurônios de entrada, adicionam um pouco de mágica, e resultam em alguma porcentagem que representa a probabilidade de aquela imagem ser de determinado objeto.



4.3 Como funciona uma rede neural artificial?

Uma rede neural é um conjunto de neurônios interligados entre si. Um neurônio nada mais é, nada menos é, que só a unidade de processamento fundamental da rede. Os neurônios são interligados a partir de conexões que possuem um peso, existe uma entrada, nessa entrada então é aplicada alguma função matemática e uma função de ativação e dependendo da mesma o neurônio é ativado ou não. Depois de toda essa loucura são fornecidos vários dados de entrada para a rede, o erro é propagado buscando obter os pesos ideias para as conexões.

Mais uma vez a sua construção é totalmente baseada em modelos matemáticos, que buscam simular o funcionamento dos neurônios. Coisa de louco não? Mas não esquenta os neurônios não, um modelo básico para a construção de uma rede neural, pode ser formulado com a seguinte receita:

Pegue um tipo de rede neural adicione muitas amostras de treinamento, as amostras são tipo café, quanto mais melhor. Passe cada amostra pela entrada multiplicando bem pelos pesos, some todo mundo. Pegue a saída da rede compare com a resposta correta e então calcule o erro, propague o erro pela rede e reajuste os pesos, forneça uma nova amostra para a rede com os pesos atualizados. Realize esse processo dezenas, centenas, milhares de vezes se necessário, faça isso até o resultado obtido ser satisfatório. Coloque em prática e está pronta sua rede.



Um bom prato não depende somente dos melhores ingredientes e da melhor receita, depende também de quem está cozinhando, a construção exigirá conhecimento por parte do *chef*, as vezes será necessário dar aquela “mexidinha” para o prato ficar mais apetitoso.

Então agora *chef* mãos a massa, na próxima seção construiremos um pequeno projeto.

5 PROJETO

Sabe que horas são? Hora de colocar a mão na massa, depois de tanta teoria chegou a hora da prática. Pegue uma garrafa de café e prepare-se! Vamos resolver um pequeno problema de classificação, utilizando técnicas de *Machine Learning* um pouco de *Data Science* e por fim, verificaremos nosso modelo.



Antes de tudo, estude um pouco de programação, para que você não fique perdido durante o desenvolvimento. Também faça download de uma *Integrated Development Environment* (IDE), em português, Ambiente Integral de Desenvolvimento de *Python*. Recomenda-se o uso do *jupyter lab/notebook*, mas utilize o que mais gostar. Instale também o interpretador da linguagem *Python*, mais uma recomendação, utilize uma versão 3, já que a versão 2 será descontinuada.

Observação: O código foi desenvolvido na IDE *jupyter lab*.

5.1 Instalando as bibliotecas necessárias

Para instala-las basta colar os comandos no *prompt* a seguir, isso se estiver no Linux, no *Windows* é necessário ir até a pasta em que se encontra o pacote PIP da linguagem *Python*. Se você estiver utilizando o *jupyter lab/notebook*, basta copiar as células a seguir.

As bibliotecas necessárias são:

- Pandas
- Matplotlib

- Seaborn
- Numpy

```
!pip3 install pandas
```

```
!pip3 install matplotlib
```

```
!pip3 install seaborn
```

```
!pip3 install numpy
```

Depois de instaladas vamos importar as bibliotecas.

```
#Importando as bibliotecas
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Observação: Os comandos precedidos de ! ou % só funcionam *no jupyter notebook\lab*

```
%matplotlib inline
# Este comando só funcionará no jupyter
# Se estiver utilizando outra IDE, utilizar o comando comentado a seguir #plt.show()
```

O conjunto de dados utilizados para treinar o computador é o conjunto de dados de íris presente na Biblioteca *Seaborn*. Certifique-se de tê-la instalada corretamente.

O conjunto de dados florais Iris ou o conjunto de dados Iris de Fisher é um conjunto de dados multivariados introduzido por Sir Ronald Fisher em 1936 como um exemplo de análise discriminante.

Nosso objetivo será classificar as írides conforme sua espécie.

5.2 Parte 1 - Análise dos dados

Nesta etapa, vamos analisar nossos dados, buscando definir se o conjunto de dados possuem dados faltantes ou se estão muito distantes uns dos outros. Além de analisarmos os possíveis métodos que podemos aplicar nos dados, para então classificá-los.

```
# Importando o dataset
iris = sns.load_dataset('iris')
```

O conjunto de dados é composto por 50 amostras de cada uma das três espécies de íris (Iris setosa, Iris virginica e Iris versicolor), de modo que temos 150 amostras totais. Foram medidas quatro características de cada amostra:

- ➔ Comprimento das sépalas: representado por `sepal_length`
- ➔ Largura das sépalas: representada por `sepal_width`
- ➔ Comprimento das pétalas: representado por `petal_length`
- ➔ Largura das pétalas: representada por `petal_width`

A espécie é aqui definida como `species` e será o rótulo para a aplicação do treinamento supervisionado.

Os dados estão organizados em um DataFrame. Um DataFrame é uma estrutura de dados em Python, bastante similar a uma planilha eletrônica, foi criada de forma a facilitar a análise de dados. Vamos utilizá-la para armazenar e manipular nossos dados.

Vamos verificar a cara dos nossos dados, para isso digite o comando. Se estiver em uma IDE diferente do *jupyter lab* escreva “print(iris)” e você verá uma tabela igual a que se encontra a seguir.

```
# Verificando a organização dos dados  
iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
10	5.4	3.7	1.5	0.2	setosa
11	4.8	3.4	1.6	0.2	setosa
12	4.8	3.0	1.4	0.1	setosa
13	4.3	3.0	1.1	0.1	setosa
14	5.8	4.0	1.2	0.2	setosa
15	5.7	4.4	1.5	0.4	setosa
16	5.4	3.9	1.3	0.4	setosa
17	5.1	3.5	1.4	0.3	setosa
18	5.7	3.8	1.7	0.3	setosa
19	5.1	3.8	1.5	0.3	setosa
20	5.4	3.4	1.7	0.2	setosa
21	5.1	3.7	1.5	0.4	setosa
22	4.6	3.6	1.0	0.2	setosa
23	5.1	3.3	1.7	0.5	setosa
24	4.8	3.4	1.9	0.2	setosa
25	5.0	3.0	1.6	0.2	setosa
			.	.	.
			.	.	.

Os dados estão organizados em linhas e colunas, bastante similar a uma planilha. Uma parte importante da análise de dados é verificar a integridade dos dados, uma das integridades é determinar se todos os campos estão devidamente preenchidos. Em Python um campo não preenchido contém o valor especial NaN. Valores faltantes são aqueles que não existem ou não foram coletados. Os valores faltantes podem impactar negativamente no nosso projeto. Uma das formas de verificar dados faltantes pode ser feito, conforme a linha a seguir:

```
# A função isnan retorna um vetor booleano contendo verdadeiro (True) se o valor é NA e falso (False) # se o valor é diferente NaN.  
pd.isnull(iris)
```

Obtemos uma tabela onde todas as colunas contêm o valor *False*.

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False

A tabela acima contém os 5 primeiros valores após aplicar a função `isnull()`.

Ok a ideia foi boa, mas gerou muita informação. Vamos filtrar o *DataFrame* buscando verificar se existe o valor *True*. Existem várias formas como por exemplo:

```
pd.isnull(iris).sum()
```

Como saída obtemos:

```
sepal_length      0  
sepal_width      0  
petal_length      0
```

```
petal_width      0  
species         0  
dtype: int64
```

Aqui apenas adicionamos a função `sum()`, que é aplicada em cada coluna do *DataFrame*, obtemos assim como retorno a coluna e o somatório daquela coluna. Nas diversas linguagens de programação, os valores *True* são representados por 1 e os valores *False* são representados por 0, *Python* não é exceção dessa regra. Portanto, se existirem valores faltante o retorno será maior ou igual 1.

Mas e se o valor fosse maior que 1? Nós teríamos um problema e teríamos que utilizar alguma técnica ninja para tratar aquele dado. Como por exemplo: Adicionar a média dos valores anteriores, chutar um valor aleatorio (só faça isso se for bom em chutes) entre outras. Como não houve nenhum resultado maior que 1, não é necessário aplicar nenhuma técnica ninja para tratar esses dados faltantes. Ainda bem! Coincidência? Eu acho que não, isso se chama estratégia.

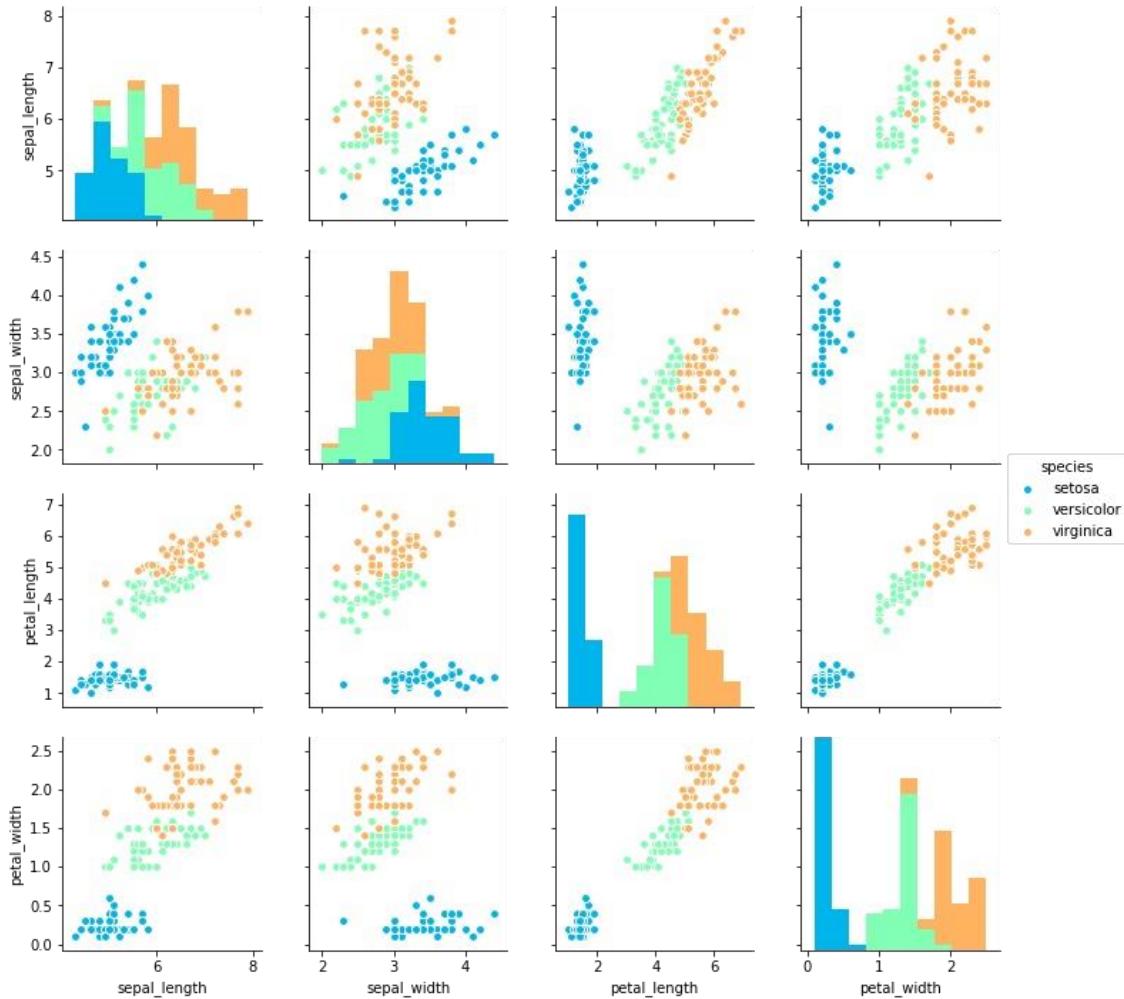
Agora que definimos que nenhum campo do *DataFrame* possui dados faltantes. Poderíamos analisar outras coisas como por exemplo: a dimensão dos dados, ou seja, se não são números muito grandes e estão muito longe uns dos outros. Mas vamos deixar isso para outra hora, na volta a gente vê. Vamos partir para uma análise mais visual dos dados, para isso, “plotaremos” um gráfico do tipo *Pairplot* que você pode encontrar na biblioteca *Seaborn*. Esse gráfico é interessante por que contém todas as combinações das variaveis em uma matriz e nos permite verificar a organização dos dados.

A função recebe como parametros o dataset, irís no caso, hue separa a coloração dos dados conforme alguma coluna do dataset, aqui separamos a partir das espécies, e o atributo palette é opcional ele apenas define a paleta de cores utilizadas.

```
# Plotando todos os gráficos possíveis com as diversas combinações das variáveis # presentes no data set
```

```
sns.pairplot(iris, hue = 'species', palette='rainbow')
```

Gráfico plotado:

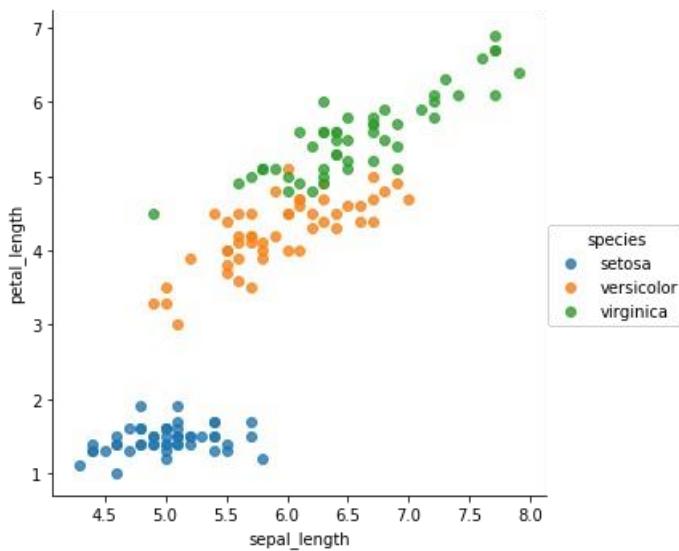


Muito gráfico e muito informação, não acha? Se analisarmos o gráfico é possível ver que os pontos que possuem a mesma cor tendem a ficar juntos. Vamos traçar um gráfico sozinho para exemplificar melhor.

Vamos usar a função lmplot, que traça uma reta de uma regressão linear (O que é uma regressão linear? Decubra nos próximos capítulos), ela recebe como parâmetro os eixos x e y, aqui passamos sepal_length como eixo x e petal_length como eixo y, data é o dataset e hue é a separação.

```
# Traçando um gráfico com as variaveis sepal_length e petal_length  
sns.lmplot(x = 'sepal_length', y = 'petal_length', data = iris,hue = 'species',
```

Gráfico traçado:



Como podemos ver os pontos azul ficaram isolados, os laranjados e verdes ficaram próximos mais é de fácil separação. Agora vamos ensinar o computador a fazer essa separação, para nós humanos é simples realiza-la.

Agora, mãos na massa e vamos montar nosso modelo.

5.3 Parte 2 - Escolha do modelo

Vamos utilizar o classificador *K Nearest Neighbours* (K vizinhos mais próximos), para classificar a qual classe as iris pertencem. Tá mas por quê, escolhemos

esse classificador? Para uma introdução ele é mais simples, vamos explorar vários conceitos discutidos e alguns *spoilers* de alguns conceitos a serem discutidos mais a frente no mangá.

Como sempre começaremos com as importações, para isso importaremos o classificador.

```
# Importando o classificador que reside na biblioteca Scikit Learning
from sklearn.neighbors import KNeighborsClassifier
```

O método se chama K vizinhos mais próximos, por que considera K pontos mais próximos de um dado ponto para realizar a classificação. A distância entre os pontos é calculado com a distância euclidiana entre dois pontos. Onde K é um número inteiro. Louco não? Mas é bem simples.

Agora que importamos o modelo, vamos criar um objeto da classe *KNeighborsClassifier*, e passar *nneighbors* (número de vizinhos) igual a 1, por quê 1? Descubra nas próximas seções.

```
knn = KNeighborsClassifier(n_neighbors=1)
```

5.4 Parte 3 - Divisão dos dados

Agora vamos dividir nosso conjunto de dados em treino e teste. Temos que fazer isso já que se utilizarmos todo o conjunto de dados para treinar nosso modelo ele se adequará tão bem aos dados que não conseguirá dizer nada quando utilizado com outros dados. Analogamente, suponhamos que alguém tenha passado a vida toda jogando futebol em games, porém um belo dia, resolve jogar futebol *in real life*. Como é de se esperar o resultado não será um dos melhores. O princípio de treinar e testar nosso

modelo é importante para que ele seja bom tanto no treino quanto na aplicação.

```
# Vamos então treinar o modelo  
# Antes de tudo, é necessário importar a função que nos ajudará com isso.  
from sklearn.model_selection import train_test_split
```

Depois de importada, separemos os dados. Queremos predizer a espécie em que um irís será utilizada, portanto a especie será o rotulo. Já as demais variaveis serão utilizadas para predizer o rótulo.

```
# Agora separemos nossos dados  
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 101,  
test_size = 0.3)  
# Para isso vamos utilizar a função train_test_split, separar as variaveis X e y,  
onde 30% (test_size=0.3) se reservados para testes  
# random_state = 101, serve para que os seus dados sejam iguais aos nossos  
(pesquisar mais sobre isso)
```

5.5 Parte 4 - Treino do modelo

Finalmente treinaremos nosso modelo. Para isso vamos utilizar a função fit e passar como parametros X_train e y_train.

```
# X = Variáveis preditoras (sepal_length, sepal_width, petal_length, petal_width)  
# y = Rotulo (species)  
X = iris.drop('species', axis = 1)  
# Na linha acima, estamos pegando todas as variáveis menos as especies, a função  
drop, deleta uma coluna de um DataFrame  
# no caso a coluna species, axis = 1, indica que estamos deletando-a no eixo das  
colunas, ou seja, estamos deletando apenas aquela coluna  
y = iris['species']  
  
# Treinando o modelo  
knn.fit(X_train, y_train)
```

Treino bem pequeno não? Isso ocorreu por que estamos utilizando uma biblioteca que nos ajuda todos os métodos já estão implementados, nossa única preocupação é passar parâmetros corretamente.

5.6 Parte 5 - Avaliação do modelo

“Legal, não vi nada de interessante...”, você deve estar com esse pensamento.

Agora começa a parte mais interessante. Certifique-se de ter uma xícara de café bem cheia e grande do seu lado.

O princípio não era predizer a qual espécie uma íris pertence? Vamos colocar isso em prática.

```
# Predizendo os valores do subconjunto de teste  
predito = knn.predict(X_test)
```

A função *predict* nos permite predizer os valores das variáveis a partir dos dados de teste passados como parâmetro.

A saída obtida é:

```
# Se você imprimir o conteúdo de predito verá apenas o nome das especies  
predito
```

```
array(['setosa', 'setosa', 'setosa', 'virginica', 'versicolor', 'virginica', 'versicolor',  
'versicolor', 'virginica', 'setosa', 'virginica', 'setosa', 'setosa', 'virginica', 'virginica',  
'versicolor', 'versicolor', 'versicolor', 'setosa', 'virginica', 'versicolor', 'setosa', 'versicolor',  
'versicolor', 'versicolor', 'versicolor', 'virginica', 'setosa', 'setosa', 'virginica',  
'versicolor', 'virginica', 'versicolor', 'virginica', 'versicolor', 'versicolor', 'versicolor',  
'versicolor', 'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor'], dtype=object)
```

Como avaliar isso? Antes de tudo, vamos a alguns conceitos matemáticos:

- **Matriz de confusão:** Trata-se de uma matriz quadrada de forma que sua diagonal principal indica o número de acertos, os demais elementos são erros para cada classe (as classes aqui são cada espécie de iris). Em Python essa métrica se chama *confusion_matrix* função presente na biblioteca *scikit learning*.

- **Reporte de classificação:** Presente na mesma biblioteca, permite a análise de vários métodos estáticos realizados sobre os dados. Vamos dar maior destaque ao campo *precision* (precisão) utilizaremos ele para mostrar o quanto bom está o modelo. Ele diz a precisão em que a classificação está ocorrendo, variando de 0 (0%) a 1 (100%).

```
# Importando de novo
from sklearn.metrics import classification_report, confusion_matrix

# A confusion_matrix (matriz de confusão), requer dois parametros, o valor de teste, valor real (y_test) # e valor predito (predito)

# A partir deles, ela avalia quantos acertos para aquele classe nosso modelo obteve, valores representados # na diagonal principal e quantos erros nosso modelo obteve, valores representados nos demais campos da

# matriz
print(confusion_matrix(y_test, predito))

# O classification_report (reporte de classificação), é uma junção de todos os métodos estatísticos

# que podem ser aplicados no conjunto de dados para sua avaliação. Recebe os mesmos parametros, os mesmos # definidos na matriz de confusão. Vamos focar no campo "precision" (precisão) para analise do nosso

# classificador.
print (classification_report(y_test, predito))
```

Como saída obtivemos:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	1.00	1.00	20
virginica	1.00	1.00	1.00	12
micro avg	1.00	1.00	1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

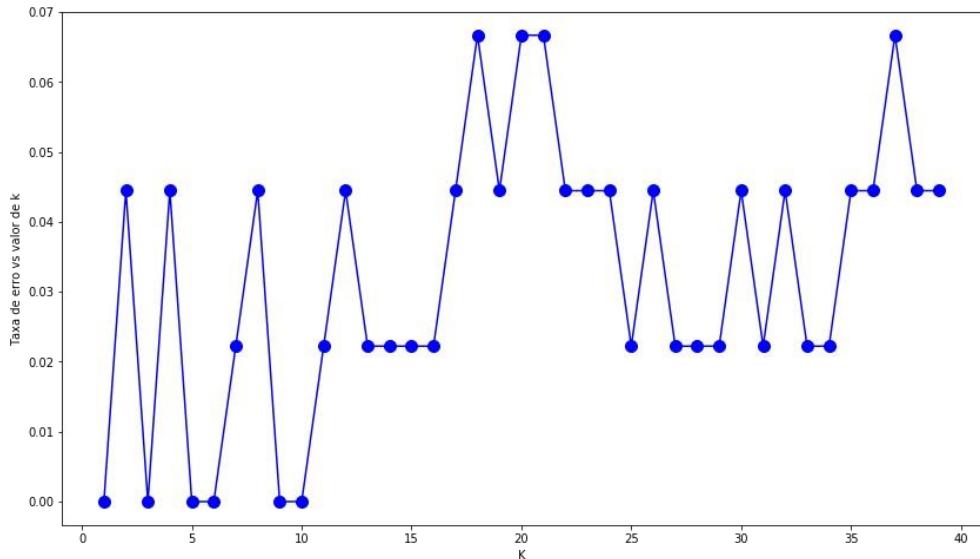
Então quer dizer que nosso modelo consegui prever a qual espécie uma flor de iris pertence com 100% de certeza? Sim isso mesmo

Mas como isso é possível?

Se você perguntasse a um mago, ele provavelmente te responderia que nunca revela seus segredos. Mas o segredo aqui é bem simples lembra-se do parâmetro n_neighbors=1 (número de vizinhos), que utilizamos a pouco? Pois então, ele foi crucial para nosso modelo possuir uma performance tão boa. Para visualizar melhor isso, vamos utilizar as linhas de código abaixo, e aplicar um método que possui o nome um tanto quanto engraçado o método do cotovelo.

```
# Encontrando o melhor k
taxa_erro = []
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    predito = knn.predict(x_test)
    taxa_erro.append(np.mean(predito != y_test))

plt.figure(figsize=(14,8))
plt.plot(range(1,40), taxa_erro, color = 'blue', marker = 'o', markersize=10)
plt.xlabel('K')
plt.ylabel('Taxa de erro vs valor de k')
```



O melhor valor de k é aquele em erro é o menor possível. O algoritmo nos ajuda a encontrar esse k. O gráfico serve como auxílio para facilitar a visualização, idealmente o valor de k ótimo é aquele que intercepta o eixo x, para este exemplo. Olha só k = 1 intercepta o eixo, sendo portanto, um k ótimo. Mágica? Não chama-se estratégia.

Se você utilizar outro valor de k que não intercepte o eixo x, vai verificar que o modelo não obtém um desempenho tão bom. Vamos colocar k = 38 e verificar o que acontece.

Obtemos como saída:

```
knn = KNeighborsClassifier(n_neighbors=38) knn.fit(X_train, y_train)
predito = knn.predict(X_test)
print(confusion_matrix(y_test, predito)) print(classification_report(y_test, predito))
```

[[13 0 0] [0 18 2] [0 0 12]]	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	13
versicolor	1.00	0.90	0.95	20
virginica	0.86	1.00	0.92	12
micro avg	0.96	0.96	0.96	45
macro avg	0.95	0.97	0.96	45
weighted avg	0.96	0.96	0.96	45

O modelo conseguiu predizer bem as duas primeiras espécies porém a espécie virginica o desempenho não foi ótimo. Viu só como é importante e legal conhecer esses conceitos. Legal não? Você também consegue fazer coisas extraordinárias, como programar uma robô para fazer o melhor café do mundo. Devote algumas horas de estudo e tudo dará certo. Até o próximo capítulo.



Hina



Toni

Lucas Macedo

Karen Jones





CONTINUA...

Curiosidades

Nomes dos personagens

A escolha de nome dos personagens da história foi uma etapa cheia de dúvidas e incertezas, tipo quando a gente vai criar nome pros nossos personagens de jogos. Todos os nomes possuem um significado, vamos a algumas curiosidades:

Hina: A pequena e nova criação da dupla dinâmica Toni e Karen. O primeiro nome a ser concebido foi o dela, depois de passar mil e uma noites acordado pensando em um nome que se destacasse na história, que tivesse uma pegada oriental e fosse fácil de memorizar e dizer, nasceu o nome Hina (“Rina”). Formado pelos caracteres iniciais das palavras:

Humanistica

Inteligência

Artificial

Humanistica vem do espirito do projeto, ser algo humano e legal de se fazer, Inteligência Artificial vem da ideia de ela ser uma inteligência artificial, literalmente. Além do mais, Hina significa menininha, já que ela é uma inteligência aritifical que acaba de nascer no mundo. Um nome simples porém com vários significados, assim como esse projeto.

Toni: O *mad scientist* e otaku da história, o nome dele possui todo um significado filosofico, computacional e científico. O nome mais brasileiro de todos os protagonistas, nasceu depois de três meses de debates e tentativas de combinação de letras que fazem alusão à várias referências a inteligência aritifical, o requisito era obter uma combinação

que ficasse de forma fluída e diferente e mais uma vez fosse de fácil pronúncia. Assim como o nome da Hina, foi formado por outras palavras:

Turing

Onisciente

Natural language

Inteligência

Turing vem de Alan Turing o pai da computação, e não é possível falar de inteligência artificial e não citar o Turing. Onisciente, vem de “aquele que sabe de tudo”, os estudos sobre inteligência artificial podem parecer querer criar um ser onisciente, mas como já diz o Paradoxo do onipotente isso é quase impossível. *Natural Language*, vem da área da Inteligência Artificial responsável por processar a linguagem natural de forma que o computador possa entender e, por fim, Inteligência vem da capacidade de pensar.

Karen Jones: A tipica programadora viciada em café, a única que possui um nome composto, pausa dramática, até agora (Ou será que não. Descubra no próximo capítulo). O nome dela ao contrário dos outros é uma homenagem a grande progenitora das buscas em sites, Karen Sparks Jones, graças a ela nossos CRT+C CRT+V, ops, digo, pesquisas acadêmico-cientificas realizadas na internet, ficam mais fáceis, já que se não fosse ela, o método de buscar algo em um site seria mais complicado.

Qual a melhor forma de se criar nomes senão procurando na internet? E se no meio dessas buscas você encontrar a pessoa responsável por tornar essa busca mais simples? É mais que necessário homenagear tal pessoa. Assim fica nossa pequena homenagem a Karen Sparks Jones, uma grande cientista mulher.

Agradecimentos

Muito obrigado a todos que leram! Esperamos que tenham gostado.

Nos envie sua opnião e também se quer receber atualizações sobre o mangá, queremos fazer um ótimo trabalho!

Email para contato: hina.ia70@gmail.com

Ansioso para o próximo capítulo?

Enquanto espera pegue um café e aproveite essas outras obras:

A história de amor de dois estudantes timidos do ensino médio. O que os espera enquanto tentam se aproximar?

História original do autor do Mangá: Vitor de Almeida, Takume x Rioka.

Acesse: <https://www.zinnes.com.br/serie/209>

O herói brasileiro que luta por uma internet melhor. Será que o capitão conseguirá salvar a sociedade brasileira das empresas de internet?

Historial original do apoiador do mangá: Alexandre Matos, Capitão Wi-Fi.

Acesse: <https://www.zinnes.com.br/serie/36>

Referências bibliográficas

ACADEMY, Data Science. Capítulo 2 – **Uma Breve História das Redes Neurais Artificiais**. Disponível em: <<http://deeplearningbook.com.br/uma-breve-historia-das-redes-neurais-artificiais/>>. Acesso em: 28 dez. 2018.

BARROS, Pedro. **Aprendizagem de Maquina: Supervisionada ou Não Supervisionada?**. Disponível em: <<https://medium.com/openanca/aprendizagem-de-maquina-supervisionada-ou-n%C3%A3o-supervisionada-7d01f78cd80a>>. Acesso em: 07 abr. 2016.

BISHOP, Christopher M.. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

BOURCHARDT, Eliezer. **Inteligência Artificial—Um pouco da história e avanços atuais**. Disponível em: <<https://medium.com/@eliezerfb/intelig%C3%A1ncia-artificial-499fc2c4aa79>>. Acesso em: 09 jan. 2018.

BROWNLEE, Jason. **Overfitting and Underfitting With Machine Learning Algorithms**. Disponível em: <<https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/>>. Acesso em: 21 mar. 2016.

FARINACCIO, Rafael. **Inteligência artificial é capaz de reconhecer até rostos disfarçados**. Disponível em: <<https://www.tecmundo.com.br/ciencia/121836-inteligencia-artificial-capaz-reconhecer-rostos-disfarados.htm>>. Acesso em: 20 nov. 2018.

FURQUIM, Cristiano. **10 Algoritmos de Aprendizagem de Máquinas (Machine Learning) que você precisa saber.** Disponível em: <<https://medium.com/@cristianofurquim/10-algoritmos-de-aprendizagem-de-m%C3%A1quinas-machine-learning-que-voc%C3%AA-precisa-saber-c49f9eef319>>. Acesso em: 05 out. 2018.

GANDHI, Rohith. **K Nearest Neighbours—Introduction to Machine Learning Algorithms.** Disponível em: <<https://towardsdatascience.com/k-nearest-neighbours-introduction-to-machine-learning-algorithms-18e7ce3d802a>>. Acesso em: 15 nov. 2018.

GHOSH, Pallab. **Inteligência artificial pode levar ao diagnóstico precoce de doenças cardíacas e câncer de pulmão.** Disponível em: <<https://www.bbc.com/portuguese/geral-42537252>>. Acesso em: 3 jan. 2019.

HELSINKI, University Of. **Elements of AI.** Disponível em: <<https://course.elementsofai.com/>>. Acesso em: 29 dez. 2018.

HONDA, Hugo; FACURE, Matheus; YAOHAO, Peng. **Os Três Tipos de Aprendizado de Máquina.** Disponível em: <<https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/>>. Acesso em: 11 nov. 2018.

JAMES, Gareth et al. **An Introduction to Statistical Learning with Applications in R.** New York: Springer, 2013.

LIMA, Edirlei Soares de. **Aprendizado Não-Supervisionado.** Rio de Janeiro: Visual, 2011. Color. Disponível em:

<http://edirlei.3dgb.com.br/aulas/ia_2011_2/IA_Aula_18_Aprendizado_Nao_Supervisionado.pdf>. Acesso em: 09 dez. 2018.

MENEZES, Paulo. **Aprendizagem Supervisionada e Aprendizagem não Supervisionada.** Disponível em: <<https://home.isr.uc.pt/~paulo/PROJ/NN95/node31.html>>. Acesso em: 25 out. 2018.

MICROSOFT. **Matriz de classificação (Analysis Services - Mineração de dados).** Disponível em: <<https://docs.microsoft.com/pt-br/sql/analysis-services/data-mining/classification-matrix-analysis-services-data-mining?view=sql-server-2017>>. Acesso em: 30 abr. 2018.

PINHEIRO, Nilcélia Aparecida Maciel. **Uma reflexão sobre a importância do conhecimento matemático para a ciência, para a tecnologia e para a sociedade.** Letras e Arte, Ponta Grossa, v. 11, n. 22, p.22-31, abr. 2003.

RUSSEL, Stuart; NOVING, Peter. **Inteligência Artificial.** Rio de Janeiro: Campus, 2013.

SHIMAKURA, Silvia Emiko. **Tipos de variáveis.** Disponível em: <<http://leg.ufpr.br/~silvia/CE055/node8.html>>. Acesso em: 20 set. 2012.

SILVA, Josenildo Costa da. **Aprendendo em uma Floresta Aleatória.** Disponível em: <<https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%ADria-3545f6babdf8>>. Acesso em: 26 nov. 2018.