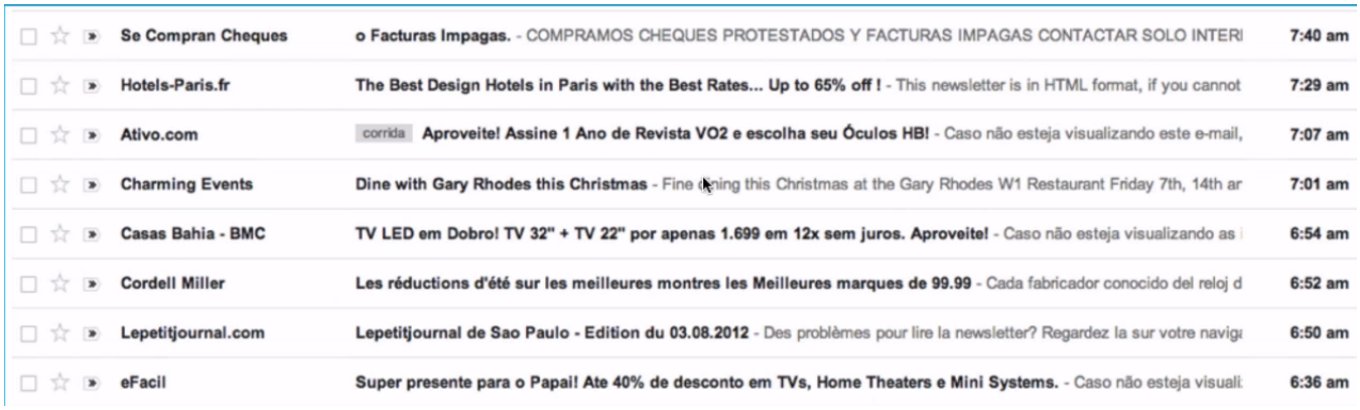
08

Classificando emails, animais e muito mais

Classificando e-mails, animais e muito mais

Nesse capítulo iremos falar de diversos problemas que nós podemos resolver com **machine learning**, com inteligência artificial, aprendizagem de máquina... Todas essas palavras bonitas que estão de alguma maneira ligadas com soluções para resolver problemas um pouco mais difíceis no nosso dia-a-dia. Vamos dar uma olhada que tipo de problema a gente pode resolver? Principalmente os primeiros que resolveremos logo no primeiro capítulo do curso. Veremos então, 3 problemas que podemos resolver logo no primeiro capítulo.

Recebi e-mails hoje de manhã, recebi vários e-mails, não sei dizer quantos e-mails você recebe, eu recebo diversos, você provavelmente também recebe diversos e nem percebe, por que você não percebe? Repare todos esses e-mails que eu recebi:



Tem e-mail que parece que não fui eu que pedi para receber, por exemplo, **Se Compran Cheques?** E-mail em espanhol? Sendo que eu nem sei falar em espanhol. **The Best Design Hotels in Paris?** E-mail francês? Realmente, estou aprendendo e por isso assinei um lugar e outro, então tudo bem... Mas e **Casas Bahia?** Não assinei e-mails da Casas Bahia, ela mandou sem eu pedir... Então repara que tem um monte de e-mail aqui que parece ser *spam*. A quantidade de *spam* que vai de um lado para o outro na internet hoje em dia é absurda e, de alguma maneira, alguém precisa classificar esses e-mails e dizer:

-Olha, isso aqui é *spam*.

-Esse e-mail aqui é *spam*!

Tem que ter alguém na internet fazendo isso, porque se esse alguém for eu, e eu tiver que olhar os 1000 e-mails que eu recebo pra dizer que desses 1000 e-mails, 990 são *spam* a minha vida acaba! E eu não quero que a minha vida acabe e você também não! A gente quer viver alegremente com o tempo para outras coisas e não passar o dia inteiro com a nossa vida só lá: "É *spam*. Não é *spam*. É *spam*. Não é *spam*...". Queremos que alguém faça por nós: "Por favor, diga pra mim se isso aí é *spam* ou se não é *spam*". E esse alguém pode ser uma pessoa! Eu posso contratar uma pessoa pra ficar falando: "Olha, isso aqui é *spam*", ou não: "Esse aqui não é *spam*". E nós pagamos essa pessoa e ela fica fazendo isso o dia inteiro... Parece trabalhoso e complicado, pode até ter soluções desse gênero para fazer uma classificação, faz sentido...

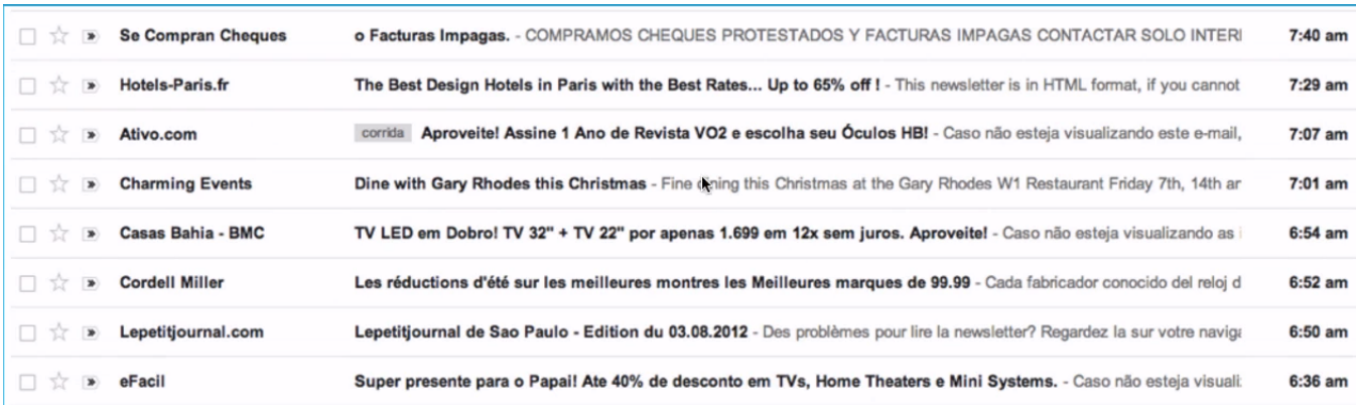
Uma outra maneira é eu falar para um programa fazer isso pra mim, falar: "O programa, por favor, classifica esse meu e-mail, se ele é *spam* e se ele não é *spam*". Mas um programa, um computador, ele sabe o que é um *spam*? A gente sabe o que é *spam*, se eu falar pra você assim:

-Olha, recebi um *spam* hoje

-Puts, o cara recebeu um *spam*...

Nós sabemos o que significa *spam*, aquele e-mail indesejado que eu não pedi pra receber e alguém me enviou... Esse é o significado da palavra *spam*. Nós sabemos, mas o computador não faz a mínima ideia o que significa a palavra *spam*. Então o que a gente precisa? Precisamos, de alguma maneira, dizer para o computador o que é *spam* e o que não é *spam*. Mas nem a palavra *spam* ele conhece, então como podemos fazer isso? Como é mesmo que o computador conhece? O computador entende apenas 0 ou 1, 0, 1, 2, 3, 4, 5... Binários, numérico etc... Lembra aquele curso onde aprendemos programação? O básico, onde vimos sobre binário. Se você não viu tem o curso de introdução a programação com Ruby que a gente fala do binário...

Então você viu lá a questão que ele só entende 0, 1, 2, 3, 4 e etc... Se ele só entende esses números o que eu vou falar para o computador é que ele me diga que se for *spam* é 1, me diz 1 se ele for *spam*! Se ele não for *spam* me diz 0, pronto! Já resolvi o problema da palavra *spam* a palavra *spam* já não existe mais para o computador, eu consigo falar para o computador assim: "E aí, é 0 ou é 1?". Já é bem melhor, certo? Já simplifiquei para o computador me entender, pois se eu explicar para o computador o que é *spam* eu estarei maluco, mas se explicar para o computador o que é 0 ou 1 ele já sabe! Só preciso falar pra ele: "Olha, esse e-mail 'Facturas Impagas' ele é 1 ou é 0?". Isso é, eu preciso agora passar por cada um desses e-mails, olhar e falar: "Todas "impagas" (*spam*) é 1", ou seja, eu preciso ensinar o computador, treinar ele! Como é que vocês sabem se esse e-mail é *spam* ou não? Porque você foi treinado! Você já viu um milhão de e-mails que eram *spam* e que não eram *spam*, quando você já bate o olho você fala: "É *spam*", você bate o olho mais uma vez e fala: "Não é *spam*". Mas como você bate o olho e fala se é *spam* ou e não é *spam*? Você foi treinado pra isso, você já viu tanto, você já fez tanto essa classificação: "*spam*", "Não *spam*","*spam*", "Não *spam*","*spam*", "Não *spam*" que hoje você faz isso facilmente. Então a mesma coisa aqui:



1º e-mail, é *spam* ou não é *spam*? É *spam*!

2º e-mail, hotel em Paris? Realmente eu estava procurando hotel em Paris, então não é *spam*.

3º e-mail, assinatura da revista não é *spam*, fui eu que realmente assinei a revista de corrida.

4º e-mail, jantar com Gary Rhodes... Quem é Gary Rhodes? Eu não tenho a menor ideia, é *spam*! Então 1.

5º e-email, TV LED em Dobro! Não, não assinei Casas Bahia, então 1, é *spam*.

6º e 7º e-mails, são 2 e-mails que eu assino então 0, não é *spam*.

8º e-mail, Super presente para o Papai! É *spam*.

Enfim, existe um monte de e-mail aqui que é *spam* e um monte que não é. Eu batendo o olho já sei dizer se é *spam* ou não e a mesma coisa eu consigo treinar o computador, treinar um programa, isso mesmo, treino um programa, para dizer se é *spam* 1, se não é *spam* 0. Mas como que eu treino? Eu dou um conjunto de e-mails que não é *spam* e um conjunto de e-mails que é

spam e falo pra ele: "Ta aí, aprenda!". Aí ele aprende da mesma forma que aprendemos, você aprendeu a dizer se é *spam* ou não, eu também aprendi, ele também vai aprender a dizer, e o que vai acontecer? Quando chegar um e-mail novo o que ele faz? Ele olha esse e-mail e fala: "É *spam*.". Aí ele olha outro e-mail e fala: "Não é spam.". E é assim que funciona um programa de classificação, ele classifica entre 0 ou 1. Ele responde qualquer pergunta do gênero que vai classificar alguma coisa entre 0 e 1, o que quer dizer 0? O que quer dizer 1? No meu caso quer dizer se é *spam* ou não, poderia querer dizer outra coisa? Com certeza! Pordemos usar a classificação entre 0 e 1 pra diversas coisas e como eu falei, só no primeiro capítulo desse curso veremos a classificação de 0 e 1 e, também, como poderemos utilizar isso em diversas coisas, primeiro foi esse caso do e-mail, classificar entre *spam* ou não, é classificar entre 1 ou 0, reduzimos o problema para classificar entre 1 ou 0 e ensinamos o programa a classificar entre 1 ou 0.

Classificamos elementos entre *spam* e não *spam*, entre 1 ou 0, independente do que signifique 1 ou 0. Vamos ver outros exemplos do dia-a-dia onde usamos a classificação? Vou dar um exemplo do dia-a-dia humano para que possamos entender e visualizar o nosso cotidiano, ou seja, como nós, seres humanos, fazemos a classificação. Como é que eu vou poder implementar um algoritmo se eu não entendo nem como eu faço uma coisa, "Ah! Quero implementar um algoritmo de soma". Ok, como fazemos a soma? "Quero implementar um algoritmo de multiplicação". Certo, como eu faço a multiplicação? Então eu quero implementar um algoritmo que diz se é *spam* ou não é *spam* se um cliente vai comprar ou se não vai comprar, se um funcionário vai se demitir ou se não vai se demitir, eu preciso saber como eu faço isso! Isto é, como eu, Guilherme, classifico isso. Então vamos pegar um exemplo do dia-a-dia, um exemplo em que eu estou andando na rua, olho para alguma coisa, vejo e classifico, e então implementamos um algoritmo similar a essa classificação. Na prática, é isso mesmo que fazemos. Vamos lá?

Queremos classificar entre 1 ou 0. No meu dia-a-dia, quando eu era criança ia na fazenda de um tio meu no interior de Minas e ele tinha diversos animais e às vezes eu encontrava um animal como este:

Que animal é esse? Bom, você acabou de classificar, eu classifico esse animal como um porco, tudo bem? Por que eu classifico ele como um porco? Olha a perninha dele, é perninha de porco, perninha curtinha... O que mais? Ele é meio gordinho, porquinho é gordinho... o que mais? Ele faz "oinc, oinc", então, pra mim, ele é um porquinho. Por essas 3 características eu classifico ele como um porco, eu dou uma olhada nessas 3, vejo se ele se encaixa e aí falo: "Sim, Ele é um porco!". Mas também numa fazenda, como é de costume, tem diversos outros animais, e às vezes eu encontrava um animal como este:

Então eu me pergunto, no momento que estou andando, eu olho para ele e falo: "Nossa que porquinho bonito e fofinho!"? Não! Quando eu olho pra ele eu falo: "Nossa, que cachorrinho bonitinho". Por que eu falava "que cachorrinho bonitinho"? Porque na minha cabeça ocorreu o processo de classificação, quando eu olhava os porquinhos eu falava: "Nossa que porquinho bonitinho". Quando eu olhada os cachorrinhos eu falava: "Nossa, que cachorrinho bonitinho". Por quê? Pois quando eu olhava o cachorrinho, eu pensava:

> Será que ele tem perninha curtinha que nem o porquinho?. **Não.**
>
> Será que ele é gordinho que nem o porquinho? **Alguns sim e outros não.**
>
> Ele faz *oinc* que nem o porquinho? **Eles não faziam *oinc*.**
>
> Então ele não é um porquinho, ele é um cachorrinho.

Então repare que, de acordo com essas 3 características que eu citei pra vocês, se é com perninha curtinha, se ele faz *oinc* ou se ele é gordinho eu classifico entre porquinho e cachorrinho. É como eu faço, e é óbvio que cada um faz de uma maneira um pouco diferente, usam algumas características um pouco diferentes umas das outras... e você tem classificações diferentes, natural, isso acontece. Eu vou seguir essa classificação minha como exemplo, ou seja, a classificação que eu faço na minha cabeça, como ser humano, e então, vou querer ensinar o computador a fazer essa classificação. Vamos lá!

Eu tenho aqui fotos de vários animais:

E eu quero saber se eles têm perna curta ou não, se eles são gordinhos ou não, se eles fazem *oinc* ou não e eu quero dizer se eles são porquinhos ou se eles são cachorrinhos, eu vou usar 1 para dizer se é um porquinho e vou usar 0 para dizer se é cachorrinho. Então vamos preencher essa tabela?

Vejamos esse primeiro animal:

Ele tem perna curta? Sim, então 1. Ele era gordinho e fazia *oinc,* estou falando do porquinho lá da fazenda, então ele é um porquinho. Então essa linha fica [1,1,1,1]. Vejamos o próximo animal:

Esse outro porquinho também tem perna curta, mas ele não era gordinho, ele era um porquinho mais magrinho, então eu vou colocar 0 para dizer que ele era mais magrinho. Ele fazia *oinc*? Sim, ele fazia *oinc* e ele era um porquinho então 1. Nossa linha fica [1,0,1,1]. Vamos verificar o próximo:

Esse outro porquinho tinha a perna curta, verdade, ele era gordinho, só que tem um porém, ele nasceu com um negócio na gartanta que ele não fazia *oinc* e nem fazia barulho e ficava de boa, então vou marcar com 0. Porém, quando eu olhava pra ele eu falava: "Você é um porquinho.". Então vamos marcar como porquinho, tá? A linha fica [1,1,0,1]. Vamos dar uma olhada como está a nossa tabela atualmente:

animal	perna curta?	gordinho?	faz oinc?	o que é?
	1	1	1	1
	1	0	1	1
	1	1	0	1

Perceba que, para cada elemento da tabela, estou pegando as características deles, ou seja, se tem perna curta, se é gordinho ou se faz *oinc*, dizendo apenas se sim ou se não usando apenas 1 e 0 para sim (1) ou não (0) e dizendo no final qual é o tipo de animal, se é porco (1) se é cachorro (0). Ainda faltam mais 3 animais para analisarmos, vamos lá!

Esse aqui tem perna curta? Esse cachorro não tinha perna curta, então 0. Ele também não era gordinho e ele também não fazia *oinc* e ele era, adivinha? Um cachorrinho, sim, porque eu olhava pra ele e era um cachorro. Então a nossa linha fica [0,0,0,0]. Agora o próximo animal:

Já esse outro era divertido, ele era um *Yorkshire* pequenininho e tinha a perna curta, só que também ele era gordinho e o mais engraçado é que quando ele ia fazer *au au*, ou seja, latir, ele fazia *oinc oinc*! Então eu vou marcar que ele fazia *oinc oinc*, tudo bem? Só que, repara, ele era um cachorro. Apesar dele ter as 3 caracteristicas parecidas com o de um porco, eu olhava pra ele e falava: "Opa! Esse aqui é um cachorro". Então a essa linha fica com esses valores [1,1,1,0]. Vamos para o último:

Esse daqui não tinha perna curta, também não era gordinho e também não fazia *oinc oinc*. Ele era um cachorrinho mesmo, eu sei dizer isso. Então marcamos assim [0,0,0,0]

O que eu estou fazendo? Eu estou pegando todos os dados dos meus elementos, isto é, desses meus animais e estou montando uma tabela, veja como ela ficou:

animal	perna curta?	gordinho?	faz oinc?	o que é?
	1	1	1	1
	1	0	1	1
	1	1	0	1
	0	0	0	0
	1	1	1	0
	0	0	0	0

Essa tabela define as características dos animais por meio dessas 3 primeiras colunas (perna curta, gordinho, faz oinc) e essa última coluna (o que é) classifica o animal, ele é cachorro ou ele é porco? Então eu estou usando as caracteristicas para classificar um animal entre dois tipos. Eu poderia usar essas caracteristicas ou outras para classificar um e-mail entre *spam* e não *spam*, por exemplo:

- tamanho do e-mail
- as palavras que aparecem
- o horário que foi enviado
- se eu conheço ou não quem enviou
- se é a primeira vez que esse e-mail vem

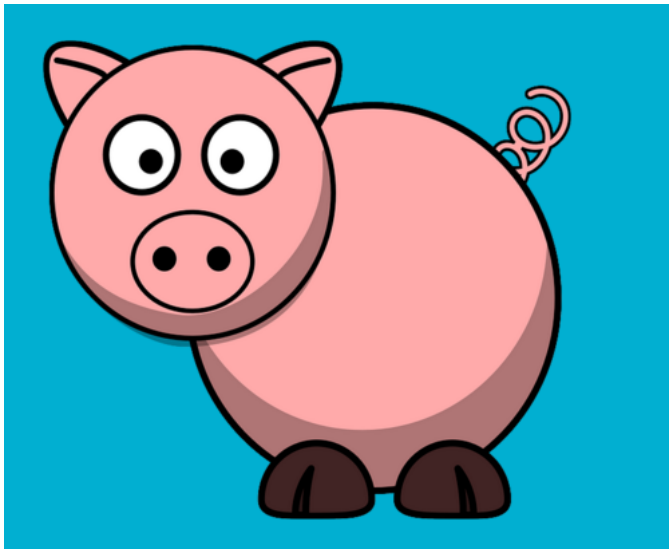
Essas são algumas caracteristicas que eu posso usar para classificar algo.

No meu dia-a-dia, lá na fazenda do meu tio, eu usava essas 3 características para classificar os animais, porém, eu poderia usar outras também:

- "Ele é rosa?"
- "Ele é preto?"
- "Ele é branco?"

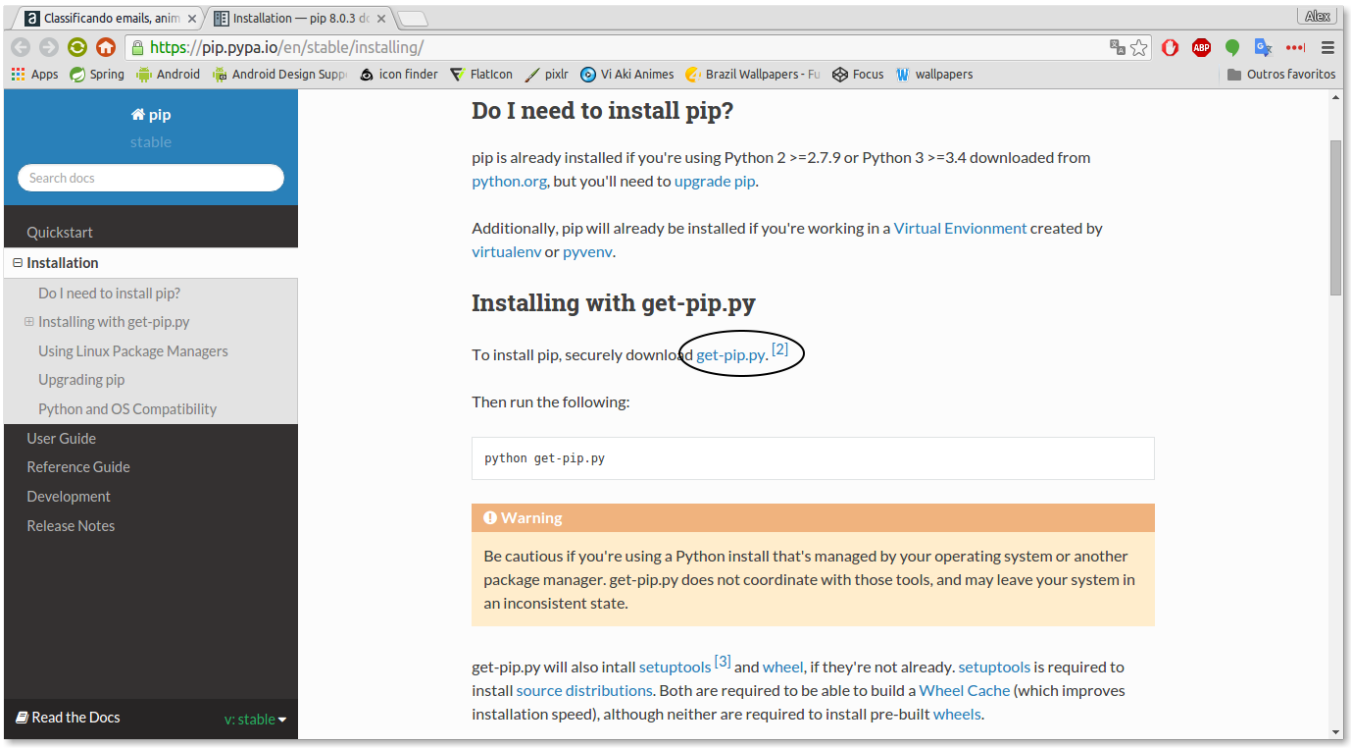
- "Ele é azul?"
- "Ele é x?"

A cor influencia para eu dizer se ele é um animal ou outro, pois tem animais de uma cor e animais de outra cor. Existem diversas características que a gente pode usar para classificar um animal. Aqui eu usei 3 características pra classificar os meus animais. Então repara que todos os problemas de classificação vão fazer isso: dado um conjunto de características e um monte de dados para eu treinar esse meu algoritmo, esse meu programa. Essa tabela de características é o meu treino, então eu falo para o meu programa: "Olha, essa tabela eu já sei, toma pra você". E também, mando a classificação: "Toma aqui, também, a classificação". E o programa fica todo feliz que ele foi treinado. Agora que o programa está treinado por essas imagens ele está feliz, o que acontece? Eu pergunto para ele, assim como eu posso perguntar para mim mesmo: "E esse aqui? Quem é ele?":



Ele é um cachorro (0) ou ele é um porco (1)? E aí, eu ou programa tomamos uma decisão baseados em toda experiência anterior que tivemos, baseados em toda experiência anterior que o programa teve, todo treino que já foi feito, ele vai olhar as caracteristicas desse animal e vai tomar a decisão se ele é um porco ou se ele é um cachorro isto é, esse animal especificamente, tem perna curta? sim (1), esse animal é gordinho? Sim (1), ele faz *oinc*? ele não faz *oinc* (0), pois isso aqui é uma imagem, ele não é um animal de verdade... Então ele não fazia *oinc*, tudo bem? Então a pergunta se resume a quê? Dadas essas caracteristicas, por favor, Guilherme, por favor, você, por favor programa, me diga se o resultado é 0 ou 1, isso significa que, eu treinei você com diversas características e classificações e, se eu te der uma linha de característica, eu quero que você me diga se ele é 0 ou 1, ou seja, eu quero que você preveja, classifique pra mim! Então dada essas caracteristicas me diga se ele é um cachorro ou um pouco. E é assim que vai funcionar os programas de classificação: Nós treinamos por meio de caracteristicas e classificações e treina. Após treinar, pedimos para ele classificar coisas que nós desconhecemos, como por exemplo, chegou um e-mail novo: "E aí, esse e-mail é um *spam* ou não é?, chegou um animal novo: "E aí, esse animal eu vou vender como porco ou como cachorro?", ou: "Vou fazer carinho como porco ou como cachorro? O que ele é?". Resumindo, os algoritmos serão baseados em caracteristicas e iremos usar essas caracteristicas entre 0 e 1, é claro que veremos outros tipos depois, para classificar se ele é ou não é um animal que esperamos, é exatamente isso que faremos com os algoritmos.

Vamos para o código. O que queremos aqui é implementar um sistema de classificação que consiga classificar alguma coisa entre 0 e 1, -1 e 1, 1 e 2, A e B, entre duas categorias diferentes! Eu tenho duas categorias, se é porco ou se é cachorro, eu quero classificá-las. Pra isso usaremos *Python* e instalaremos nele o *pip*, que é o instalador simples de pacotes. Vá até a página de instalação (https://pip.pypa.io/en/stable/installing/) do pip e clique em `get-pip.py` :



Ele vai abrir o arquivo na página atual, salve em um diretório de fácil acesso via terminal. A seguir, no terminal, entre no diretório que você salvou o arquivo e digite o comando:

```
> python get-pip.py
```

Caso você esteja no linux será necessário utilizar o `sudo` para permissão de super usuário.

Pronto! O pip foi instalado. Na própria página de instalação existem outras alternativas de instalação, como por exemplo, no Ubuntu, podemos utilizar um package manager como o `apt-get` . Utilizamos o método demonstrado, pois funciona em qualquer sistema operacional!

Agora que nós temos o pip, podemos instalar as bibliotecas do python de nosso interesse para trabalhar com as informações científicas que queremos, como por exemplo, o machine learning e a classificação. A biblioteca que usaremos é o `scikit-learn` . Então vamos pedir para o pip instalar para nós por meio do comando:

```
> pip install scikit-learn
```

Da mesma forma que aconteceu na instalação do *pip*, caso esteja no Linux, será necessário utilizar o `sudo` .

Bacana, instalei a biblioteca do `python` , o `scikit-learn` . Além da biblioteca, nós temos acesso à documentação (http://scikit-learn.org/stable/documentation.html) para consultarmos tudo o que ela disponibiliza para nós. O que queremos fazer, a princípio, é a classificação entre diversos porcos e cachorros, depois iremos fazer do nosso mundo web. Então, abriremos um editor de texto para escrevermos o nosso código python. Utilize um de sua preferência.

Primeiro vamos criar um arquivo chamado `classificao.py` e vamos salvá-lo, lembre-se de salvar o arquivo dentro de um diretório de fácil acesso ao terminal. E nesse arquivo iremos definir todos os porcos e cachorros que já conhecemos, por exemplo, nós queremos representar o nosso primeiro porquinho, então escrevemos:

```
porco1
```

E o que era esse porquinho? Ele era um *array* de 3 valores, você lembra? Quais são os valores? 0 ou 1, ou seja, se ele tem ou não tem uma característica. Mas, e quais são as 3 características que eu, Guilherme, decidi usar agora?

1 - Se ele é gordinho. 2 - Se ele tem perninha curta. 3 - Se ele faz *auau*.

Então vamos lá, ele é gordinho? Sim, ele é gordinho:

```
porco1 = [1]
```

Ele tem perninha curta? Sim, ele tem perninha curta:

```
porco1 = [1, 1]
```

Esse porquinho faz *auau*? Não, ele não faz *auau*:

```
porco1 = [1, 1, 0]
```

Perceba que eu mudei, agora estou modelando o meu sistema de uma maneira diferente, estou modelando ele de qual maneira? Estamos dizendo que:

```
[1, 1, 0] -> [tem perna curta, é gordinho, não faz *auau*]
```

Agora eu vou colocar o meu segundo porco. Esse porquinho também é gordinho, também tem perna curta e também não faz *auau*:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
```

Por fim eu vou representar o meu terceiro porquinho que também é gordinho, tem perna curta e também não faz *auau*:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
```

Cadastrei os 3 porquinhos, agora eu preciso cadastrar 3 cachorros que eu já conheço. O primeiro cachorro ele é gordinho, ele tem perninha curta e ele faz auau:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro1 = [1, 1, 1]
```

Os outros 2 cachorros não são gordinhos, tem perninhas curtas e fazem *auau*:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro1 = [1, 1, 1]
cachorro2 = [0, 1, 1]
cachorro3 = [0, 1, 1]
```

Esses são os 6 elementos que eu, Guilherme, conheço. Então vamos comentar o que cada uma dessas posições se refere, você lembra?

```
# [é gordinho?, tem perninha curta?, faz auau?]
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro1 = [1, 1, 1]
cachorro2 = [0, 1, 1]
cachorro3 = [0, 1, 1]
```

Dentre esses 6 elementos já sabemos que do primeiro ao terceiro são porquinhos e que do quarto até o sexto são cachorros. A grande sacada é dizer o que já sabemos o que eles são, ou seja, dizer se eles são 1 ou 0, se são 1 ou -1, se é cachorro ou porco... Eu posso escolher o que eu quiser. Precisamos de todos esses elementos, e chamaremos todos eles de `dados`:

```
# [é gordinho?, tem perninha curta?, faz auau?]
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro1 = [1, 1, 1]
cachorro2 = [0, 1, 1]
cachorro3 = [0, 1, 1]

dados = [porco1, porco2, porco3,
cachorro1, cachorro2, cachorro3]
```

Agora que temos os nossos dados agrupados em um *array*, precisamos fazer alguma marcação para indicar o que cada um desses elementos são, usaremos então uma variável chamada `marcacoes` para indicar o que cada um representa:

```
marcacoes
```

Sabemos que do primeiro ao terceiro, são porcos e eu vou marcar com 1, tudo bem?

```
marcacoes = [1, 1, 1]
```

E para marca como cachorro eu vou querer marcar com um outro número, eu posso marcar com 0, -1, 10000... Eu posso marcar com o que eu quiser. Para dar ênfase na distinção dos 2 elementos, ou seja, dizer que um é oposto do outro, eu irei usar 1 positivo(1) para porcos e 1 negativo(-1) para cachorro:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
```

```
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]
```

Marcamos todos os nosso elementos, porém, nós temos agora um elemento misterioso, quem é esse misterioso? Bom, eu tenho um colega que é gordinho, tem perninha curta e faz *auau*:

```
misterioso = [1, 1, 1]
```

E agora? Esse colega é um porco ou um cachorro? Esse é o nosso elemento misterioso... O que nós queremos saber é: "Será que esse elemento misterioso é um cachorro ou é um porco?". Então nós queremos treinar um algoritmo de classificação baseado nesses dados para que, baseado nesses dados, ele me diga se é um cachorro ou é um porco. Vamos lá?

O que precisamos fazer agora é treinar um algoritmo baseado nesses dados. Então nós precisamos importar da biblioteca do `sklearn` uma parte dela que faz o treinamento baseado no algoritmo que se chama *bayesiano*, chamado `naive_bayes`. A partir dessa biblioteca iremos importar o algoritmo `MultinomialNB`, NB de *naive bayes*:

```
from sklearn.naive_bayes import MultinomialNB
```

O Multinomial é o algoritmo que usaremos para treinar o nosso modelo que diz se os nosso elementos são cachorros ou porcos. Para treinar esse modelo nós precisamos dos nossos dados e nossas marcações. E como fazemos para criar um modelo? basta apenas fazer uma chamada para o `MultinomialNB`:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

misterioso = [1, 1, 1]

from sklearn.naive_bayes import MultinomialNB

modelo = MultinomialNB()
```

Criamos um modelo! Vamos tentar rodar o nosso código? Salve o arquivo, abra o terminal, vá até o diretório onde salvou o arquivo e execute:

```
> python classificacao.py
>
```

Não aconteceu nada? Claro que não aconteceu nada! Apenas criamos o modelo! Nem mandamos ele treinar... Como nós, como ser humanos, treinamos? Olhamos para um elemento e classificamos como porquinho, olhamos para outro elemento e classificamos como cachorrinho, isto é, baseado nos dados e marcações nós treinamos o modelo do nosso cérebro, e precisamos fazer a mesma coisa para o nosso modelo:

```
modelo = MultinomialNB()
modelo.treinar(dados, marcacoes)
```

Porém, não existe o método `treinar`, então como faremos isso? Observe que o nosso modelo precisa se adequar aos nossos `dados` e `marcacoes`, ou seja, os dados do porco1 ao porco3 é 1 os dados do cachorro4 ao cachorro6 é -1. É dessa maneira que precisamos **adequar** o nosso modelo, e para isso usaremos o método `fit`:

```
modelo.fit(dados, marcacoes)
```

Nesse momento do código estamos dizendo: "Por favor, adeque-se a essas informações". Vamos testar?

```
> python classificacao.py
>
```

Não aconteceu nada... Nesse momento nós apenas treinamos! E o que precisamos é pedir para que o modelo **preveja** quem é o elemento misterioso utilizando o método `predict`:

```
modelo.predict(misterioso)
```

Esse método vai devolver se é um cachorro ou um porco, então precisamos imprimir o valor com o método `print`:

```
# [é gordinho?, tem perninha curta?, faz auau?]
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

misterioso = [1, 1, 1]

from sklearn.naive_bayes import MultinomialNB

modelo = MultinomialNB()
modelo.fit(dados, marcacoes)
print(modelo.predict(misterioso))
```

Testando novamente o nosso algoritmo:

```
> python classificacao.py
> /usr/local/lib/python2.7/dist-packages/sklearn/utils/validation.py:386: DeprecationWarning: Passir
  DeprecationWarning)
[-1]
```

Ele devolveu um *warning*, porém ele imprimiu o resultado que foi um *array* e, dentro desse *array*, devolveu -1, logo veremos como resolver esse *warning*. E o que significa o -1 pra ele? Significa um cachorro! Como será que ele deduziu que era um cachorro? Se nós olharmos a nossa experiência passada:

```
# [é gordinho?, tem perninha curta?, faz auau?]
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

# restante do código
```

Todos que faziam *auau* eram cachorros. Com base nesses dados, fazendo uma análise pelo nosso cérebro, provavelmente faríamos a mesma classificação. Não sabemos o que aconteceu por trás dos panos, só sabemos que, quando rodamos, ele nos informou que o elemento misterioso é um cachorro! Repare que, nós passamos vários dados para o nosso programa, o treinamos e pedimos para ele verificar um único caso que foi o elemento misterioso, mas será que nós queremos prever apenas um único caso? Não! Nós queremos prever muitos outros casos! Então precisamos de outros elementos misteriosos para que o programa classifique para nós, como por exemplo o `misterioso2` que é gordinho, não tem perninha curta e não faz *auau*:

```
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
```

Bom, ele não é gordinho e não tem perna curta, porém ele não faz *auau*, eu imagino que seja um porco, pois todos que não fazem *auau* são porquinhos. Então agora, em vez de prever o `misterioso1` , pediremos para que ele preveja o `misterioso2` :

```
print(modelo.predict(misterioso2))
```

Se rodarmos novamente:

```
> python classificacao.py
> /usr/local/lib/python2.7/dist-packages/sklearn/utils/validation.py:386: DeprecationWarning: Passir
  DeprecationWarning)
[1]
```

Na interpretação dele, a maior chance que ele dá é que esse `misterioso2` seja um porquinho, bom, se observarmos o `misterioso2` vemos que ele é gordinho, não tem perna longa e não faz *auau*... Eu chutaria que é um porquinho também! Porém nós não prevemos todos de uma vez só! Para isso precisamos adicionar todos os misteriosos dentro de um array e atribuir a uma variável que vamos chamar de `teste` :

```
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]

teste = [misterioso1, misterioso2]
```

E então passamos apenas o `teste` no `predict` :

```
print(modelo.predict(teste))
```

Repara que o parâmetro do método `predict` recebe um array, com vários elementos que queremos testar e é por isso que ele deu aquele warning daquela vez, antigamente ele recebia apenas um valor, porém essa solução foi *depreciada*, e logo não haverá mais suporte para ela. No nosso caso ele deu um *warning*, pode ser que em versões futuras ele nem funcione! Então, mesmo que for um *array* de uma única posição, precisamos passar um array! Vamos testar novamente?

```
> python classificacao.py
> [-1  1]
```

Agora ele imprime dois resultados e diz que o primeiro elemento misterioso é um cachorro e o segundo é um porco. Se verificarmos nossos dois elementos misteriosos:

```
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
```

Realmente, o `misterioso1` é um cachorro e o `misterioso2` é um porco. Então como funciona o algoritmo Bayesiano Multinominal para classificação?

- Treinamos o nosso modelo pedindo para se adequar aos dados e marcações utilizando o método `fit` .
- Preveja pra mim o que eles são por meio do método `predict` .

E então ele vai prevendo se esses elementos são porcos ou cachorros. Mas agora eu estou pensando, será que esse código que criamos funciona apenas para porco ou cachorro? E se, ao invés de cachorro ou porco, esses elementos fossem *spam* ou não *spam*? E se fosse um cliente que não paga as dívidas e um cliente que paga as dívidas em dia? E se esses 0 e 1 classificassem qualquer outra coisa? Eu mudaria alguma coisa no código? O que você acha? É justamente esse exemplo que eu queria demonstrar para você: como podemos implementar a mesma coisa que vimos com tabelas e números e traduzimos isso pra código, extremamente simples! E quais são os passos mesmo?

- Para cada um dos elementos a gente define as variáveis, por exemplo `porco1 = [1, 1, 0]` , `porco2 = [1, 1, 0]` e assim por diante.
- Cada variável quer dizer alguma coisa, no nosso exemplo foi: `[é gordinho?, tem perninha curta?, faz auau?]` .
- Marcamos esses elementos para diferenciar um do outro, no nosso exemplo utilizamos 1 para porco e -1 para cachorro: `marcacoes = [1, 1, 1, -1, -1, -1]` .
- Então treinamos o nosso modelo: `modelo.fit(dados, marcacoes)` .

- E, por fim, pedimos para ele imprimir os nossos testes: `print(modelo.predict(teste))` .

Repare que as marcações foram feitas de uma forma diferente da qual vimos na tabela: em vez de usar 0 e 1 eu usei -1 e 1. Eu optei por essa escolha, pois dessa forma dá uma ênfase melhor de distinção. Poderíamos testar com 0 e 1 sem problema algum:

```
marcacoes = [1, 1, 1, 0, 0, 0]
```

Se tentarmos rodar novamente:

```
> python classificacao.py
> [0 1]
```

Mas vamos deixar -1 e 1, justamente pra dar essa ênfase de um ser o oposto do outro no momento em que eu tenho duas categorias!

Vimos um algoritmo que dados vários números para ele, devolve um único número, parece uma coisa de nerd... Mas perceba que é genial poder dizer se um animal é um ou outro, se um e-mail é *spam* ou não é, se uma pessoa, um ser humano, vai se comportar de uma maneira ou de outra de acordo com o que aconteceu no passado, ou seja, de acordo com experiências anteriores, nós podemos classificar grupos para podermos tomar decisões.

Isso não significa que iremos tomar decisões negativas ou ruins, pois podemos tomar decisões positivas, por exemplo, poder dizer se um vídeo que está subindo para o Youtube é um vídeo legal, isso é, uma vídeo sem violência ou sem qualquer conteúdo que não pode ser exibido. Provavelmente, eu não terei pessoas assistindo todos os vídeos existentes no Youtube, eu preciso de fato, de um programa que consiga verificar se o vídeo contém um conteúdo permitido ou não para depois ter um filtro por um ser humano, isso significa que o primeiro filtro precisa ser por uma máquina para que ela consiga verificar todas as regras que precisam ser aplicadas para um vídeo novo que será publicado.

Um outro exemplo é uma música que está subindo, eu preciso verificar se ela é pirata ou não. Nós precisamos verificar se o usuário está praticando pirataria, se ele tem os direitos para baixar o conteúdo ou não. Tudo isso significa que precisamos classificar baseados nas características que nós temos, para cada caso usaremos características diferentes. Mas também existe uma coisa muito importante que é comum em todos esses algoritmos que utilizamos, como também é comum a nós: **nem sempre nós temos certeza da nossa classificação**, vejamos uma exemplo:

- Vamos supor que eu peguei todos os vídeos do Youtube.
- Agora eu vou tentar classificar todos esses vídeos verificando se aparece um gatinho nele ou não.

Então esse algoritmo classifica todos os vídeos que contém um gatinho. E aí a gente precisa parar e pensar que todo algoritmo contém uma margem de erro e precisamos nos atentar a toda margem de erro que o nosso algoritmo pode apresentar, por exemplo, ele pode classificar alguma coisa como gatinho, porém não era um gatinho, ou então, classificar um gatinho como não gatinho, existem diversos tipo de erros que podem acontecer num algoritmo de classificação e precisamos ficar atento ao erro para que possamos controlá-lo e saber a nossa taxa de erro, compreender qual é a taxa de erro que iremos aceitar no nosso algoritmo.

Nós iremos nos atentar a esse margem de erro e vamos aprender como calcular para poder dizer se esse erro está sendo bom ou se nós estamos melhorando o nossa margem atual. Podemos concluir que todos os nossos algoritmos apresentarão erros, porém, o nosso papel é, a partir dos erros, verificar se são aceitáveis ou não, por exemplo: é aceitável que a minha foto seja

classificada como gatinho? Não existe uma regra fixa para sabermos se isso é aceitável ou não, pois, dependendo do ponto de vista pode ser que sim ou pode ser que não, ou seja, depende muito do contexto para podermos dizer se o nosso erro é aceitável ou não é aceitável ou se eu preciso de uma margem de erro menor ou se está dentro do padrão que esperamos.

Vimos que lá no começo do capítulo que nós tinhamos 3 porcos e 3 cachorros, cada um com suas variáveis, ou seja, suas características que definiam se eles eram gordinhos (1) ou não (0), se tinham perninhas curtas (1) ou não (0) e, por fim, verificamos se eles faziam *auau* (1) ou não (0) e representamos como nossos elementos:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]
```

E nós tínhamos agrupados todos esses elementos em um *array* para representar os nossos dados:

```
dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]
```

E também fizemos as marcações para indicar quais desses elementos eram porcos ou cachorros, marcando com 1 para porco e -1 para cachorro:

```
marcacoes = [1, 1, 1, -1, -1, -1]
```

Então treinamos o nosso algoritmo:

```
modelo = MultinomialNB()
modelo.fit(dados, marcacoes)
```

E por fim nós testamos 2 elementos misteriosos:

```
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]

teste = [misterioso1, misterioso2]

print(modelo.predict(teste))
```

Mas existe um detalhe no processo que fizemos, pois não adianta nós rodarmos o nosso algoritmo e não fazer ideia de quão bom ele é. Quando treinamos o nosso cérebro, chegamos a ver 300, ou mais, porcos e cachorros na nossa vida, porém nós não temos certeza se estamos bons o suficiente para distinguir entre um cachorro e um porco ou então *spam* e não *spam ou pessoas que vão me pagar e que não vão me pagar...

E como podemos ter certeza se estamos bons ou não para fazer essas distinções? Precisamos testar! Sim, testar no mundo real, com algum valor que desconhecemos, se iremos funcionar conforme o esperado para verificar o quão bons estamos sendo com esse algoritmo. E como podemos testar? Podemos criar um cenário de teste utilizando os simulando os nossos elementos reais (porcos ou cachorros) por elementos misteriosos:

```
# [é gordinho?, tem perninha curta?, faz auau?]
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
misterioso3 = [0, 0, 1]

teste = [misterioso1, misterioso2, misterioso3]
```

Agora que temos o nosso cenário de teste, podemos informar os resultados esperados para cada um desses elementos misteriosos, ou seja, as marcações de teste:

```
marcacoes_teste
```

Agora, precisamos classificar cada elemento misterioso e indicar nas nossas marcações de teste. Sabemos que o `misterioso1` é um cachorro(-1), o `misterioso2` é um porco e o `misterioso3` é um cachorro(-1):

```
marcacoes_teste = [-1, 1, -1]
```

Por fim, para ficar mais claro, vamos atribuir o retorno do método `predict()`, que são os nossos resultados, para uma variável chamada `resultado`:

```
resultado = modelo.predict(teste)
print(resultado)
```

Agora repare o nosso código:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

from sklearn.naive_bayes import MultinomialNB

modelo = MultinomialNB()
modelo.fit(dados, marcacoes)

misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
misterioso3 = [0, 0, 1]

teste = [misterioso1, misterioso2, misterioso3]

marcacoes_teste = [-1, 1, -1]

resultado = modelo.predict(teste)
```

```
print(resultado)
```

Será que agora ele vai prever conforme as nossas marcações de teste? Ou seja, falar que os elementos misteriosos são: cachorro, porco e cachorro? Vamos testar:

```
> python classificacao.py
> [-1  1 -1]
```

Será que o nosso código se saiu bem? Vamos verificar os valores que estávamos esperando nas nossas marcações de teste:

```
marcacoes_teste = [-1, 1, -1]
```

De acordo com as nossas marcações, o nosso código funcionou muito bem, ele conseguiu prever! Mas é importante lembrar que no mundo real, ou seja, na prática, esse resultado não costuma ser 100% e o próximo exemplo que veremos logo mais, de compras na web, não iremos conseguir acertar sempre. Porém, observe que estamos testando isso manualmente, ou seja, estamos verificando se o teste passou a olho nu, para o nosso caso atual, não tem problema, mas no mundo real, iremos realizar testes gigantes, como por exemplo, 1000 elementos! E não faz sentido nós ficarmos olhando 1 a 1 para verificar o quanto ele acertou ou o quanto ele errou... O que precisamos saber é: quantos elementos valores da variável `resultado` (resultados que o algoritmo classificou) são diferentes da variável `marcacoes_teste` (resultado que esperamos dos elementos). Mas como podemos fazer para verificar os valores que foram diferentes? Podemos subtrair os 2 arrays:

```
print(resultado - marcacoes_teste)
```

Parece estranho mas essa abordagem funciona da seguinte maneira:

- Resultado e marcação: 1, resultará em: 1 - 1 = 0.
- Resultado: 1 e marcação: -1, resultará em: 1 - (-1) -> 1 + 1 = 2.
- Resultado: -1 e marcação: -1, resultará em: -1 + 1 = 0.
- Resultado: -1 e marcação: 1, resultará em: -1 - 1 = -2

Resumindo todas as possibilidades que fizemos, nos casos em que os valores forem iguais o resultado será 0, ou seja, quando resultar em 0, saberemos que o algoritmo acertou. Vamos testar o nosso código?

```
> python classificacao.py
> [0 0 0]
```

O resultado foi tudo 0, então acertou 100% novamente! Mas o ideial seria que o nosso algoritmo mostrasse o percentual ao invés de só a diferença entre o `resultado` e a `marcacoes_teste`. Então vamos fazer com que a diferença dessas duas variáveis sejam retornadas para uma variável chamada `diferencas`:

```
diferencas = resultado - marcacoes_teste
```

Mas como iremos fazer para verificar os acertos? Sabemos que qualquer valor igual a 0 é um acerto, ou seja, qualquer valor da variável `diferencas` que seja 0. Isso significa que, para cada diferença dentro da variável `diferencas` :

```
for d in diferencas
```

Se o valor for igual a 0:

```
for d in diferencas if d == 0
```

retorne:

```
acertos = for d in diferencas if d == 0
```

Porém, nós precisamos passar isso para um array, então fazemos:

```
acertos = [d for d in diferencas if d == 0]
```

Se imprimirmos os acertos:

```
acertos = [d for d in diferencas if d == 0]
print(acertos)
```

Resultado:

```
> python classificacao.py
> [0, 0, 0]
```

Observe que foi impresso foi um array com todos os 0 contidos no array `diferencas` , ou seja, todos os valores, porém, não queremos os valores, precisamos saber a quantidade de elementos que existem nesse array, e como fazemos para retornar a quantidade de elementos existentes de um array? Utilizamos a função `len()` que retorna o tamanho do array:

```
len(acertos)
```

E esse será o nosso total de acertos, então vamos representar por uma variável chamada `total_de_acertos` :

```
total_de_acertos = len(acertos)
```

Agora vamos imprimir o `total_de_acertos` :

```
total_de_acertos = len(acertos)
print(total_de_acertos)
```

Se rodarmos novamente:

```
> python classificacao.py
> [0, 0, 0]
> 3
```

Porém, para verificarmos a porcentagem, nós precisamos também, do total de elementos que foram testados, podemos extrair esse valor a partir do nosso array `teste` que contém todos os elementos que foram testados:

```
teste = [misterioso1, misterioso2, misterioso3]

marcacoes_teste = [-1, 1, -1]

resultado = modelo.predict(teste)

diferencas = resultado - marcacoes_teste

print(diferencas)

acertos = [d for d in diferencas if d == 0]

total_de_acertos = len(acertos)

print(total_de_acertos)

total_de_elementos = len(teste)

print(total_de_elementos)
```

Testando o nosso código:

```
> [0 0 0]
> 3
> 3
```

Agora que temos tanto o total de acertos quanto o total de elementos, precisamos apenas fazer a divisão de `total_de_acertos` por `total_de_elementos` para extraírmos a nossa taxa de acerto:

```
total_de_acerto / total_de_elementos
```

Então atribuímos para uma variável chamada `taxa_de_acerto` e a imprimimos:

```
taxa_de_acerto = total_de_acerto / total_de_elementos
print(taxa_de_acerto)
```

Verificando o resultado:

```
> python classificacao.py
> [0 0 0]
> 3
```

```
> 3
> 1
```

Vamos melhorar a nossa impressão, vamos retirar todas as impressões. Então imprimimos primeiro o nosso resultado: `print(resultado)`, depois a diferença entre o resultado e as marcações de teste: `print(diferenca)`, por fim, faremos a impressão da taxa de acerto: `print(taxa_de_acerto)`, porém vamos multiplicar por 100.0: `taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos`, para que seja apresentado como percentual, vejamos o resultado do nosso código final:

```
porco1 = [1, 1, 0]
porco2 = [1, 1, 0]
porco3 = [1, 1, 0]
cachorro4 = [1, 1, 1]
cachorro5 = [0, 1, 1]
cachorro6 = [0, 1, 1]

dados = [porco1, porco2, porco3, cachorro4, cachorro5, cachorro6]

marcacoes = [1, 1, 1, -1, -1, -1]

from sklearn.naive_bayes import MultinomialNB

modelo = MultinomialNB()
modelo.fit(dados, marcacoes)

misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
misterioso3 = [0, 0, 1]

teste = [misterioso1, misterioso2, misterioso3]

marcacoes_teste = [-1, 1, -1]

resultado = modelo.predict(teste)

diferencas = resultado - marcacoes_teste

acertos = [d for d in diferencas if d == 0]

total_de_acertos = len(acertos)
total_de_elementos = len(teste)

taxa_de_acerto = 100.0 * total_de_acertos / total_de_elementos

print(resultado)
print(diferencas)
print(taxa_de_acerto)
```

Rodando o nosso algoritmo, obtemos o seguinte resultado:

```
> python classificacao.py
> [-1  1 -1]
> [0 0 0]
> 100.0
```

100.0! Porém, é sempre válido lembrar que 100.0 é um número muito difícil de acontecer no mundo real. Um exemplo bem simples que demonstra o nosso algoritmo errando seria modificar a nossa `marcacoes_teste` informando que o último elemento misterioso era um porco que fazia *auau*, ou seja, um porquinho que faz *auau*:

```
misterioso1 = [1, 1, 1]
misterioso2 = [1, 0, 0]
misterioso3 = [0, 0, 1]

teste = [misterioso1, misterioso2, misterioso3]

marcacoes_teste = [-1, 1, 1]
```

Vamos verificar se o nosso algoritmo vai adivinhar que esse porquinho que faz *auau* é um porquinho? Vejamos o resultado:

```
> python classificacao.py
> [-1  1 -1]
> [ 0  0 -2]
> 66.6666666667
```

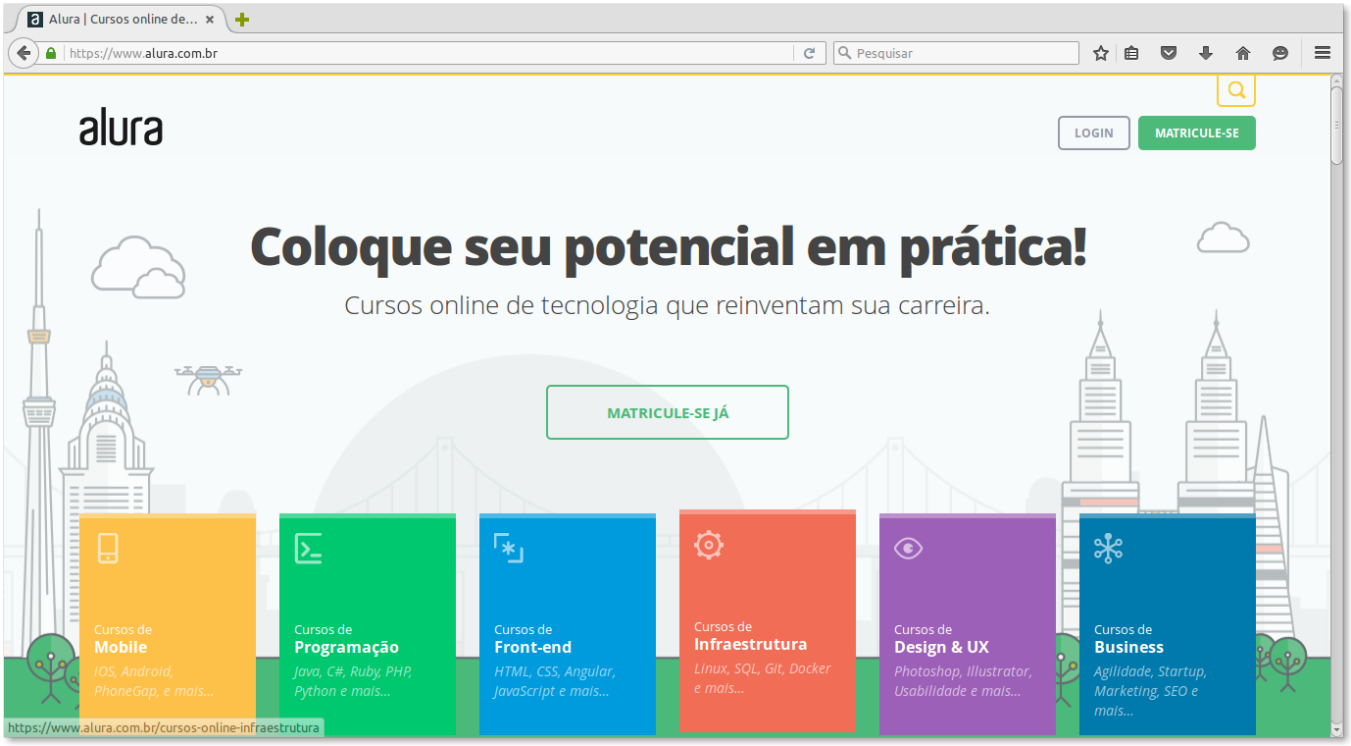
Como vimos, ele errou! perceba que o úlitmo valor da `diferenca` foi diferente de 0, ou seja, um erro! Por fim, a nossa taxa de acerto foi de 66%.

Repara que a taxa de acerto é fundamental para que o nosso algoritmo, no momento em que recebe novos dados, nesse caso, animais novos com características novas, seja possível verificar o quão bom ele foi em uma situação do mundo real, como por exemplo, nesse último teste, ele foi capaz de acertar 66%.

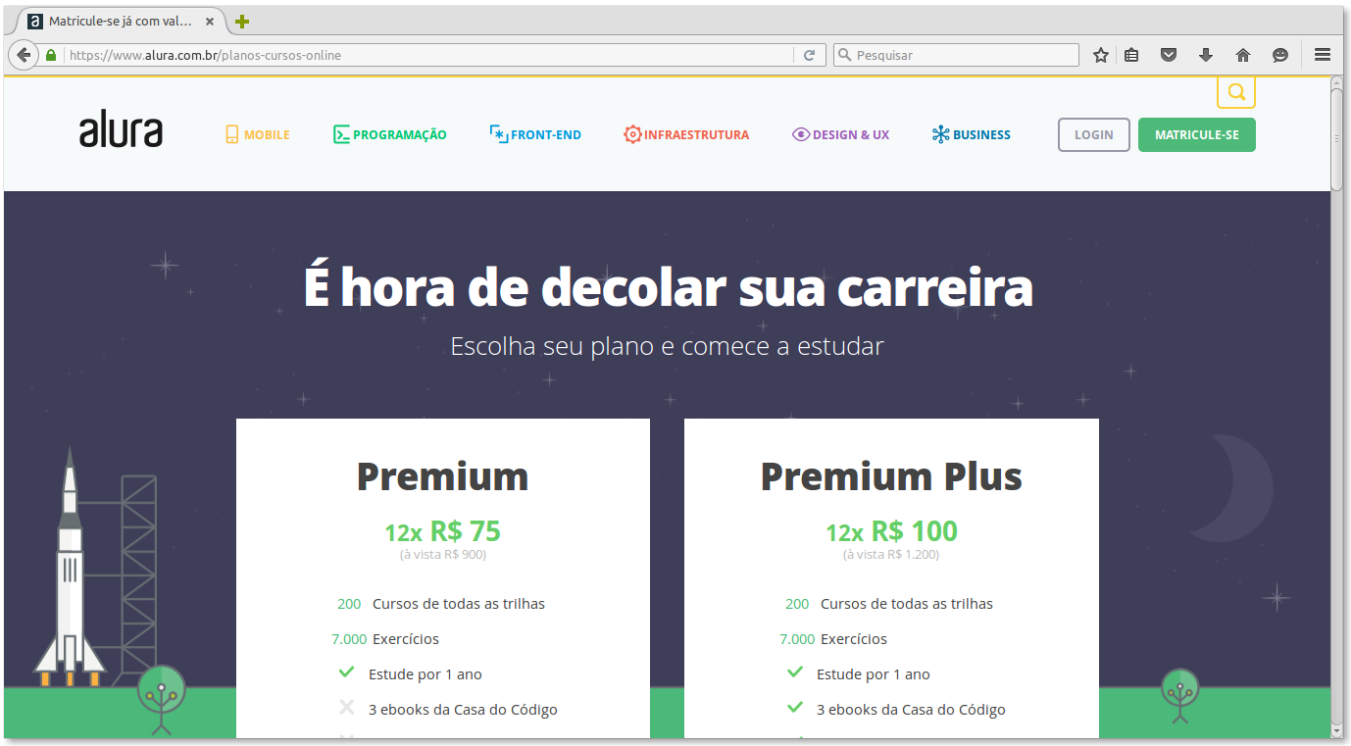
Daqui a pouco a nossa situação será diferente, iremos utilizar um exemplo real em uma aplicação web onde iremos pegar muito mais valores para realizarmos o nosso teste com diversas características e um melhor calculo para a taxa de acerto. E nesse mesmo exemplo, veremos como a nossa taxa de acerto dificilmente acertará 100%.

Até agora fizemos uma classificação entre um porco e um cachorro, porém, o que queremos no nosso dia-a-dia, é realizar uma classificação, por exemplo, do mundo web. Vamos tentar fazer essa analise para esse cenário? Veremos um exemplo do mundo web que pode acontecer:

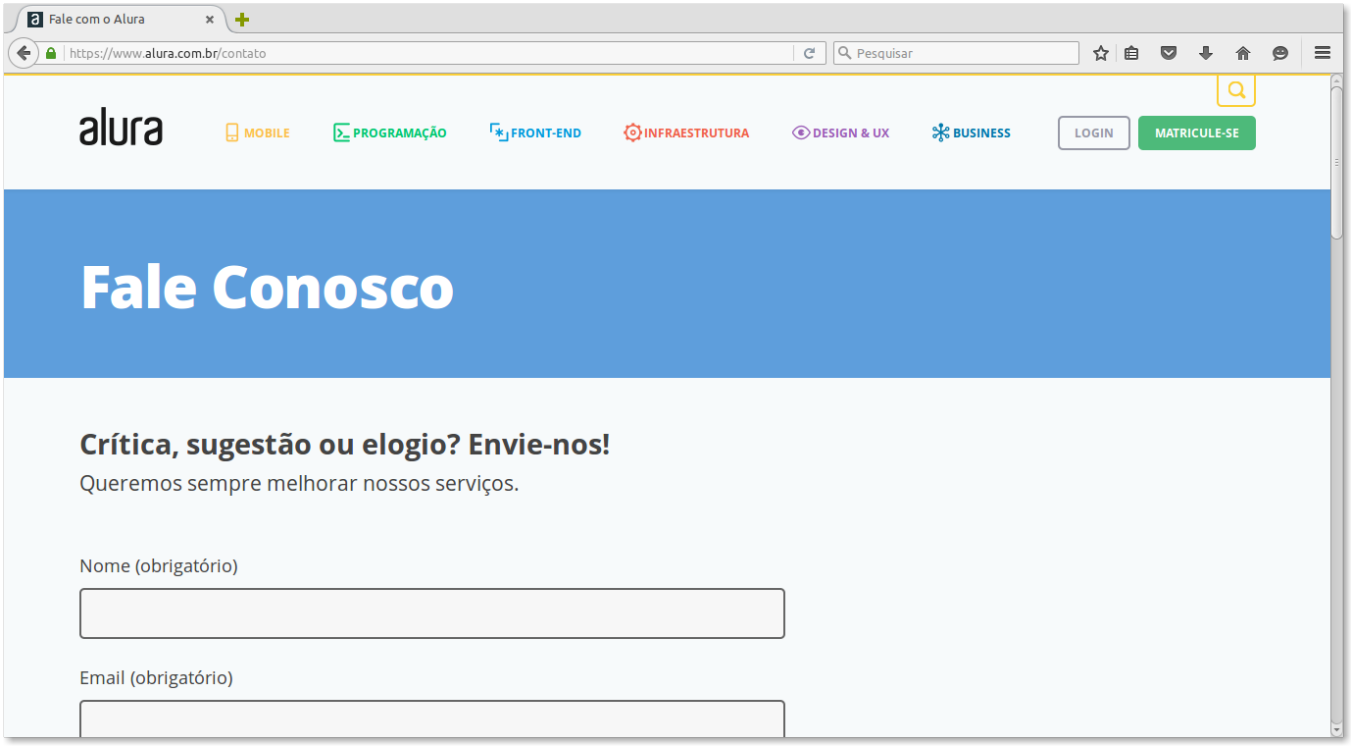
Um dos exemplos seria um usuário que entra num site, nesse exemplo no site do Alura (https://www.alura.com.br/):



Então ele entra numa [segunda página (https://www.alura.com.br/cursos-online-programacao)]:



Por fim ele chega nessa [terceira página (https://www.alura.com.br/curso-online-logica-de-programacao)]:



Visto apenas essas 3 páginas, surgem a pergunta, será que o usuário vai ou não comprar? Percebeu que é uma pergunta de classificação? Ou seja, as perguntas de classificação estão em todos os lugares, já citei alguns exemplos, um deles seria:

Eu tenho um funcionário que trabalhou x horas no ano passado, teve y projetos, na avaliação entre pessoas ele ficou com a média z, o chefe dele deu um aumento, ele não tirou férias... E agora vem a pergunta: "E aí? Ele vai pedir para sair no próximo ano ou não?".

Tudo isso é um problema de classificação, e são problemas reais usado no mundo a fora, por exemplo, essa situação para detectar se um funcionário vai ou não continuar. A Accenture (https://www.accenture.com) é uma das empresas muito famosa por usar esse tipo de classficação. Um outro exemplo, poderia ser esse que fizemos do usuário que acessou as página do site, quais serão as chances desse usuário comprar ou não? Se ele não comprar, não seria melhor eu ajudar ele para verificar se ele tem interesse pelo produto? Pois, se ele não tiver, tudo bem, ele vai procurar outra coisa, mas se ele tiver interesse, provavelmente ele está com dúvida sobre o produto e precisa de ajuda.

Veja o quão interessante é utilizar a classificação em diferentes contextos, ou seja, não utilizamos a classficação para classificar apenas cachorros ou porcos, também não utilizamos apenas para as coisas mais loucas do Google (https://www.google.com.br/), e também, não é só para fazer com que um carro dirija sozinho... Queremos vender um produto! Queremos ajudar um cliente, queremos verificar se um aluno vai bombar na matéria esse ano... Em todos esses casos eu tenho uma classificação, e aqui eu estou querendo classificar se ele vai comprar ou não.

Repara que, se ele vai comprar ou não, o que podemos analisar? Vamos verificar:

- Ele visitou a página 1? Sim(1)
- Ele visitou a página 2? Sim(1)
- Ele visitou a página 3? Sim(1)

Na nossa analise ele visitou as 3 páginas [1, 1, 1], ele vai comprar ou não? Podemos fazer diversas variações, como por exemplo: Ele visitou apenas a página 1 e página 3 [1, 0, 1], será que ele vai comprar? Agora ele visitou a página 2 e a terceira apenas [0, 1, 1], será que ele vai comprar?

Percebeu que estamos caindo no mesmo problema de sempre? Isso é, dada uma matriz de características de cada usuário do passado e seus comportamentos, ou seja, suas classificações, como um usuário novo no site vai se comportar? Essa é a classificação que queremos fazer! Nesse caso, a nossa dúvida é: "Será que que ele vai comprar ou não?". Então repara que sempre caímos nessa situação de classificação, baseado em características!

Nesse primeiro capítulo, nós escrevemos um código em que temos os nossos elementos, ou seja, os nossos dados:

```
dados = [porco1, porco2, porco3,
         cachorro4, cachorro5, cachorro6]
```

E as nossas marcações:

```
marcacoes = [1, 1, 1, -1, -1, -1]
```

E da mesma maneira que temos os nossos dados e marcações, também temos os nossos testes que representam os nossos dados com elementos de teste:

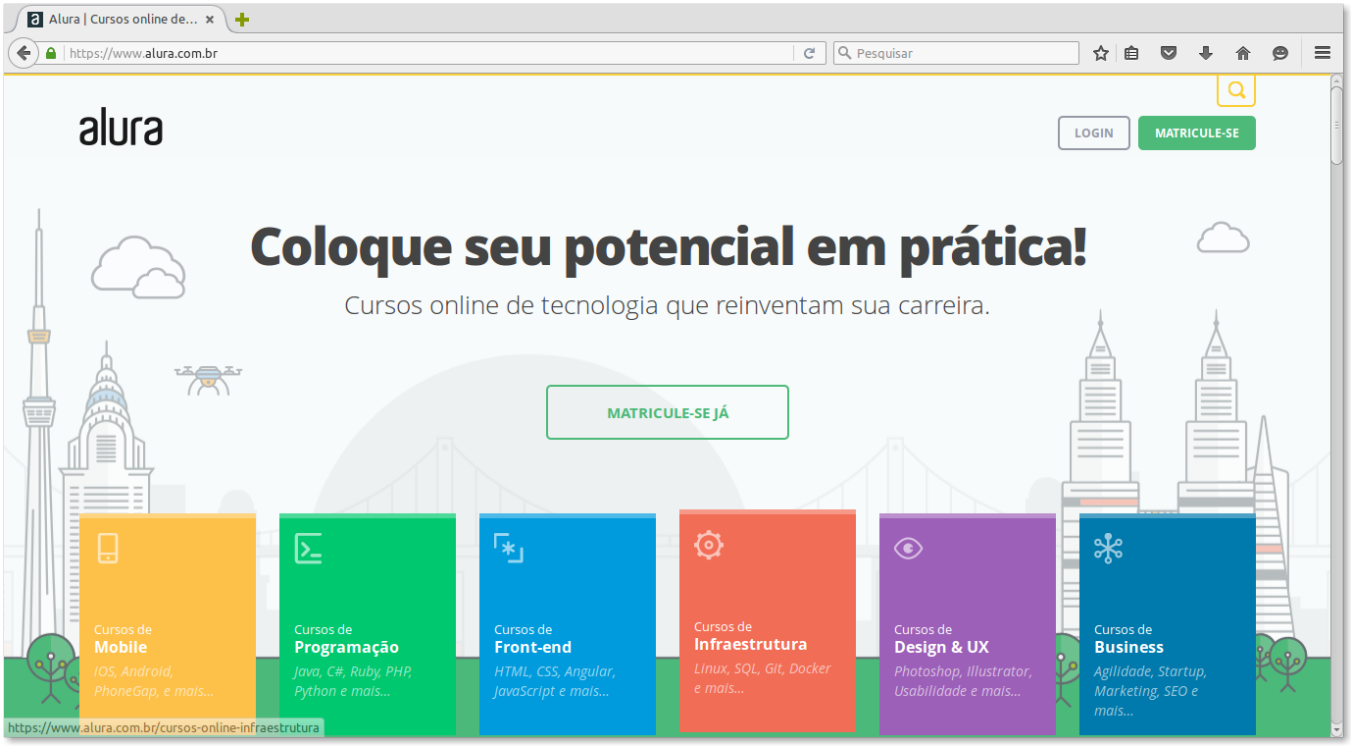
```
teste = [misterioso1, misterioso2, misterioso3]
```

E as nossas marcações de teste:

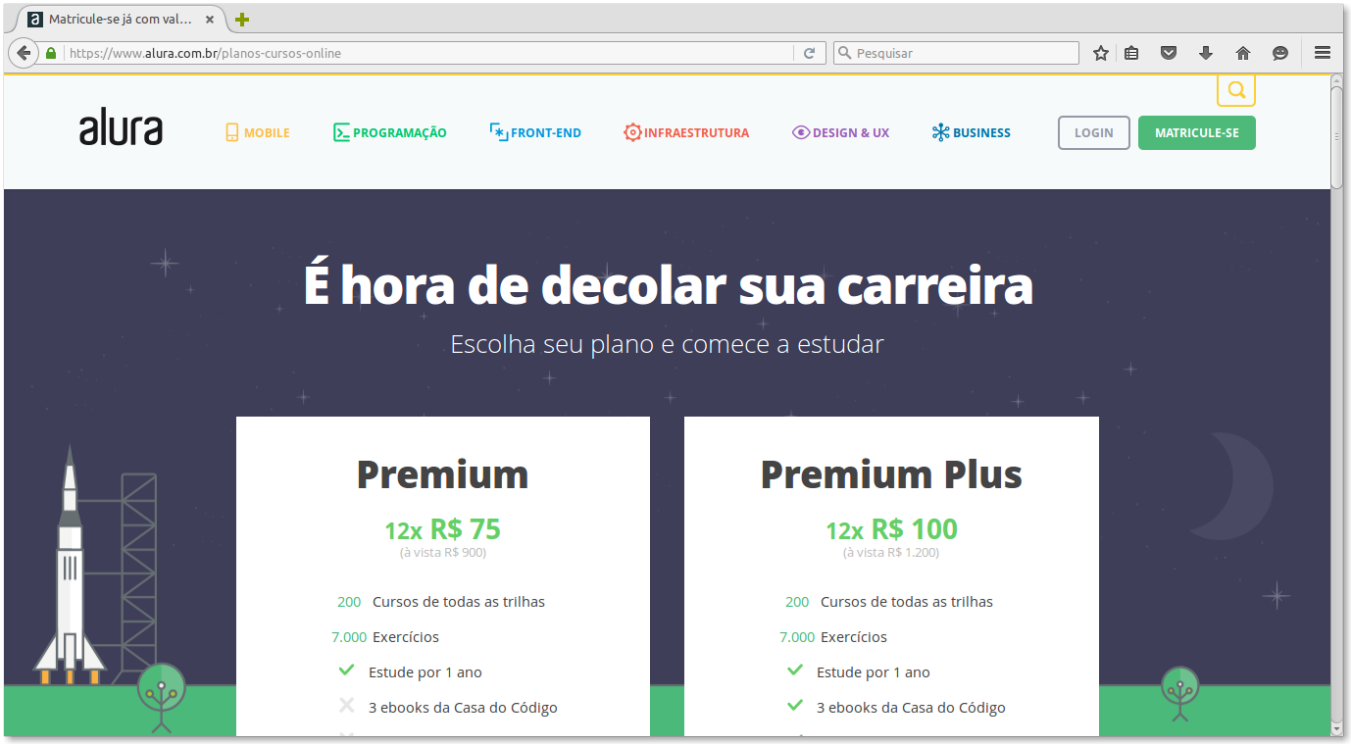
```
marcacoes_teste = [-1, 1, 1]
```

E tudo isso representava o que? Os animais que estávamos classificando entre -1 e 1 que eram porcos e cachorros, porém, poderíamos representar alguma outra coisa, e vimos que podemos levar isso para diversos mundos! Vamos trazer isso para o mundo web?

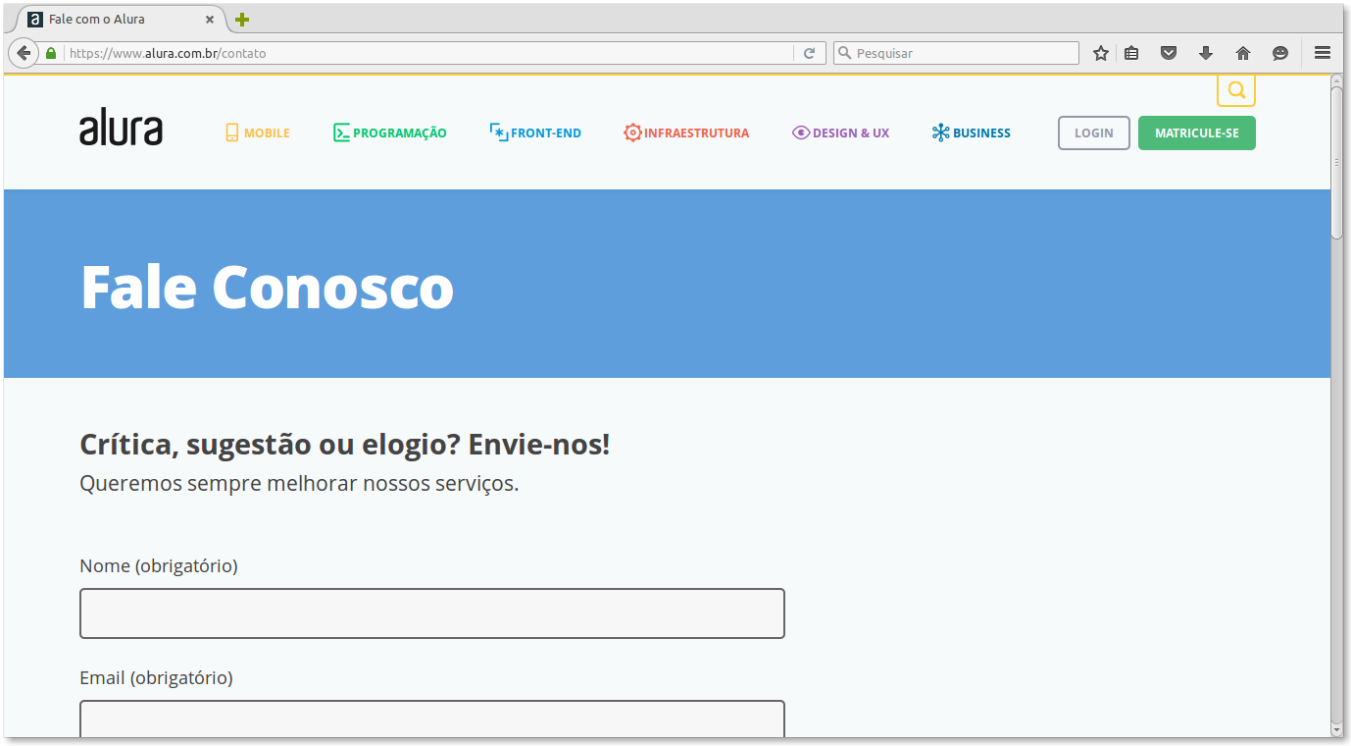
No mundo web, temos diversos usuários acessando meu sistema, então teremos um log de acesso, vamos utilizar a página do Alura (https://www.alura.com.br/) como exemplo. A primeira é página principal:



E então nós temos a uma página de uma categoria de cursos:



Por fim, chegamos a uma página de um curso:

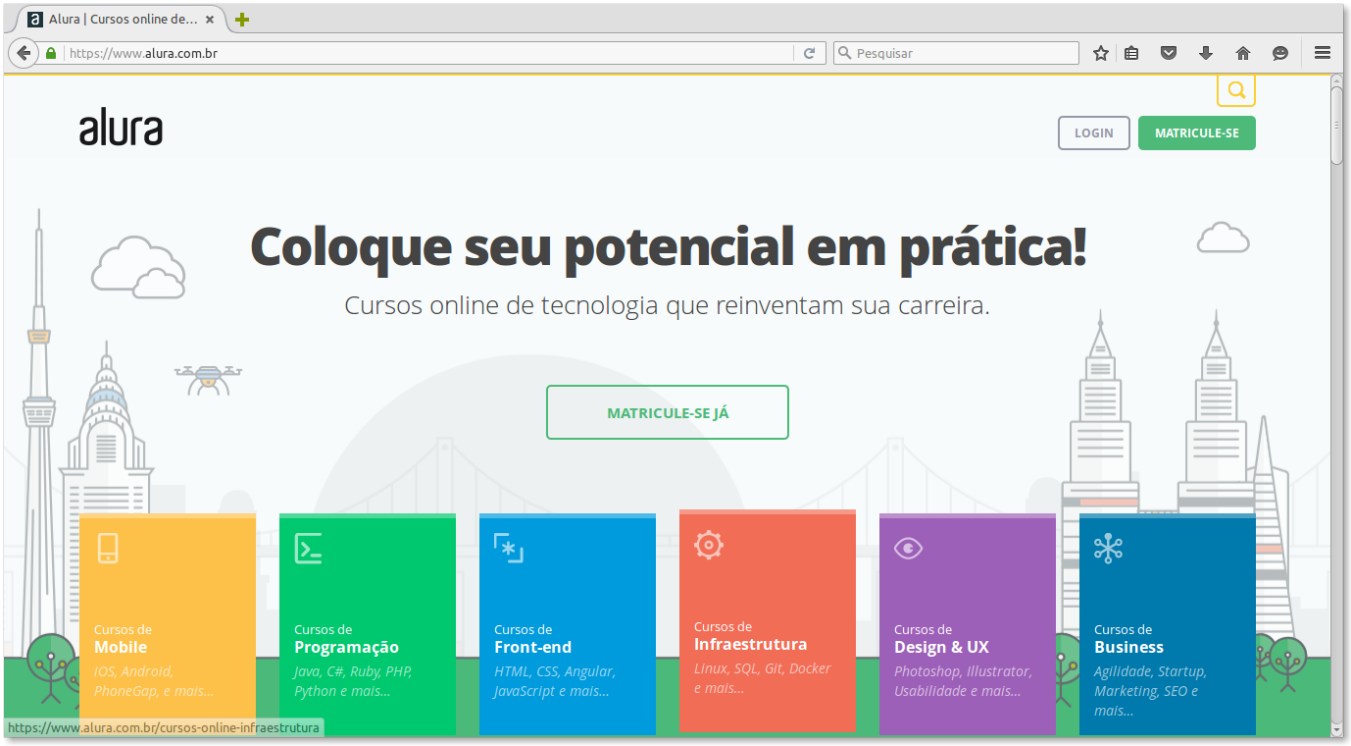


E, de alguma forma, sabemos quais foram as páginas que o usuário acessou, ou seja, se acessou as 3 páginas, se acessou a página 1 e a 3 se acessou a 2 e 3 e assim por diante. É claro que num cenário real, a analise é feita mais 100 páginas ou mais, porém, inicialmente, faremos com 3 páginas, 3 características! Assim como fizemos com os animais (cachorros e porcos). Tudo bem?

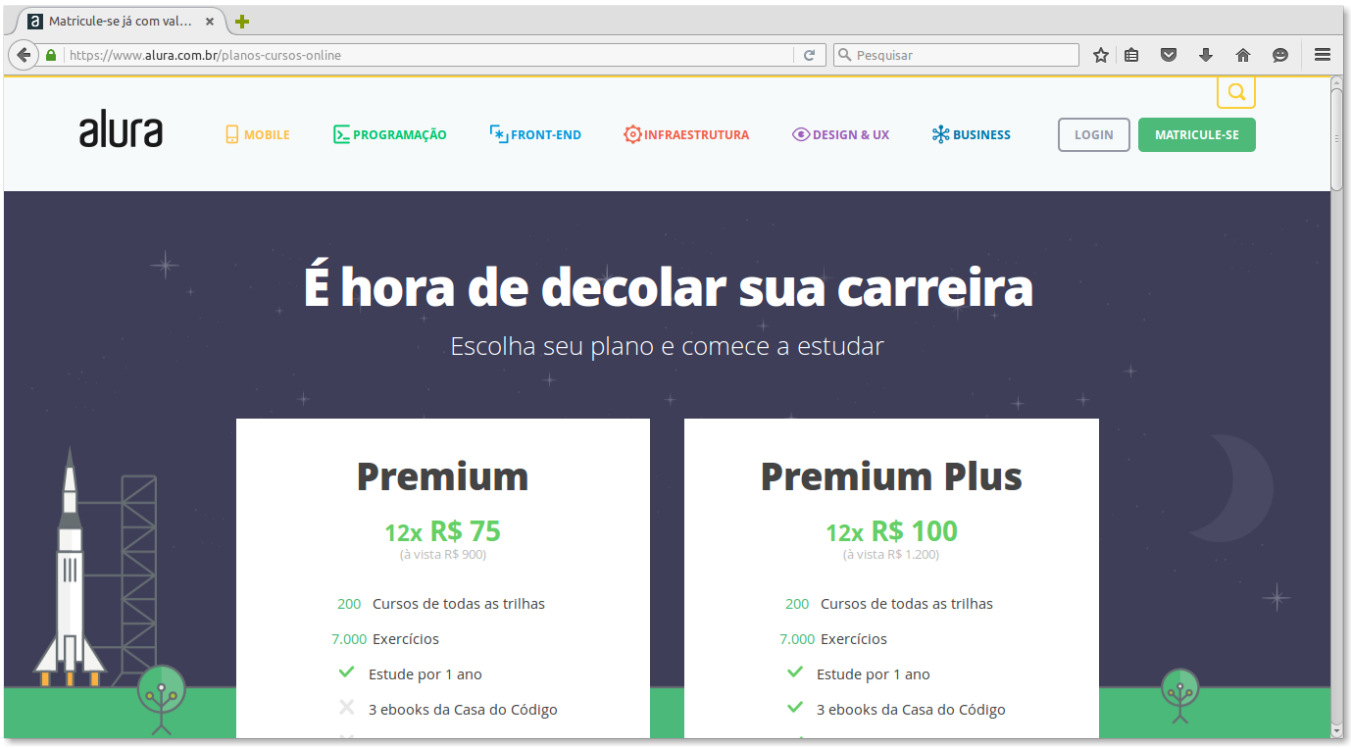
Sabemos quais páginas o usuário acessou, então o que precisamos saber? Queremos saber se esse mesmo usuário vai comprar ou não, se vai assinar o meu produto ou não vai, se ele vai virar um cliente ou não, se ele vai entrar em contato ou não, tudo isso se resume entre 0 e 1, ou seja, é um tipo de classificação! Estamos classificando um usuário que está acessando o meu site de acordo com quais características? 3 características:

- Se ele visitou a primeira página.
- Se ele visitou a segunda página
- Se ele visitou a terceira página

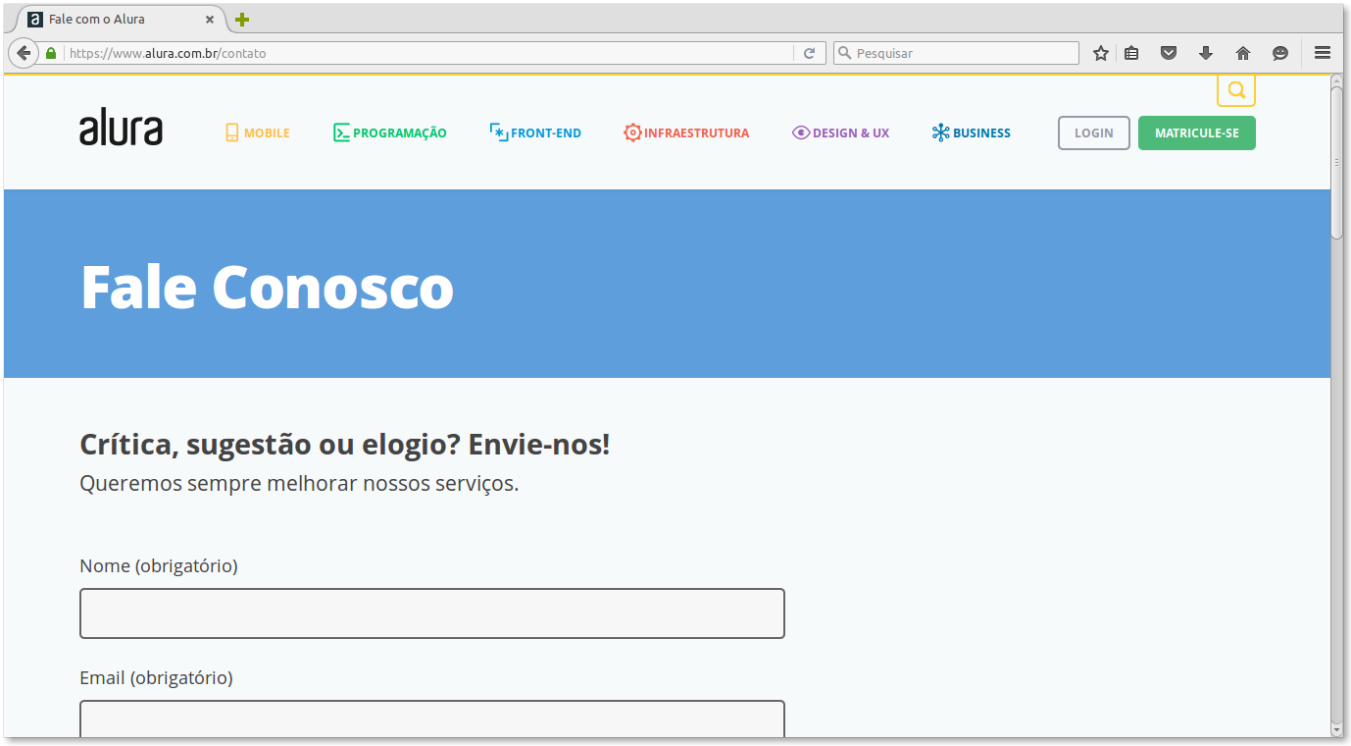
Quais páginas poderia ser essas? Podemos utilizar a página principal (https://www.alura.com.br/):



Será que o usuário entrou na página de como funciona (https://www.alura.com.br/planos-cursos-online)?



Por fim, será que o usuário entrou na página de contato (https://www.alura.com.br/contato)?



Será que ele entrou nessas 3 páginas? Vamos utilizar 0 para indicar que não entrou e 1 para indicar que entrou. Então vamos criar um novo arquivo. Agora vamos preencher as características do usuário.

O Primeiro usuário acessou a página principal(1), porém, não acessou a página de como funciona(0) e nem a página de contato(0), mas esse cara comprou(1):

```
1,0,1,1
```

O segundo usuário acessou por um link interno, ou seja, nem passou pela home(0) e foi direto ao como funciona(1), porém, ele não se interessou, não entrou em contato(0) e não comprou(0):

```
1,0,1,1
0,1,0,0
```

Um terceiro usuário entrou na página principal(1), foi para a página de como funciona(1), não entrou na página de contato(0) e comprou(1).

```
1,0,1,1
0,1,0,0
1,1,0,1
```

As 3 primeiras variáveis representa cada acesso que o usuário teve no nosso sistema e por fim, indicamos se ele comprou ou não. Vamos deixar mais claro o que está acontecendo:

```
acessou_home, acessou_como_funciona,
acessou_contato, comprou
1,0,1,1
0,1,0,0
1,1,0,1
```

Observe que estamos analisando passo-a-passo as ações do usuário para prever se ele vai comprar ou não no nosso site, mas por que estamos fazendo isso? Digamos que prevemos que o usuário não vai comprar, dessa forma, podemos entrar em contato com ele para entender o motivo dele não ter comprado, de repente algo não o agradou do que estamos fornecendo ou então o que ele procura ainda não está disponível e, então, iremos correr atrás para disponibilizar o conteúdo, por exemplo. Ou então, é só uma dúvida que ele tem e não conseguiu tirar com ninguém e nós mesmo iremos tirar a dúvida para que ele adquira o produto... Então eu estou demonstrado um exemplo de como podemos classificar no mundo da web. Independentemente se você natura ou se é um e-commerce que vende de tudo ou é um blog e você precisa saber se ele acessou determinadas páginas e se ele vai clicar em um propaganda ou não, se ele vai voltar daqui a x dias ou não. Ou seja, não importa a qual necessidade estamos fazendo a classificação, o importante é, dadas as características do usuário eu quero saber se ele comprou ou não. Observe que na primeira linha, as 3 primeiras colunas: `acessou_home, acessou_como_funciona, acessou_contato` do no nosso novo arquivo, são as características que desejamos analisar dos nossos usuários e a última: `comprou` é a nossa marcação! Estamos classificando os usuários como 1 e 0, ou seja, ele comprou ou não?

Temos todos os nossos dados do histórico de acesso dos usuários. Porém, repare que cada uma das linhas são separadas por vírgula, ou seja, de acordo com o cabeçalho:

```
acessou_home, acessou_como_funciona,
acessou_contato, comprou
```

Esse tipo de arquivo chamamos de **comma separated value** (valor separado por vírgula) mais conhecido por `csv` , esse arquivo separa todos os valores por vírgula. Esse tipo de arquivo, nós conseguimos ler pelo python facilmente por meio de uma função que carrega esse arquivo e então tratamos os dados de uma maneira bem fácil e é exatamente isso que faremos no próximo capítulo. Iremos utilizar esse padrão para a leitura dos dados, pois é um padrão bem simples de manter.