

[VER PLANOS](#)[PROGRAMAÇÃO](#)[FRONT-END](#)[DATA SCIENCE](#)[INTELIGÊNCIA ARTIFICIAL](#)[DEVOPS](#)[UX & DESIGN](#)[MOBILE](#)[INOVAÇÃO & GESTÃO](#)

Alura > Cursos de Data Science > Cursos de Data Science >
Conteúdos de Data Science >
Primeiras aulas do curso Pandas: limpeza e tratamento de dados

Pandas: limpeza e tratamento de dados

Conhecendo os dados - Apresentação

0:00 / 1:31

Boas-vindas ao curso **Pandas: Limpeza e tratamento de dados!** Meu nome é Bruno Raphael e serei o seu instrutor ao longo deste curso.

Bruno Raphael é um homem que apresenta pele de cor negra e olhos castanhos. Seus cabelos são castanhos, cacheados e curtos, e ele tem um bigode e um cavanhaque. Usa óculos de grau com armação preta e arredondada. Veste uma camiseta preta e está sentado em uma cadeira preta e verde. Ao fundo, do lado esquerdo do instrutor, há uma estante de livros e do lado direito um guarda-roupas. A parede atrás é branca e há uma luz gradiente na cor azul.

O que vamos aprender?

Este curso é ideal para quem deseja adquirir conhecimentos sobre como lidar com dados incompletos, incluindo dados nulos, e torná-los prontos para análise de forma mais consistente e precisa. Além disso, aprenderemos a inserir essa base de dados em um modelo de machine learning (traduzindo, aprendizado de máquina).

Tratamentos na base de dados

Vamos pegar uma base de dados e tratar seus registros, aplicando diversos tratamentos, como:

Dados nulos

Dados duplicados

Features ou colunas categóricas

*Ou seja, preparar a base de dados para ser inserida em **modelo de aprendizado de máquina**.*

Com a realização desse curso, nós estaremos habilitados a realizar a limpeza e tratamento de qualquer base de dados, visando torná-la mais consistente e pronta para ser utilizada em análises futuras.

É importante ressaltar que o curso não tem o intuito de nos auxiliar no desenvolvimento de um modelo de machine learning, mas sim em preparar a base de dados para que ela possa ser utilizada em tais projetos de forma mais precisa e eficiente.

Em suma, o curso é voltado para o processo de limpeza e tratamento dos dados, com o objetivo de deixá-los prontos para o desenvolvimento de modelos de machine learning.

Pré-requisitos para realizar este curso

Para melhor aproveitarmos o curso, é necessário que possuamos conhecimentos prévios em:

Python

Bibliotecas Numpy e Pandas

Agrupamento, seleção e manipulação de dados.

Iremos analisar como faremos para limpar e tratar nossos dados.

Vamos lá?

Conhecendo os dados - Entendendo o problema

0:00 / 8:12



Como cientistas de dados, recebemos a seguinte tarefa:

Limpar a base de dados para a construção de modelos de machine learning (ML).

A base de dados em questão diz respeito à **taxa de churn** de clientes em uma empresa de telecomunicações. Mas, afinal, o que é churn?

Churn

Ocorre quando clientes deixam de **fazer negócios** com uma empresa

Representa a perda de receita

Pode ser um **indicador de satisfação** de clientes em relação aos **produtos/serviços**.

O termo *churn* é amplamente utilizado no mundo dos negócios e se refere à situação em que as pessoas **deixam de fazer negócios** com uma empresa. Como cientistas de dados, sabemos que o *churn* está diretamente relacionado à **receita da empresa**, já que quanto mais clientes deixam de fazer negócios, mais dinheiro a empresa perde.

Além disso, o *churn* pode ser utilizado como um **indicador de satisfação do cliente**, pois se muitas pessoas deixam de fazer negócios, pode indicar que o serviço oferecido não está atendendo às expectativas dos clientes.

Estratégias para reduzir a taxa de churn

Melhorar a qualidade de produtos e serviços

Oferecer suporte de qualidade a clientes

Criar programas de fidelidade

Por isso, as empresas buscam alternativas para baixar o churn, oferecendo serviços de maior qualidade, suporte mais rápido e com melhor qualidade, criando programas de fidelidade, entre outras estratégias.

Qual base de dados vamos tratar?

O modelo que iremos construir a partir da base de dados que nos foi enviada é destinado a **prever se um determinado cliente cancelará ou não** o serviço da empresa de telecomunicações.

Com base nessa previsão, poderemos elaborar estratégias para reduzir o churn. No entanto, a base de dados que recebemos não está pronta para ser utilizada diretamente em modelos de *machine learning*. Nossa tarefa é exatamente limpá-la e tratá-la para esse propósito.

Vamos aprender como fazer isso?

Começando o tratamento da base de dados

Vamos utilizar o **Google Colab** como ambiente de desenvolvimento deste curso.

Optamos por essa ferramenta porque ela apresenta diversas vantagens, como a facilidade de uso e a dispensa de preocupações com instalação de bibliotecas.

Caso não conheça, temos um artigo sobre na Alura:

[Google Colab: o que é, tutorial de como usar e criar códigos](#)

O nome que o instrutor deu para esse notebook foi Limpeza_tratamento_churn.

Precisamos procurar a base de dados e trazê-la para o nosso ambiente de desenvolvimento. Para realizar isso, vamos clicar no quarto ícone de pasta referente aos **arquivos** na barra lateral esquerda do Colab, esperar alocar uma máquina e quando aparecer uma pasta chamada de sample-data, entenderemos que foi alocada.

Depois, vamos clicar no primeiro ícone de pasta referente ao *upload* para o armazenamento da sessão e na nossa máquina selecionamos o arquivo dataset-telecon. Na sequência clicamos no botão "Abrir".

No Colab, será exibido um pop-up com a seguinte mensagem:

AVISO

Confira se os arquivos foram salvos em outro lugar. Os arquivos deste ambiente de execução vão ser excluídos quando ele for encerrado.

Basta clicarmos no botão "Ok", localizado no canto inferior direito do pop-up.

Observe que do lado esquerdo, em "Arquivos", abaixo da pasta sample-data agora temos outra pasta nomeada de dataset-telecon.json. Agora, precisamos instalar a ferramenta que iremos utilizar para trabalhar com esses dados, que é o **Pandas**.

Para instalar o Pandas, na primeira célula escrevemos o seguinte comando:

```
import pandas as pd
```

Sendo "pd" o apelido da biblioteca. Podemos executar essa célula teclando "Shift + Enter" ou clicando no botão de *play* "►" do lado esquerdo da célula. Ao lado do botão de *play*, aparecerá um *check* na cor verde, indicando que a célula foi executada com sucesso.

Na segunda célula, **faremos a leitura desses dados**. Ao analisar o nome do arquivo dataset-telecon.json, identificamos que o formato é JSON. Felizmente, o Pandas nos oferece formas de realizar essa leitura em arquivos desse tipo. Para tal, usamos o seguinte comando:

```
dados_churn = pd.read_json("dataset-telecon.json")
```

Entre parênteses e aspas duplas, está o **caminho** do arquivo. Como o arquivo está na pasta raiz, basta copiar o nome dele e damos o nome dados_churn para os dados que vamos ler. Em seguida, na próxima linha da célula, utilizamos dados_churn.head() para visualizar as primeiras cinco amostras e executamos a célula teclando "Shift + Enter".

Como retorno, obtemos a tabela:

#	id_cliente	Churn	cliente	telefone
0	0002-ORFBO	nao	{"genero':'feminino','idoso':0,'parceiro':...}	{"servico_t
1	0003-MKNFE	nao	{"genero':'masculino','idoso':0,'parceiro':...}	{"servico_t
2	0004-TLHLJ	sim	{"genero':'masculino','idoso':0,'parceiro':...}	{"servico_t
3	0011-IGKFF	sim	{"genero':'masculino','idoso':1,'parceiro':...}	{"servico_t
4	0013-EXCHZ	sim	{"genero':'feminino','idoso':1,'parceiro':...}	{"servico_t

A primeira coisa que notamos é que há 6 colunas:

id_cliente

Churn

cliente

telefone

internet

conta

As colunas `id_cliente` e `Churn` vieram em um formato padrão, mas as outras colunas vieram em um formato que parece ser um JSON aninhado. Não conseguimos trabalhar com os dados dessa forma no Google Colab usando o Pandas, ou seja, não podemos obter muitos insights a partir dessas informações. Então, precisamos transformar esses dados em um formato que possamos trabalhar.

Para visualizarmos melhor esse conteúdo, vamos imprimir a primeira amostra da coluna `conta`. Para isso, na próxima célula digitamos o comando `dados_churn['conta'][0]`, ou seja, selecionamos a coluna `conta` e a primeira amostra.

```
dados_churn['conta'][0]
```

Como retorno, obtemos:

```
{'contrato': None,  
 'faturamento_eletronico': None,  
 'metodo_pagamento': None,  
 'cobranca': {'mensal': None, 'Total': None}}
```

São as mesmas informações que temos na nossa coluna. Abrimos um JSON que possui a chave **contrato** com valor nulo, e as chaves **faturamento eletronico** e **método pagamento** também com valor nulo. Dentro da chave **cobrança**, temos outra chave chamada **mensal** e uma chamada **total**.

Dessa forma, não conseguimos trabalhar de forma eficaz com esses dados. Felizmente, nós podemos contar com a ajuda do Pandas, que possui um método capaz de nos auxiliar a fazer essa normalização no formato de colunas desejado, o `json_normalize()`.

Método `json_normalize()`

Na próxima célula, usaremos esse método passando a coluna que desejamos normalizar como argumento para `pd.json_normalize()`. Vamos selecionar a coluna de conta (`dados_churn['conta']`) e, em seguida, adicionar `.head()` para visualizar as cinco primeiras amostras.

```
pd.json_normalize(dados_churn['conta']).head()
```

Como retorno, obtemos um formato melhor para a visualização:

#	contrato	faturamento_eletronico	metodo_pagamento	cobranca.mensalidade
0	None	None	None	NaN
1	mes a mes	nao	cheque pelo correio	59.9
2	mes a mes	sim	cheque eletronico	73.9
3	mes a mes	sim	cheque eletronico	98.0
4	mes a mes	sim	cheque pelo correio	83.9

Observamos que cada chave do JSON se tornou uma coluna, totalizando 5 colunas. Além disso, existem outras colunas que possuem chaves dentro, como é o caso da coluna de telefone.

Podemos copiar o código que aplicamos na coluna conta e ajustar para a telefone:

```
pd.json_normalize(dados_churn['telefone']).head()
```

Como retorno, obtemos:

#	servico_telefone	varias_linhas
0	sim	nao
1	sim	sim
2	sim	nao
3	sim	nao
4	sim	nao

Conseguimos normalizar as informações da coluna de telefone e transformá-las em colunas separadas, mas isso nos trouxe um problema: se tivéssemos muito mais dados, **como poderíamos organizar tudo em colunas?**

Até mesmo no conjunto de dados da primeira tabela que geramos, se quisermos um *dataframe* totalmente normalizado em uma tabela que possamos trabalhar, **precisamos normalizar todas as colunas, concatenar** as novas colunas e, em seguida, **remover as colunas originais**. Só assim teríamos um *dataframe* pronto para ser trabalhado no Pandas.

Será que não há uma forma melhor de aplicarmos isso? É isso que aprenderemos nos próximos vídeos.

Conhecendo os dados - Transformando dados em uma tabela

0:00 / 6:06

Relembrando

Aprendemos como aplicar o método `json_normalize()` em nossos *dataframes* para obter colunas com valores em formato JSON normalizado. Isso nos permite expandir as chaves que contêm os valores em nossos *dataframes* em várias colunas. No entanto, essa abordagem nos apresenta uma dificuldade: ter que fazer isso manualmente para cada coluna.

Vamos aprender uma forma "pythonica" de aplicar isso de maneira mais automatizada.

"Pythonico" é um código que segue as diretrizes e filosofia da linguagem Python, sendo fácil de ler, escrever e manter, além de ser eficiente e expressivo.

Método `json_normalize()`

Podemos aplicar o método `json_normalize()` em um *dataframe* ou em um objeto JSON. A principal diferença é que, ao normalizar uma coluna específica, aplicamos o método no *dataframe*. Já ao normalizar todo o JSON de uma vez, aplicamos o método no objeto JSON.

Em um DataFrame → normaliza apenas uma coluna

Em um objeto JSON → normaliza todas as colunas aninhadas de uma só vez

É desta última forma que vamos aprender a fazer. Vamos utilizar o Google Colab!

Transformando dados em uma tabela

O instrutor inseriu uma célula em markdown com o título "Transformando dados em uma tabela". Isso para deixar o projeto mais organizado.

Primeiro, vamos **importar a biblioteca JSON**. Para isso, rodamos ("Shift + Enter") o seguinte comando na primeira célula dessa nova seção:

```
import json
```

Com essa biblioteca, podemos trabalhar mais facilmente com objetos JSON, convertendo objetos Python em JSON e vice-versa.

Com a biblioteca importada em nosso ambiente, vamos agora abrir o arquivo dataset-telecon.json em nosso ambiente de desenvolvimento. Para realizar isso, utilizaremos a palavra reservada do Python **with**, que garante que o arquivo será fechado após o uso da instrução.

Método with()

Para utilizar o método with(), na próxima célula, digitaremos with open(""), passando o nome do arquivo dataset-telecon.json, e, em seguida, criaremos uma variável nomeada como f utilizando o termo as.

```
with open("dataset-telecon.json") as f:
```

Nessa mesma célula, criaremos uma variável chamada json_bruto para armazenar nosso JSON original, utilizando o método json.load(f) e passando f, que é o arquivo que está sendo aberto com o método open().

```
with open("dataset-telecon.json") as f:  
    json_bruto = json.load(f)
```

Teclamos "Shift + Enter" para rodar a célula.

Agora vamos **visualizar o conteúdo** da variável `json_bruto` digitando somente isso na célula e teclando mais uma vez "Shift + Enter".

```
json_bruto
```

Como retorno, obtemos a seguinte estrutura do conjunto de dados:

A estrutura abaixo foi parcialmente transcrita. Para conferi-la na íntegra, execute o código na sua máquina.

```
{'id_cliente': '1399-UBQIU',
 'Churn': 'nao',
 'cliente': {'genero': 'masculino'},
 'idoso': 0,
 'parceiro': 'nao,' ,
 'tempo_servico': 11},
 'Telefone': {'servico_telefone': 'sim', 'varias_linhas': 'nao'},
 'Internet': {'servico_internet': 'DSL', 'Seguranca_online': 'nao'}

// estrutura de retorno omitida

}
```

Temos o `id_cliente` e o `Churn`, além disso temos a estrutura JSON de `cliente` sendo aberta, cujo valor é um novo JSON. Também temos a chave `telefone` que é outro novo JSON, e assim por diante.

Com isso, notamos que temos vários **JSONs aninhados**. Percebemos também que já temos o JSON como objeto do Python, e agora precisamos passar esse objeto para o método `json_normalize()`. Vamos clicar no ícone de xis da estrutura, localizado no canto esquerdo e superior para limpar a saída do comando.

Na próxima célula, digitamos `pd.json_normalize()`, mas em vez de passarmos uma coluna do DataFrame, passamos `json_bruto`, que é nosso objeto JSON. Vamos salvar tudo isso em uma variável chamada `dados_normalizados`. Também adicionamos na mesma célula o comando `dados_normalizados.head()` para visualizar apenas as primeiras cinco amostras.

```
dados_normalizados = pd.json_normalize(json_bruto)
dados_normalizados.head()
```

Ao executarmos, obtemos a seguinte tabela com 5 linhas e 21 colunas:

A tabela abaixo foi parcialmente transcrita. Para conferi-la na íntegra, execute o código na sua máquina.

#	id_cliente	Churn	cliente.genero	cliente.idoso	cliente.parceiro	
0	0002-ORFBO	nao	feminino	0	sim	
1	0003-MKNFE	nao	masculino	0	nao	
2	0004-TLHLJ	sim	masculino	0	nao	
...	

Ao analisarmos os dados normalizados, notamos que as primeiras colunas são as mesmas do objeto JSON original, como `id_cliente` e `Churn`. Porém, a partir da terceira coluna, há diferenças significativas. Por exemplo, a coluna `cliente.genero` era uma chave do JSON de cliente original que continha outro JSON dentro dela.

Agora, ela é uma coluna por si só, sendo que o valor correspondente é extraído do JSON interno. Isso se repete para as outras colunas que antes estavam aninhadas em outros JSONs.

Conclusão

Com isso, normalizamos o nosso JSON todo de uma vez, sem precisar aplicar coluna por coluna. Com essa estrutura, podemos trabalhar melhor com os dados, obter insights, gerar novos dados e realizar a tratativa necessária. A partir de agora, faremos alguns tratamentos iniciais nessa base de dados.

Espero você no próximo vídeo!

Sobre o curso Pandas: limpeza e tratamento de dados

O [curso Pandas: limpeza e tratamento de dados](#) possui **125 minutos de vídeos**, em um total de **52 atividades**. Gostou? Conheça nossos outros [cursos de Data Science](#) em [Data Science](#), ou leia nossos [artigos de Data Science](#).

Matricule-se e comece a estudar com a gente hoje! Conheça outros tópicos abordados durante o curso:

Conhecendo os dados

Transformação inicial dos dados

Utilizando dados duplicados e nulos

Lidando com os outliers

Trabalhando com variáveis categóricas

Escolha a duração do seu plano

1 ano **2 anos** ATÉ 30% OFF

O maior desconto!

PLUS

Impulsiona a sua carreira com os melhores cursos e faça parte da maior comunidade tech.

12x **R\$109**

1 ano de Alura

À vista R\$1.308

MATRICULE-SE

Assine o PLUS e garanta:

- Acesso a TODOS os cursos por 1 ano
- Certificado
- Mentorias em grupo com especialistas
- Comunidade exclusiva
- Acesso ao conteúdo das Imersões
- App Android e iOS para estudar onde quiser

PRO

Acelere o seu aprendizado com a IA da Alura e prepare-se para o mercado internacional.

12x **R\$149**

1 ano de Alura

À vista R\$1.788

MATRICULE-SE

Todos os benefícios do PLUS e mais vantagens exclusivas:

Luri, a inteligência artificial da Alura

Alura Língua - Inglês e Espanhol

Preço especial de lançamento 

ULTRA LAB

Para estudantes ultra comprometidos atingirem seu objetivo mais rápido.

R\$249**12x R\$229,08****1 ano** de Alura

OFERTA LANÇAMENTO

À vista R\$2.748,96

MATRICULE-SE**Todos os benefícios do PRO e mais vantagens exclusivas:****Luri**, com mensagens ILIMITADAS**Luri Vision**, a IA que enxerga suas dúvidas**6 Ebooks** da Casa do Código**Talent Lab**

Pague com cartão de crédito em até 12x, PayPal, NuPay ou Pix

Garantimos cancelamento gratuito em até 7 dias

[Conheça os Planos para Empresas](#)

Nossas redes e apps



Institucional

- Sobre nós
- Carreiras Alura
- Para Empresas
- Para Sua Escola
- Política de Privacidade
- Compromisso de Integridade
- Termos de Uso
- Documentos Institucionais
- Status
- Uma empresa do grupo Alun

A Alura

- Como Funciona
- Formações
- Plataforma
- Depoimentos
- Instrutores(as)
- Dev em <T>
- Luri, a inteligência artificial da Alura
- IA Conference 2025
- Cursos imersivos
- Certificações

Conteúdos

- Alura Cases
- Imersões
- Artigos
- Podcasts
- Artigos de educação corporativa
- Imersão Dados com Python

Fale Conosco

- Email e telefone
- Perguntas frequentes

Novidades e Lançamentos

bins.br@gmail.com

CURSOS

Cursos de Programação

Lógica | Python | PHP | Java | .NET | Node JS | C | Computação | Jogos | IoT

Cursos de Front-end

HTML, CSS | React | Angular | JavaScript | jQuery

Cursos de Data Science

Ciência de dados | BI | SQL e Banco de Dados | Excel | Machine Learning | NoSQL | Estatística

Cursos de Inteligência Artificial

IA para Programação | IA para Dados

Cursos de DevOps

AWS | Azure | Docker | Segurança | IaC | Linux

Cursos de UX & Design

Usabilidade e UX | Vídeo e Motion | 3D

Cursos de Mobile

Flutter | iOS e Swift | Android, Kotlin | Jogos

Cursos de Inovação & Gestão

Métodos Ágeis | Softskills | Liderança e Gestão | Startups | Vendas

CURSOS UNIVERSITÁRIOS FIAP

Graduação | Pós-graduação | MBA