escola britânica de artes criativas & tecnologia

Aula | Análise Exploratória de Dados com Python: State of Data Brazil

Professor **André Perez** (LinkedIn)

Tópicos

1. Introdução;
2. Exploração;
3. Processamento;
4. Agrupamentos e Agregações;
5. Visualização;
6. Finalizando o projeto;
7. Divulgação.

+ Code + Text

Aulas

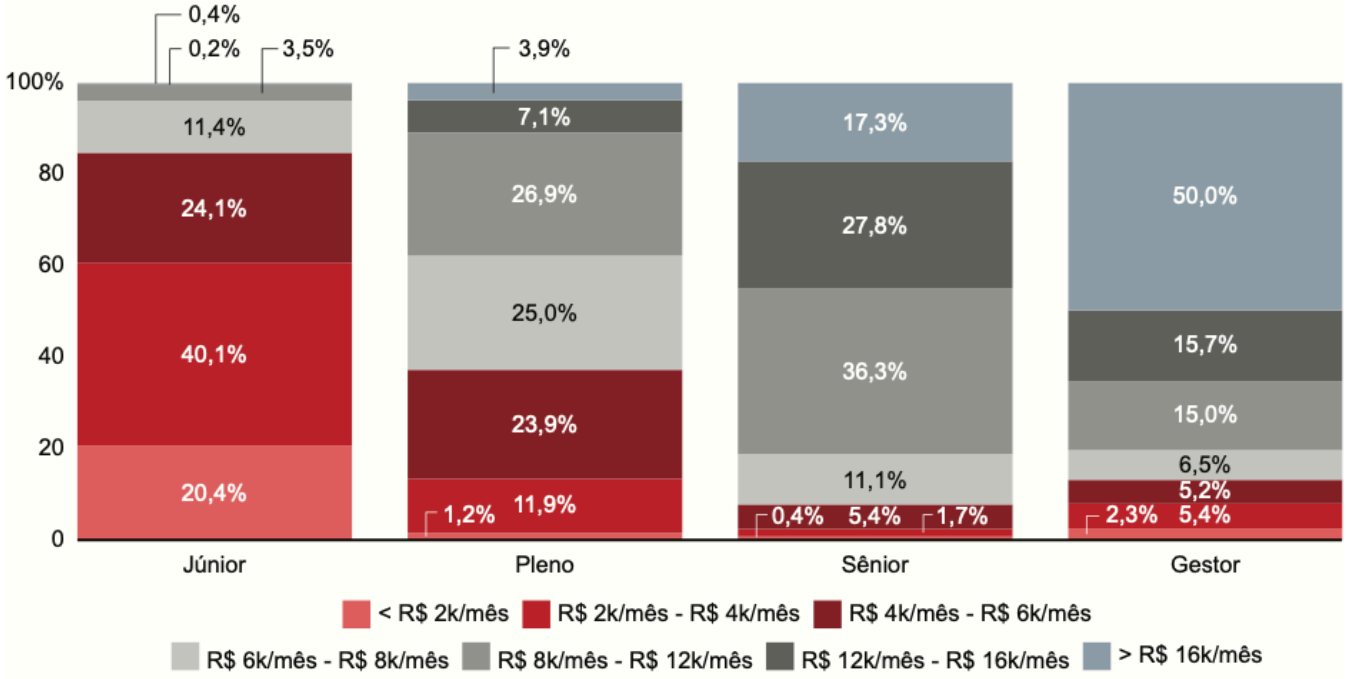
1. Introdução

Nesta aula, vamos entender o contexto de negócio que estamos inseridos e aprender a utilizar o Google Colab.

1.1. Contexto

O **State of Data Brazil** (link), realizada pelo Data Hackers e pela Bain & Company, é uma das maiores pesquisas sobre o mercado brasileiro de dados. Dentre os assuntos pesquisado na última pesquisa (2022), foi mapeado o perfil atual das três maiores profissões da área de dados do Brasil: **engenheiro** de dados, **cientista** de dados e **analista** de dados. O relatório completo pode ser acessado neste link.

> **Figura 24**: Remuneração em 2022 por nível de cargo.



Nota: Desconsidera respondentes que se consideram desempregados, apenas estudantes e que preferiram não informar a situação atual de trabalho
Fonte: Pesquisa State of Data Brasil 2022

1.2. Google Colab

> Ferramenta web autogerênciada de cadernos (*notebooks*).

Ferramenta web

- Crie uma conta Google em gmail.com;
- Acesse o Google Colab através do endereço colab.research.google.com.

Autogerênciada

- A Google provisiona uma máquina virtual para você;
- A máquina virtual dura no máximo 12h.

Cadernos (*notebooks*)

Um **caderno** é um documento *web* composto por um conjunto de elementos (células) de texto e código:

- Células de **texto** podem ser editados com o editor da ferramenta, HTML ou Markdown;
- Células de **código** são exclusivamente para a linguagem de programação Python.

```
print("olá, mundo!")
```

2. Exploração

Nesta aula vamos explorar os dados da pesquisa.

2.1. Dados

Os dados da pesquisa foram disponibilizados no site do Kaggle (link). O arquivo está no formato **CSV** (do inglês valores separados por vírgula), possuí **353 colunas** e **4271 linhas** e "pesa" por volta de **10 MB**.

2.2. Upload

Acesse o Kaggle (link), faça o *download* do arquivo no formato CSV e então, o seu *upload* no ambiente do Google Colab.

2.3. Exploração

Na etapa de exploração, encontrar as colunas de interesse: **cargo**, **experiência** e **salário**. Vamos começar extraindo o cabeçalho do arquivo como um texto, ou *string* em Python.

```
with open(file="State_of_data_2022.csv", mode="r") as fp:
  header = fp.readline()
```

```
type(header)
```

```
print(header[0:200])
```

Agora, vamos transformar o texto do cabeçalho em uma lista de textos e listar apenas os elementos que representam as perguntas da segunda parte (carreira), ou seja, aqueles elementos de texto que apresentam as letras **P2**.

```
header = header.split(sep='","')
```

```
type(header)
```

```
print(header[1])
```

```
for column, column_name in enumerate(header):
  if 'P2' in column_name:
    print(str(column) + ' ' + column_name)
```

Perfeito, encontramos as colunas de interesse:

Info	Coluna	Coluna (nome)
Cargo	21	('P2_f ', 'Cargo Atual')
Experiência	22	('P2_g ', 'Nivel')
Salário	23	'P2_h ', 'Faixa salarial'

Podemos conferir o tipo do dado presente em cada coluna.

```
with open(file="State_of_data_2022.csv", mode="r") as fp:
  header = fp.readline()

  for _ in range(10):

    line = fp.readline() # linhas como texto
    line = line.split(sep=',') # linha como lista
    print({"Cargo": line[21], "Experiencia": line[22], "Salário": line[23]})
```

3. Processamento

Nesta aula vamos selecionar e limpar as colunas de interesse.

```
data = list()

with open(file="State_of_data_2022.csv", mode="r") as fp:
  header = fp.readline()
  line = fp.readline() # linhas como texto

  while line:
    line = line.split(sep=',') # linha como lista
    data.append({"Cargo": line[21], "Experiencia": line[22], "Salário": line[2
    line = fp.readline() # linhas como texto
```

```
for datum in data[0:10]:
  print(datum)
```

```
len(data)
```

3.1. Pandas

Vamos utilizar o `Pandas` (documentação), o pacote **Python** mais utilizado para processamento de dados.

```
import pandas as pd
```

3.2. DataFrame

A abstração base do `Pandas` é o `DataFrame`, uma estrutura de dados que representa uma tabela de duas dimensões.

```
data_table = pd.DataFrame(data)
```

```
data_table.head(n=15)
```

```
data_table.shape
```

```
data_table.to_csv("data.csv", header=True, index=False)
```

3.3. Limpeza

Uma boa análise de dados só é possível com dados consistentes, portanto a limpeza do conjunto de dados se faz necessária.

- **Linhas vazias**

Existem diversas estratégias para lidar com dados faltantes, como a inputação ("forçar valor") ou a deleção (remover a linha toda). Vamos utilizar a última.

```
data_table = data_table.replace('', pd.NA)
```

```
data_table.head(n=15)
```

```
data_table = data_table.dropna()
```

```
data_table.head(n=15)
```

```
data_table.shape
```

- **Linhas corrompidas**

Linhas corrompidas podem derivar de diversos fatores: erros de processamento, erros humanos, etc. Podemos manual ou automaticamente corrigi-las, ou ainda aplicar a técnica de deleção vista anteriormente. Vamos utilizar a última.

```
data_table['Salário'] = data_table['Salário'].apply(lambda row: row if 'R$' in
```

```
data_table.head(n=15)
```

```
data_table = data_table.dropna()
```

```
data_table.head(n=15)
```

```
data_table.shape
```

- **Linhas de interesse**

A seleção dos dados de interesse também pode ocorrer a nível de linha. Vamos selecionar apenas as proofissões de interesse: **analista**, **cientista** e **engenheira** de dados.

```
data_table['Cargo'].unique()
```

```
positions = ['Analista de Dados/Data Analyst', 'Cientista de Dados/Data Scient
```

```
data_table['Cargo'] = data_table['Cargo'].apply(lambda row: row if row in posi
```

```
data_table.head(n=15)
```

```
data_table = data_table.dropna()
```

```
data_table.head(n=15)
```

```
data_table.shape
```

Para simplificar um pouco mais, podemos renomear as profissões.

```
labels = ['Analista', 'Cientista', 'Engenheira']
```

```
labels = dict(zip(positions, labels))
```

```
print(labels)
```

```
data_table['Cargo'] = data_table['Cargo'].apply(lambda row: labels[row])
```

```
data_table.head(n=15)
```

Para conferir a qualidade do conjunto, podemos listar os valores únicos presentes em cada coluna de interesse. Se os valores estiverem dentro do esperado, a etapa de limpeza de dados foi realizada com sucesso.

```
data_table['Cargo'].unique()
```

```
data_table['Experiencia'].unique()
```

```
data_table['Salário'].unique()
```

Por fim, pode-se salvar o conjunto de dados selecionado e limpo em um outro arquivo CSV.

```
data_table.to_csv("data.csv", header=True, index=False)
```

4. Agrupamentos e Agregações

Nesta etapa vamos agrupar os dados com uma operação matemática de soma em diversos níveis de agregação.

```
data_table.shape
```

```
data_table.head(15)
```

```
data_table['Quantidade'] = 1
```

```
data_table.head(15)
```

4.1. Por salário

No primeiro nível de agregação, vamos contar quantos profissionais estão em cada faixa de salário.

```
salario_agg = data_table[['Salário', 'Quantidade']].groupby('Salário').agg('su
```

```
salario_agg.head(15)
```

Uma operação comum sobre dados agregados é a ordenação. Como a coluna Salário não é numericamente ordenável (textos são ordenados por ordem alfabética da primeira letra), podemos manualmente definir uma coluna artificial de ordenamento.

```
salario_ordem = {
    'Menos de R$ 1.000/mês': 0,
    'de R$ 1.001/mês a R$ 2.000/mês': 1,
    'de R$ 2.001/mês a R$ 3.000/mês': 2,
    'de R$ 3.001/mês a R$ 4.000/mês': 3,
    'de R$ 4.001/mês a R$ 6.000/mês': 4,
    'de R$ 6.001/mês a R$ 8.000/mês': 5,
    'de R$ 8.001/mês a R$ 12.000/mês': 6,
    'de R$ 12.001/mês a R$ 16.000/mês': 7,
    'de R$ 16.001/mês a R$ 20.000/mês': 8,
    'de R$ 20.001/mês a R$ 25.000/mês': 9,
    'de R$ 25.001/mês a R$ 30.000/mês': 10,
    'de R$ 30.001/mês a R$ 40.000/mês': 11,
    'Acima de R$ 40.001/mês': 12
}
```

```
salario_agg['Ordem'] = salario_agg['Salário'].apply(lambda row: salario_ordem[
```

```
salario_agg.head(15)
```

Agora, ordena-se o salário corretamente.

```
salario_agg = salario_agg.sort_values(by='Ordem', ascending=True)
```

```
salario_agg.head(15)
```

4.2. Por salário e cargo

No segundo nível de agregação, vamos contar quantos profissionais estão em cada faixa de salário por cargo.

```
salario_cargo_agg = data_table[['Cargo', 'Salário', 'Quantidade']].groupby(['C
```

```
salario_cargo_agg['Ordem'] = salario_cargo_agg['Salário'].apply(lambda row: sa
```

```
salario_cargo_agg = salario_cargo_agg.sort_values(by=['Cargo', 'Ordem'], ascen
```

```
salario_cargo_agg.head(100)
```

A informação agregada permite a extração de *insights*. Por exemplo, vamos analisar o salário dos analistas de dados.

```
salario_cargo_agg[salario_cargo_agg['Cargo'] == 'Analista'].head(15)
```

4.3. Por salário, cargo e experiência

Por fim, no terceiro nível de agregação, vamos contar quantos profissionais estão em cada faixa de salário por cargo e nivel de experiência.

```
salario_cargo_experiencia_agg = data_table[['Cargo', 'Experiencia', 'Salário',
```

```
salario_cargo_experiencia_agg['Ordem'] = salario_cargo_experiencia_agg['Salári
```

```
salario_cargo_experiencia_agg = salario_cargo_experiencia_agg.sort_values(by=[
```

```
salario_cargo_experiencia_agg.head(15)
```

Vamos analisar mais a fundo o salário dos analistas de dados.

```
salario_cargo_experiencia_agg[salario_cargo_experiencia_agg['Cargo'] == 'Anali
```

5. Visualização

Nesta aula vamos visualizar os dados agregados.

```
salario_cargo_experiencia_agg[salario_cargo_experiencia_agg['Cargo'] == 'Anali
```

5.1. Seaborn

Vamos utilizar o `Seaborn` (documentação), um dos pacotes **Python** mais utilizado para visualização de dados.

```
import seaborn as sns
```

5.2. Por salário

No primeiro nível, vamos utilizar um gráfico de barras. No eixo **x** temos o **salário** (representado pela sua ordem) e no eixo **y**, a **quantidade** de profissionais. A cor das barras e a legenda também representam o **salário**.

```
salario_agg.head(15)
```

```
with sns.axes_style('whitegrid'):

  plot = sns.barplot(data=salario_agg, x='Ordem', y='Quantidade', hue='Salário
  plot.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

5.3. Por salário e cargo

No segundo nível de agregação, incrementa-se o gráfico anterior com a informação do **cargo**.

```
salario_cargo_agg.head(15)
```

```
with sns.axes_style('whitegrid'):

  plot = sns.barplot(data=salario_cargo_agg, x='Cargo', y='Quantidade', hue='S
  plot.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

5.3. Por salário, cargo e experiência

Por fim, no terceiro nível, aplicamos a mesma técnica do gráfico anterior, mas agora distribuídos por **experiência** (eixo **x**) e **cargo** (`DataFrame` dedicado).

- **Analista**

```
data = salario_cargo_experiencia_agg[salario_cargo_experiencia_agg['Cargo'] ==
```

```
with sns.axes_style('whitegrid'):
  plot = sns.barplot(data=data, x='Experiencia', y='Quantidade', hue='Salário'
  plot.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

- **Cientista**

```
data = salario_cargo_experiencia_agg[salario_cargo_experiencia_agg['Cargo'] ==
```

```
with sns.axes_style('whitegrid'):
  plot = sns.barplot(data=data, x='Experiencia', y='Quantidade', hue='Salário'
  plot.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

- **Engenheira**

```
data = salario_cargo_experiencia_agg[salario_cargo_experiencia_agg['Cargo'] ==
```

```
with sns.axes_style('whitegrid'):
  plot = sns.barplot(data=data, x='Experiencia', y='Quantidade', hue='Salário'
  plot.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

6. Finalizando o projeto

> Contar histórias (*storytelling*) de dados é a arte de apresentar dados com uma narrativa contextual.

6.1. Recomendação

- **Livro**: Storytelling com dados: um guia sobre visualização de dados para profissionais de negócios;
- **Autora**: Cole Nussbaumer Knaflic;
- **Link**: Amazon BR.

6.2. Dicas práticas

Estrutura (Notebook)

- **Introdução**: Contexto e objetivos.
- **Dados**: Descrição e carregamento;
- **Processamento**: Limpeza, filtros, etc.;
- **Insights**: Agregações, visualizações e resultados.

Texto

- Use o caracter `#` para criar sessões na tabela de conteúdo;
- Use *itálico* para palavras estrangeiras;
- Formate como `código` nome de variáveis, pacotes, etc: `Pandas`, `DataFrame`.
- Use ferramentas de *proofreading* (Browser, Grammarly, etc.).

Gráficos

- Adicione um título;
- Edite o nome dos eixos;
- Adicione uma legenda (abaixo) que torne o gráfico auto-explicativo.

7. Divulgação

Nesta aula vamos aprender a expor o nossa análise de dados.

7.1. Notebook

Vamos conferir o notebook finalizado.

7.2. Kaggle

Kaggle é a maior comunidade online de ciência de dados e aprendizado de máquina. A plataforma permite que usuários encontrem e publiquem **conjuntos de dados**, construam e compartilhem **notebooks** (como este do Google Colab) e participem de **competições** (que pagam muito dinheiro as vezes).

> Vamos publicar nosso **notebook** na plataforma web do Kaggle para que você possa compartilhar tudo o que você aprendeu nestas aulas e compor o seu portfólio.

- Crie uma conta no Kaggle;
- Faça o *download* do *notebook* do Google Colab;
- Faça o *upload* do *notebook* no Kaggle.