

3

Lab03 (EDA)

- O EDA (Exploratory Data Analysis) é uma etapa fundamental em qualquer projeto de análise de dados ou machine learning. Durante o EDA, você explora e visualiza os dados para entender melhor suas características, relacionamentos e possíveis padrões. Isso envolve a criação de gráficos, resumos estatísticos e a identificação de anomalias nos dados. Técnicas comuns de EDA:

1. **Resumo Estatístico:** Calcule estatísticas descritivas básicas, como média, mediana, desvio padrão, máximo, mínimo e quartis. Isso pode fornecer uma visão geral das propriedades numéricas dos dados.

2. **Visualizações Unidimensionais:** Histogramas: Representam a distribuição de uma variável numérica, dividindo-a em intervalos e contando a frequência de observações em cada intervalo.

Gráficos de barras: Adequados para variáveis categóricas, mostrando a contagem ou proporção de cada categoria.

3. **Visualizações Bidimensionais:** Gráficos de dispersão: Mostram a relação entre duas variáveis numéricas, ajudando a identificar padrões, tendências ou correlações.

Box plots: Exibem a distribuição de uma variável numérica em diferentes grupos ou categorias, destacando medidas de tendência central, dispersão e possíveis outliers.

4. **Exploração de Relações entre Variáveis:** Matriz de correlação: Calcula a correlação entre todas as combinações de variáveis numéricas, ajudando a identificar associações lineares entre elas.

Gráficos de pares: Mostram a relação entre todas as combinações de variáveis em um conjunto de dados, facilitando a identificação de padrões ou agrupamentos.

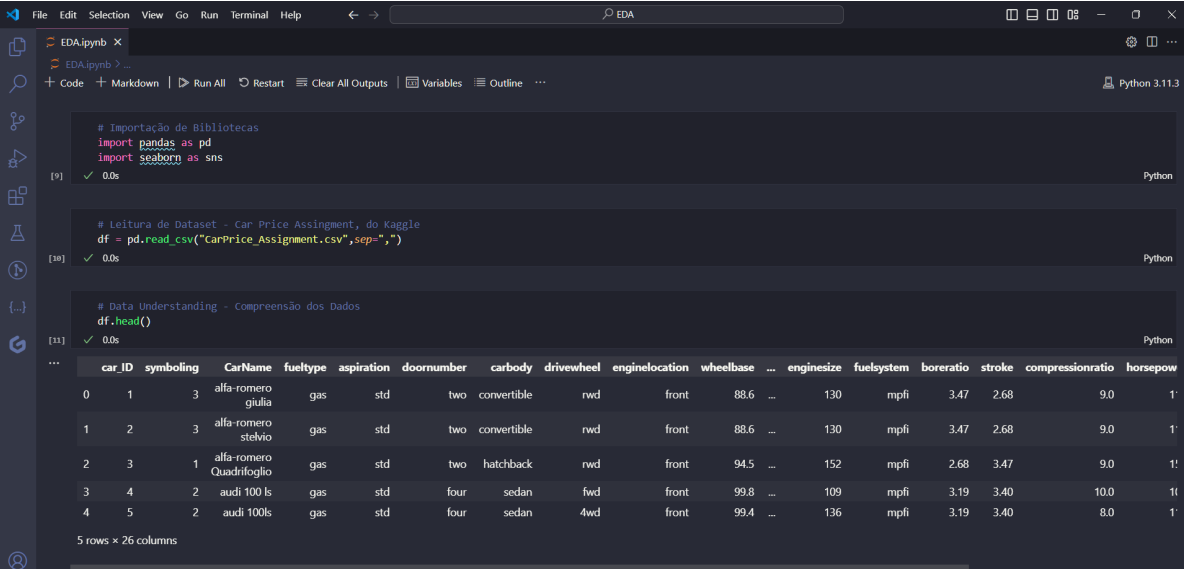
5. **Tratamento de Dados Ausentes ou Anômalos:** Identifique e lide com valores ausentes ou anômalos nos dados, seja removendo-os, imputando valores ou aplicando técnicas de limpeza de dados.

1. Leitura de Dataset:

- `df = pd.read_csv("CarPrice_Assignment.csv", sep=",")` : Lê um arquivo CSV chamado "CarPrice_Assignment.csv" e o carrega em um DataFrame do Pandas chamado `df`. O parâmetro `sep=", "` especifica que o separador no arquivo CSV é uma vírgula.

2. Data Understanding (Compreensão dos Dados):

- `df.head()` : Exibe as primeiras linhas do DataFrame `df`, permitindo uma rápida inspeção visual dos dados.



```
# Importação de Bibliotecas
import pandas as pd
import seaborn as sns

# Leitura de Dataset - Car Price Assignment, do Kaggle
df = pd.read_csv("CarPrice_Assignment.csv", sep=",")

# Data Understanding - Compreensão dos Dados
df.head()
```

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	engineLocation	wheelbase	...	engineSize	fuelSystem	boreRatio	stroke	compressionRatio	horsepow
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1'
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	1'
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	1!
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	1!
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	1'

5 rows x 26 columns

- `df.info()` : Fornece informações sobre o DataFrame, incluindo o número de entradas, o tipo de dados de cada coluna e a quantidade de memória usada.

```
df.info()
✓ 0.0s

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
#   Column                Non-Null Count  Dtype  
---  -
0   car_ID                205 non-null   int64  
1   symboling             205 non-null   int64  
2   CarName               205 non-null   object  
3   fueltype              205 non-null   object  
4   aspiration             205 non-null   object  
5   doornumber            205 non-null   object  
6   carbody               205 non-null   object  
7   drivewheel            205 non-null   object  
8   enginelocation         205 non-null   object  
9   wheelbase             205 non-null   float64 
10  carlength             205 non-null   float64 
11  carwidth              205 non-null   float64 
12  carheight             205 non-null   float64 
13  curbweight            205 non-null   int64  
14  enginetype            205 non-null   object  
15  cylindernumber        205 non-null   object  
16  enginesize            205 non-null   int64  
17  fuelsystem            205 non-null   object  
18  boreratio             205 non-null   float64 
19  stroke                205 non-null   float64 
...
24  highwaympg           205 non-null   int64  
25  price                205 non-null   float64 
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

- `df.describe()` : Gera estatísticas descritivas para colunas numéricas, como média, desvio padrão, quartis, etc.
- `df.describe(include='O')` : Gera estatísticas descritivas para colunas categóricas.

```
df.describe()
✓ 0.0s
Python
```

	car_ID	symboling	wheelbase	carlength	carwidth	carheight	curbweight	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	3.329756	3.255415	10.142537	104.117073	5125.121951	25.219512	30.75122
std	59.322565	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	0.270844	0.313597	3.972040	39.544167	476.985643	6.542142	6.88644
min	1.000000	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000	4150.000000	13.000000	16.00000
25%	52.000000	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	3.150000	3.110000	8.600000	70.000000	4800.000000	19.000000	25.00000
50%	103.000000	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000	5200.000000	24.000000	30.00000
75%	154.000000	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	3.580000	3.410000	9.400000	116.000000	5500.000000	30.000000	34.00000
max	205.000000	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	288.000000	6600.000000	49.000000	54.00000

```
df.describe(include='O')
✓ 0.0s
Python
```

	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	enginetype	cylindernumber	fuelsystem
count	205	205	205	205	205	205	205	205	205	205
unique	147	2	2	2	5	3	2	7	7	8
top	toyota corona	gas	std	four	sedan	fwd	front	ohc	four	mpfi
freq	6	185	168	115	96	120	202	148	159	94

- `list(df.carbody.unique())` : Lista os valores únicos na coluna 'carbody'.
- `print(df['carbody'].value_counts())` : Conta o número de ocorrências de cada valor na coluna 'carbody'.
- `print(df['fuelsystem'].value_counts())` : Conta o número de ocorrências de cada valor na coluna 'fuelsystem'.

```
list(df.carbody.unique())
✓ 0.0s
['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop']

print(df['carbody'].value_counts())
✓ 0.0s
carbody
sedan      96
hatchback  70
wagon      25
hardtop     8
convertible 6
Name: count, dtype: int64

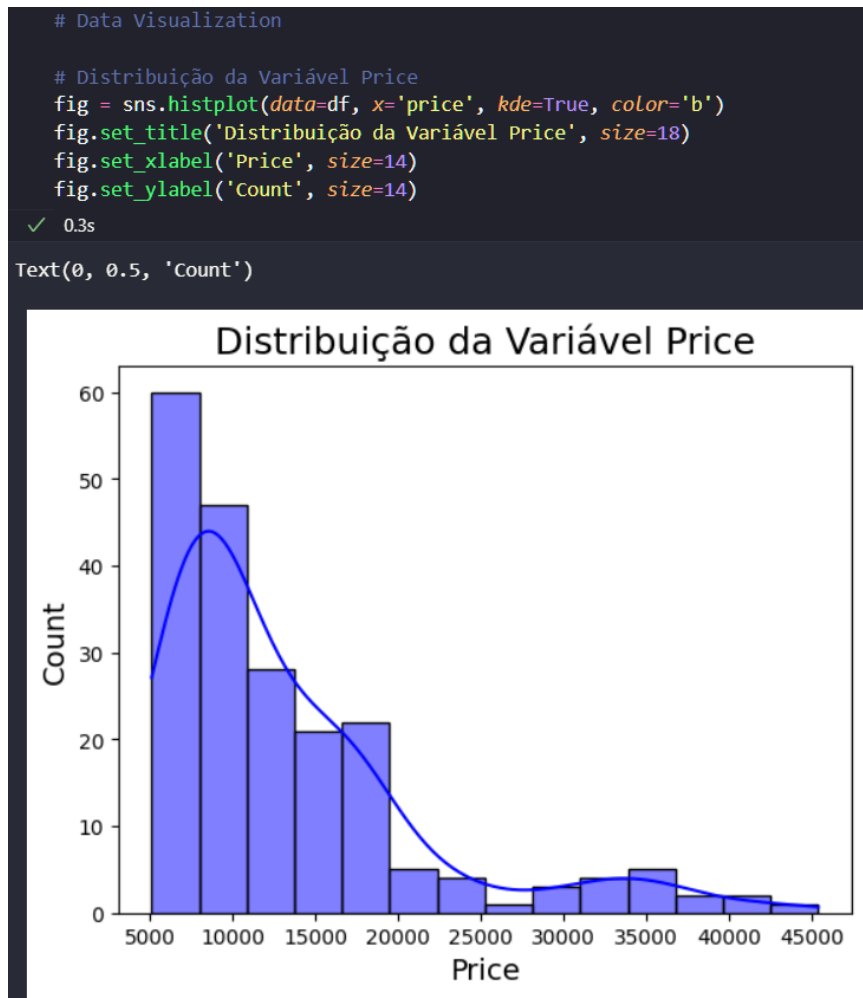
print(df['fuelsystem'].value_counts())
✓ 0.0s
fuelsystem
mpfi      94
2bbl      66
idi       20
1bbl      11
spdi       9
4bbl       3
mfi        1
spfi        1
Name: count, dtype: int64
```

3. Data Preparation (Preparação dos Dados):

- `df.isnull().sum()` : Calcula o total de valores nulos em cada coluna do DataFrame.
- `df[df.duplicated(keep='first')]` : Identifica linhas duplicadas no DataFrame.
 - `df.drop_duplicates(keep='first', inplace=True)` : remove linhas duplicadas, se houver

4. Data visualization (visualização de dados)

- `fig = sns.histplot(data=df, x='price', kde=True, color='b')` : Este trecho cria um histograma da variável 'price' usando a função `histplot` da biblioteca Seaborn. O parâmetro `data=df` especifica o DataFrame a ser utilizado, `x='price'` indica que os valores da variável 'price' serão plotados no eixo x do histograma. O parâmetro `kde=True` adiciona uma estimativa de densidade do kernel para suavizar a distribuição, e `color='b'` define a cor do gráfico como azul.
- `fig.set_title('Distribuição da Variável Price', size=18)` : Define o título do gráfico como "Distribuição da Variável Price" com um tamanho de fonte de 18.
- `fig.set_xlabel('Price', size=14)` : Define o rótulo do eixo x como "Price" com um tamanho de fonte de 14.
- `fig.set_ylabel('Count', size=14)` : Define o rótulo do eixo y como "Count" com um tamanho de fonte de 14.

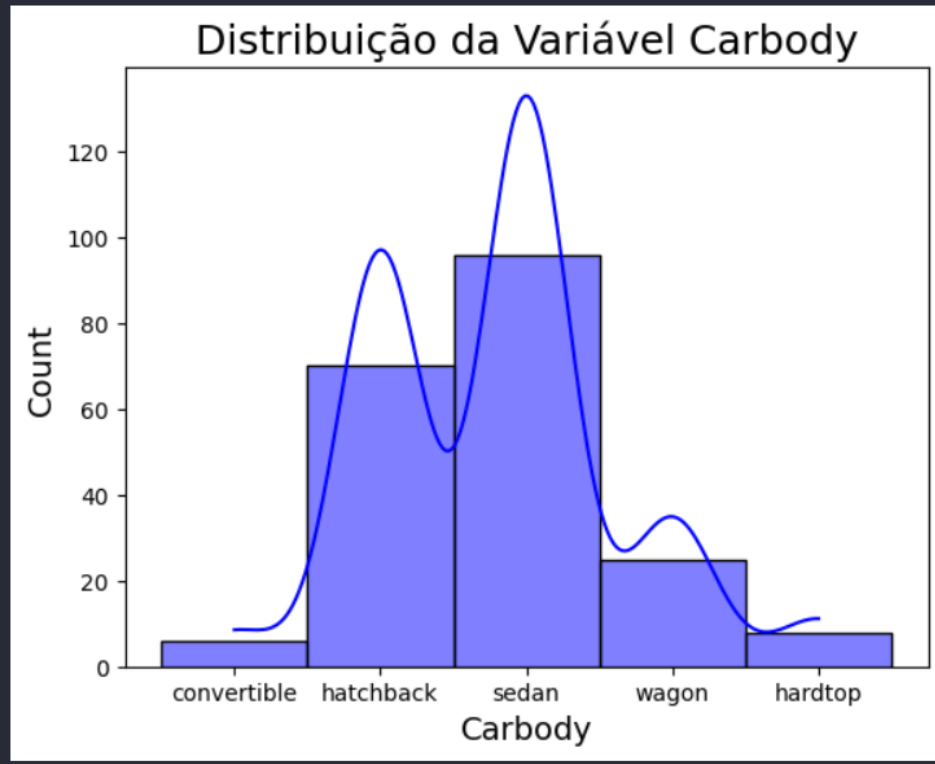


- Este trecho é semelhante ao anterior, mas agora estamos criando um histograma para a variável 'carbody'. As funções são as mesmas, apenas os nomes das variáveis e o título do gráfico são diferentes.

```
# Distribuição da Variável Carbody
fig = sns.histplot(data=df, x='carbody', kde=True, color='b')
fig.set_title('Distribuição da Variável Carbody', size=18)
fig.set_xlabel('Carbody', size=14)
fig.set_ylabel('Count', size=14)
```

✓ 0.1s

Text(0, 0.5, 'Count')

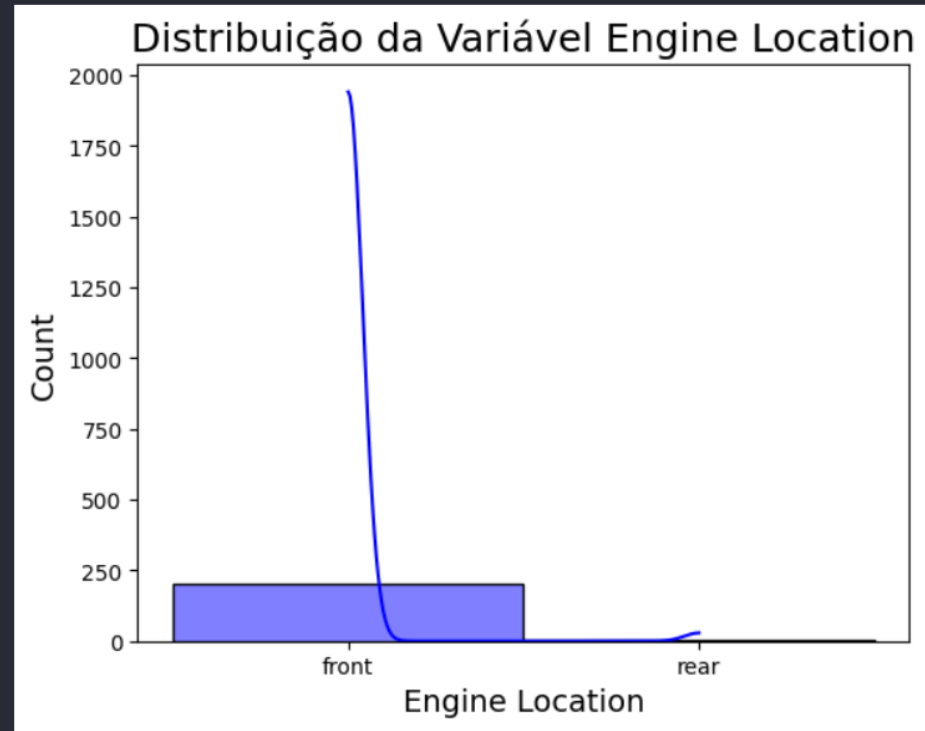


- Novamente, este trecho é semelhante aos anteriores, mas agora estamos criando um histograma para a variável 'engineloation'. As funções e parâmetros são os mesmos, apenas os nomes das variáveis e o título do gráfico são diferentes.

```
# Distribuição da Variável EngineLocation
fig = sns.histplot(data=df, x='engineLocation', kde=True, color='b')
fig.set_title('Distribuição da Variável Engine Location', size=18)
fig.set_xlabel('Engine Location', size=14)
fig.set_ylabel('Count', size=14)
```

✓ 0.1s

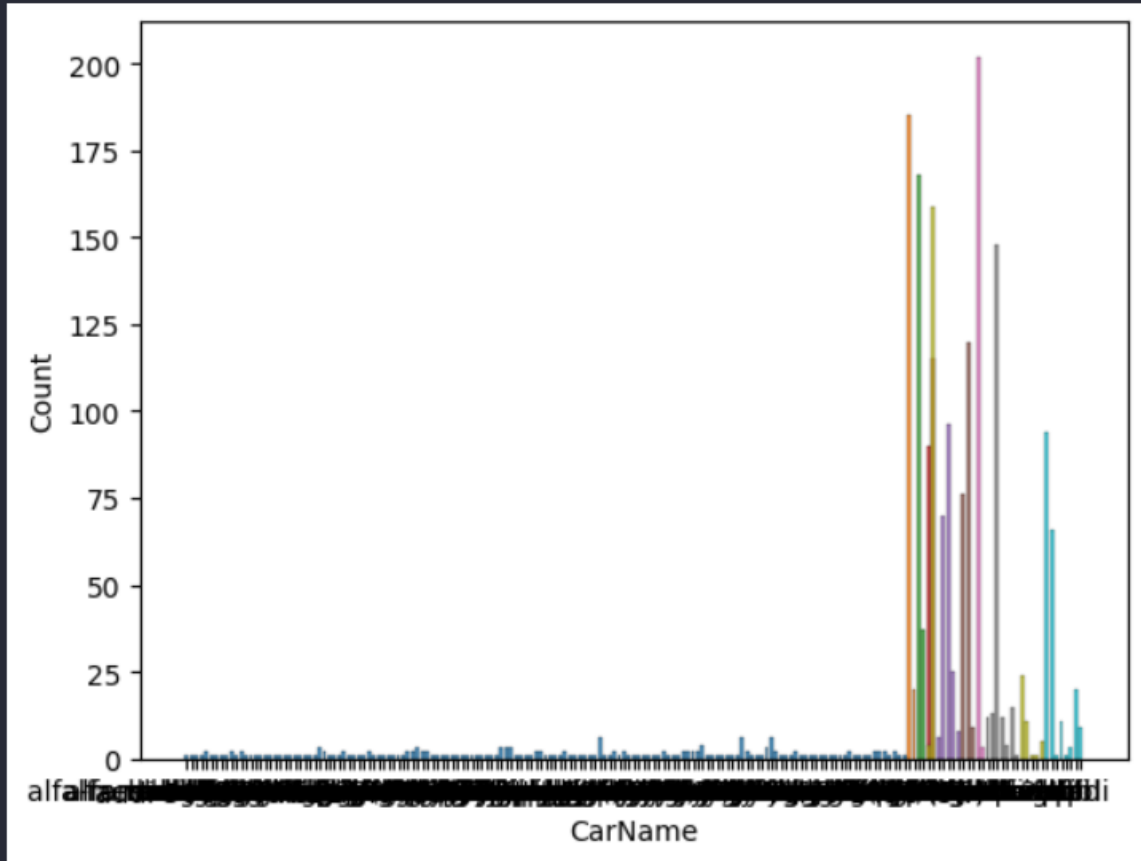
Text(0, 0.5, 'Count')



- O gráfico bagunçou a exibição com sobreposições de labels do CarName.


```
df_categorical = df.select_dtypes(include = 'object').columns
for i in df_categorical:
    fig = sns.histplot(data=df, x = i, shrink=.8)
```

✓ 1.3s



- O problema de sobreposição de labels do eixo x pode ocorrer quando há muitos valores únicos na variável categórica e o espaço no eixo x é limitado. Isso faz com que os labels se sobreponham, tornando o gráfico difícil de ler.

Uma solução para esse problema é ajustar o tamanho dos gráficos ou os intervalos entre os labels no eixo x para evitar a sobreposição. Você pode tentar as seguintes abordagens:

1. Aumentar o tamanho do gráfico:

- Definir uma largura maior para o gráfico pode ajudar a fornecer mais espaço para os labels no eixo x. Você pode experimentar definindo a largura do gráfico usando o parâmetro `figsize` na

função `plt.subplots()` ou ajustando o tamanho da figura com `plt.figure(figsize=(largura, altura))`.

2. Rotacionar os labels do eixo x:

- Rotacionar os labels do eixo x pode ajudar a evitar a sobreposição. Você pode fazer isso usando o parâmetro `rotation` na função `plt.xticks()` ou `ax.set_xticklabels()` para rotacionar os labels em um ângulo específico.

3. Reduzir o número de labels mostrados:

- Se houver muitos valores únicos na variável categórica, pode ser útil reduzir o número de labels mostrados no eixo x. Você pode fazer isso selecionando apenas um subconjunto dos valores únicos para exibir ou agrupando os valores em intervalos.