Projeto4-Solucao

November 16, 2022

1 Data Science Academy

2 Análise de Dados com Linguagem Python

2.1 Projeto 4

2.2 Análise de Dados Para Campanhas de Marketing de Instituições Financeiras

Não tenha pressa de chegar ao final. O aprendizado não está no final. O aprendizado está na jornada. Aproveite a jornada!

2.3 Pré-Requisitos

Recomendamos que você tenha concluído pelo menos os 5 primeiros capítulos do curso gratuito de Python Fundamentos Para Análise de Dados.

2.4 Instalando e Carregando os Pacotes

[1]:
```
# Versão da Linguagem Python
from platform import python_version
print('Versão da Linguagem Python Usada Neste Jupyter Notebook:',␣
 ↪python_version())
```

```
Versão da Linguagem Python Usada Neste Jupyter Notebook: 3.8.8
```

[2]:
```
# Para atualizar um pacote, execute o comando abaixo no terminal ou prompt de␣
 ↪comando:
# pip install -U nome_pacote

# Para instalar a versão exata de um pacote, execute o comando abaixo no␣
 ↪terminal ou prompt de comando:
# !pip install nome_pacote==versão_desejada

# Depois de instalar ou atualizar o pacote, reinicie o jupyter notebook.

# Instala o pacote watermark.
# Esse pacote é usado para gravar as versões de outros pacotes usados neste␣
 ↪jupyter notebook.
# !pip install -q -U watermark
```

[3]:
```
# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

[4]:
```
# Versões dos pacotes usados neste jupyter notebook
%reload_ext watermark
%watermark -a "Data Science Academy" --iversions
```

```
Author: Data Science Academy

seaborn   : 0.11.1
matplotlib: 3.4.3
numpy     : 1.21.2
pandas    : 1.3.3
```

2.5 Carregando os Dados

[5]:
```
# Carrega o dataset
df = pd.read_csv("dados/dataset.csv")
```

[6]:
```
# Shape
df.shape
```

[6]:
```
(45211, 19)
```

[7]:
```
# Amostra
df.head()
```

[7]:

	customerid	age	salary	balance	marital	jobedu	targeted	default	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	response
0	1	58.0	100000.0	2143	married	management,tertiary	yes	no	yes	no	unknown	5	may, 2017	261 sec	1	-1	0	unknown	no
1	2	44.0	60000.0	29	single	technician,secondary	yes	no	yes	no	unknown	5	may, 2017	151 sec	1	-1	0	unknown	no
2	3	33.0	120000.0	2	married	entrepreneur,secondary	yes	no	yes	yes	unknown	5	may, 2017	76 sec	1	-1	0	unknown	no
3	4	47.0	20000.0	1506	married	blue-collar,unknown	no	no	yes	no	unknown	5	may, 2017	92 sec	1	-1	0	unknown	no
4	5	33.0	0.0	1	single	unknown,unknown	no	no	no	no	unknown	5	may, 2017	198 sec	1	-1	0	unknown	no

2.6 Análise Exploratória

[8]:
```
# Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 19 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   customerid  45211 non-null  int64
 1   age         45191 non-null  float64
 2   salary      45185 non-null  float64
 3   balance     45211 non-null  int64
```

```
 4   marital     45211 non-null  object
 5   jobedu      45211 non-null  object
 6   targeted    45211 non-null  object
 7   default     45211 non-null  object
 8   housing     45211 non-null  object
 9   loan        45211 non-null  object
 10  contact     45211 non-null  object
 11  day         45211 non-null  int64
 12  month       45161 non-null  object
 13  duration    45211 non-null  object
 14  campaign    45211 non-null  int64
 15  pdays       45211 non-null  int64
 16  previous    45211 non-null  int64
 17  poutcome    45211 non-null  object
 18  response    45181 non-null  object
dtypes: float64(2), int64(6), object(11)
memory usage: 6.6+ MB
```

[9]:
```
# Temos valores nulos?
df.isna().any()
```

[9]:
```
customerid    False
age            True
salary         True
balance       False
marital       False
jobedu        False
targeted      False
default       False
housing       False
loan          False
contact       False
day           False
month          True
duration      False
campaign      False
pdays         False
previous      False
poutcome      False
response       True
dtype: bool
```

[10]:
```
# Temos valores nulos?
df.isna().sum()
```

[10]:
```
customerid     0
age           20
salary        26
```

```
balance        0
marital        0
jobedu         0
targeted       0
default        0
housing        0
loan           0
contact        0
day            0
month         50
duration       0
campaign       0
pdays          0
previous       0
poutcome       0
response      30
dtype: int64
```

[11]:
```
# Não usaremos a coluna ID. Vamos removê-la.
df.drop(["customerid"], axis = 1, inplace = True)
```

[12]:
```
# Colunas
df.columns
```

[12]:
```
Index(['age', 'salary', 'balance', 'marital', 'jobedu', 'targeted', 'default',
       'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign',
       'pdays', 'previous', 'poutcome', 'response'],
      dtype='object')
```

Exercício 1: A coluna "jobedu" parece ter duas informações. Vamos separar em duas colunas.

[13]:
```
df.head()
```

[13]:

	age	salary	balance	marital	jobedu	targeted	default
0	58.0	100000.0	2143	married	management,tertiary	yes	no
1	44.0	60000.0	29	single	technician,secondary	yes	no
2	33.0	120000.0	2	married	entrepreneur,secondary	yes	no
3	47.0	20000.0	1506	married	blue-collar,unknown	no	no
4	33.0	0.0	1	single	unknown,unknown	no	no

	housing	loan	contact	day	month	duration	campaign	pdays	previous
0	yes	no	unknown	5	may, 2017	261 sec	1	-1	0
1	yes	no	unknown	5	may, 2017	151 sec	1	-1	0
2	yes	yes	unknown	5	may, 2017	76 sec	1	-1	0
3	yes	no	unknown	5	may, 2017	92 sec	1	-1	0
4	no	no	unknown	5	may, 2017	198 sec	1	-1	0

```
  poutcome response
0  unknown       no
1  unknown       no
2  unknown       no
3  unknown       no
4  unknown       no
```

[14]:
```
# Fazemos o split da coluna jobedu e criamos a coluna job com o primeiro␣
 ↪elemento antes da vírgula
df['job'] = df["jobedu"].apply(lambda x:x.split(",")[0])
```

[15]:
```
df.head()
```

[15]:
```
    age    salary  balance  marital                jobedu targeted default  \
0  58.0  100000.0     2143  married     management,tertiary      yes      no
1  44.0   60000.0       29   single    technician,secondary      yes      no
2  33.0  120000.0        2  married  entrepreneur,secondary      yes      no
3  47.0   20000.0     1506  married     blue-collar,unknown       no      no
4  33.0       0.0        1   single         unknown,unknown       no      no

  housing loan  contact  day      month duration  campaign  pdays  previous  \
0     yes   no  unknown    5  may, 2017  261 sec         1     -1         0
1     yes   no  unknown    5  may, 2017  151 sec         1     -1         0
2     yes  yes  unknown    5  may, 2017   76 sec         1     -1         0
3     yes   no  unknown    5  may, 2017   92 sec         1     -1         0
4      no   no  unknown    5  may, 2017  198 sec         1     -1         0

  poutcome response           job
0  unknown       no    management
1  unknown       no    technician
2  unknown       no  entrepreneur
3  unknown       no   blue-collar
4  unknown       no       unknown
```

[16]:
```
# Fazemos o split da coluna jobedu e criamos a coluna education com o segundo␣
 ↪elemento antes da vírgula
df['education'] = df["jobedu"].apply(lambda x:x.split(",")[1])
```

[17]:
```
df.head()
```

[17]:
```
    age    salary  balance  marital                jobedu targeted default  \
0  58.0  100000.0     2143  married     management,tertiary      yes      no
1  44.0   60000.0       29   single    technician,secondary      yes      no
2  33.0  120000.0        2  married  entrepreneur,secondary      yes      no
3  47.0   20000.0     1506  married     blue-collar,unknown       no      no
4  33.0       0.0        1   single         unknown,unknown       no      no
```

```
  housing loan  contact  day      month duration  campaign  pdays  previous  \
0     yes   no  unknown    5  may, 2017  261 sec         1     -1         0
1     yes   no  unknown    5  may, 2017  151 sec         1     -1         0
2     yes  yes  unknown    5  may, 2017   76 sec         1     -1         0
3     yes   no  unknown    5  may, 2017   92 sec         1     -1         0
4      no   no  unknown    5  may, 2017  198 sec         1     -1         0

  poutcome response           job  education
0  unknown       no    management   tertiary
1  unknown       no    technician  secondary
2  unknown       no  entrepreneur  secondary
3  unknown       no   blue-collar    unknown
4  unknown       no       unknown    unknown
```

[18]:
```
# Drop da coluna "jobedu"
df.drop(["jobedu"], axis = 1, inplace = True)
```

[19]:
```
df.head()
```

[19]:
```
    age    salary  balance  marital targeted default housing loan  contact  \
0  58.0  100000.0     2143  married      yes      no     yes   no  unknown
1  44.0   60000.0       29   single      yes      no     yes   no  unknown
2  33.0  120000.0        2  married      yes      no     yes  yes  unknown
3  47.0   20000.0     1506  married       no      no     yes   no  unknown
4  33.0       0.0        1   single       no      no      no   no  unknown

   day      month duration  campaign  pdays  previous poutcome response  \
0    5  may, 2017  261 sec         1     -1         0  unknown       no
1    5  may, 2017  151 sec         1     -1         0  unknown       no
2    5  may, 2017   76 sec         1     -1         0  unknown       no
3    5  may, 2017   92 sec         1     -1         0  unknown       no
4    5  may, 2017  198 sec         1     -1         0  unknown       no

            job  education
0    management   tertiary
1    technician  secondary
2  entrepreneur  secondary
3   blue-collar    unknown
4       unknown    unknown
```

2.7 Tratamento de Valores Ausentes

Vamos primeiro tratar a variável que representa a idade.

[20]:
```
# Valores ausentes no dataframe
df.isna().any()
```

[20]:
```
age           True
salary        True
balance      False
marital      False
targeted     False
default      False
housing      False
loan         False
contact      False
day          False
month         True
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
response      True
job          False
education    False
dtype: bool
```

[21]:
```
# Valores ausentes da variável age
df.age.isnull().sum()
```

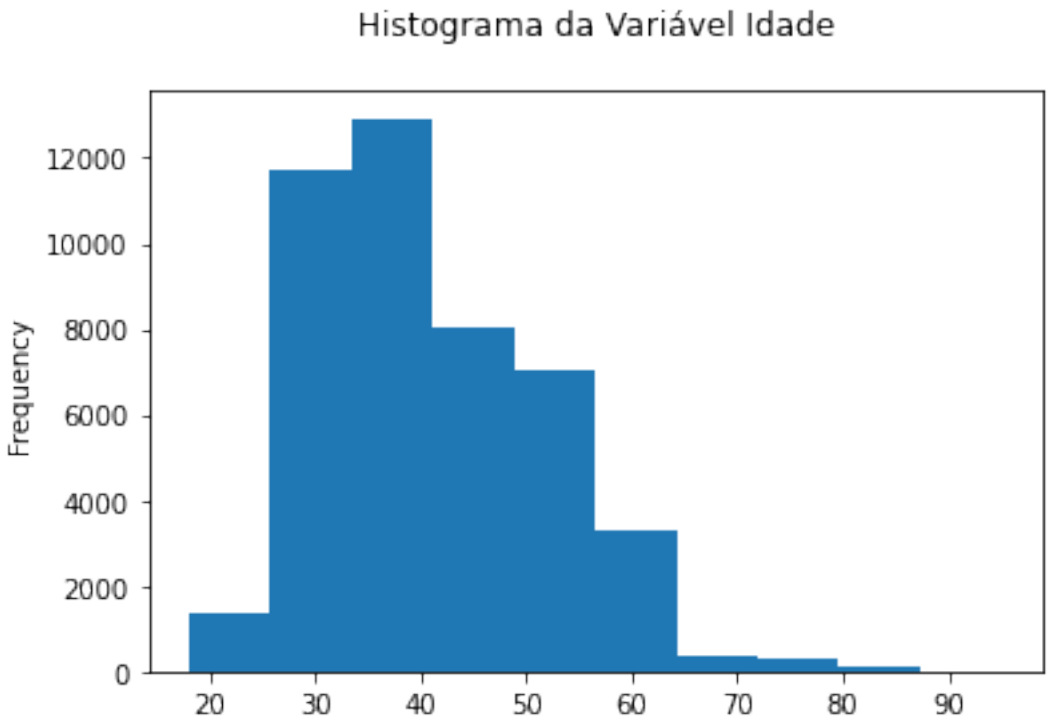
[21]:
```
20
```

[22]:
```
# Calcula o percentual de valores ausentes na variável age
df.age.isnull().mean()*100
```

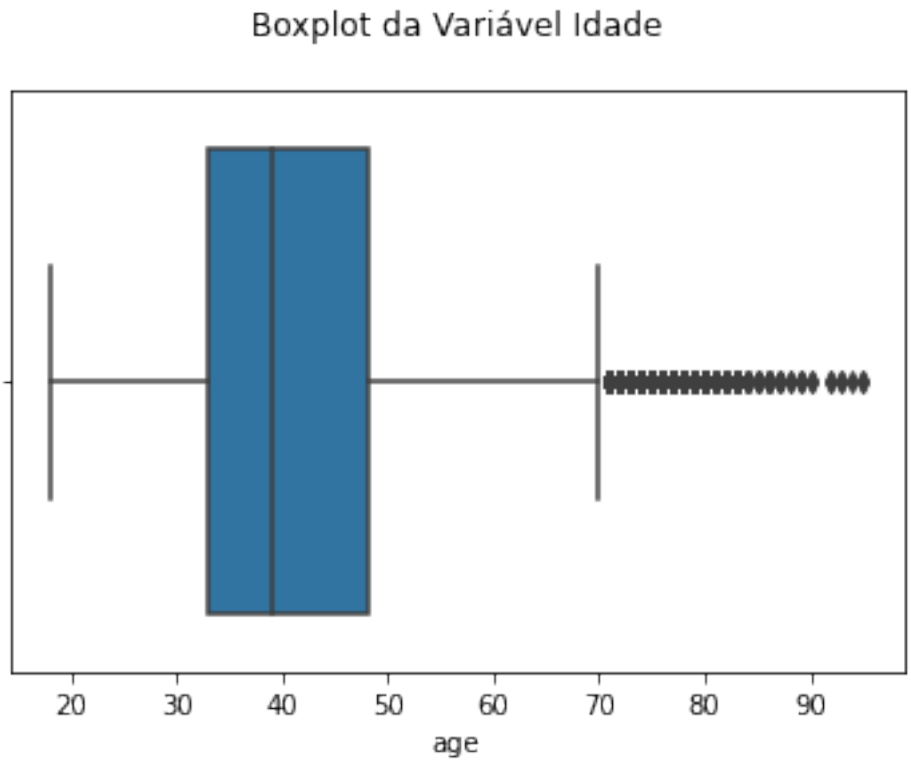
[22]:
```
0.0442370219636814
```

Como o percentual é baixo não podemos eliminar a coluna. Podemos então eliminar os registros com valores ausentes (nesse caso perderíamos 20 linhas no dataset) ou podemos aplicar imputação. Vamos usar a segunda opção.

[23]:
```
# Histograma
df.age.plot(kind = "hist")
plt.title("Histograma da Variável Idade\n")
plt.show()
```



[24]:
```
# Boxplot
sns.boxplot(df.age)
plt.title("Boxplot da Variável Idade\n")
plt.show()
```



[25]:
```
# Vamos verificar qual é a média de idade.
df.age.mean()
```

[25]:
```
40.93565090394105
```

[26]:
```
# Vamos verificar qual é a mediana, valor do meio da distribuição quando os␣
↪dados estão ordenados.
df.age.median()
```

[26]:
```
39.0
```

[27]:
```
# Vamos verificar qual é a moda, o valor que aparece com mais frequência.
df.age.mode()
```

[27]:
```
0    32.0
dtype: float64
```

Exercício 2: Vamos imputar os valores ausentes da variável age com uma medida de tendência central. Escolha uma das medidas, aplique a imputação e justifique sua escolha. Deixamos a variável como float ou como int? Se convertemos, fazemos isso antes ou depois da imputação?

[28]:
```
# Vamos preencher com a moda pois são poucos valores ausentes e assim alteramos␣
↪muito pouco o padrão nos dados.
df.age.fillna("32", inplace = True)
```

[29]:
```
# Agora convertemos para int
df.age = df.age.astype("int")
```

[30]:
```
# Tipo da variável
df.age.dtypes
```

[30]:
```
dtype('int64')
```

[31]:
```
# Média
df.age.mean()
```

[31]:
```
40.93169803808808
```

[32]:
```
# Mediana
df.age.median()
```

[32]:
```
39.0
```

[33]:
```
# Percentual de valores ausentes
df.age.isnull().mean()*100
```

[33]:
```
0.0
```

2.8 Tratamento de Valores Ausentes

Vamos agora tratar a variável que representa o mês.

[34]:
```
# Valores ausentes no dataframe
df.isna().any()
```

[34]:
```
age          False
salary        True
balance      False
marital      False
targeted     False
default      False
housing      False
loan         False
contact      False
day          False
month         True
duration     False
campaign     False
pdays        False
```

```
previous     False
poutcome     False
response      True
job          False
education    False
dtype: bool
```

[35]:
```
# Valores ausentes na variável
df.month.isnull().sum()
```

[35]:
```
50
```

[36]:
```
# Percentual de valores ausentes
df.month.isnull().mean()*100
```

[36]:
```
0.11059255490920351
```

Como o percentual é menor que 30% não podemos eliminar a coluna. Podemos então eliminar os registros com valores ausentes (nesse caso perderíamos 50 linhas no dataset) ou podemos aplicar imputação. Vamos usar a segunda opção.

[37]:
```
# Tipo da variável
df.month.dtypes
```

[37]:
```
dtype('O')
```

[38]:
```
# Categorias da variável
df.month.value_counts()
```

[38]:
```
may, 2017    13747
jul, 2017     6888
aug, 2017     6240
jun, 2017     5335
nov, 2017     3968
apr, 2017     2931
feb, 2017     2646
jan, 2017     1402
oct, 2017      738
sep, 2017      576
mar, 2017      476
dec, 2017      214
Name: month, dtype: int64
```

Exercício 3: Vamos imputar os valores ausentes da variável month. Escolha uma estratégia e aplique no dataset.

[39]:
```
# Vamos imputar com a moda, o valor mais frequente da variável, pois são poucos␣
↪registros
df.month.mode()
```

[39]:
```
0    may, 2017
dtype: object
```

[40]:
```
# Imputação com a moda
df.month.fillna("may, 2017", inplace = True)
```

[41]:
```
# Valores ausentes tratados com sucesso
df.month.isnull().sum()
```

[41]:
```
0
```

2.9 Tratamento de Valores Ausentes

Vamos agora tratar a variável que representa o salário.

[42]:
```
# Valores ausentes no dataframe
df.isna().any()
```

[42]:
```
age          False
salary        True
balance      False
marital      False
targeted     False
default      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
response      True
job          False
education    False
dtype: bool
```

[43]:
```
# Valores ausentes na variável
df.salary.isnull().sum()
```

[43]:
```
26
```

[44]:
```
# Calcula o percentual de valores ausentes na variável salary
df.salary.isnull().mean()*100
```

[44]:
```
0.05750812855278583
```

Como o percentual é baixo não podemos eliminar a coluna. Podemos então eliminar os registros com valores ausentes (nesse caso perderíamos 26 linhas no dataset) ou podemos aplicar imputação. Vamos usar a segunda opção.

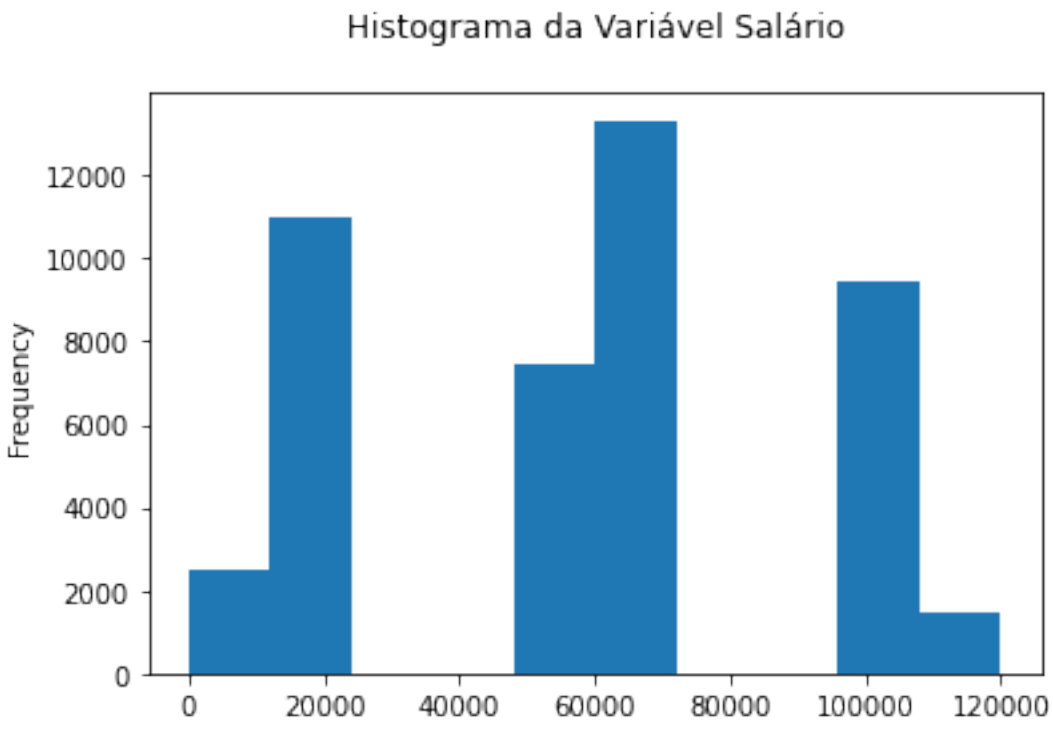
Mas espere. Vamos checar algo aqui.

[45]:
```
df.head()
```

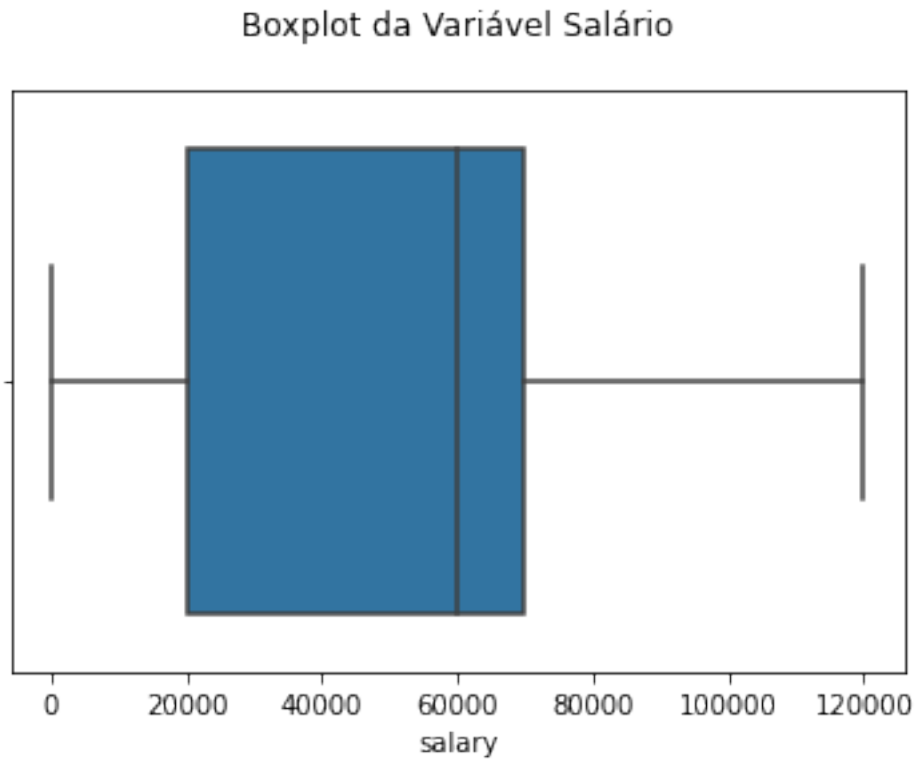
[45]:
```
   age    salary  balance  marital targeted default housing loan  contact  \
0   58  100000.0     2143  married      yes      no     yes   no  unknown
1   44   60000.0       29   single      yes      no     yes   no  unknown
2   33  120000.0        2  married      yes      no     yes  yes  unknown
3   47   20000.0     1506  married       no      no     yes   no  unknown
4   33       0.0        1   single       no      no      no   no  unknown

   day      month duration  campaign  pdays  previous poutcome response  \
0    5  may, 2017  261 sec         1     -1         0  unknown       no
1    5  may, 2017  151 sec         1     -1         0  unknown       no
2    5  may, 2017   76 sec         1     -1         0  unknown       no
3    5  may, 2017   92 sec         1     -1         0  unknown       no
4    5  may, 2017  198 sec         1     -1         0  unknown       no

            job  education
0    management   tertiary
1    technician  secondary
2  entrepreneur  secondary
3   blue-collar    unknown
4       unknown    unknown
```

Existe salário igual a zero? Não. O valor zero é provavelmente um outlier (confirmar com a área de negócio).

[46]:
```
# Histograma
df.salary.plot(kind = "hist")
plt.title("Histograma da Variável Salário\n")
plt.show()
```



[47]:
```
# Boxplot
sns.boxplot(df.salary)
plt.title("Boxplot da Variável Salário\n")
plt.show()
```



[48]:
```
# Vamos verificar qual é a média de idade.
df.salary.mean()
```

[48]:
```
57008.65331415293
```

[49]:
```
# Vamos verificar qual é a mediana.
df.salary.median()
```

[49]:
```
60000.0
```

[50]:
```
# Vamos verificar qual é a moda.
df.salary.mode()
```

[50]:
```
0    20000.0
dtype: float64
```

Exercício 4: Vamos imputar os valores ausentes da variável salary com uma medida de tendência central. Precisamos também tratar os valores iguais a zero. Escolha sua estratégia, aplique a imputação e justifique sua escolha.

[51]:
```
# Vamos preencher com a mediana pois os dados parecem assimétricos (nesse caso␣
↪a média não pode ser usada)
```

```
# e o valor mais frequente está muito abaixo da média e da mediana (por isso
 ↪não usaremos a moda)
df.salary.fillna("60000", inplace = True)
```

[52]:
```
df.head()
```

[52]:

	age	salary	balance	marital	targeted	default	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	response	job	education
0	58	100000.0	2143	married	yes	no	yes	no	unknown	5	may, 2017	261 sec	1	-1	0	unknown	no	management	tertiary
1	44	60000.0	29	single	yes	no	yes	no	unknown	5	may, 2017	151 sec	1	-1	0	unknown	no	technician	secondary
2	33	120000.0	2	married	yes	no	yes	yes	unknown	5	may, 2017	76 sec	1	-1	0	unknown	no	entrepreneur	secondary
3	47	20000.0	1506	married	no	no	yes	no	unknown	5	may, 2017	92 sec	1	-1	0	unknown	no	blue-collar	unknown
4	33	0.0	1	single	no	no	no	no	unknown	5	may, 2017	198 sec	1	-1	0	unknown	no	unknown	unknown

[53]:
```
# Histograma (vai gerar erro)
df.salary.plot(kind = "hist")
plt.title("Histograma da Variável Salário\n")
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/dc/lqrc3k5j4438r150cbrdr_000000gn/T/ipykernel_10787/936846111.py i
 ↪<module>
      1 # Histograma (vai gerar erro)
----> 2 df.salary.plot(kind = "hist")
      3 plt.title("Histograma da Variável Salário\n")
      4 plt.show()

~/opt/anaconda3/lib/python3.8/site-packages/pandas/plotting/_core.py in
 ↪__call__(self, *args, **kwargs)
    970                     data.columns = label_name
    971
--> 972         return plot_backend.plot(data, kind=kind, **kwargs)
    973
```

```
    974     __call__.__doc__ = __doc__

~/opt/anaconda3/lib/python3.8/site-packages/pandas/plotting/_matplotlib/__init__.
 ↪py in plot(data, kind, **kwargs)
     69             kwargs["ax"] = getattr(ax, "left_ax", ax)
     70     plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71     plot_obj.generate()
     72     plot_obj.draw()
     73     return plot_obj.result

~/opt/anaconda3/lib/python3.8/site-packages/pandas/plotting/_matplotlib/core.py
 ↪in generate(self)
    284     def generate(self):
    285         self._args_adjust()
--> 286         self._compute_plot_data()
    287         self._setup_subplots()
    288         self._make_plot()

~/opt/anaconda3/lib/python3.8/site-packages/pandas/plotting/_matplotlib/core.py
 ↪in _compute_plot_data(self)
    451         # no non-numeric frames or series allowed
    452         if is_empty:
--> 453             raise TypeError("no numeric data to plot")
    454
    455         self.data = numeric_data.apply(self._convert_to_ndarray)

TypeError: no numeric data to plot
```

[54]:
```
# Tipo da variável
df.salary.dtypes
```

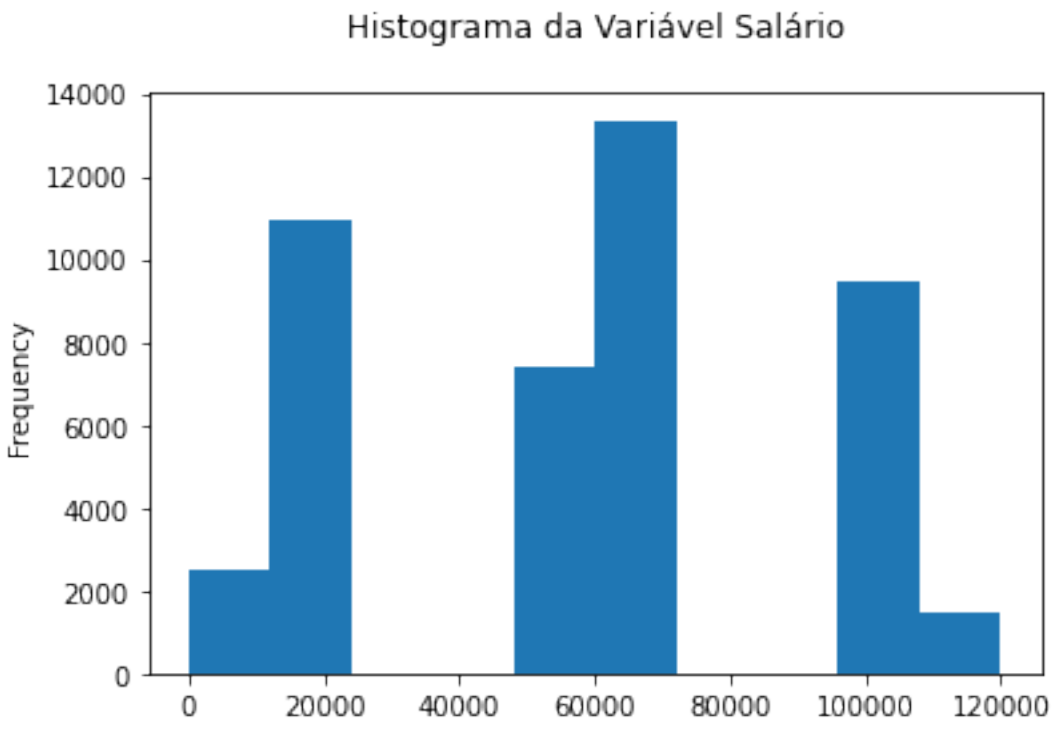
[54]:
```
dtype('O')
```

[55]:
```
# Convertemos para o tipo float
df.salary = df.salary.astype("float")
```

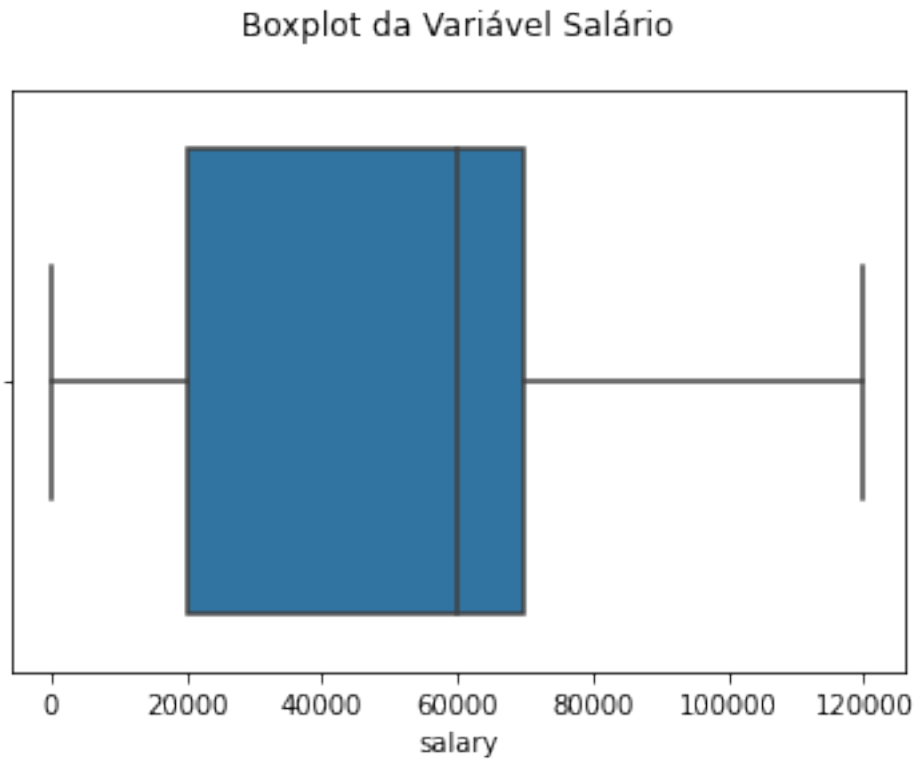
[56]:
```
# Tipo da variável
df.salary.dtypes
```

[56]:
```
dtype('float64')
```

[57]:
```
# Histograma
df.salary.plot(kind = "hist")
plt.title("Histograma da Variável Salário\n")
plt.show()
```



[58]:
```
# Boxplot
sns.boxplot(df.salary)
plt.title("Boxplot da Variável Salário\n")
plt.show()
```



[59]:
```
# Registros para cada salário
df.salary.value_counts()
```

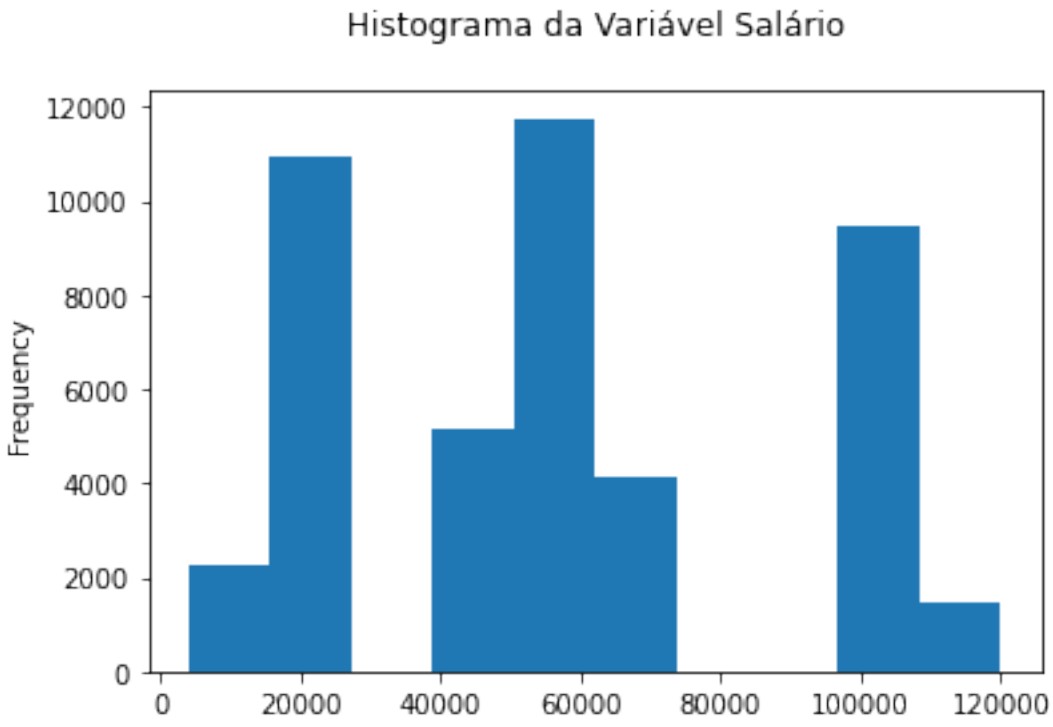
[59]:
```
20000.0     9725
100000.0    9454
60000.0     9195
50000.0     5167
70000.0     4153
55000.0     2264
120000.0    1486
8000.0      1303
16000.0     1239
4000.0       937
0.0          288
Name: salary, dtype: int64
```

[60]:
```
# Replace do zero pela mediana
df['salary'] = df['salary'].replace(0, df['salary'].median())
```

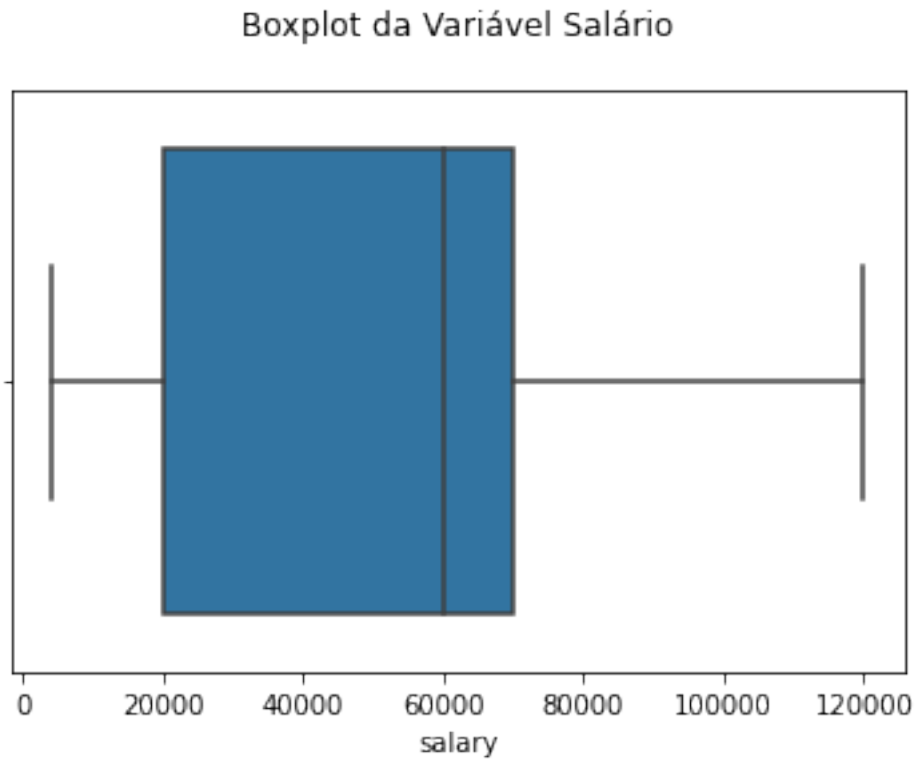
[61]:
```
# Registros para cada salário
df.salary.value_counts()
```

[61]:
```
20000.0     9725
60000.0     9483
100000.0    9454
50000.0     5167
70000.0     4153
55000.0     2264
120000.0    1486
8000.0      1303
16000.0     1239
4000.0       937
Name: salary, dtype: int64
```

[62]:
```
# Histograma
df.salary.plot(kind = "hist")
plt.title("Histograma da Variável Salário\n")
plt.show()
```



[63]:
```
# Boxplot
sns.boxplot(df.salary)
plt.title("Boxplot da Variável Salário\n")
plt.show()
```



[64]:
```
# Calcula o percentual de valores ausentes na variável salary
df.salary.isnull().mean()*100
```

[64]:
```
0.0
```

[65]:
```
df.isna().any()
```

[65]:
```
age          False
salary       False
balance      False
marital      False
targeted     False
default      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
```

```
poutcome     False
response      True
job          False
education    False
dtype: bool
```

2.10 Tratamento de Valores Ausentes

Vamos agora tratar a variável que representa a resposta (variável alvo).

[66]:
```
df.head()
```

[66]:
```
   age    salary  balance  marital targeted default housing loan  contact  \
0   58  100000.0     2143  married      yes      no     yes   no  unknown
1   44   60000.0       29   single      yes      no     yes   no  unknown
2   33  120000.0        2  married      yes      no     yes  yes  unknown
3   47   20000.0     1506  married       no      no     yes   no  unknown
4   33   60000.0        1   single       no      no      no   no  unknown

   day      month duration  campaign  pdays  previous poutcome response  \
0    5  may, 2017  261 sec         1     -1         0  unknown       no
1    5  may, 2017  151 sec         1     -1         0  unknown       no
2    5  may, 2017   76 sec         1     -1         0  unknown       no
3    5  may, 2017   92 sec         1     -1         0  unknown       no
4    5  may, 2017  198 sec         1     -1         0  unknown       no

            job  education
0    management   tertiary
1    technician  secondary
2  entrepreneur  secondary
3   blue-collar    unknown
4       unknown    unknown
```

[67]:
```
# Valores ausentes
df.response.isnull().sum()
```

[67]:
```
30
```

[68]:
```
# Calcula o percentual
df.response.isnull().mean()*100
```

[68]:
```
0.0663555329455221
```

Como o percentual é baixo (e a variável é o alvo da nossa análise) não podemos eliminar a coluna. Podemos então eliminar os registros com valores ausentes (nesse caso perderíamos 30 linhas no dataset) ou podemos aplicar imputação.

Exercício 5: Escolha sua estratégia, aplique e justifique sua escolha.

[69]:
```
# Não devemos aplicar imputação na variável de estudo (variável resposta ou␣
↪variável alvo)
# Vamos dropar os registros
df.dropna(subset = ["response"], inplace = True)
```

[70]:
```
# Verifca valores NA
df.isnull().sum()
```

[70]:
```
age          0
salary       0
balance      0
marital      0
targeted     0
default      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
response     0
job          0
education    0
dtype: int64
```

2.11 Tratamento de Valores Ausentes

Vamos agora tratar a variável pdays.

[71]:
```
# Valores ausentes
df.pdays.isnull().sum()
```

[71]:
```
0
```

[72]:
```
# Describe
df.pdays.describe()
```

[72]:
```
count    45181.000000
mean        40.198601
std        100.134050
min         -1.000000
25%         -1.000000
50%         -1.000000
75%         -1.000000
```

```
max        871.000000
Name: pdays, dtype: float64
```

-1 indica valor ausente

[73]:
```
# Vamos fazer relace de -1 por NaN
df.pdays = df.pdays.replace({-1.0:np.NaN})
```

[74]:
```
# Valores ausentes
df.isnull().sum()
```

[74]:
```
age              0
salary           0
balance          0
marital          0
targeted         0
default          0
housing          0
loan             0
contact          0
day              0
month            0
duration         0
campaign         0
pdays        36930
previous         0
poutcome         0
response         0
job              0
education        0
dtype: int64
```

[75]:
```
# Calcula o percentual
df.pdays.isnull().mean()*100
```

[75]:
```
81.73789867422147
```

Exercício 6: Escolha sua estratégia, aplique e justifique sua escolha.

[76]:
```
# Drop da coluna "pdays" pois tem mais de 30% dos valores ausentes
df.drop(["pdays"], axis = 1, inplace = True)
```

[77]:
```
# Valores ausentes
df.isnull().sum()
```

[77]:
```
age          0
salary       0
balance      0
marital      0
```

```
targeted     0
default      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
previous     0
poutcome     0
response     0
job          0
education    0
dtype: int64
```

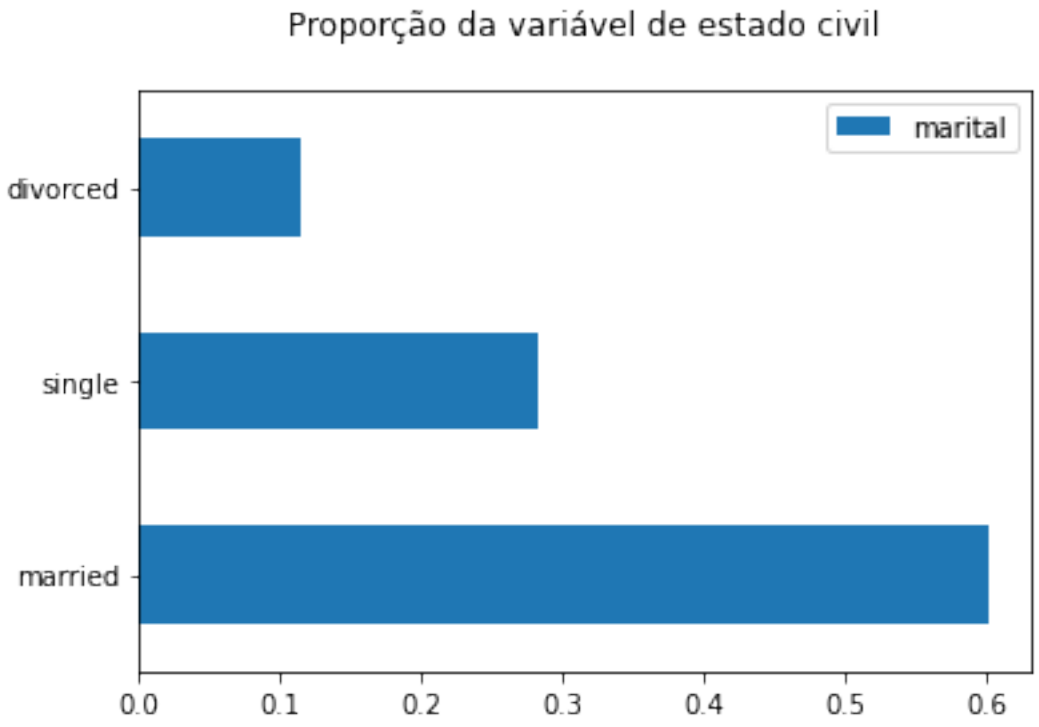
2.12 Conclusão e Análise dos Dados

2.12.1 Análise Univariada

[78]:
```
# Proporção da variável de estado civil
df.marital.value_counts(normalize = True)
```

[78]:
```
married     0.601912
single      0.282907
divorced    0.115181
Name: marital, dtype: float64
```

[79]:
```
# Plot
df.marital.value_counts(normalize = True).plot(kind = "barh")
plt.title("Proporção da variável de estado civil\n")
plt.legend()
plt.show()
```

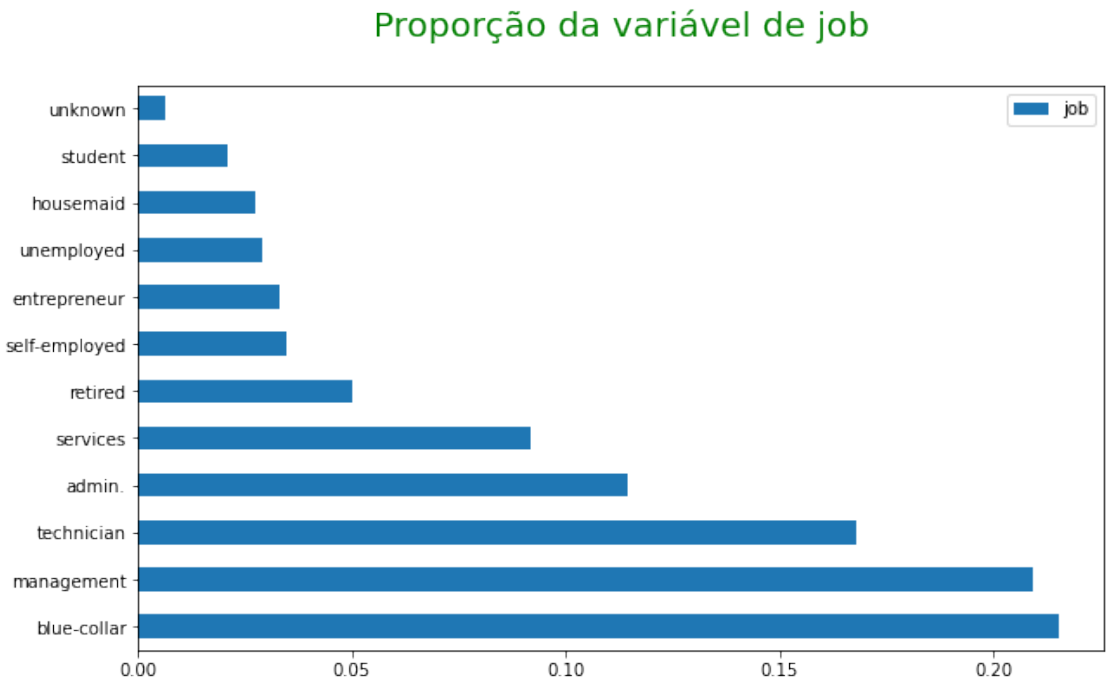


[80]:
```
# Proporção da variável de job
df.job.value_counts(normalize = True)
```

[80]:
```
blue-collar      0.215290
management       0.209247
technician       0.168035
admin.           0.114340
services         0.091853
retired          0.050087
self-employed    0.034860
entrepreneur     0.032890
unemployed       0.028840
housemaid        0.027423
student          0.020761
unknown          0.006374
Name: job, dtype: float64
```

[81]:
```
# Plot
plt.figure(figsize = (10,6))
df.job.value_counts(normalize = True).plot(kind = "barh")
plt.title("Proporção da variável de job\n", fontdict = {'fontsize': 20,␣
 ↪'fontweight' : 5, 'color' : 'Green'})
plt.legend()
```

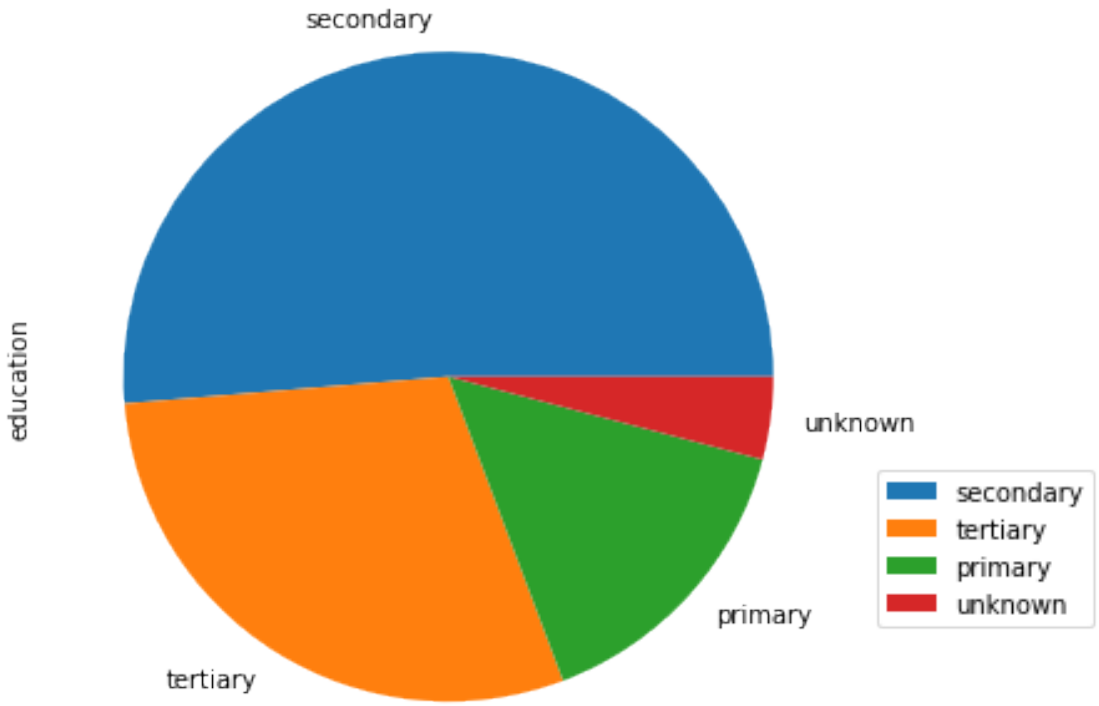
```
plt.show()
```



[82]:
```
# Proporção da variável de education
df.education.value_counts(normalize = True)
```

[82]:
```
secondary    0.513247
tertiary     0.294194
primary      0.151480
unknown      0.041079
Name: education, dtype: float64
```

[83]:
```
# Plot
plt.figure(figsize = (10,6))
df.education.value_counts(normalize = True).plot(kind = "pie")
plt.title("Proporção da variável de education\n", fontdict = {'fontsize': 20,␣
 ↪'fontweight' : 5, 'color' : 'Green'})
plt.legend()
plt.legend(bbox_to_anchor=(1.31,0.4))
plt.show()
```



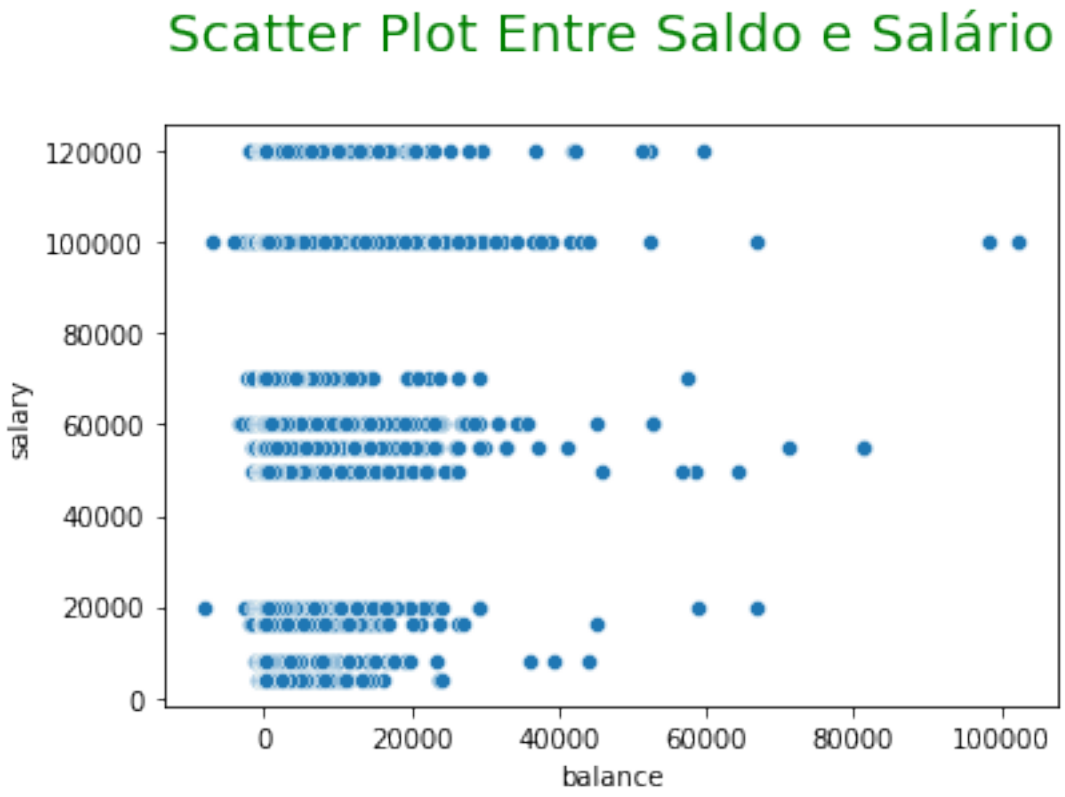
[84]:
```
# Proporção da variável response
df.response.value_counts(normalize = True)
```

[84]:
```
no     0.882982
yes    0.117018
Name: response, dtype: float64
```

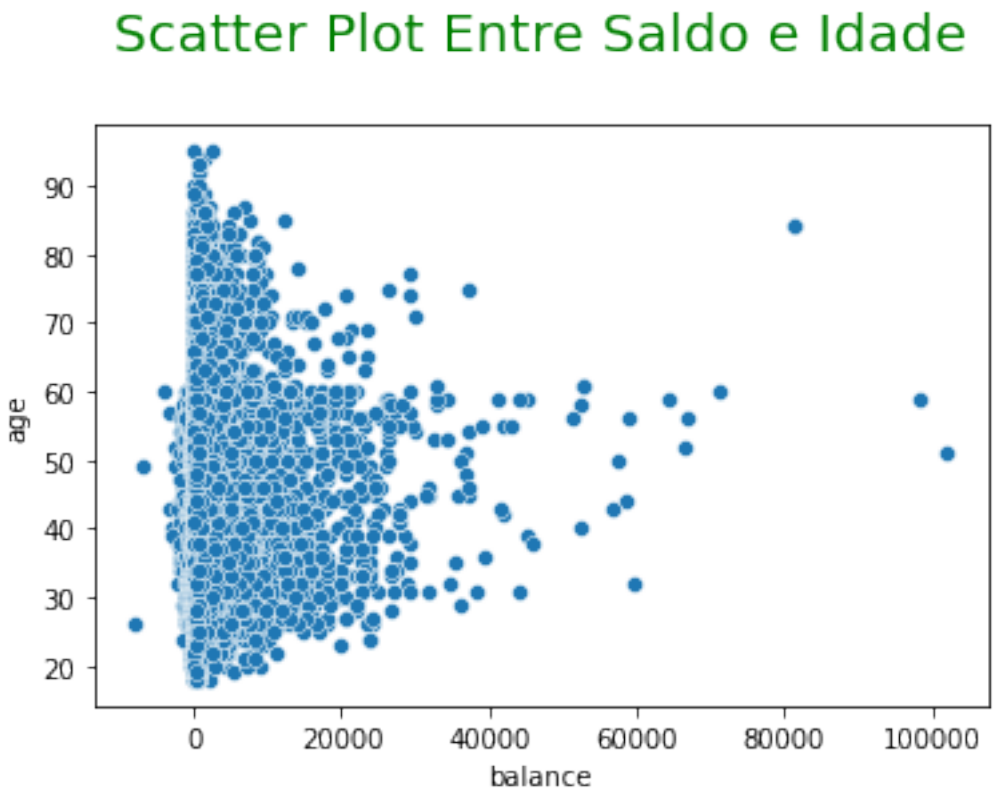
[85]:
```
# Plot
plt.figure(figsize = (10,6))
df.response.value_counts(normalize = True).plot(kind = "pie")
plt.title("Proporção da variável response\n", fontdict = {'fontsize': 20,␣
 ↪'fontweight' : 5, 'color' : 'Green'})
plt.legend()
plt.show()
```

2.13 Análise Multivariada

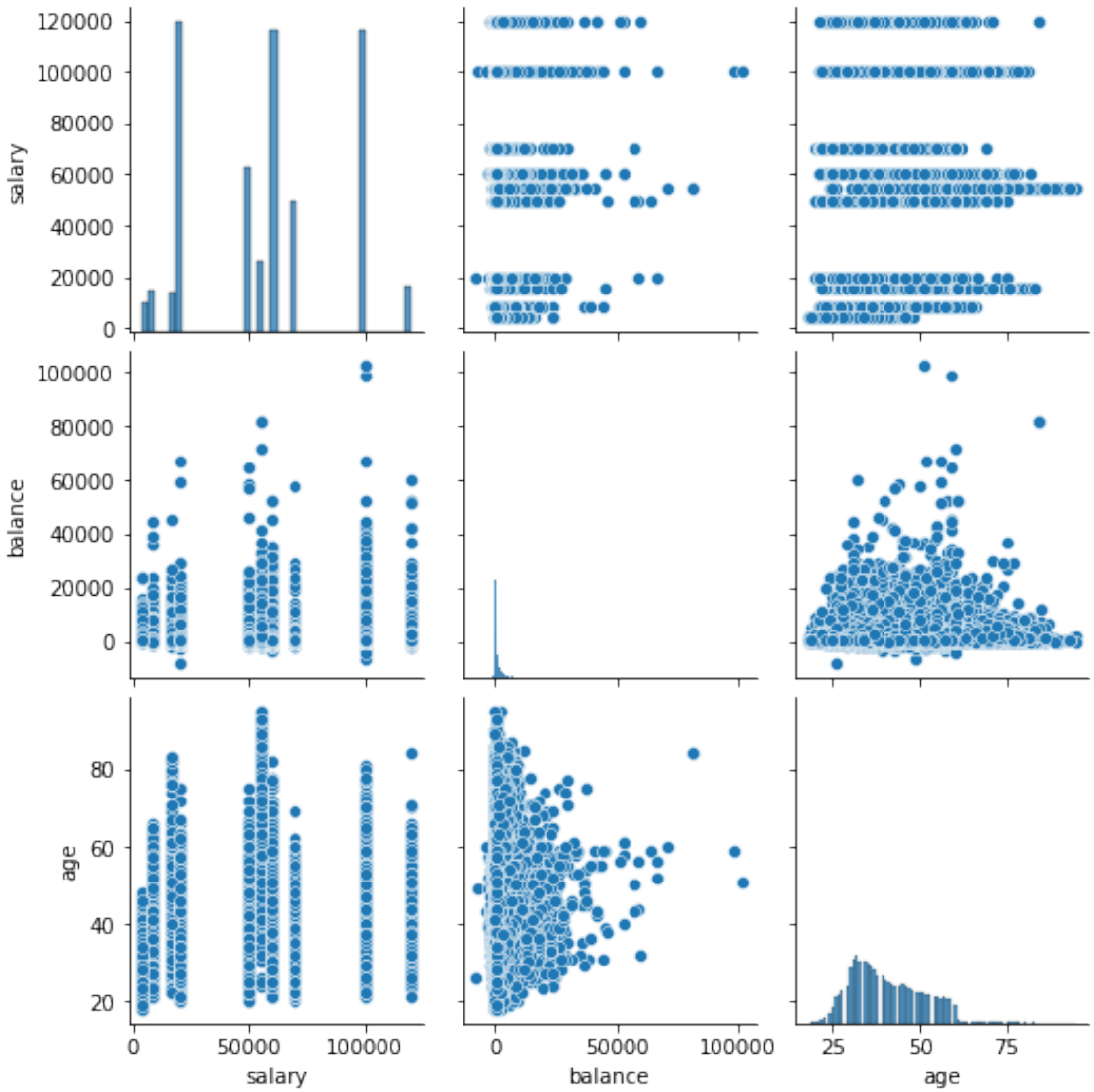
[86]:
```
# Scatter Plot
sns.scatterplot(df["balance"], df["salary"])
plt.title("Scatter Plot Entre Saldo e Salário\n", fontdict = {'fontsize': 20,␣
 ↪'fontweight' : 5, 'color' : 'Green'})
plt.show()
```



[87]:
```
# Scatter Plot
sns.scatterplot(df["balance"], df["age"])
plt.title("Scatter Plot Entre Saldo e Idade\n", fontdict = {'fontsize': 20,␣
  ↪'fontweight' : 5, 'color' : 'Green'})
plt.show()
```

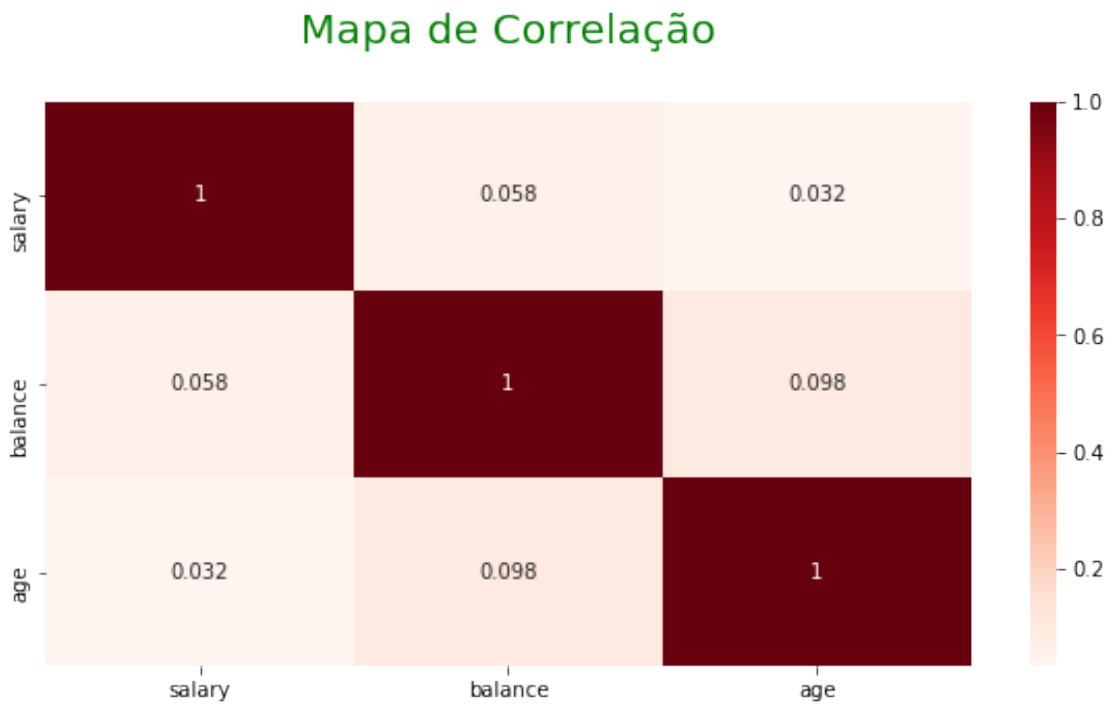


[88]:
```
# Pair Plot
sns.pairplot(df[["salary","balance","age"]])
plt.show()
```



[89]:
```
# Calcula a correlação
res = df[["salary", "balance", "age"]].corr()
```

[90]:
```
# Mapa de Correlação
plt.figure(figsize = (10,5))
sns.heatmap(res, annot = True, cmap = "Reds")
plt.title("Mapa de Correlação\n", fontdict = {'fontsize': 20, 'fontweight' : 5,␣
  ↪'color' : 'Green'})
plt.show()
```



2.13.1 Numérico x Categórico

[91]:
```
# Agrupa o salário pela variável resposta e calcula a média
df.groupby(by = ["response"])["salary"].mean()
```

[91]:
```
response
no     57157.692886
yes    59157.556270
Name: salary, dtype: float64
```

[92]:
```
# Agrupa o salário pela variável resposta e calcula a mediana
df.groupby(by = ["response"])["salary"].median()
```

[92]:
```
response
no     60000.0
yes    60000.0
Name: salary, dtype: float64
```

[93]:
```
# Boxplot
plt.figure(figsize = (10,5))
sns.boxplot(df["response"], df["salary"])
plt.title("Salário x Resposta\n", fontdict = {'fontsize': 20, 'fontweight' : 5,␣
 ↪'color' : 'Green'})
plt.show()
```

[94]:
```
# Agrupa educação por salário e calcula a média
df.groupby(by = ["education"])["salary"].mean()
```

[94]:
```
education
primary      34697.106955
secondary    49922.420113
tertiary     83041.077340
unknown      50708.512931
Name: salary, dtype: float64
```

[95]:
```
# Cria a variável response_flag como tipo numérico onde response "yes"= 1,␣
  ↪"no"= 0
df["response_flag"] = np.where(df["response"] == "yes",1,0)
df.head()
```

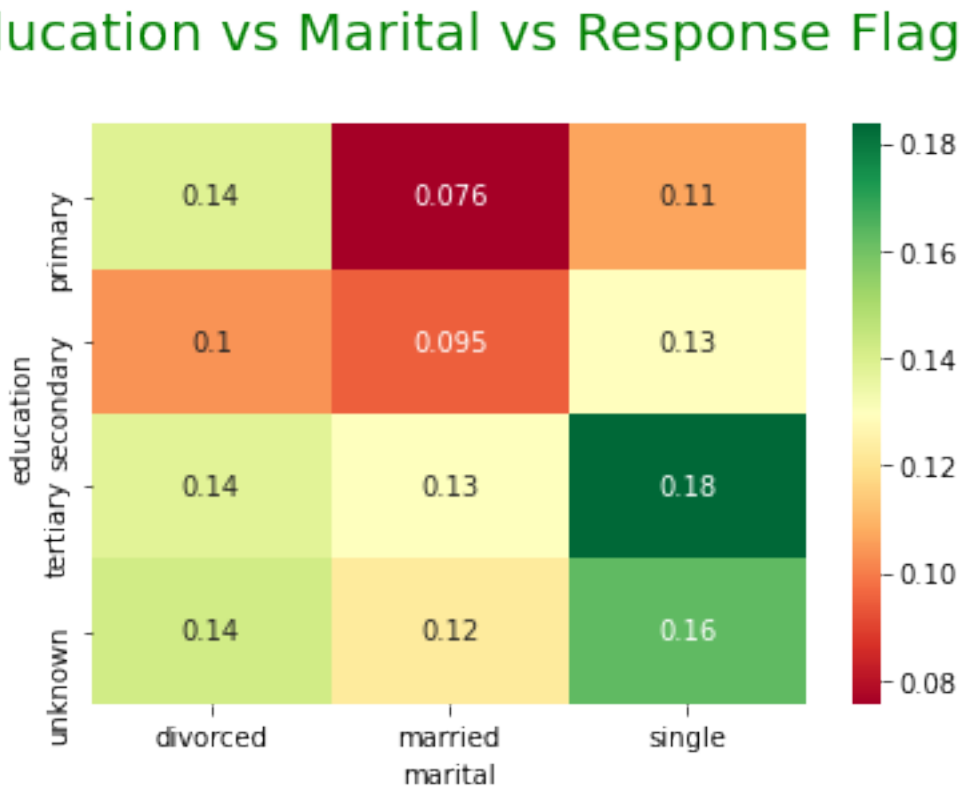
[95]:

	age	salary	balance	marital	targeted	default	housing	loan	contact	\
0	58	100000.0	2143	married	yes	no	yes	no	unknown	
1	44	60000.0	29	single	yes	no	yes	no	unknown	
2	33	120000.0	2	married	yes	no	yes	yes	unknown	
3	47	20000.0	1506	married	no	no	yes	no	unknown	
4	33	60000.0	1	single	no	no	no	no	unknown	

	day	month	duration	campaign	previous	poutcome	response	\
0	5	may, 2017	261 sec	1	0	unknown	no	
1	5	may, 2017	151 sec	1	0	unknown	no	
2	5	may, 2017	76 sec	1	0	unknown	no	

```
3     5  may, 2017    92 sec         1         0  unknown       no
4     5  may, 2017   198 sec         1         0  unknown       no

             job  education  response_flag
0     management   tertiary              0
1     technician  secondary              0
2   entrepreneur  secondary              0
3    blue-collar    unknown              0
4        unknown    unknown              0
```

[96]:
```
# Mapa de correlação
res1 = df.pivot_table(index = "education", columns = "marital", values =␣
 ↪"response_flag", aggfunc = "mean")
sns.heatmap(res1, annot = True, cmap = "RdYlGn")
plt.title("Education vs Marital vs Response Flag\n", fontdict = {'fontsize':␣
 ↪20, 'fontweight' : 5, 'color' : 'Green'})
plt.show()
```



3 Fim