Projeto2

November 16, 2022

1 Data Science Academy

2 Análise de Dados com Linguagem Python

2.1 Projeto 2

2.2 Análise de Dados de RH (Recursos Humanos)

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]:
```
!pip install -q missingno
```

[4]:
```
!pip install -q category_encoders
```

[5]:
```
!pip install -q plotly
```

[6]:
```
# Imports

# Manipulação de dados
import pandas as pd
import numpy as np

# Visualização
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import missingno

# Estatística
import scipy
from scipy.stats import normaltest
from scipy.stats import chi2_contingency
```

```
# Engenharia de Atributos
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder
from sklearn.compose import ColumnTransformer
import category_encoders as ce

# Ignore Warning
import sys
import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

[7]:
```
# Versões dos pacotes usados neste jupyter notebook
%reload_ext watermark
%watermark -a "Data Science Academy" --iversions
```

```
Author: Data Science Academy

seaborn          : 0.11.1
category_encoders: 2.2.2
sys              : 3.8.8 (default, Apr 13 2021, 12:59:45)
[Clang 10.0.0 ]
missingno        : 0.5.0
pandas           : 1.3.3
plotly           : 5.3.1
scipy            : 1.6.2
matplotlib       : 3.4.3
numpy            : 1.21.2
```

2.5 Carregando os Dados

[8]:
```
# Carrega o dataset
df = pd.read_csv("dataset/aug_train.csv")
```

[9]:
```
# Shape
df.shape
```

[9]:
```
(19158, 14)
```

[10]:
```
# Colunas
df.columns
```

[10]:
```
Index(['enrollee_id', 'city', 'city_development_index', 'gender',
       'relevent_experience', 'enrolled_university', 'education_level',
       'major_discipline', 'experience', 'company_size', 'company_type',
```

```
       'last_new_job', 'training_hours', 'target'],
      dtype='object')
```

[11]:
```
# Amostra dos dados
df.head()
```

[11]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	NaN	1	36	1.0
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	Pvt Ltd	>4	47	0.0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	NaN	never	83	0.0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	Pvt Ltd	never	52	1.0
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded Startup	4	8	0.0

[12]:
```
# Info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
```

```
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```

2.6 Análise Exploratória de Dados

[13]:
```
# Descrevendo os dados não numéricos
df.describe(include = object)
```

[13]:

	city	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job
count	19158	14650	19158	18772	18698	16345	19093	13220	13018	18735
unique	123	3	2	3	5	6	22	8	6	6
top	city_103	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	50-99	Pvt Ltd	1
freq	4355	13221	13792	13817	11598	14492	3286	3083	9817	8040

[14]:
```
# Descrevendo os dados numéricos
df.describe().drop(columns = ['enrollee_id', 'target'])
```

[14]:

	city_development_index	training_hours
count	19158.000000	19158.000000
mean	0.828848	65.366896
std	0.123362	60.058462
min	0.448000	1.000000
25%	0.740000	23.000000
50%	0.903000	47.000000
75%	0.920000	88.000000
max	0.949000	336.000000

- Em **city_development_index** (CDI), os valores médios são 0,828, mediana 0,903 e std

0,123. Isso significa que a maioria dos candidatos é de cidades bem desenvolvidas.

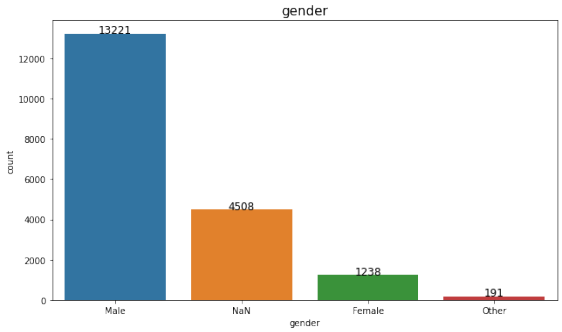
- Em **training_hours**, os valores médios são 65,367, mediana 47 e max 336. Isso significa que há mais candidatos com poucas horas de treinamento, mas alguns candidatos gastam muito tempo para fazer o treinamento.

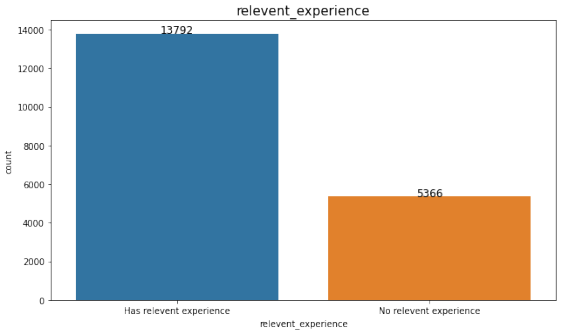
2.6.1 Visualizando as Variáveis Categóricas

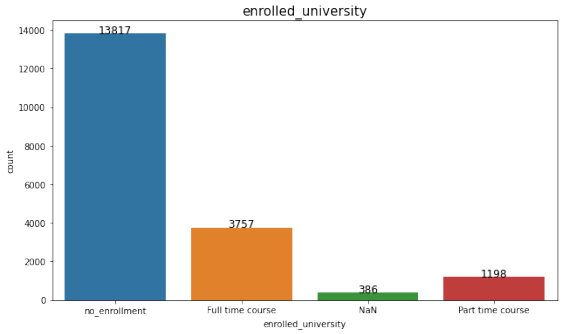
[15]:
```
list(df.columns.values)[3:12]
```

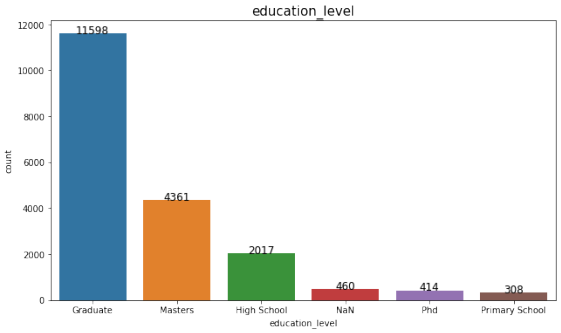
[15]:
```
['gender',
 'relevent_experience',
 'enrolled_university',
 'education_level',
 'major_discipline',
 'experience',
 'company_size',
 'company_type',
 'last_new_job']
```

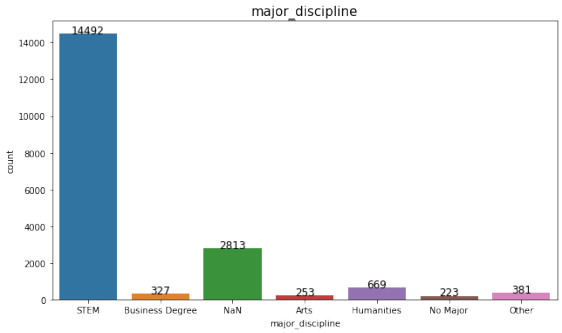
[16]:
```
# Plot

# Tamanho da figura
plt.figure(figsize = (18,30))

# Lista de colunas
column_list = list(df.columns.values)[3:12]

# Contador
A = 0

# Loop
for i in column_list:
    A += 1
    plt.subplot(5, 2, A)
    ax = sns.countplot(data = df.fillna('NaN'), x = i)
    plt.title(i, fontsize = 15)
    for p in ax.patches:
        ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha␣
 ↪= 'center', color = 'black', size = 12)
    if A >= 7:
        plt.xticks(rotation = 45)

# Layout
plt.tight_layout(h_pad = 2)
```

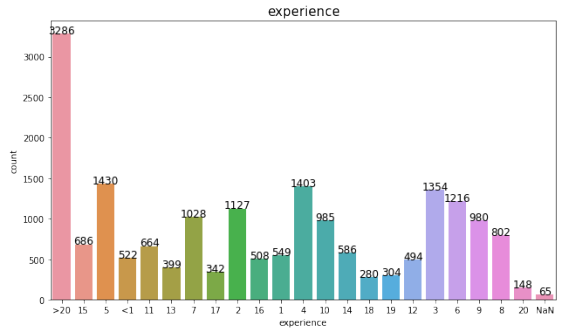

gender
13221
4508
1238
191
Male
NaN
Female
Other
gender
count

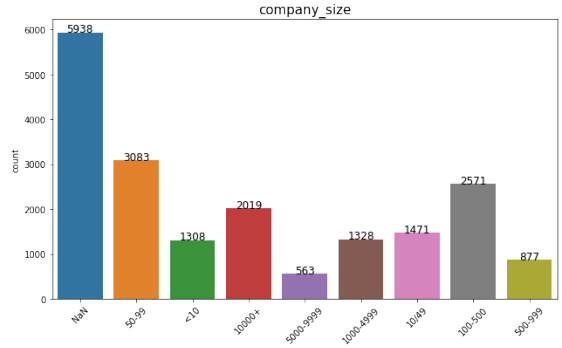

relevent_experience
13792
5366
Has relevent experience
No relevent experience
relevent_experience
count

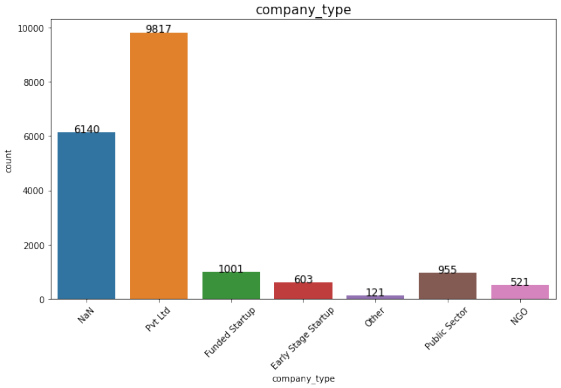

enrolled_university
13817
3757
386
1198
no_enrollment
Full time course
NaN
Part time course
enrolled_university
count

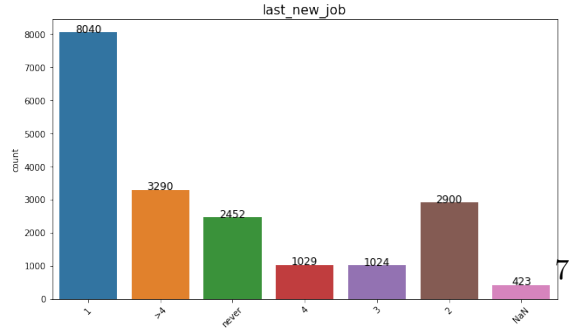

education_level
11598
4361
2017
460
414
308
Graduate
Masters
High School
NaN
Phd
Primary School
education_level
count


major_discipline
14492
327
2813
253
669
223
381
STEM
Business Degree
NaN
Arts
Humanities
No Major
Other
major_discipline
count


experience
3286
686
1430
522
664
399
1028
342
1127
508
549
1403
985
586
280
304
494
1354
1216
980
802
148
65
>20
15
5
<1
11
13
7
17
2
16
1
4
10
14
18
19
12
3
6
9
8
20
NaN
experience
count


company_size
5938
3083
1308
2019
563
1328
1471
2571
877
NaN
50-99
<10
10000+
5000-9999
1000-4999
10-49
100-500
500-999
company_size
count


company_type
9817
6140
1001
603
121
955
521
NaN
Pvt Ltd
Funded Startup
Early Stage Startup
Other
Public Sector
NGO
company_type
count


last_new_job
8040
3290
2452
1029
1024
2900
423
1
>4
never
4
3
2
NaN
last_new_job
count

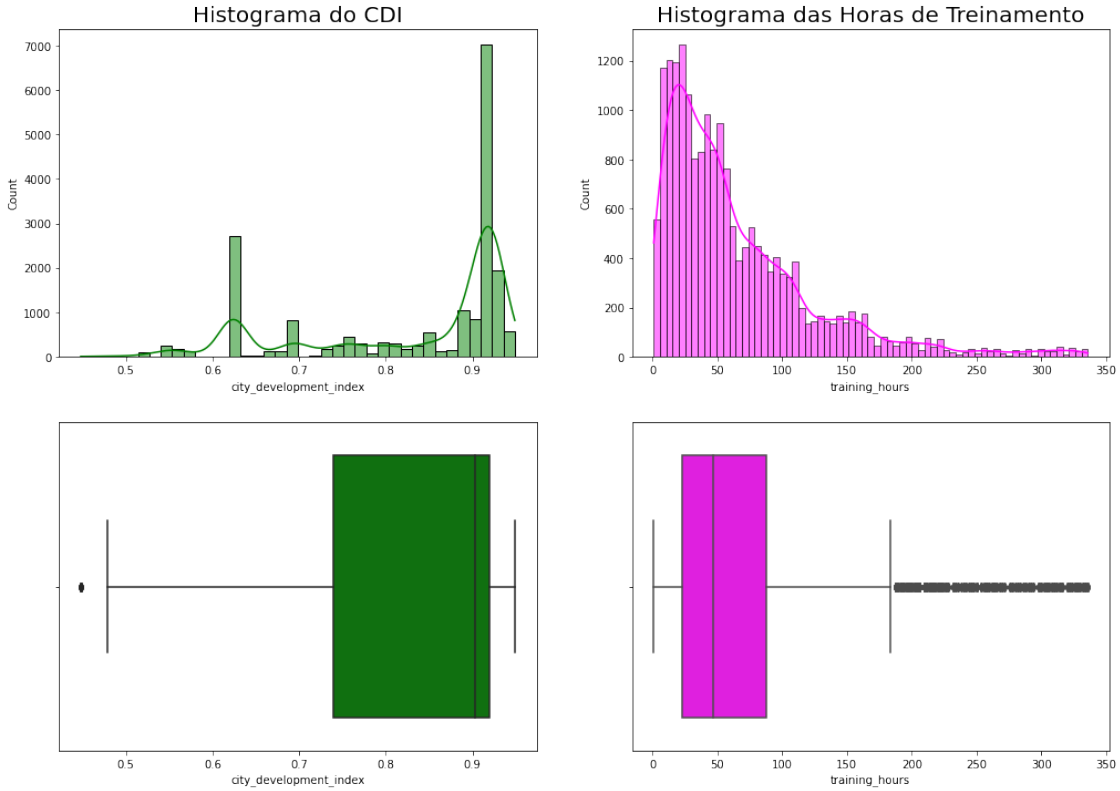
2.6.2 Verificando a Distribuição das Variáveis Numéricas

[17]:
```
# Descrevendo os dados numéricos
df.describe().drop(columns = ['enrollee_id', 'target'])
```

[17]:

	city_development_index	training_hours
count	19158.000000	19158.000000
mean	0.828848	65.366896
std	0.123362	60.058462
min	0.448000	1.000000
25%	0.740000	23.000000
50%	0.903000	47.000000
75%	0.920000	88.000000
max	0.949000	336.000000

[18]:
```
# Figura
plt.figure(figsize = (17,12))

# Subplots com histogramas
plt.subplot(221)
sns.color_palette("hls", 8)
sns.histplot(df['city_development_index'], kde = True, color = "green")
plt.title('Histograma do CDI', fontsize = 20)

plt.subplot(222)
sns.histplot(df['training_hours'], kde = True, color = "magenta")
plt.title('Histograma das Horas de Treinamento', fontsize = 20)

# Subplots com boxplots
plt.subplot(223)
sns.boxplot(df['city_development_index'], color = "green")

plt.subplot(224)
sns.boxplot(df['training_hours'], color = "magenta")

plt.show()
```



Em Estatística, a distribuição normal é uma das distribuições de probabilidade mais utilizadas para modelar fenômenos naturais. Isso se deve ao fato de que um grande número de fenômenos naturais apresenta sua distribuição de probabilidade tão proximamente normal, que a ela pode ser com sucesso referida, e, portanto, com adequado acerto por ela representada como se normal fosse.

A distribuição normal, também conhecida como distribuição gaussiana, é uma curva simétrica em torno do seu ponto médio, apresentando assim seu famoso formato de sino.

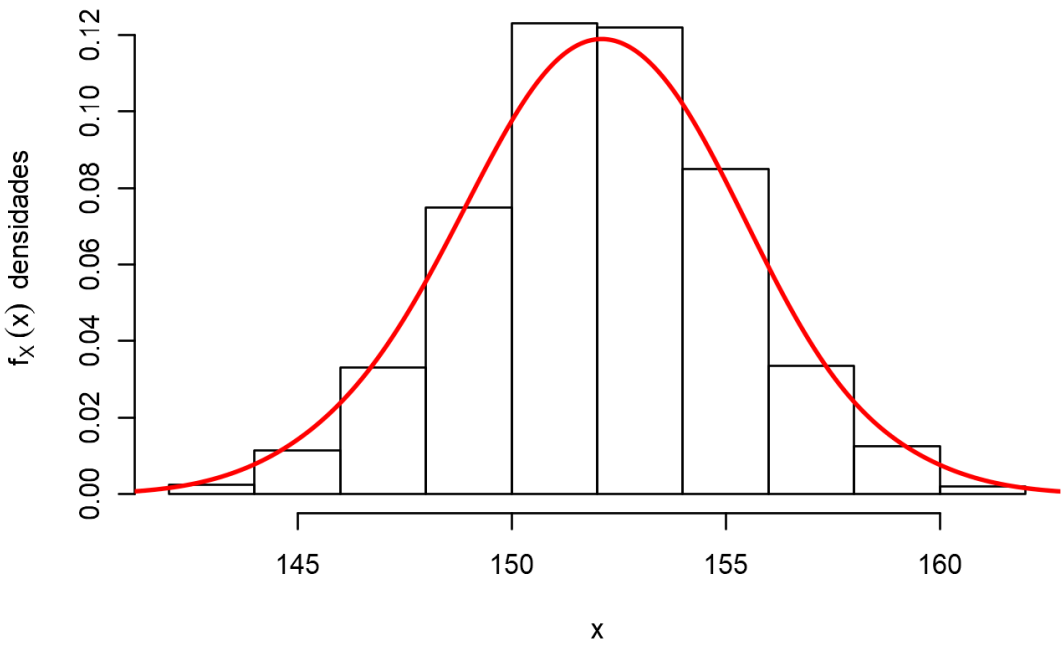
Uma distribuição estatística é uma função que define uma curva, e a área sob essa curva determina a probabilidade de ocorrer o evento por ela correlacionado.

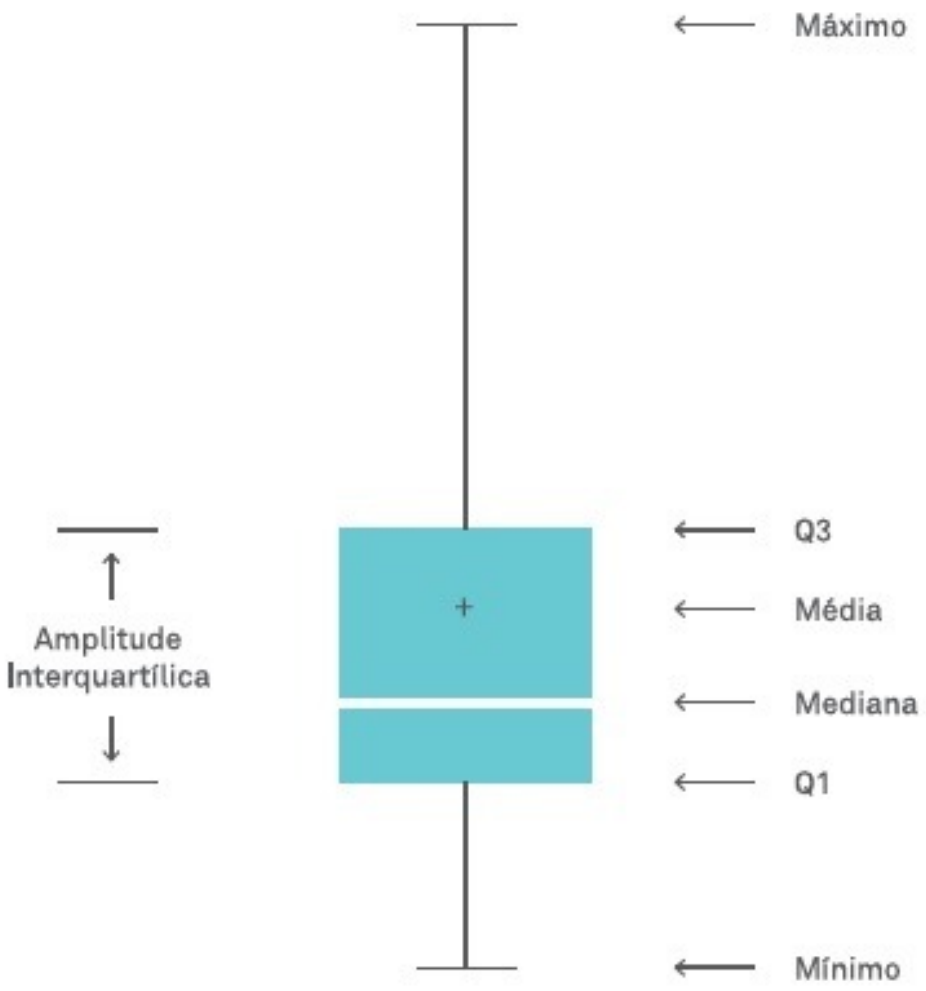
E o que é distribuição normal? É a mais importante dentre as distribuições estatísticas.

A curva de distribuição normal representa o comportamento de diversos processos nas empresas e muitos fenômenos comuns, como por exemplo, altura ou peso de uma população, a pressão sanguínea de um grupo de pessoas, o tempo que um grupo de estudantes gasta para realizar uma prova.

A distribuição normal pode ser usada para aproximar distribuições discretas de probabilidade, como por exemplo a distribuição binomial. Além disso, a distribuição normal serve também como base para a inferência estatística clássica.

Nela, a média, mediana e moda dos dados possuem o mesmo valor.


$f_X(x)$ densidades
0.00
0.02
0.04
0.06
0.08
0.10
0.12
145
150
155
160
x



[19]:
```
# Teste de Normalidade da Distribuição

# Lista com as variáveis numéricas
numerical_feature = ['city_development_index', 'training_hours']

# Loop
for i in numerical_feature:

    # Calcula a normalidade
    stats, pval = normaltest(df[i])

     # Checar valor-p
    if pval > 0.05:
        print(i, ': Distribuição Normal')
    else:
```

```
        print(i, ': Distribuição Não Normal')
```

```
city_development_index : Distribuição Não Normal
training_hours : Distribuição Não Normal
```

- As variáveis **city_development_index** e **training_hours** não seguem a distribuição normal. Para a análise numérica, usaremos o método não paramétrico.
- Dados de **training_hours** estão localizados principalmente no lado esquerdo do histograma. É um comportamento esperado porque as pessoas geralmente ficam animadas ao fazer o treinamento no início, mas nem todos que começam conseguem terminar uma maratona. ;-)

2.6.3 Correlação dos Dados

Correlação de Spearman Entre Variáveis Numéricas Analisamos a correlação entre as variáveis numéricas e entre as variáveis numéricas e a variável alvo (o que estamos querendo analisar).

[20]:
```
df.head()
```

[20]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	NaN	1	36	1.0
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	Pvt Ltd	>4	47	0.0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	NaN	never	83	0.0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	Pvt Ltd	never	52	1.0
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded Startup	4	8	0.0

[21]:
```
df.columns
```

[21]:
```
Index(['enrollee_id', 'city', 'city_development_index', 'gender',
       'relevent_experience', 'enrolled_university', 'education_level',
       'major_discipline', 'experience', 'company_size', 'company_type',
       'last_new_job', 'training_hours', 'target'],
      dtype='object')
```

[22]:
```
# Criamos uma cópia do dataframe original
df_numerical = df.copy()
```

[23]:
```
df_numerical["experience"].value_counts()
```

[23]:
```
>20    3286
5      1430
4      1403
3      1354
6      1216
2      1127
7      1028
10      985
9       980
8       802
15      686
11      664
14      586
1       549
<1      522
16      508
12      494
13      399
17      342
19      304
18      280
20      148
Name: experience, dtype: int64
```

[24]:
```
# Convertemos a variável experience para numérica
df_numerical["experience"] = np.where(df_numerical["experience"] == "<1", 1,␣
 ↪df_numerical["experience"])
df_numerical["experience"] = np.where(df_numerical["experience"] == ">20", 21,␣
 ↪df_numerical["experience"])
df_numerical["experience"] = df_numerical["experience"].astype(float)
```

[25]:
```
df_numerical["experience"].value_counts()
```

[25]:
```
21.0    3286
5.0     1430
4.0     1403
```

```
3.0     1354
6.0     1216
2.0     1127
1.0     1071
7.0     1028
10.0     985
9.0      980
8.0      802
15.0     686
11.0     664
14.0     586
16.0     508
12.0     494
13.0     399
17.0     342
19.0     304
18.0     280
20.0     148
Name: experience, dtype: int64
```

[26]:
```
df_numerical["last_new_job"].value_counts()
```

[26]:
```
1        8040
>4       3290
2        2900
never    2452
4        1029
3        1024
Name: last_new_job, dtype: int64
```

[27]:
```
# Convertemos a variável last_new_job para numérica
df_numerical["last_new_job"] = np.where(df_numerical["last_new_job"] ==␣
 ↪"never", 0, df_numerical["last_new_job"])
df_numerical["last_new_job"] = np.where(df_numerical["last_new_job"] == ">4",␣
 ↪5, df_numerical["last_new_job"])
df_numerical["last_new_job"] = df_numerical["last_new_job"].astype(float)
```

[28]:
```
df_numerical["last_new_job"].value_counts()
```

[28]:
```
1.0    8040
5.0    3290
2.0    2900
0.0    2452
4.0    1029
3.0    1024
Name: last_new_job, dtype: int64
```

[29]:
```
df_numerical.head()
```

[29]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	21.0	NaN	NaN	1.0	36	1.0
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15.0	50-99	Pvt Ltd	5.0	47	0.0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5.0	NaN	NaN	0.0	83	0.0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	1.0	NaN	Pvt Ltd	0.0	52	1.0
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	21.0	50-99	Funded Startup	4.0	8	0.0

[30]:
```
df_numerical.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  float64
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
```

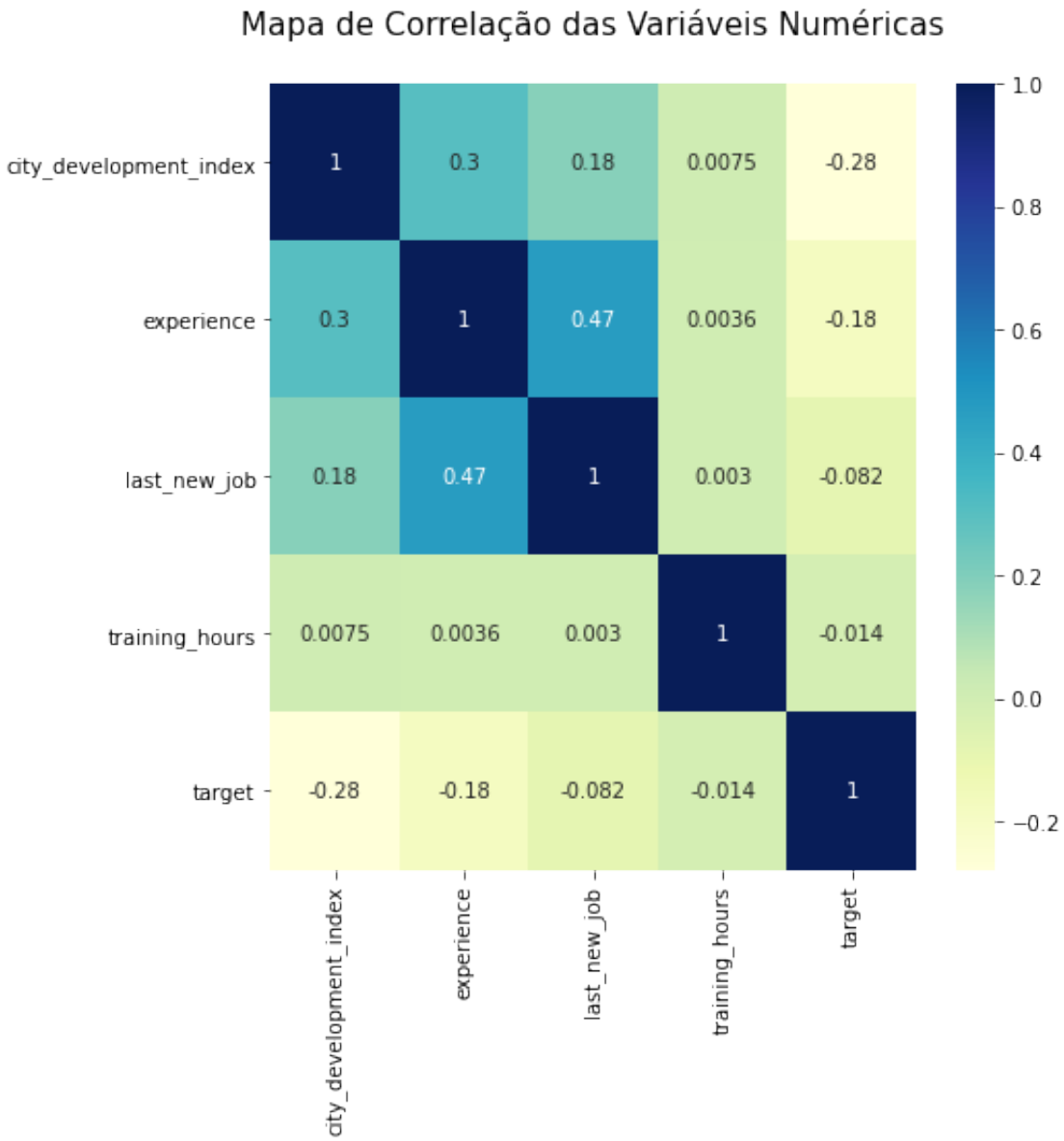
```
 11  last_new_job            18735 non-null  float64
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
dtypes: float64(4), int64(2), object(8)
memory usage: 2.0+ MB
```

[31]:
```
df_numerical.drop("enrollee_id", axis = 1).corr("spearman")
```

[31]:

	city_development_index	experience	last_new_job	training_hours	target
city_development_index	1.000000	0.300997	0.182698	0.007491	-0.279165
experience	0.300997	1.000000	0.473284	0.003569	-0.183721
last_new_job	0.182698	0.473284	1.000000	0.002959	-0.082045
training_hours	0.007491	0.003569	0.002959	1.000000	-0.014126
target	-0.279165	-0.183721	-0.082045	-0.014126	1.000000

[32]:
```
# Heatmap
plt.figure(figsize = (7,7))
sns.heatmap(df_numerical.drop("enrollee_id", axis = 1).corr("spearman"), annot␣
 ↪= True, cmap = "YlGnBu")
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```



Pela correlação de spearman, last_new_job e experience têm correlação média (0,473), enquanto outros têm correlação fraca.

As variáveis last_new_job e training_hours tem correlação próxima de zero com a variável alvo e poderiam ser descartadas.

Para a relação entre as variáveis categóricas e a variável alvo, usaremos WOE e IV.

2.6.4 Weight of Evidence (WOE) e Information Value (IV)

Este conjunto de dados contém mais dados com tipo categórico do que tipo numérico. Usaremos recursos categóricos nominais para WOE e IV.

Interpretamos o resultado de IV assim:

```
Information Value, Poder de Previsão

< 0.02, não deve ser usado para previsão
0.02 - 0.1, preditor fraco
0.1 - 0.3, preditor médio
0.3 - 0.5, preditor forte
> 0.5, parece bom demais para ser verdade
```

[33]:
```
# Loop
for i in df.drop(columns = ['target',
                            'enrollee_id',
                            'city',
                            'city_development_index',
                            'training_hours',
                            'experience',
                            'last_new_job',
                            'company_size']).columns:

    df_woe_iv = (pd.crosstab(df[i], df['target'], normalize = 'columns')
                 .assign(woe = lambda dfx: np.log(dfx[1] / dfx[0]))
                 .assign(iv = lambda dfx: np.sum(dfx['woe'] * (dfx[1]-dfx[0]))))


  ␣
 ↪print(df_woe_iv,'\n------------------------------------------------------------')
```

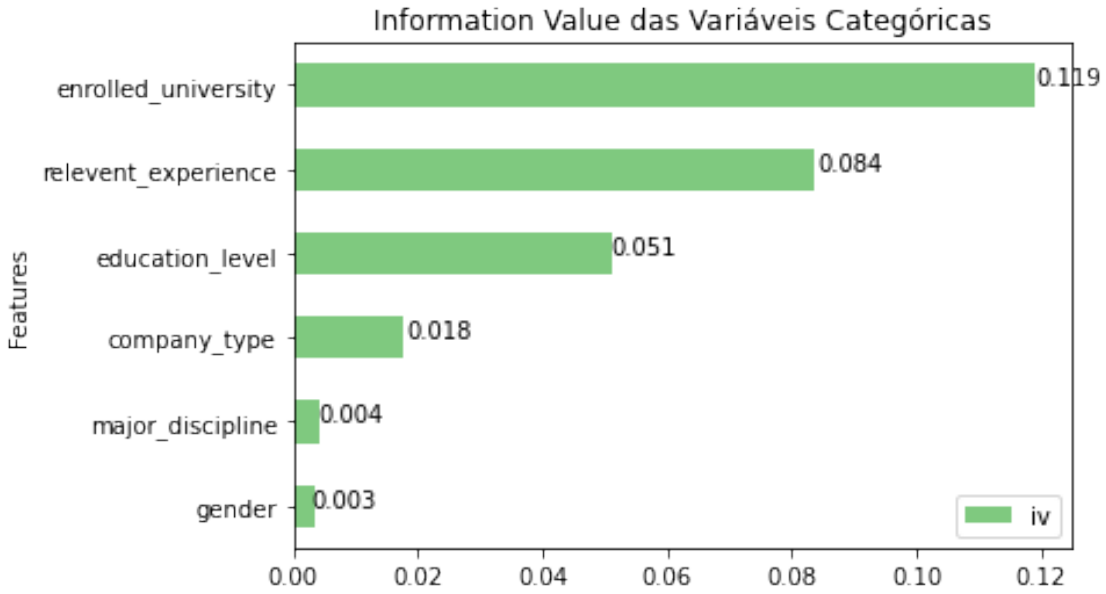
```
target      0.0       1.0       woe        iv
gender
Female  0.08098  0.096222  0.172452  0.003337
Male    0.90650  0.889020 -0.019471  0.003337
Other   0.01252  0.014758  0.164458  0.003337
------------------------------------------------------------
target                         0.0       1.0       woe        iv
relevent_experience
Has relevent experience   0.753147  0.619845 -0.194790  0.083523
No relevent experience    0.246853  0.380155  0.431784  0.083523
------------------------------------------------------------
target                      0.0       1.0       woe        iv
enrolled_university
Full time course       0.164754  0.307477  0.623947  0.118886
Part time course       0.063465  0.064890  0.022210  0.118886
no_enrollment          0.771781  0.627632 -0.206746  0.118886
------------------------------------------------------------
target                  0.0       1.0       woe       iv
education_level
Graduate           0.595579  0.694415  0.153535  0.05117
High School        0.115722  0.084314 -0.316640  0.05117
Masters            0.244278  0.200086 -0.199562  0.05117
```

```
Phd              0.025383  0.012412 -0.715448  0.05117
Primary School   0.019037  0.008774 -0.774636  0.05117
------------------------------------------------------------
target                0.0       1.0       woe        iv
major_discipline
Arts             0.016506  0.012535 -0.275145  0.004148
Business Degree  0.019889  0.020341  0.022431  0.004148
Humanities       0.043575  0.033349 -0.267456  0.004148
No Major         0.013865  0.013009 -0.063750  0.004148
Other            0.023026  0.024125  0.046641  0.004148
STEM             0.883139  0.896641  0.015173  0.004148
------------------------------------------------------------
target                    0.0       1.0       woe        iv
company_type
Early Stage Startup  0.043388  0.059340  0.313091  0.017772
Funded Startup       0.081035  0.058504 -0.325790  0.017772
NGO                  0.039906  0.040535  0.015639  0.017772
Other                0.008659  0.012119  0.336169  0.017772
Public Sector        0.070118  0.087756  0.224385  0.017772
Pvt Ltd              0.756894  0.741747 -0.020215  0.017772
------------------------------------------------------------
```

[34]:
```
# Plot do Information Value

# Variáveis categóricas
columns_cat = df.drop(columns = ['target',
                                 'enrollee_id',
                                 'city',
                                 'city_development_index',
                                 'training_hours',
                                 'experience',
                                 'last_new_job',
                                 'company_size']).columns

# Lista para o IV
iv = []

# Loop
for i in columns_cat:
    df_woe_iv = (pd.crosstab(df[i], df['target'], normalize = 'columns')
                 .assign(woe = lambda dfx: np.log(dfx[1] / dfx[0]))
                 .assign(iv = lambda dfx: np.sum(dfx['woe']*(dfx[1]-dfx[0]))))
    iv.append(df_woe_iv['iv'][0])

# Dataframe
df_iv = pd.DataFrame({'Features':columns_cat,'iv':iv}).set_index('Features').
 ↪sort_values(by = 'iv')
```

```
# Plot
# Figura
plt.figure(figsize = (10,12))
df_iv.plot(kind = 'barh', title = 'Information Value das Variáveis␣
 ↪Categóricas', colormap = "Accent")
for index, value in enumerate(list(round(df_iv["iv"],3))):
    plt.text((value), index, str(value))
plt.legend(loc  = "lower right")
plt.show()
```

```
<Figure size 720x864 with 0 Axes>
```

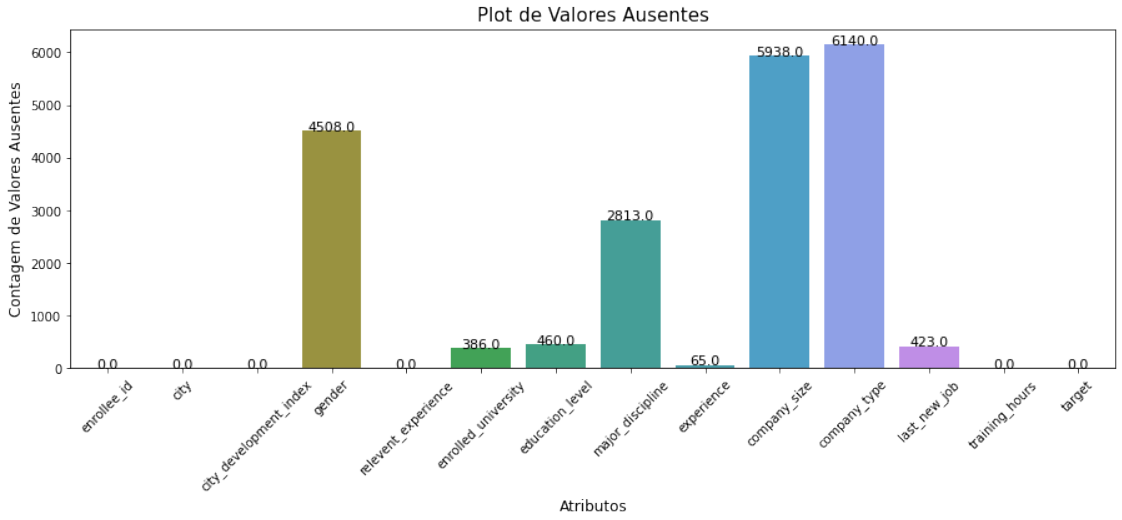


- No gráfico acima, podemos ver a ordem dos recursos com base em seu poder preditivo em relação ao alvo.
- Com base em seu valor IV, `enrolled_university` é um preditor médio, `relevent_experience` e `education_level` são preditores fracos e os outros são inúteis para a previsão.

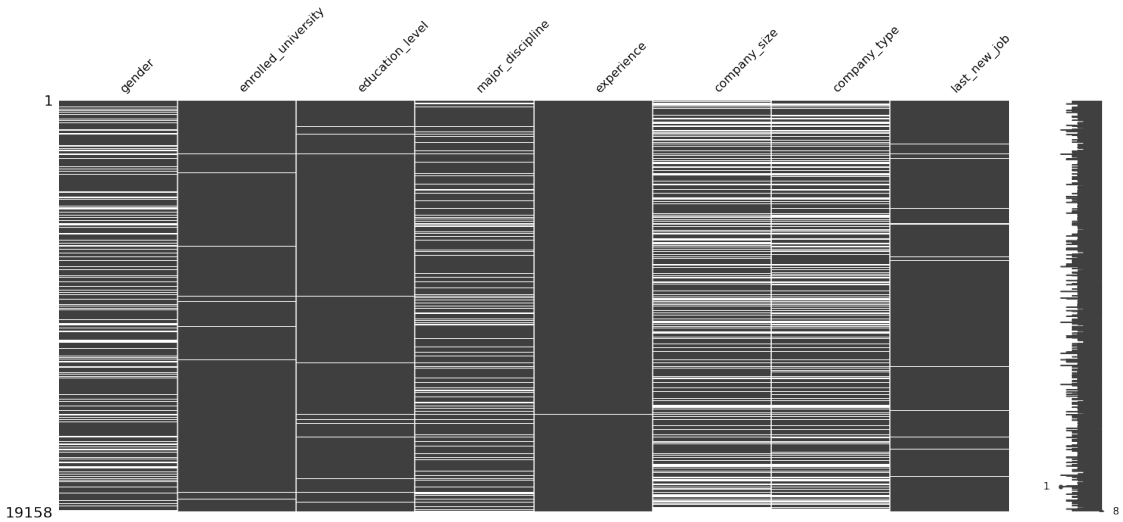
2.6.5 Identificando Valores Ausentes

[35]:
```
# Valores ausentes por coluna
null_df = df.isna().sum().reset_index()

# Figura
ax = plt.figure(figsize = (15,5))

# Barplot
```

```
ax = sns.barplot(null_df['index'], null_df[0], palette = 'husl')
plt.xlabel('Atributos', fontsize = 12)
plt.ylabel('Contagem de Valores Ausentes', fontsize = 12)
plt.xticks(rotation = 45)
plt.title("Plot de Valores Ausentes", fontsize = 15)

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, (p.get_height())), ha =␣
 ↪'center', color = 'black', size = 11)

plt.show()
```



[36]:
```
# Gera a visualização

# Dataframe
df_nan = pd.DataFrame(df.isna().sum())

# Plot - Mapa de Valores Ausentes
if df.isna().any(axis = None):
    missingno.matrix(df[df_nan[df_nan[0]>0].index])
    plt.show()
```



Valores ausentes em cada coluna têm um padrão aleatório.

2.6.6 Identificando Valores Duplicados

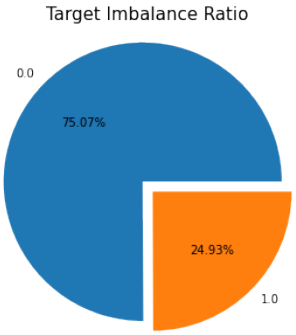
[37]:
```
# Checando valores duplicados (não há)
df['enrollee_id'].duplicated().sum()
```

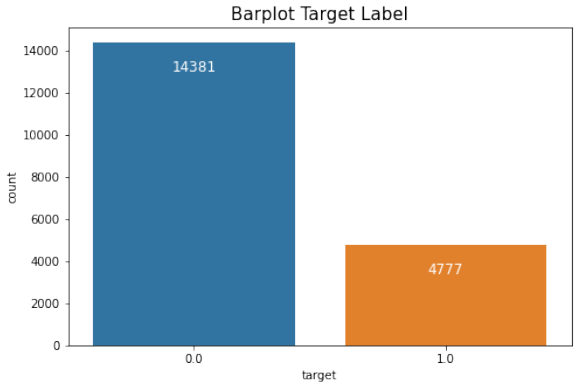
[37]: 0

2.6.7 Identificando Dados Desbalanceados

[38]:
```
# Figura
plt.figure(figsize = (17,(100)/20))

plt.subplot(121)

plt.pie(round(df['target'].value_counts() / len(df) * 100, 2),
        labels = list(df['target'].value_counts().index),
        autopct = "%.2f%%",
        explode = (0,0.1))

plt.axis("equal")
plt.title("Target Imbalance Ratio", size = 15)

plt.subplot(122)
ax = sns.countplot(data = df, x = 'target')
plt.title("Barplot Target Label", fontsize = 15)
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}',
                (p.get_x()+0.4,
```

```
                 p.get_height()),
                ha = 'center',
                va = 'top',
                color = 'white',
                size = 12)
```





Os dados estão desbalanceados em relação à variável target.

2.7 Limpeza e Processamento dos Dados

Em que momento fazemos a limpeza dos dados?

2.7.1 Tratando os Valores Ausentes

[39]:
```
df.columns
```

[39]:
```
Index(['enrollee_id', 'city', 'city_development_index', 'gender',
       'relevent_experience', 'enrolled_university', 'education_level',
       'major_discipline', 'experience', 'company_size', 'company_type',
       'last_new_job', 'training_hours', 'target'],
      dtype='object')
```

[40]:
```
colunas_manter = ['city_development_index',
                  'experience',
                  'enrolled_university',
                  'relevent_experience',
                  'education_level',
                  'company_type',
                  'major_discipline',
                  'target']
```

[41]:
```
new_df = df[colunas_manter]
```

[42]:
```
new_df.head()
```

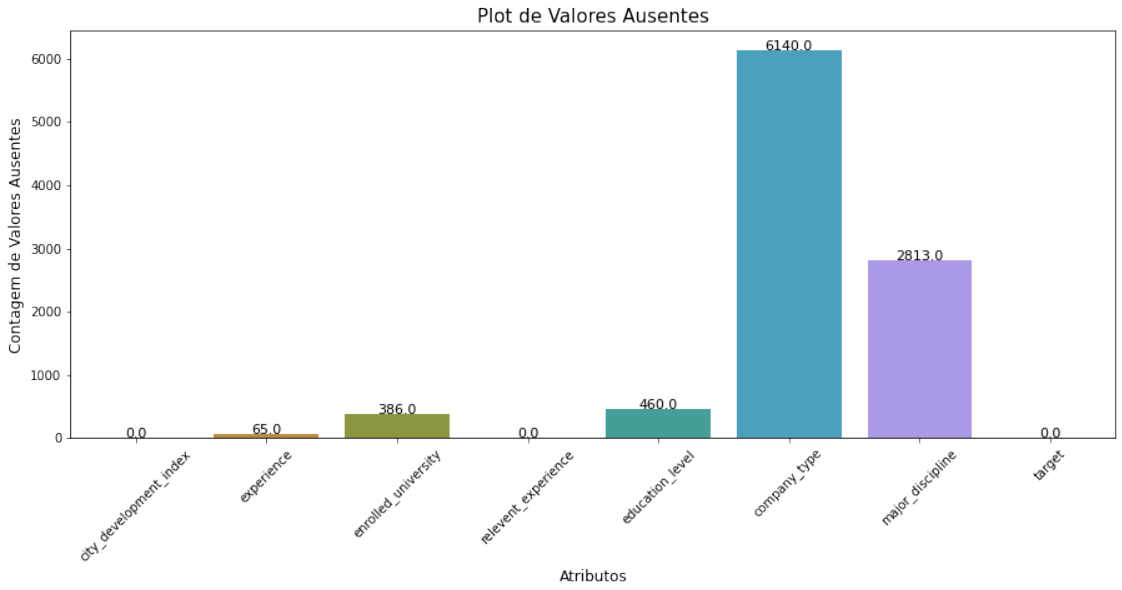
[42]:
```
   city_development_index experience enrolled_university  \
0                   0.920        >20       no_enrollment
1                   0.776         15       no_enrollment
2                   0.624          5    Full time course
3                   0.789         <1                 NaN
4                   0.767        >20       no_enrollment

       relevent_experience education_level    company_type major_discipline  \
0  Has relevent experience        Graduate             NaN             STEM
1   No relevent experience        Graduate         Pvt Ltd             STEM
2   No relevent experience        Graduate             NaN             STEM
3   No relevent experience        Graduate         Pvt Ltd  Business Degree
4  Has relevent experience         Masters  Funded Startup             STEM

   target
0     1.0
1     0.0
2     0.0
3     1.0
4     0.0
```

[43]:
```
df.head()
```

[43]:
```
   enrollee_id      city  city_development_index gender  \
0         8949  city_103                   0.920   Male
1        29725   city_40                   0.776   Male
2        11561   city_21                   0.624    NaN
3        33241  city_115                   0.789    NaN
4          666  city_162                   0.767   Male

       relevent_experience enrolled_university education_level  \
0  Has relevent experience       no_enrollment        Graduate
1   No relevent experience       no_enrollment        Graduate
2   No relevent experience    Full time course        Graduate
3   No relevent experience                 NaN        Graduate
4  Has relevent experience       no_enrollment         Masters

  major_discipline experience company_size    company_type last_new_job  \
0             STEM        >20          NaN             NaN            1
1             STEM         15        50-99         Pvt Ltd           >4
2             STEM          5          NaN             NaN        never
3  Business Degree         <1          NaN         Pvt Ltd        never
4             STEM        >20        50-99  Funded Startup            4

   training_hours  target
0              36     1.0
1              47     0.0
```

```
2              83     0.0
3              52     1.0
4               8     0.0
```

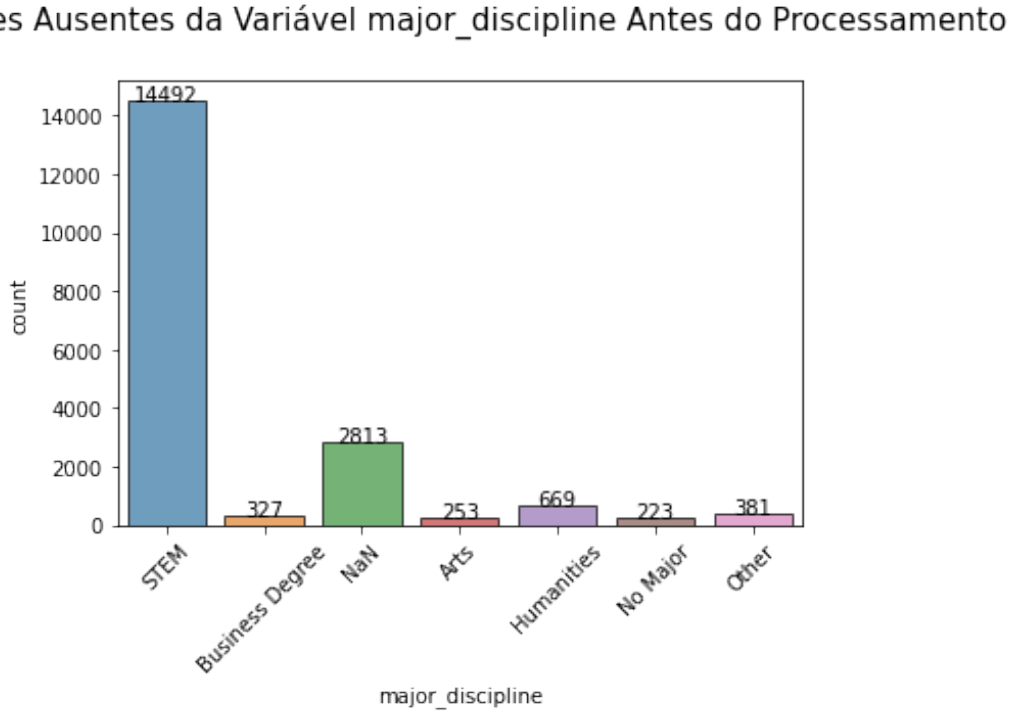
[44]:
```
# Valores ausentes por coluna
null_df = new_df.isna().sum().reset_index()

# Figura
ax = plt.figure(figsize = (15,6))

# Barplot
ax = sns.barplot(null_df['index'], null_df[0], palette = 'husl')
plt.xlabel('Atributos', fontsize = 12)
plt.ylabel('Contagem de Valores Ausentes', fontsize = 12)
plt.xticks(rotation = 45)
plt.title("Plot de Valores Ausentes", fontsize = 15)

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, (p.get_height())), ha =␣
 ↪'center', color = 'black', size = 11)

plt.show()
```



Variável major_discipline

[45]:
```
# Valores Ausentes da Variável major_discipline Antes do Processamento
sns.countplot(data = new_df.fillna('NaN'), x = 'major_discipline', alpha = 0.7,␣
 ↪edgecolor = 'black')
```

```
plt.xticks(rotation = 45)
bound = ax.get_xbound()
ax = plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title("Valores Ausentes da Variável major_discipline Antes do␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```



[46]:
```
# Relação entre major_discipline x education_level
print('\nTotal de Valores Ausentes na Variável major_discipline:',␣
 ↪new_df['major_discipline'].isna().sum())
print('\nProporção de Valores Ausentes na Variável education_level:')
new_df[new_df['major_discipline'].isna()]['education_level'].
 ↪value_counts(dropna = False)
```

```
Total de Valores Ausentes na Variável major_discipline: 2813

Proporção de Valores Ausentes na Variável education_level:
```

[46]:
```
High School       2017
NaN                460
```

```
Primary School     308
Graduate            22
Masters              6
Name: education_level, dtype: int64
```

Dentre os registros com valores ausentes, vemos a proporção na variável education_level.

Preencheremos os valores NA da variável `major_discipline` com `Non Degree`.

[47]:
```
# Cria o índice
nan_index = (new_df[(new_df['major_discipline'].isna()) &␣
 ↪((new_df['education_level']=='High School') | (new_df['education_level'].
 ↪isna()) | (new_df['education_level']=='Primary School'))]).index
```

[48]:
```
len(nan_index)
```

[48]:
```
2785
```

[49]:
```
# Imputação do valor ausente
new_df['major_discipline'][nan_index] = 'Non Degree'
```

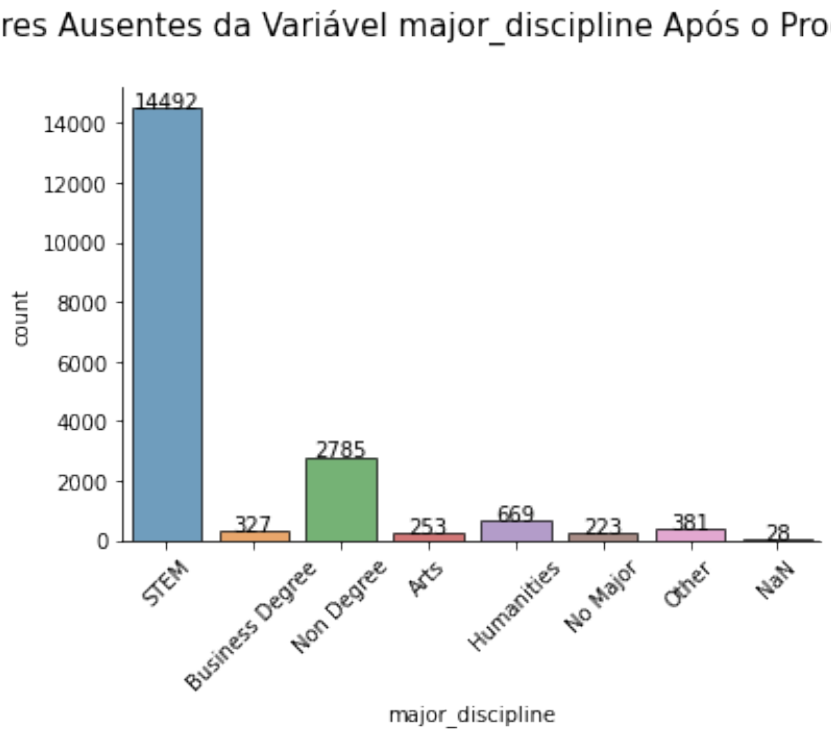
[50]:
```
print('Total de Valores Ausentes na Variável major_discipline:',␣
 ↪new_df['major_discipline'].isna().sum())
new_df['major_discipline'].value_counts(dropna = False)
```

```
Total de Valores Ausentes na Variável major_discipline: 28
```

[50]:
```
STEM               14492
Non Degree          2785
Humanities           669
Other                381
Business Degree      327
Arts                 253
No Major             223
NaN                   28
Name: major_discipline, dtype: int64
```

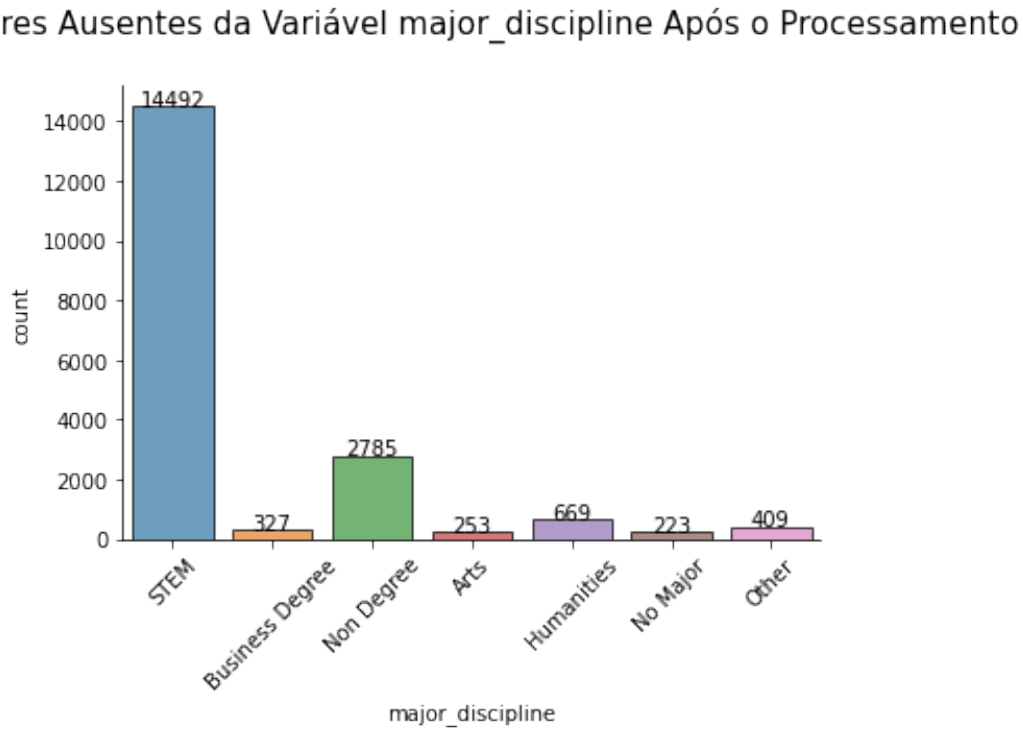
[51]:
```
# Valores Ausentes da Variável major_discipline Após o Processamento
sns.countplot(data = new_df.fillna('NaN'), x = 'major_discipline', alpha = 0.7,␣
 ↪edgecolor = 'black')
sns.despine()
plt.xticks(rotation=45)
bound=ax.get_xbound()
ax=plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title(" Valores Ausentes da Variável major_discipline Após o␣
 ↪Processamento\n", fontsize = 15)
```

```
plt.show()
```



[52]:
```
# Para os valores ausentes restantes preenchemos com 'Other'
new_df[new_df['major_discipline'].isna()]  = 'Other'
```

[53]:
```
# Valores Ausentes da Variável major_discipline Após o Processamento
sns.countplot(data = new_df.fillna('NaN'), x = 'major_discipline', alpha = 0.7,␣
 ↪edgecolor = 'black')
sns.despine()
plt.xticks(rotation=45)
bound=ax.get_xbound()
ax=plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title(" Valores Ausentes da Variável major_discipline Após o␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```

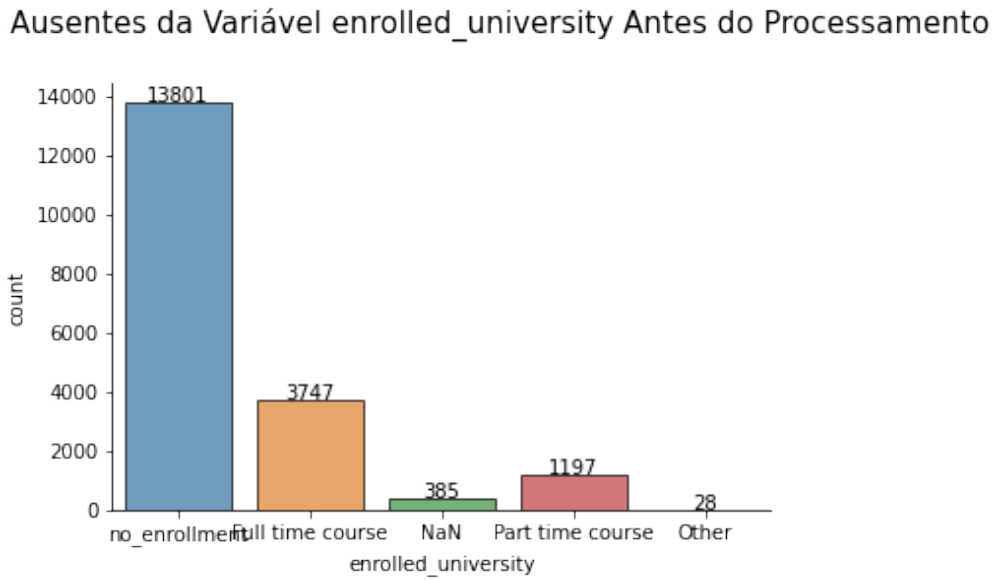


[54]:
```
new_df.head()
```

[54]:
```
   city_development_index experience enrolled_university  \
0                    0.92        >20       no_enrollment
1                   0.776         15       no_enrollment
2                   0.624          5    Full time course
3                   0.789         <1                 NaN
4                   0.767        >20       no_enrollment

        relevent_experience education_level    company_type major_discipline  \
0  Has relevent experience        Graduate             NaN             STEM
1   No relevent experience        Graduate         Pvt Ltd             STEM
2   No relevent experience        Graduate             NaN             STEM
3   No relevent experience        Graduate         Pvt Ltd  Business Degree
4  Has relevent experience         Masters  Funded Startup             STEM

  target
0    1.0
1    0.0
2    0.0
3    1.0
4    0.0
```

Variável enrolled_university

[55]:
```
# Plot
sns.countplot(data = new_df.fillna('NaN'), x = 'enrolled_university', alpha = 0.
 ↪7, edgecolor = 'black')
sns.despine()
plt.xticks()
bound=ax.get_xbound()
ax=plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title("Valores Ausentes da Variável enrolled_university Antes do␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```



[56]:
```
print('\nTotal de Valores Ausentes na Variável enrolled_university:',␣
 ↪new_df['enrolled_university'].isna().sum())
print('\nProporção de Valores Ausentes na Variável education_level:')
new_df[new_df['enrolled_university'].isna()]['education_level'].
 ↪value_counts(dropna = False)
```

```
Total de Valores Ausentes na Variável enrolled_university: 385

Proporção de Valores Ausentes na Variável education_level:
```

[56]:
```
Graduate          170
NaN               100
Masters            53
High School        47
Primary School      9
Phd                 6
Name: education_level, dtype: int64
```

Candidatos com `Primary School` como `education_level` não estão qualificados para ingressar na universidade. Portanto, preencheremos valores `NaN` em `education_level` para `Primary Grad`.

[57]:
```
# Prepara o índice
nan_index = (new_df[(new_df['enrolled_university'].isna()) &␣
 ↪(new_df['education_level']=='Primary School')]).index
```

[58]:
```
len(nan_index)
```

[58]:
```
9
```

[59]:
```
# Imputação de valores ausentes
new_df['enrolled_university'][nan_index] = 'Primary Grad'
```

[60]:
```
print('Total de Valores Ausentes:', new_df['enrolled_university'].isna().sum())
new_df[new_df['enrolled_university'].isna()]['education_level'].
 ↪value_counts(dropna = False)
```

```
Total de Valores Ausentes: 376
```

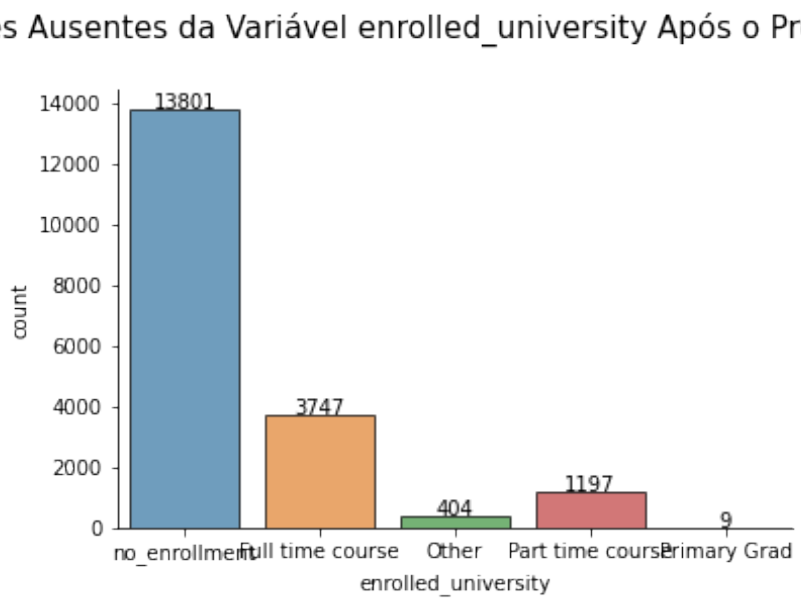
[60]:
```
Graduate       170
NaN            100
Masters         53
High School     47
Phd              6
Name: education_level, dtype: int64
```

[61]:
```
# Prepara o índice
nan_index = new_df[(new_df['enrolled_university'].isna())].index
```

[62]:
```
# O restante colocamos como 'Other'
new_df['enrolled_university'][nan_index] = 'Other'
```

[63]:
```
# Plot
sns.countplot(data = new_df.fillna('NaN'), x = 'enrolled_university', alpha = 0.
 ↪7, edgecolor = 'black')
sns.despine()
plt.xticks()
bound=ax.get_xbound()
ax=plt.gca()
for p in ax.patches:
```

```
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title("Valores Ausentes da Variável enrolled_university Após o␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```



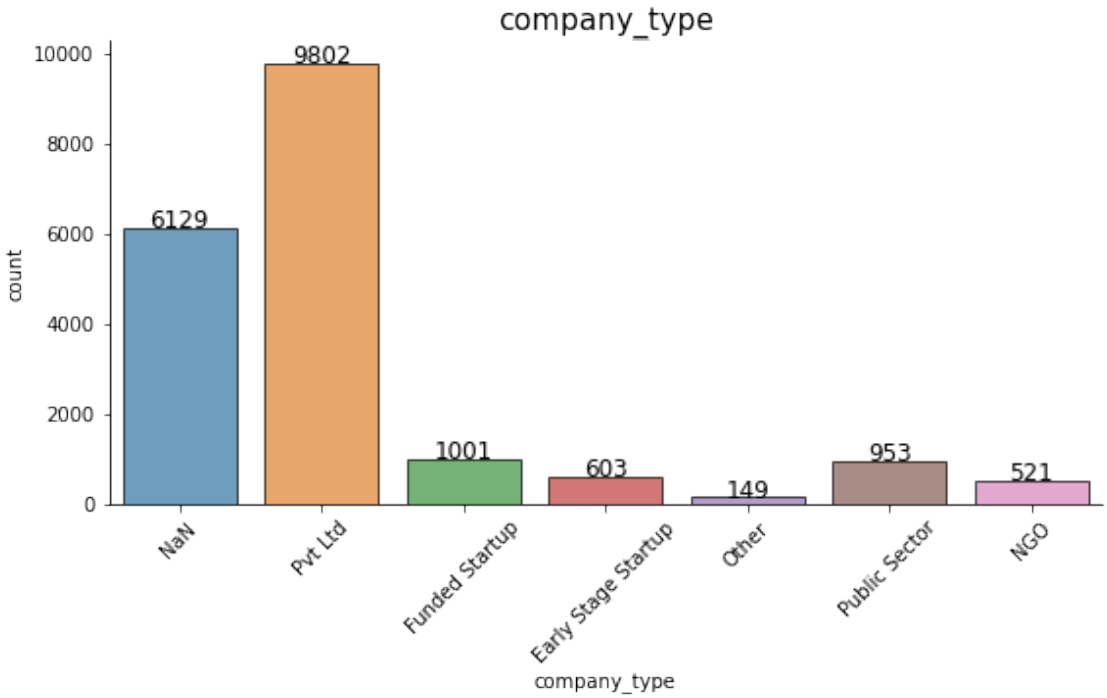
[64]:
```
new_df.head()
```

[64]:
```
  city_development_index experience enrolled_university  \
0                   0.92        >20       no_enrollment
1                  0.776         15       no_enrollment
2                  0.624          5    Full time course
3                  0.789         <1              Other
4                  0.767        >20       no_enrollment

       relevent_experience education_level    company_type major_discipline  \
0  Has relevent experience        Graduate             NaN             STEM
1   No relevent experience        Graduate         Pvt Ltd             STEM
2   No relevent experience        Graduate             NaN             STEM
3   No relevent experience        Graduate         Pvt Ltd  Business Degree
4  Has relevent experience         Masters  Funded Startup             STEM

  target
0    1.0
1    0.0
2    0.0
3    1.0
```

```
4    0.0
```

Variável company_type

[65]:
```
# Plot
plt.figure(figsize = (20, 20))
column_list = ['company_type']
A = 0
for i in column_list:
    A+=1
    plt.subplot(4,2,A)
    ax = sns.countplot(data = new_df.fillna('NaN'), x = i, alpha = 0.7,␣
 ↪edgecolor = 'black')
    sns.despine()
    plt.title(i, fontsize = 15)
    for p in ax.patches:
        ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha␣
 ↪= 'center', color = 'black', size = 12)
    if A >=0:
        plt.xticks(rotation = 45)
```



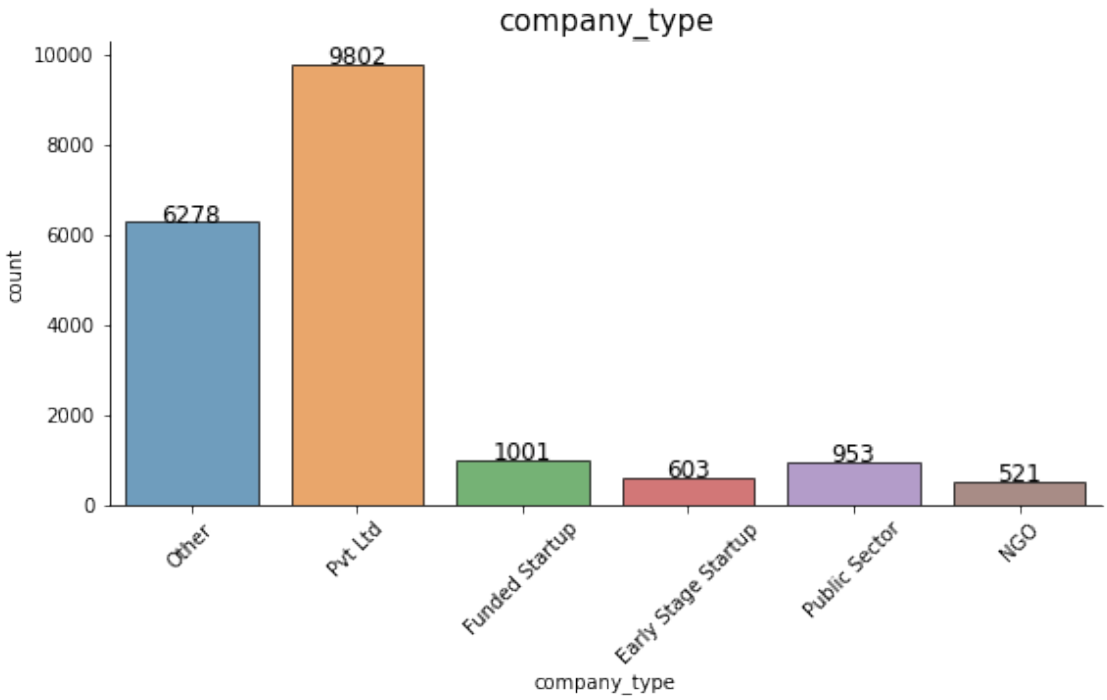
[66]:
```
new_df['company_type'].value_counts(dropna = False)
```

[66]:
```
Pvt Ltd                9802
NaN                    6129
Funded Startup         1001
Public Sector           953
Early Stage Startup     603
NGO                     521
Other                   149
Name: company_type, dtype: int64
```

[67]:
```
# Índice
nan_index = new_df[(new_df['company_type'].isna())].index
```

[68]:
```
# Imputação dos valores NaN com 'Other'
new_df['company_type'][nan_index] = 'Other'
```

[69]:
```
# Plot
plt.figure(figsize = (20, 20))
column_list = ['company_type']
A = 0
for i in column_list:
    A+=1
    plt.subplot(4,2,A)
    ax = sns.countplot(data = new_df.fillna('NaN'), x = i, alpha = 0.7,␣
 ↪edgecolor = 'black')
    sns.despine()
    plt.title(i, fontsize = 15)
    for p in ax.patches:
        ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha␣
 ↪= 'center', color = 'black', size = 12)
    if A >=0:
        plt.xticks(rotation = 45)
```



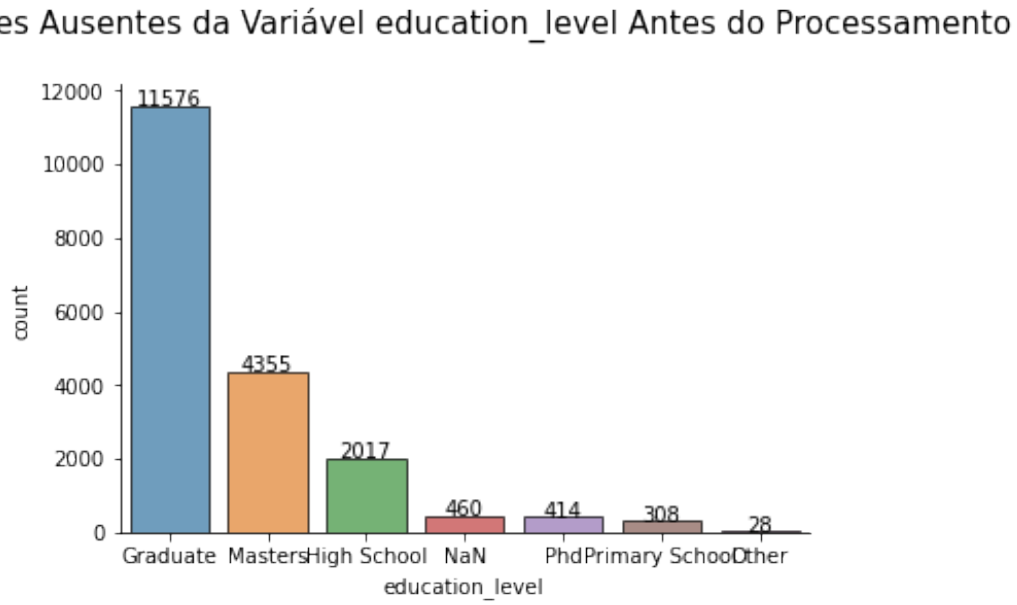
[70]:
```
new_df.head()
```

[70]:

	city_development_index	experience	enrolled_university	relevent_experience	education_level	company_type	major_discipline	target
0	0.92	>20	no_enrollment	Has relevent experience	Graduate	Other	STEM	1.0
1	0.776	15	no_enrollment	No relevent experience	Graduate	Pvt Ltd	STEM	0.0
2	0.624	5	Full time course	No relevent experience	Graduate	Other	STEM	0.0
3	0.789	<1	Other	No relevent experience	Graduate	Pvt Ltd	Business Degree	1.0
4	0.767	>20	no_enrollment	Has relevent experience	Masters	Funded Startup	STEM	0.0

Variável education_level

[71]:
```
# Plot
sns.countplot(data = new_df.fillna('NaN'), x = 'education_level', alpha = 0.7,␣
 ↪edgecolor = 'black')
sns.despine()
plt.xticks()
bound=ax.get_xbound()
ax=plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title("Valores Ausentes da Variável education_level Antes do␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```

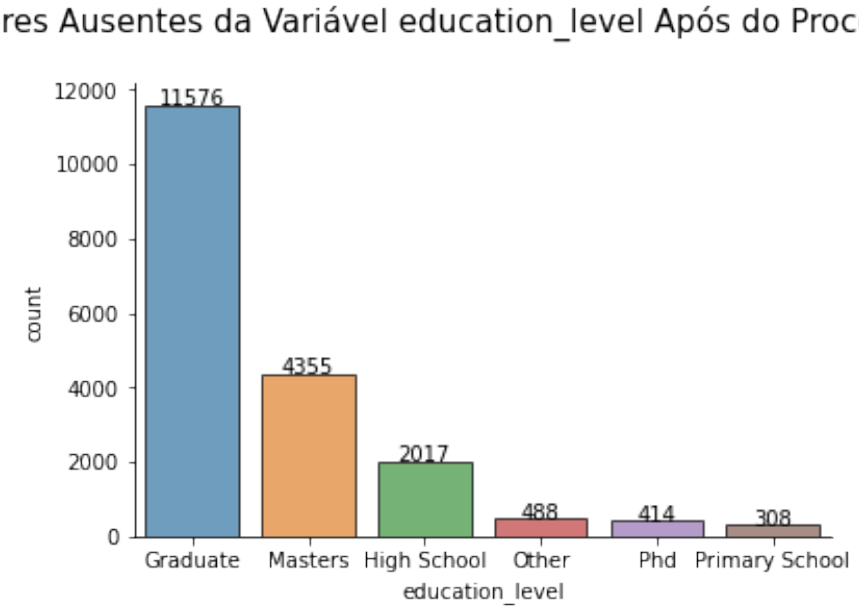


[72]:
```
# Índice
nan_index = new_df[(new_df['education_level'].isna())].index
```

[73]:
```
# Imputação dos valores NaN com 'Other'
new_df['education_level'][nan_index] = 'Other'
```

[74]:
```
# Plot
sns.countplot(data = new_df.fillna('NaN'), x = 'education_level', alpha = 0.7,␣
 ↪edgecolor = 'black')
sns.despine()
plt.xticks()
bound=ax.get_xbound()
```

```
ax=plt.gca()
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha =␣
 ↪'center', color = 'black', size = 10)
plt.title("Valores Ausentes da Variável education_level Após do␣
 ↪Processamento\n", fontsize = 15)
plt.show()
```



[75]:
```
new_df.head()
```

[75]:

	city_development_index	experience	enrolled_university	relevent_experience	education_level	company_type	major_discipline	target
0	0.92	>20	no_enrollment	Has relevent experience	Graduate	Other	STEM	1.0
1	0.776	15	no_enrollment	No relevent experience	Graduate	Pvt Ltd	STEM	
2	0.624	5	Full time course	No relevent experience	Graduate	Other	STEM	
3	0.789	<1	Other	No relevent experience	Graduate	Pvt Ltd	Business Degree	
4	0.767	>20	no_enrollment	Has relevent experience	Masters	Funded Startup	STEM	

```
1    0.0
2    0.0
3    1.0
4    0.0
```

Variável experience

[76]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   city_development_index  19158 non-null  object
 1   experience              19095 non-null  object
 2   enrolled_university     19158 non-null  object
 3   relevent_experience     19158 non-null  object
 4   education_level         19158 non-null  object
 5   company_type            19158 non-null  object
 6   major_discipline        19158 non-null  object
 7   target                  19158 non-null  object
dtypes: object(8)
memory usage: 1.2+ MB
```

[77]:
```
new_df['experience'].value_counts(dropna = False)
```

[77]:
```
>20      3281
5        1427
4        1402
3        1351
6        1214
2        1124
7        1028
10        982
9         980
8         802
15        685
11        664
14        586
1         549
<1        520
16        506
12        494
13        399
17        342
19        304
18        279
```

```
20        148
NaN        63
Other      28
Name: experience, dtype: int64
```

[78]:
```
# Percentual de valores ausentes
percent_missing = new_df.isnull().sum() / len(new_df) * 100
percent_missing
```

[78]:
```
city_development_index    0.000000
experience                0.328844
enrolled_university       0.000000
relevent_experience       0.000000
education_level           0.000000
company_type              0.000000
major_discipline          0.000000
target                    0.000000
dtype: float64
```

[79]:
```
new_df['experience'].isnull().sum()
```

[79]:
```
63
```

[80]:
```
new_df.shape
```

[80]:
```
(19158, 8)
```

[81]:
```
new_df = new_df.dropna()
```

[82]:
```
new_df.shape
```

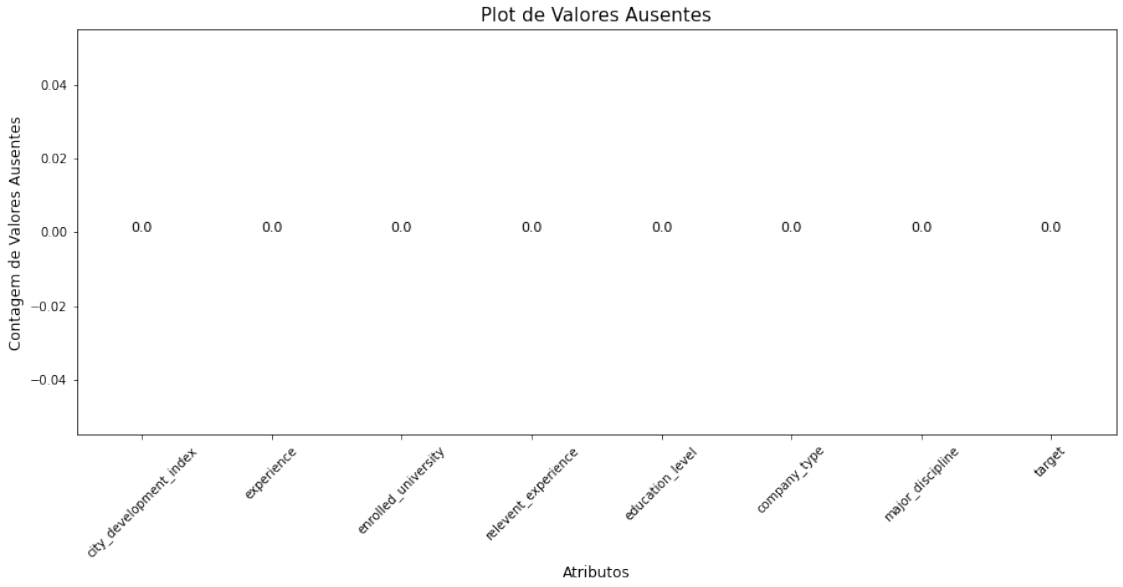
[82]:
```
(19095, 8)
```

[83]:
```
percent_missing = new_df.isnull().sum() * 100 / len(new_df)
percent_missing
```

[83]:
```
city_development_index    0.0
experience                0.0
enrolled_university       0.0
relevent_experience       0.0
education_level           0.0
company_type              0.0
major_discipline          0.0
target                    0.0
dtype: float64
```

[84]:
```
# Valores ausentes por coluna
null_df = new_df.isna().sum().reset_index()
```

```
# Figura
ax = plt.figure(figsize = (15,6))

# Barplot
ax = sns.barplot(null_df['index'], null_df[0], palette = 'husl')
plt.xlabel('Atributos', fontsize = 12)
plt.ylabel('Contagem de Valores Ausentes', fontsize = 12)
plt.xticks(rotation = 45)
plt.title("Plot de Valores Ausentes", fontsize = 15)

for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, (p.get_height())), ha =␣
 ↪'center', color = 'black', size = 11)

plt.show()
```



2.7.2 Ajustes Finais

[85]:
```
new_df.head()
```

[85]:

	city_development_index	experience	enrolled_university	\
0	0.92	>20	no_enrollment	
1	0.776	15	no_enrollment	
2	0.624	5	Full time course	
3	0.789	<1	Other	
4	0.767	>20	no_enrollment	

```
       relevent_experience education_level    company_type major_discipline  \
0  Has relevent experience        Graduate           Other              STEM
1   No relevent experience        Graduate         Pvt Ltd              STEM
2   No relevent experience        Graduate           Other              STEM
3   No relevent experience        Graduate         Pvt Ltd  Business Degree
4  Has relevent experience         Masters  Funded Startup              STEM

  target
0    1.0
1    0.0
2    0.0
3    1.0
4    0.0
```

[86]:
```
# Ajustando os dados
new_df['enrolled_university'] = new_df['enrolled_university'].
 ↪replace('no_enrollment', 'No enrollment')
```

[87]:
```
new_df.head()
```

[87]:
```
  city_development_index experience enrolled_university  \
0                   0.92        >20       No enrollment
1                  0.776         15       No enrollment
2                  0.624          5    Full time course
3                  0.789         <1               Other
4                  0.767        >20       No enrollment

       relevent_experience education_level    company_type major_discipline  \
0  Has relevent experience        Graduate           Other              STEM
1   No relevent experience        Graduate         Pvt Ltd              STEM
2   No relevent experience        Graduate           Other              STEM
3   No relevent experience        Graduate         Pvt Ltd  Business Degree
4  Has relevent experience         Masters  Funded Startup              STEM

  target
0    1.0
1    0.0
2    0.0
3    1.0
4    0.0
```

[88]:
```
x = new_df.drop(columns = ['target'])
y = new_df['target']
```

2.8 Relatório Final

Com base em nossa análise observamos que as variáveis mais relevantes para identificar um bom candidato são:

- Índice de Desenvolvimento da cidade onde mora o candidato.
- Tempo de experiência profissional.
- Se está ou não matriculado em um curso universitário.
- Se tem ou não experiência relevante.
- O nível educacional.
- O tipo de empresa que o candidato trabalhou ou trabalha atualmente.
- A especialização na graduação (quando for o caso).

Não são relevantes para a análise:

- O ID do candidato.
- O código da cidade do candidato.
- O gênero.
- A última vez que o candidato esteve empregado.
- O tamanho da empresa (quando for o caso).
- Total de horas de treimamento.

Recomendações do Analista de Dados:

- O RH pode desenvolver um método de coleta de dados para obter outros recursos a fim de melhorar a qualidade dos dados e tornar o trabalho de análise mais preciso.
- O RH pode procurar candidatos que vêm de cidades com índice de desenvolvimento urbano mais baixo, sem experiência relevante, nível de educação superior e menor experiência de trabalho para ter maior chance de encontrar candidatos que estão procurando um emprego.
- O RH pode tornar o treinamento mais compacto porque muitas pessoas não precisam de muito tempo para concluir o treinamento.

3 Fim