eda

November 29, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import plotly.express as px
```

[2]:
```
df = pd.read_csv('original_data.csv').drop(columns=['Unnamed: 0'])
```

0.0.1 Pedido da Entrega S02

Critérios principais: - Estatísticas Descritivas (média, mediana, moda, desvio padrão e outros); - Pelo menos 3 gráficos relevantes (histogramas, boxplot ou outro) - O relatório deve ser salvo em formato Jupyter Notebook

0.0.2 Relatório de Análise Exploratória dos Dados (EDA)

Colunas numéricas:

`Idade, Pressão Geral, Satisfação Geral, Tempo Trabalho/Estudo e Estresse Financeiro`

Colunas Categóricas (transformadas):

`Teve Pensamento Suicida?, Histórico Familiar Psiquiatrico?, Depressão (futura coluna alvo)`

Idade dos Indivíduos e a relação com a pesquisa A mediana e a média de idade dos entrevistados (de ambos os gêneros) é de 39 anos. A moda dos entrevistados é bimodal, sendo 28 e 56 as idades mais recorrentes na pesquisa.

A média e a mediana das idades mais afetadas (ambos os gêneros) pela depressão roda a faixa dos 25/26 anos A moda das idades mais afetadas pela depressão é unimodal com 18 anos tendo destaque

Profissão dos Indivíduos e a relação com a pesquisa O cargo/ocupação com maior número de pessoas depressivas é o "Unemployed", sendo seguido de longe pelos professores e designers Unemployed (673) tem 45% do seu público aparentemente depressivo Designers/Professores (53) tem 26% do seu público depressivo A maioria das pessoas neste cargo (Unemployed) tem entre 18-22 anos

Correlação e Covalência Pressão Externa Geral e Pensamentos Suicídas são as duas informações com maior covalência e correlação

[3]:
```
cols = ['academic_pressure', 'work_pressure', 'job_satisfaction',␣
 ↪'study_satisfaction']

for col in cols:
    df[col].fillna(0, inplace=True)

df['general_pressure'] = df['academic_pressure'] + df['work_pressure']
df['general_satisfaction'] = df['job_satisfaction'] + df['study_satisfaction']
```

```
/tmp/ipykernel_53231/3931243690.py:4: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].fillna(0, inplace=True)
```

[4]:
```
df['profession'].fillna('Unemployed', inplace=True)
df.drop(columns=cols).head()
```

```
/tmp/ipykernel_53231/3920238238.py:1: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['profession'].fillna('Unemployed', inplace=True)
```

[4]:

	name	gender	age	city	working_professional_or_student	\
0	Pooja	Female	37	Ghaziabad	Working Professional	
1	Reyansh	Male	60	Kalyan	Working Professional	
2	Manvi	Female	42	Bhopal	Working Professional	
3	Isha	Female	44	Thane	Working Professional	
4	Aarav	Male	48	Indore	Working Professional	

```
          profession  cgpa sleep_duration dietary_habits degree  \
0            Teacher   NaN      7-8 hours       Moderate     MA
1  Financial Analyst   NaN      5-6 hours      Unhealthy  B.Com
2            Teacher   NaN      5-6 hours       Moderate  M.Com
3            Teacher   NaN      7-8 hours        Healthy     MD
4     UX/UI Designer   NaN      7-8 hours       Moderate     BE

  have_you_ever_had_suicidal_thoughts_?  work/study_hours  financial_stress  \
0                                    No                 6                 2
1                                   Yes                 0                 4
2                                    No                 0                 2
3                                   Yes                 1                 2
4                                   Yes                 6                 5

  family_history_of_mental_illness depression  general_pressure  \
0                               No         No              2.0
1                              Yes         No              4.0
2                               No         No              2.0
3                              Yes         No              3.0
4                              Yes         No              4.0

   general_satisfaction
0                   4.0
1                   3.0
2                   3.0
3                   5.0
4                   3.0
```

[5]:
```
df['working_professional_or_student'].replace({
    'Working Professional': 1,
    'Student': 0
}, inplace=True)

df.rename(columns={
    'working_professional_or_student': 'is_working'
}, inplace=True)
```

```
/tmp/ipykernel_53231/3827138957.py:1: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

```
  df['working_professional_or_student'].replace({
/tmp/ipykernel_53231/3827138957.py:1: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df['working_professional_or_student'].replace({
```

[6]:
```
categoricalCols = [
    'have_you_ever_had_suicidal_thoughts_?',
    'family_history_of_mental_illness',
    'depression'
]

def categorial_to_bool_to_int(df, spec_type, col, replace_val: dict = None):
    if not replace_val:
        replace_val = {
            "Yes": True,
            "No": False
        }

    df[col].replace(
        replace_val
    , inplace=True)

    return df[col].astype(spec_type)

for cols in categoricalCols:
    df[cols] = categorial_to_bool_to_int(df, int, cols)
```

```
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].replace(
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
```

```
  df[col].replace(
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].replace(
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df[col].replace(
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df[col].replace(
/tmp/ipykernel_53231/880504184.py:14: FutureWarning: Downcasting behavior in
`replace` is deprecated and will be removed in a future version. To retain the
old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to
the future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df[col].replace(
```

[7]:
```python
def rearrangeMean(value):
    return float(f'{(value * 100):.2f}')

# 48.9% Já tiveram
meanSuicThoughts = rearrangeMean(df['have_you_ever_had_suicidal_thoughts_?'].
 ↪mean())
# 48.7% Tem familiares com registro psicoterapeutico
meanFamPsychHist = rearrangeMean(df['family_history_of_mental_illness'].mean())
# 17.8% Tem depressão
```

```
meanDepresIndice = rearrangeMean(df['depression'].mean())
```

[8]:
```
# A correlação de depressão e pensamentos suicidas é de 28%
df[['have_you_ever_had_suicidal_thoughts_?', 'depression']].corr()
```

[8]:
```
                                    have_you_ever_had_suicidal_thoughts_?  \
have_you_ever_had_suicidal_thoughts_?                                1.000000
depression                                                           0.281669

                                    depression
have_you_ever_had_suicidal_thoughts_?   0.281669
depression                              1.000000
```

[9]:
```
# A covariancia entre depressão e pensamentos suicidas é pouco positiva
df[['have_you_ever_had_suicidal_thoughts_?', 'depression']].cov()
```

[9]:
```
                                    have_you_ever_had_suicidal_thoughts_?  \
have_you_ever_had_suicidal_thoughts_?                                0.249969
depression                                                           0.053880

                                    depression
have_you_ever_had_suicidal_thoughts_?   0.053880
depression                              0.146381
```

[10]:
```
# A correlação entre depressão e histórico familiar é de 2%
df[['depression', 'family_history_of_mental_illness']].corr()
```

[10]:
```
                                depression  family_history_of_mental_illness
depression                        1.000000                          0.019182
family_history_of_mental_illness  0.019182                          1.000000
```

[11]:
```
# A covariancia entre depressão e histórico familiar é quase nula
df[['depression', 'family_history_of_mental_illness']].cov()
```

[11]:
```
                                depression  family_history_of_mental_illness
depression                        0.146381                          0.003669
family_history_of_mental_illness  0.003669                          0.249931
```

[12]:
```
# A correlação de depressão e pressão externa geral é de 24%
df[['depression', 'general_pressure']].corr()
```

[12]:
```
                  depression  general_pressure
depression          1.000000          0.242483
general_pressure    0.242483          1.000000
```

[13]:
```
# A covariancia entre ambos é positiva
df[['depression', 'general_pressure']].cov()
```

[13]:

	depression	general_pressure
depression	0.146381	0.130972
general_pressure	0.130972	1.993008

[14]:
```
# A correlação entre depressão e estresse financeiro é de 16%
df[['depression', 'financial_stress']].corr()
```

[14]:

	depression	financial_stress
depression	1.000000	0.165669
financial_stress	0.165669	1.000000

[15]:
```
# A covariância entre ambos é pouco positiva
df[['depression', 'financial_stress']].cov()
```

[15]:

	depression	financial_stress
depression	0.146381	0.089723
financial_stress	0.089723	2.003717

[16]:
```
# Idade geral do público é 39
meanPublicAge = int(df['age'].mean())

# Idade geral dos homens é 38
meanMaleAge = int(
    df.loc[
        df['gender'] == 'Male',
        'age'
    ].mean()
)

# Idade geral das mulheres é 39
meanFemaleAge = int(
    df.loc[
        df['gender'] == 'Female',
        'age'
    ].mean()
)
```

[17]:
```
# 26 anos
meanDepMaleAge = int(
    df.loc[
        (df['gender'] == 'Male')
        &
        (df['depression'] == 1)
        ,
        'age'
    ].mean()
)
```

```
# 26 anos
meanDepFemaleAge = int(
    df.loc[
        (df['gender'] == 'Female')
        &
        (df['depression'] == 1)
        ,
        'age'
    ].mean()
)
```

[18]:
```
# A mediana das idades é 39
medianAges = df['age'].sort_values().median()
```

[19]:
```
# As idades que mais se repetem são 28 e 56 anos
modeAges = df['age'].mode()
```

[20]:
```
# O desvio padrão das idades é de 12 em média
stdAges = df['age'].std()
```

[21]:
```
# A variancia é de 150
varAges = df['age'].var()
```

[22]:
```
useCols = ['age', 'financial_stress', 'general_pressure',␣
 ↪'general_satisfaction', 'depression', 'profession']
greaterAgeDepIndices = df[useCols].groupby("age").agg({
    'age': 'count',
    'financial_stress': 'mean',
    'general_pressure': 'mean',
    'general_satisfaction': 'mean',
    'depression': 'mean'
}).nlargest(n=5, columns='depression')

greaterAgeDepIndices
```

[22]:

age	age	financial_stress	general_pressure	general_satisfaction	depression
18	60	2.800000	3.466667	3.533333	0.733333
21	53	3.264151	2.830189	3.037736	0.584906
20	65	3.261538	3.215385	3.215385	0.584615
19	47	3.042553	2.702128	3.297872	0.574468
23	53	2.754717	2.924528	2.867925	0.452830

[23]:
```
lowerAgeDepIndices = df[useCols].groupby("age").agg({
    'age': 'count',
    'financial_stress': 'mean',
```

```
    'general_pressure': 'mean',
    'general_satisfaction': 'mean',
    'depression': 'mean'
}).nsmallest(n=5, columns='depression')

lowerAgeDepIndices
```

[23]:

age	financial_stress	general_pressure	general_satisfaction	depression
age				
44	51	2.980392	2.627451	3.117647	0.0
52	46	2.891304	2.826087	2.847826	0.0
53	63	2.968254	3.000000	2.984127	0.0
54	59	3.118644	2.915254	2.644068	0.0
55	51	2.862745	3.058824	3.254902	0.0

[24]:
```
greaterProfDepIndices = df[useCols].groupby("profession").agg({
    'profession': 'count',
    'financial_stress': 'mean',
    'general_pressure': 'mean',
    'general_satisfaction': 'mean',
    'depression': 'mean'
}).nlargest(n=5, columns='depression')

greaterProfDepIndices
```

[24]:

profession	financial_stress	general_pressure	general_satisfaction	depression
profession				
Unemployed	673	2.950966	3.037147	3.101040	0.459138
Graphic Designer	26	3.230769	3.230769	2.730769	0.230769
Data Scientist	42	3.333333	3.309524	3.166667	0.142857
HR Manager	84	3.273810	3.130952	2.892857	0.142857
Judge	42	2.809524	3.095238	3.000000	0.142857

[25]:
```
lowerProfDepIndices = df[useCols].groupby("profession").agg({
    'profession': 'count',
    'financial_stress': 'mean',
    'general_pressure': 'mean',
    'general_satisfaction': 'mean',
    'depression': 'mean'
```

```
}).nsmallest(n=5, columns='depression')

lowerProfDepIndices
```

[25]:

profession	profession	financial_stress	general_pressure	general_satisfaction	depression
Pharmacist	75	3.000000	3.040000	3.000000	0.013333
Entrepreneur	63	2.777778	3.126984	3.095238	0.015873
Chemist	59	2.847458	3.169492	2.983051	0.016949
Travel Consultant	46	3.000000	3.130435	3.478261	0.021739
Software Engineer	34	2.911765	2.117647	3.117647	0.029412

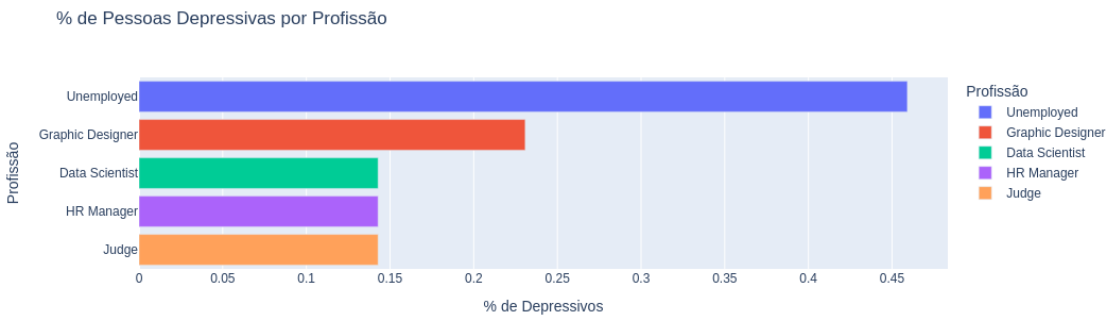
[26]:
```
df.loc[
    df['depression'] == 1,
    'profession'
].value_counts()
```

[26]:
```
profession
Unemployed                309
Teacher                    28
HR Manager                 12
Architect                   9
Business Analyst            6
Data Scientist              6
Judge                       6
Graphic Designer            6
Chef                        5
Consultant                  5
Lawyer                      5
Financial Analyst           4
Mechanical Engineer         4
Educational Consultant      4
Content Writer              4
Doctor                      4
Marketing Manager           3
Pilot                       3
Civil Engineer              3
Plumber                     3
Manager                     3
```

```
Electrician                 2
Researcher                  2
Research Analyst            2
Finanancial Analyst         2
Accountant                  2
Customer Support            2
UX/UI Designer              2
Sales Executive             2
Investment Banker           1
Chemist                     1
Entrepreneur                1
Digital Marketer            1
Travel Consultant           1
Pharmacist                  1
Software Engineer           1
Name: count, dtype: int64
```

0.0.3 Gráficos sobre o tema:

[27]:
```
px.bar(
    greaterProfDepIndices,
    x='depression',
    y=greaterProfDepIndices.index,
    title="% de Pessoas Depressivas por Profissão",
    labels={
        'index': 'Profissão',
        'depression': "% de Depressivos"
    },
    color=greaterProfDepIndices.index
)
```

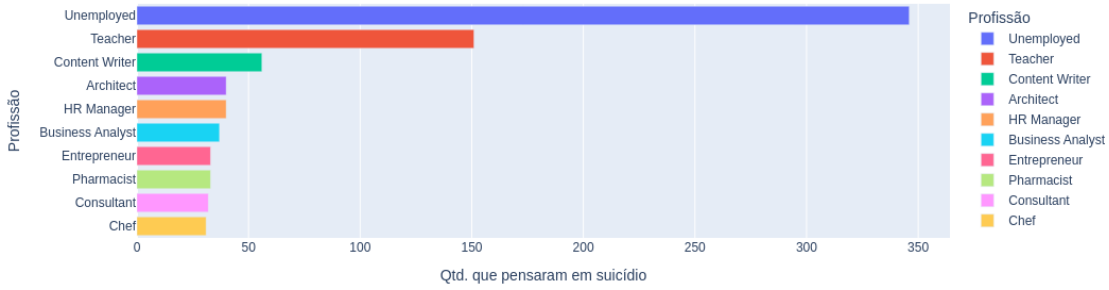


[28]:
```
px.box(
    df,
```

```
    x='depression',
    y='age',
    labels={
        'age': 'Idade',
        'depression': 'É depressivo?'
    },
    color='depression'
)
```

[29]:
```
relCols = ['profession', 'financial_stress', 'general_pressure',␣
 ↪'general_satisfaction', 'have_you_ever_had_suicidal_thoughts_?']
greaterHadSuicThoughts = df[relCols].groupby('profession').agg({
    'profession': 'count',
    'financial_stress': 'mean',
    'general_pressure': 'mean',
    'general_satisfaction': 'mean',
    'have_you_ever_had_suicidal_thoughts_?': 'sum'
}).nlargest(columns='have_you_ever_had_suicidal_thoughts_?', n=10)

px.bar(
    greaterHadSuicThoughts,
    x='have_you_ever_had_suicidal_thoughts_?',
    y=greaterHadSuicThoughts.index,
    color=greaterHadSuicThoughts.index,
    hover_data='profession',
    labels={
        'index': 'Profissão',
        'have_you_ever_had_suicidal_thoughts_?': "Qtd. que pensaram em␣
 ↪suicídio",
        'profession': 'Profissionais Entrevistados'
    }
)
```


Profissão
Unemployed
Teacher
Content Writer
Architect
HR Manager
Business Analyst
Entrepreneur
Pharmacist
Consultant
Chef
0
50
100
150
200
250
300
350
Qtd. que pensaram em suicídio
Profissão
Unemployed
Teacher
Content Writer
Architect
HR Manager
Business Analyst
Entrepreneur
Pharmacist
Consultant
Chef