Indicium - Desafio Cientista de Dados

Importação das Bibliotecas

In [3]:
```
# Módulos de manipulação de dados
import numpy as np
import pandas as pd
```

In [4]:
```
# Pacotes gráficos
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
# Pacotes de modelagem
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
```

Importação da base de dados

In [7]:
```
df = pd.read_csv('desafio_indicium_imdb.csv', sep=',')
```

In [8]:
```
df.head(2)
```

Out[8]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Sta
0	1	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Dia Keat
1	2	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Micha Cai

Preparação dos Dados

Missing Values

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Unnamed: 0         0
Series_Title       0
Released_Year      0
Certificate      101
Runtime            0
Genre              0
IMDB_Rating        0
Overview           0
Meta_score       157
Director           0
Star1              0
Star2              0
Star3              0
Star4              0
No_of_Votes        0
Gross            169
dtype: int64
```

Há 101 dados faltantes na coluna "Certificate", 157 dados faltantes na coluna "Meta_score" e 169 dados faltantes na coluna "Gross". É sempre muito importante alinhar com os analistas de negócio como tratar esses missing values. Para que as nossas análises e modelos não sejam prejudicados, iremos eliminar da nossa base de dados as linhas com dados faltantes.

Eliminando os valores missing

In [14]:
```
df.dropna(inplace=True)
```

In [15]:
```
df.isnull().sum()
```

Out[15]:
```
Unnamed: 0       0
Series_Title     0
Released_Year    0
Certificate      0
Runtime          0
Genre            0
IMDB_Rating      0
Overview         0
Meta_score       0
Director         0
Star1            0
Star2            0
Star3            0
Star4            0
No_of_Votes      0
Gross            0
dtype: int64
```

Duplicações

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
0
```

Não há valores duplicados na nossa base de dados

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 713 entries, 0 to 996
Data columns (total 16 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Unnamed: 0     713 non-null    int64  
 1   Series_Title   713 non-null    object 
 2   Released_Year  713 non-null    object 
 3   Certificate    713 non-null    object 
 4   Runtime        713 non-null    object 
 5   Genre          713 non-null    object 
 6   IMDB_Rating    713 non-null    float64
 7   Overview       713 non-null    object 
 8   Meta_score     713 non-null    float64
 9   Director       713 non-null    object 
 10  Star1          713 non-null    object 
 11  Star2          713 non-null    object 
 12  Star3          713 non-null    object 
 13  Star4          713 non-null    object 
 14  No_of_Votes    713 non-null    int64  
 15  Gross          713 non-null    object 
dtypes: float64(2), int64(2), object(12)
memory usage: 94.7+ KB
```

Transformação de Variáveis

A coluna "Runtime" está classificada como object, mas queremos tratá-la como número inteiro. Iremos retirar a palavra "min" das linhas desta coluna e iremos convertê-la para sendo como do tipo int64.

A coluna "Gross" está classificada como object, mas queremos tratá-la como número decimal. Assim, iremos convertê-la para sendo como do tipo float64.

Nota-se que a coluna "Genre" apresenta, em algumas linhas, diversos gêneros para o mesmo filme. Iremos criar uma nova coluna, nomeada de "Genre_New" com apenas o primeiro gênero que aparece em cada linha. Partiremos do pressuposto que esse será o gênero dominante do filme. Essa nova coluna será importante para avaliarmos o poder de separação desta variável qualitativa com outras variáveis da nossa base de dados

Removendo a palavra "min" das linhas da coluna "Runtime"

In [23]:
```
df['Runtime'] = df['Runtime'].str.replace('min', '')
```

Convertendo a coluna "Runtime" para o tipo int64

In [25]:
```
df['Runtime'] = df['Runtime'].astype('int64')
```

In [26]:
```
df.dtypes
```

Out[26]:
```
Unnamed: 0         int64
Series_Title      object
Released_Year     object
Certificate       object
Runtime            int64
Genre             object
IMDB_Rating      float64
Overview          object
Meta_score       float64
Director          object
Star1             object
Star2             object
Star3             object
Star4             object
No_of_Votes        int64
Gross             object
dtype: object
```

Convertendo a coluna "Gross" para o tipo float64

In [28]:
```
df['Gross'] = df['Gross'].str.replace(',', '').astype(float)
```

In [29]:
```
df.dtypes
```

Out[29]:
```
Unnamed: 0         int64
Series_Title      object
Released_Year     object
Certificate       object
Runtime            int64
Genre             object
IMDB_Rating      float64
Overview          object
Meta_score       float64
Director          object
Star1             object
Star2             object
Star3             object
Star4             object
No_of_Votes        int64
Gross            float64
dtype: object
```

Criando a nova coluna "Genre_New" com apenas o primeiro gênero de cada filme

In [31]:
```
# Função para extrair a primeira palavra do texto que está separado por vírgulas
def extrair_primeira_palavra(texto_com_virgulas):
    texto = texto_com_virgulas.split(',')
    return texto[0] if len(texto) > 0 else ''

# Aplicando a função à coluna e criando uma nova coluna no DataFrame
df['Genre_New'] = df['Genre'].apply(extrair_primeira_palavra)

df.head(2)
```

Out[31]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Sta
0	1	The Godfather	1972	A	175	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Dia Keat
1	2	The Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Micha Cai

Análises de Correlações e Associações

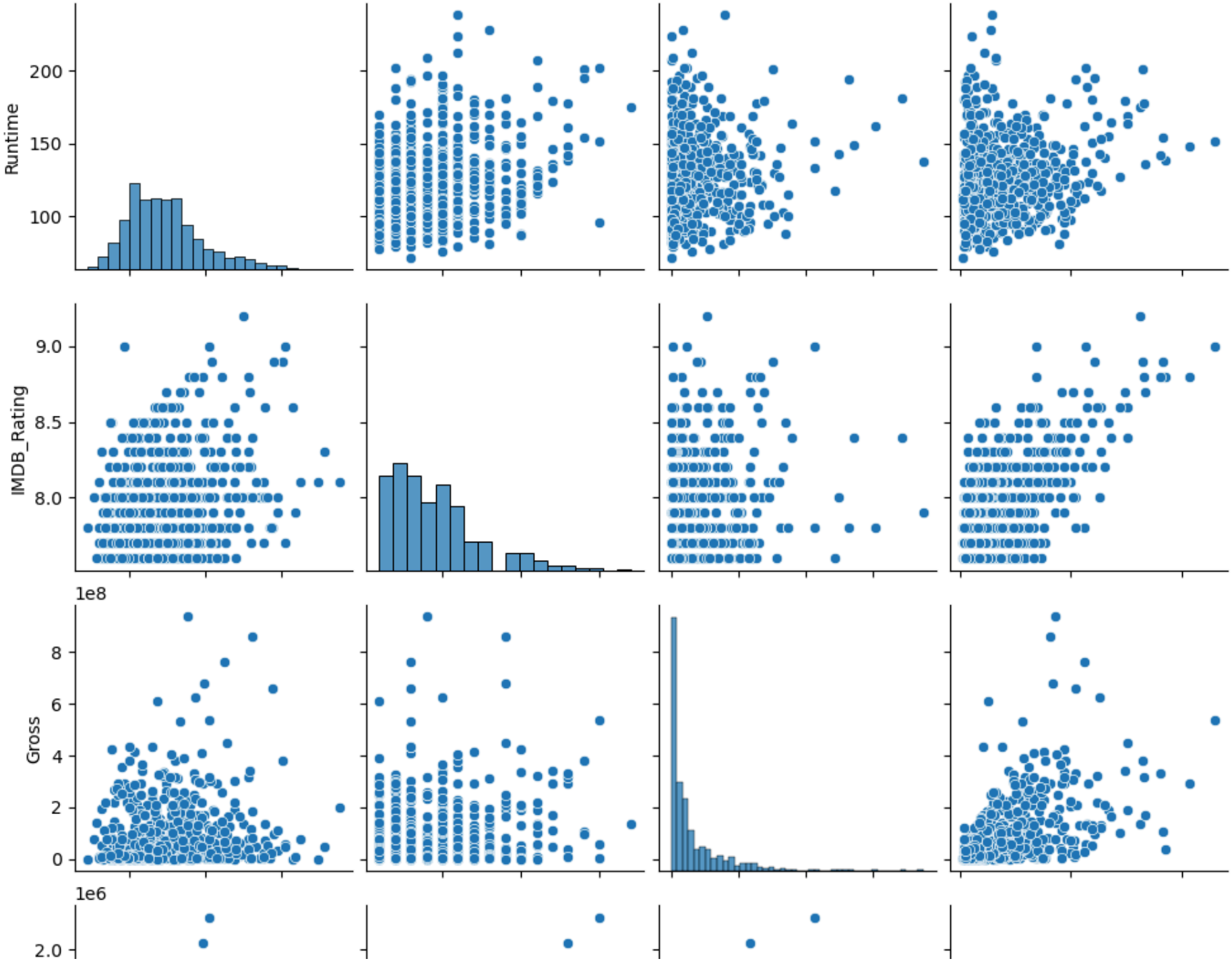
Análises de Correlações entre variáveis quantitativas

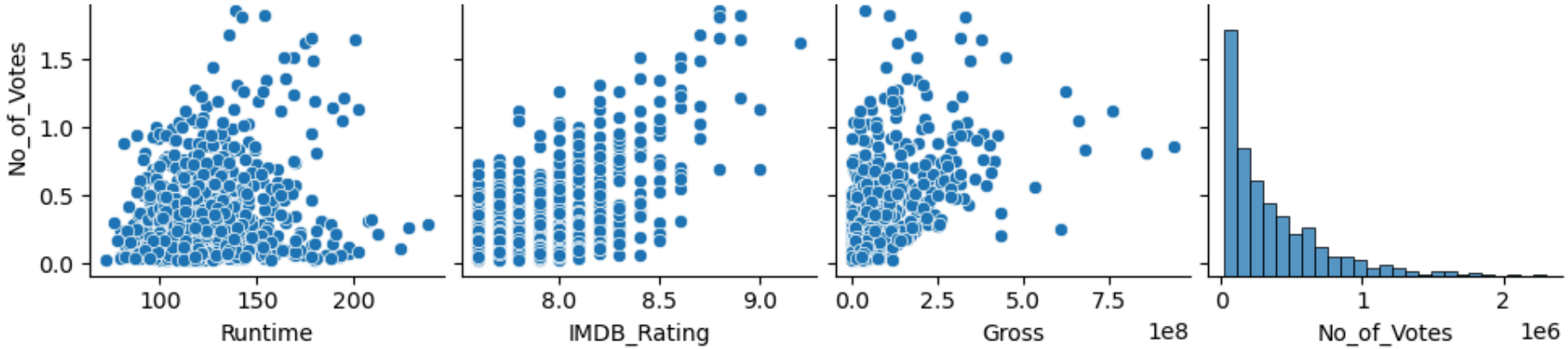
In [34]:

```
import matplotlib
matplotlib.use('module://ipykernel.pylab.backend_inline')

sns.pairplot(df[['Runtime', 'IMDB_Rating','Gross','No_of_Votes']]);
```

```
C:\Users\mateu\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\mateu\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\mateu\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\mateu\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Runtime
200
150
100
IMDB_Rating
9.0
8.5
8.0
1e8
Gross
8
6
4
2
0
1e6
2.0



Correlação entre Runtime e IMDB_Rating

In [36]:
```
correlacao = round(df['Runtime'].corr(df['IMDB_Rating']),2)

print(f"A correlação entre Runtime e IMDB_Rating é: {correlacao}")
```

```
A correlação entre Runtime e IMDB_Rating é: 0.26
```

A correlação entre as Runtime e IMDB_Rating foi fraca, o que sinaliza que a duração do filme não impacta muito na nota recebida pelo filme.

Correlação entre Runtime e Gross

In [39]:
```
correlacao = round(df['Runtime'].corr(df['Gross']),2)

print(f"A correlação entre Runtime e Gross é: {correlacao}")
```

```
A correlação entre Runtime e Gross é: 0.17
```

A correlação entre as Runtime e Gross foi fraca, o que sinaliza que a duração do filme não impacta muito no faturamento obtido pelo filme.

Correlação entre Runtime e No_of_Votes

In [42]:
```
correlacao = round(df['Runtime'].corr(df['No_of_Votes']),2)

print(f"A correlação entre Runtime e No_of_Votes é: {correlacao}")
```

```
A correlação entre Runtime e No_of_Votes é: 0.21
```

A correlação entre as Runtime e No_of_Votes foi fraca, o que sinaliza que a duração do filme não impacta muito na quantidade de votos que o filme recebeu.

Correlação entre Gross e IMDB_Rating

In [45]:
```
correlacao = round(df['Gross'].corr(df['IMDB_Rating']),2)

print(f"A correlação entre Gross e IMDB_Rating é: {correlacao}")
```

```
A correlação entre Gross e IMDB_Rating é: 0.13
```

A correlação entre Gross e IMDB_Rating foi fraca, o que sinaliza que o faturamento obtido pelo filme impacta pouco na nota recebida pelo filme.

O filme pode gerar um faturamento alto, mas isso não implica que ele receberá uma boa avaliação.

Correlação entre Gross e No_of_Votes

In [48]:
```
correlacao = round(df['Gross'].corr(df['No_of_Votes']),2)

print(f"A correlação entre Gross e No_of_Votes é: {correlacao}")
```

```
A correlação entre Gross e No_of_Votes é: 0.56
```

A correlação entre Gross e No_of_Votes foi muito mais alta que a correlação entre Gross e IMDB_Rating.

Ou seja, filmes que geram muito faturamento são assistidos por muitas pessoas. Esses tipo de filme leva muitas pessoas a votarem no IMDB. Mas muitos desses votos apresentam avaliações baixas.

Correlação entre IMDB_Rating e No_of_Votes

In [51]:
```
correlacao = round(df['IMDB_Rating'].corr(df['No_of_Votes']),2)

print(f"A correlação entre IMDB_Rating e No_of_Votes é: {correlacao}")
```

```
A correlação entre IMDB_Rating e No_of_Votes é: 0.61
```

A correlação entre IMDB_Rating e No_of_Votes pode ser classificada como próxima de forte. Foi a mais alta das correlações entre as variáveis quantitativas.

Ela demonstra que filmes que possuem mais votos podem tender a ter uma nota mais alta no IMDB.

Análises de Associações entre variáveis qualitativas

Iremos analisar a associação de variáveis qualitativas da nossa base de dados.

Para isso, iremos utilizar o IV (Information Value) que é uma medida capaz de nos passsar o poder de separação de uma variável qualitativa em relação a outra variável de duas categorias (variável binária).

Iremos criar uma variável binária com base na nota recebida pelo filme. Ou seja, com base na coluna "IMDB_Rating". Essa variável será nomeada como "IDB_Classification".

O critério para essa variável binária será o seguinte: Filmes que tiverem notas posicionadas acima do 3º quartil irão receber a classificação 1. Filmes que não tiverem notas posicionadas acima do 3º Quartil receberão a classificação 0.

Também iremos criar uma variável binária com base no faturamento do filme. Ou seja, com base na coluna "Gross". Essa variável será nomeada como "Gross_Classification".

O critério para essa variável binária será o seguinte: Filmes que tiverem o faturamento posicionado acima do 3º quartil irão receber a classificação 1. Filmes que não tiverem o faturamento posicionado acima do 3º Quartil receberão a classificação 0.

In [55]:
```
df.describe()
```

Out[55]:

	Unnamed: 0	Runtime	IMDB_Rating	Meta_score	No_of_Votes	Gross
count	713.000000	713.000000	713.000000	713.000000	7.130000e+02	7.130000e+02
mean	519.300140	123.690042	7.935203	77.154278	3.533480e+05	7.858395e+07
std	295.416331	25.896632	0.288999	12.409392	3.462212e+05	1.150433e+08
min	1.000000	72.000000	7.600000	28.000000	2.522900e+04	1.305000e+03
25%	263.000000	104.000000	7.700000	70.000000	9.582600e+04	6.153939e+06
50%	527.000000	120.000000	7.900000	78.000000	2.363110e+05	3.500000e+07
75%	778.000000	136.000000	8.100000	86.000000	5.059180e+05	1.025158e+08
max	997.000000	238.000000	9.200000	100.000000	2.303232e+06	9.366622e+08

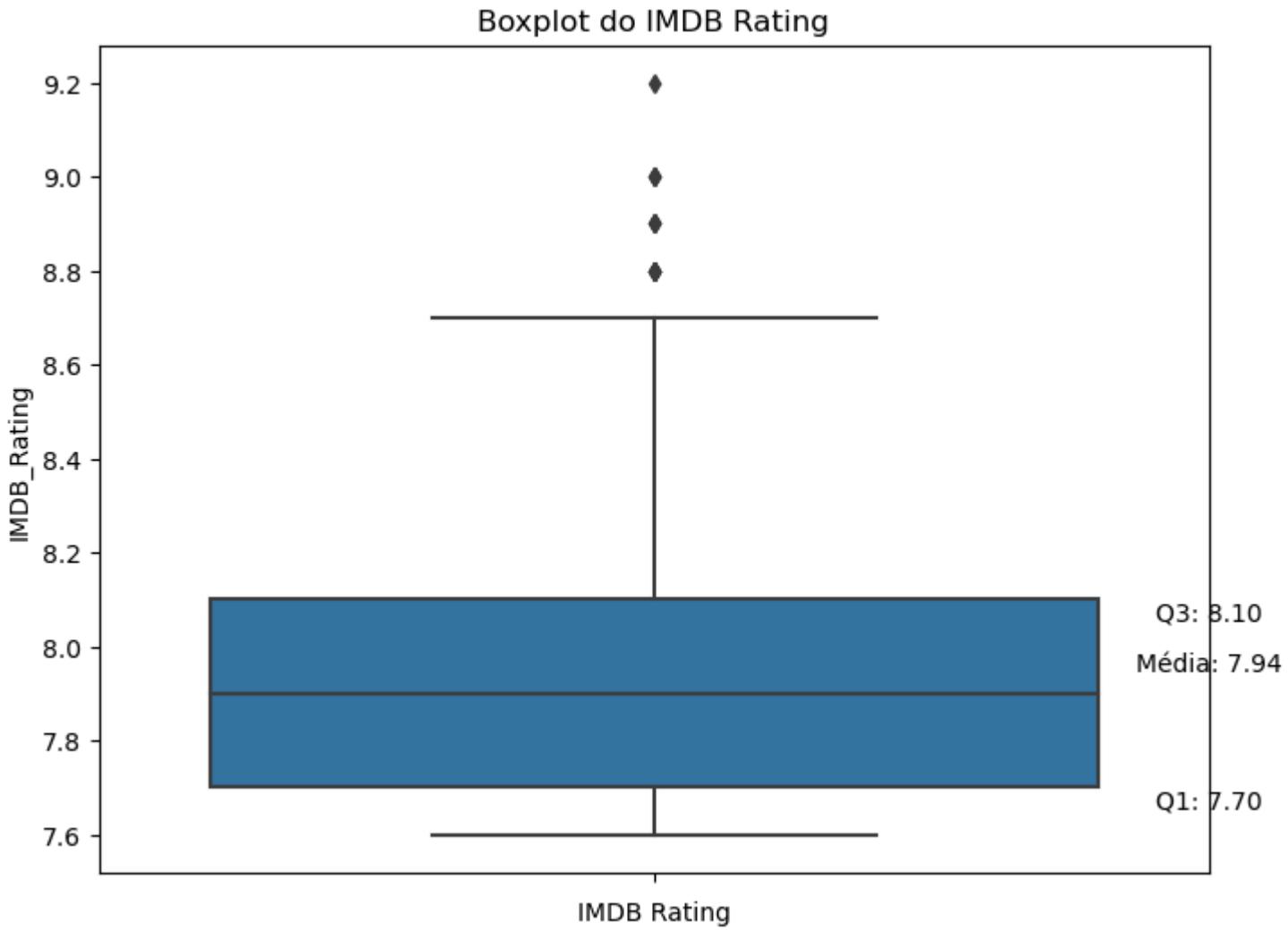
Elaborando o Boxplot da variável IMDB_Rating

In [57]:
```
mean_value = df['IMDB_Rating'].mean()
q1_value = df['IMDB_Rating'].quantile(0.25)
q3_value = df['IMDB_Rating'].quantile(0.75)

plt.figure(figsize=(8, 6))
sns.boxplot(y='IMDB_Rating', data=df)
plt.title('Boxplot do IMDB Rating')
plt.xlabel('IMDB Rating')

plt.text(0.5, mean_value, f'Média: {mean_value:.2f}', color='black', ha='center', va='bottom')
plt.text(0.5, q1_value, f'Q1: {q1_value:.2f}', color='black', ha='center', va='top')
plt.text(0.5, q3_value, f'Q3: {q3_value:.2f}', color='black', ha='center', va='top')
```

Out[57]:
```
Text(0.5, 8.1, 'Q3: 8.10')
```



Como podemos ver na tabela e no Boxplot acima, o 3º Quartil da variável IMDB_Rating foi de 8,1.

Essa será nossa referência para a criação da coluna "IMDB_Classification".

In [59]:
```
df['IMDB_Classification'] = np.where(df['IMDB_Rating'] >= 8.1, 1, 0)
```

In [60]:
```
df.head(2)
```

Out[60]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Sta
0	1	The Godfather	1972	A	175	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Dia Keat
1	2	The Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Micha Cai

In [61]:
```
df.tail(2)
```

Out[61]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	
993	994	A Hard Day's Night	1964	U	87	Comedy, Music, Musical	7.6	Over two "typical" days in the life of The Bea...	96.0	Richard Lester	John Lennon	Paul McCartney	G Ha
996	997	From Here to Eternity	1953	Passed	118	Drama, Romance, War	7.6	In Hawaii in 1941, a private is cruelly punish...	85.0	Fred Zinnemann	Burt Lancaster	Montgomery Clift	Del

Elaborando o cálculo do Information Value (IV)

In [63]:
```
class analise_iv:

    def __get_tab_bivariada(self, var_escolhida):
```

```
        # Cria a contagem de Target_1 e Target_0
        df_aux = self.df.copy()
        df_aux['target2'] = self.df[self.target]
        df2 = df_aux.pivot_table(values='target2',
                                 index=var_escolhida,
                                 columns=self.target,
                                 aggfunc='count')

        df2 = df2.rename(columns={0:'#Target_0',
                                  1:'#Target_1'})
        df2.fillna(0, inplace=True)

        # Cria as demais colunas da tabela bivariada
        df2['Total'] = (df2['#Target_0'] + df2['#Target_1'])
        df2['%Freq'] = (df2['Total'] / (df2['Total'].sum()) * 100).round(decimals=2)
        df2['%Target_1'] = (df2['#Target_1'] / (df2['#Target_1'].sum()) * 100).round(decimals=2)
        df2['%Target_0'] = (df2['#Target_0'] / (df2['#Target_0'].sum()) * 100).round(decimals=2)
        df2['%Target_0'] = df2['%Target_0'].apply(lambda x: 0.01 if x == 0 else x) #corrige problema do log indeterminado
        df2['%Taxa_de_Target_1'] = (df2['#Target_1'] / df2['Total'] * 100).round(decimals=2)
        df2['Odds'] = (df2['%Target_1'] / df2['%Target_0']).round(decimals=2)
        df2['Odds'] = df2.Odds.apply(lambda x: 0.01 if x == 0 else x) #corrige problema do log indeterminado
        df2['LN(Odds)'] = np.log(df2['Odds']).round(decimals=2)
        df2['IV'] = (((df2['%Target_1'] / 100 - df2['%Target_0'] / 100) * df2['LN(Odds)'])).round(decimals=2)
        df2['IV'] = np.where(df2['Odds'] == 0.01, 0 , df2['IV'])

        df2 = df2.reset_index()
        df2['Variavel'] = var_escolhida
        df2 = df2.rename(columns={var_escolhida: 'Var_Range'})
        df2 = df2[['Variavel','Var_Range', '#Target_1','#Target_0', 'Total', '%Freq', '%Target_1', '%Target_0',
       '%Taxa_de_Target_1', 'Odds', 'LN(Odds)', 'IV']]

        # Guarda uma cópia da tabela no histórico
        self.df_tabs_iv = pd.concat([self.df_tabs_iv, df2], axis = 0)

        return df2

    def get_bivariada(self, var_escolhida='all_vars'):

        if var_escolhida == 'all_vars':

            #vars = self.df.drop(self.target,axis = 1).columns
            vars = self.get_lista_iv().index
            for var in vars:
```

```
            tabela = self.df_tabs_iv[self.df_tabs_iv['Variavel'] == var]
            print('==> "{}" tem IV de {}'.format(var,self.df_tabs_iv[self.df_tabs_iv['Variavel'] == var]['IV'].sum().round(de
            # printa a tabela no Jupyter
            display(tabela)

        return

    else:
        print('==> "{}" tem IV de {}'.format(var_escolhida,self.df_tabs_iv[self.df_tabs_iv['Variavel'] == var_escolhida]['IV'
        return self.df_tabs_iv[self.df_tabs_iv['Variavel'] == var_escolhida]


def get_lista_iv(self):


    # agrupa a lista de IV's em ordem descrescente
    lista = (self.df_tabs_iv.groupby('Variavel').agg({'IV':'sum'})).sort_values(by=['IV'],ascending=False)

    return lista



def __init__(self, df, target, nbins=10):

    self.df = df.copy()
    self.target = target

    #lista de variaveis numericas
    df_num = self.df.loc[:,((self.df.dtypes == 'int32') |
                            (self.df.dtypes == 'int64') |
                            (self.df.dtypes == 'float64')
                           )
                         ]

    vars = df_num.drop(target,axis = 1).columns

    for var in vars:
        nome_var = 'fx_' + var
        df_num[nome_var] = pd.qcut(df_num[var],
                                   q=nbins,
                                   precision=2,
                                   duplicates='drop')
        df_num = df_num.drop(var, axis = 1)
        df_num = df_num.rename(columns={nome_var: var})
```

```
        #lista de variaveis qualitativas
        df_str = self.df.loc[:,((self.df.dtypes == 'object') |
                                (self.df.dtypes == 'category') |
                                (self.df.dtypes == 'bool'))]


        self.df = pd.concat([df_num,df_str],axis = 1)


         # inicializa tab historica
        self.df_tabs_iv = pd.DataFrame()

        vars = self.df.drop(self.target,axis = 1).columns
        for var in vars:
            self.__get_tab_bivariada(var);

        # remove tabs de iv duplicadas
        self.df_tabs_iv = self.df_tabs_iv.drop_duplicates(subset=['Variavel','Var_Range'], keep='last')
```

In [64]:
```
df_iv = analise_iv(df,
                   'IMDB_Classification',
                           nbins=5)
```

```
C:\Users\mateu\AppData\Local\Temp\ipykernel_14776\1774483247.py:86: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_num[nome_var] = pd.qcut(df_num[var],
```

In [65]:
```
df_iv.get_bivariada(var_escolhida='Genre_New')
```

```
==> "Genre_New" tem IV de 0.11
```

Out[65]:

	IMDB_Classification	Variavel	Var_Range	#Target_1	#Target_0	Total	%Freq	%Target_1	%Target_0	%Taxa_de_Target_1	Odds	LN(Odds)	IV
0	Genre_New	Action	34.0	93.0	127.0	17.81	16.67	18.27	26.77	0.91	-0.09	0.00
1	Genre_New	Adventure	17.0	41.0	58.0	8.13	8.33	8.06	29.31	1.03	0.03	0.00
2	Genre_New	Animation	18.0	45.0	63.0	8.84	8.82	8.84	28.57	1.00	0.00	-0.00
3	Genre_New	Biography	17.0	56.0	73.0	10.24	8.33	11.00	23.29	0.76	-0.27	0.01
4	Genre_New	Comedy	23.0	81.0	104.0	14.59	11.27	15.91	22.12	0.71	-0.34	0.02
5	Genre_New	Crime	26.0	48.0	74.0	10.38	12.75	9.43	35.14	1.35	0.30	0.01
6	Genre_New	Drama	58.0	133.0	191.0	26.79	28.43	26.13	30.37	1.09	0.09	0.00
7	Genre_New	Family	0.0	2.0	2.0	0.28	0.00	0.39	0.00	0.01	-4.61	0.00
8	Genre_New	Film-Noir	1.0	0.0	1.0	0.14	0.49	0.01	100.00	49.00	3.89	0.02
9	Genre_New	Horror	3.0	6.0	9.0	1.26	1.47	1.18	33.33	1.25	0.22	0.00
10	Genre_New	Mystery	4.0	3.0	7.0	0.98	1.96	0.59	57.14	3.32	1.20	0.02
11	Genre_New	Western	3.0	1.0	4.0	0.56	1.47	0.20	75.00	7.35	1.99	0.03

O IV (Information Value) do Gênero do Filme transformado (coluna "Genre_New") em relação à Classificação das Notas (coluna "IMDB_Classification") foi de 0,11 o que é considerado um médio poder de separação entre as variáveis. Assim, o gênero do filme apresenta um certo poder de separação para sabermos se o filme apresentará um bom desempenho na nota que receberá.

Quando Odds da categoria de Gênero de filme é maior que 1, é sinal que temos mais chance de aquele gênero apresentar notas altas (notas posicionadas acima do 3º quartil) do que notas baixas. O gênero "Crime", por exemplo, foi responsável por 12,75% das avaliações altas e 9,43% das avaliações baixas, resultando em um Odds de 1,35. Ou seja, filmes do gênero "Crime" possum mais chance de ter notas posicionadas acima do 3º quartil do que notas posicionadas abaixo do 3º quartil.

In [67]:
```
df_iv.get_bivariada(var_escolhida='Certificate')
```

```
==> "Certificate" tem IV de 0.21
```

Out[67]:

	IMDB_Classification	Variavel	Var_Range	#Target_1	#Target_0	Total	%Freq	%Target_1	%Target_0	%Taxa_de_Target_1	Odds	LN(Odds)	IV
0	Certificate	A	65.0	108.0	173.0	24.26	31.86	21.22	37.57	1.50	0.41	0.04
1	Certificate	Approved	4.0	2.0	6.0	0.84	1.96	0.39	66.67	5.03	1.62	0.03
2	Certificate	G	3.0	6.0	9.0	1.26	1.47	1.18	33.33	1.25	0.22	0.00
3	Certificate	GP	0.0	1.0	1.0	0.14	0.00	0.20	0.00	0.01	-4.61	0.00
4	Certificate	PG	4.0	15.0	19.0	2.66	1.96	2.95	21.05	0.66	-0.42	0.00
5	Certificate	PG-13	3.0	35.0	38.0	5.33	1.47	6.88	7.89	0.21	-1.56	0.08
6	Certificate	Passed	5.0	4.0	9.0	1.26	2.45	0.79	55.56	3.10	1.13	0.02
7	Certificate	R	27.0	104.0	131.0	18.37	13.24	20.43	20.61	0.65	-0.43	0.03
8	Certificate	TV-PG	0.0	1.0	1.0	0.14	0.00	0.20	0.00	0.01	-4.61	0.00
9	Certificate	U	57.0	126.0	183.0	25.67	27.94	24.75	31.15	1.13	0.12	0.00
10	Certificate	U/A	0.0	1.0	1.0	0.14	0.00	0.20	0.00	0.01	-4.61	0.00
11	Certificate	UA	36.0	106.0	142.0	19.92	17.65	20.83	25.35	0.85	-0.16	0.01

O IV (Information Value) da Classificação Etária (coluna "Certificate") em relação à Classificação das Notas (coluna "IMDB_Classification") foi de 0,21 o que é considerado um médio poder de separação entre as variáveis. Foi um poder de separação acima do apresentado pelo Gênero do filme. Assim, a classificação etária do filme apresenta um certo poder de separação para sabermos se o filme apresentará um bom desempenho na nota que receberá.

Quando Odds da categoria da classificação etária do filme é maior que 1, é sinal que temos mais chance de aquele gênero apresentar notas altas (notas posicionadas acima do 3º quartil) do que notas baixas. A classificação etária "U", por exemplo, foi responsável por 27,94% das avaliações altas e 24,75% das avaliações baixas, resultando em um Odds de 1,13. Ou seja, filmes da classificação etária "U" possum mais chance de ter notas posicionadas acima do 3º quartil do que notas posicionadas abaixo do 3º quartil.

In [69]:
```
df.describe()
```

Out[69]:

	Unnamed: 0	Runtime	IMDB_Rating	Meta_score	No_of_Votes	Gross	IMDB_Classification
count	713.000000	713.000000	713.000000	713.000000	7.130000e+02	7.130000e+02	713.000000
mean	519.300140	123.690042	7.935203	77.154278	3.533480e+05	7.858395e+07	0.286115
std	295.416331	25.896632	0.288999	12.409392	3.462212e+05	1.150433e+08	0.452261
min	1.000000	72.000000	7.600000	28.000000	2.522900e+04	1.305000e+03	0.000000
25%	263.000000	104.000000	7.700000	70.000000	9.582600e+04	6.153939e+06	0.000000
50%	527.000000	120.000000	7.900000	78.000000	2.363110e+05	3.500000e+07	0.000000
75%	778.000000	136.000000	8.100000	86.000000	5.059180e+05	1.025158e+08	1.000000
max	997.000000	238.000000	9.200000	100.000000	2.303232e+06	9.366622e+08	1.000000

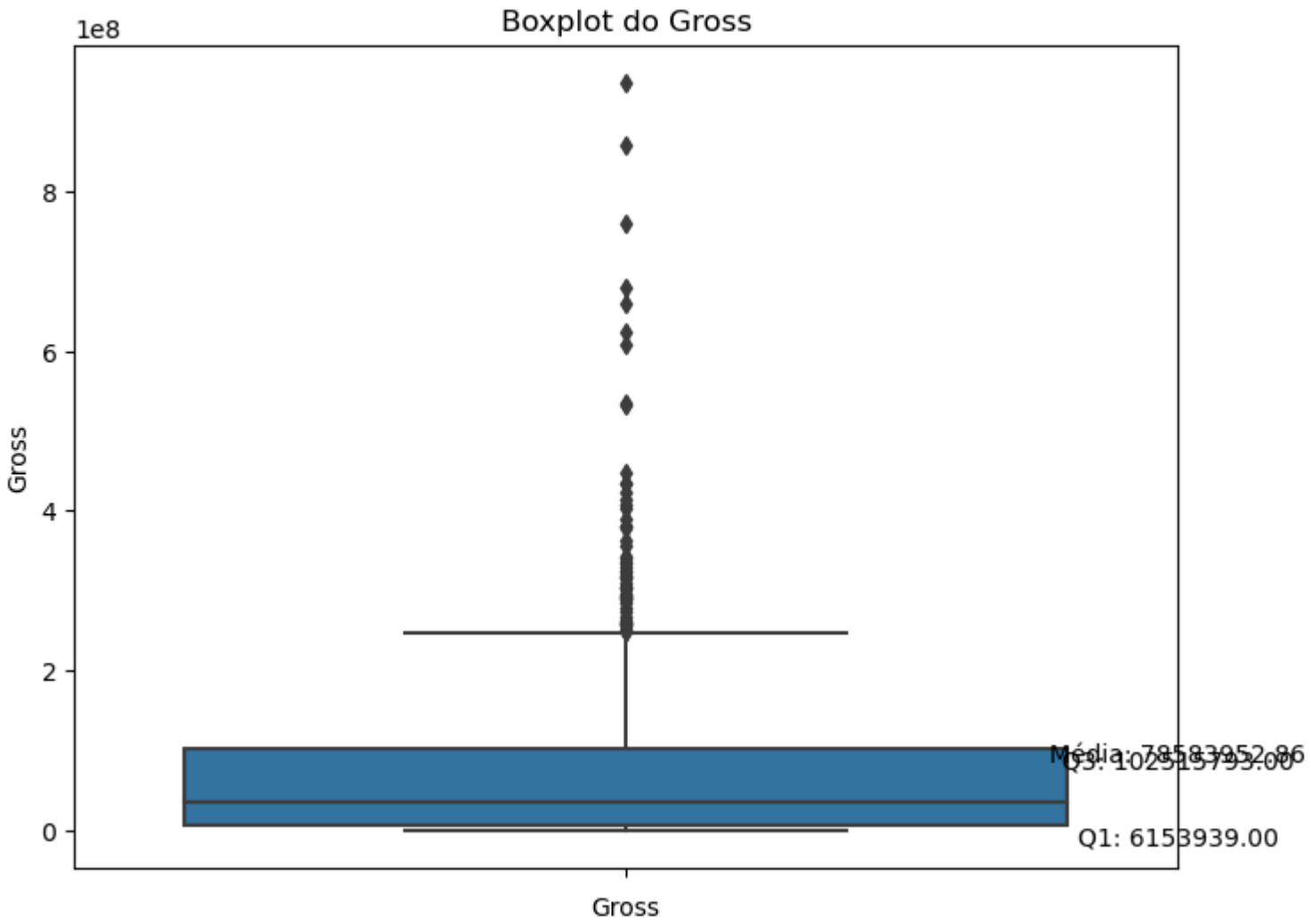
Elaborando o Boxplot da variável Gross

In [71]:
```
mean_value = df['Gross'].mean()
q1_value = df['Gross'].quantile(0.25)
q3_value = df['Gross'].quantile(0.75)

plt.figure(figsize=(8, 6))
sns.boxplot(y='Gross', data=df)
plt.title('Boxplot do Gross')
plt.xlabel('Gross')

plt.text(0.5, mean_value, f'Média: {mean_value:.2f}', color='black', ha='center', va='bottom')
plt.text(0.5, q1_value, f'Q1: {q1_value:.2f}', color='black', ha='center', va='top')
plt.text(0.5, q3_value, f'Q3: {q3_value:.2f}', color='black', ha='center', va='top')
```

Out[71]:
```
Text(0.5, 102515793.0, 'Q3: 102515793.00')
```



Como podemos ver na tabela e no Boxplot acima, o 3º Quartil da variável Gross foi de $ 102.515.793.

Essa será nossa referência para a criação da coluna "Gross_Classification".

In [73]:
```
df['Gross_Classification'] = np.where(df['Gross'] >= 102515793, 1, 0)
```

In [74]:
```
df.head(2)
```

Out[74]:

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Sta
0	1	The Godfather	1972	A	175	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Dia Keat
1	2	The Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Micha Cai

In [75]:
```
df_iv_gross = analise_iv(df,
                  'Gross_Classification',
                          nbins=5)
```

```
C:\Users\mateu\AppData\Local\Temp\ipykernel_14776\1774483247.py:86: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_num[nome_var] = pd.qcut(df_num[var],
```

In [76]:
```
df_iv_gross.get_bivariada(var_escolhida='Genre_New')
```

```
==> "Genre_New" tem IV de 0.59
```

Out[76]:

	Gross_Classification	Variavel	Var_Range	#Target_1	#Target_0	Total	%Freq	%Target_1	%Target_0	%Taxa_de_Target_1	Odds	LN(Odds)	IV
0	Genre_New	Action	60.0	67.0	127.0	17.81	33.52	12.55	47.24	2.67	0.98	0.21
1	Genre_New	Adventure	18.0	40.0	58.0	8.13	10.06	7.49	31.03	1.34	0.29	0.01
2	Genre_New	Animation	31.0	32.0	63.0	8.84	17.32	5.99	49.21	2.89	1.06	0.12
3	Genre_New	Biography	14.0	59.0	73.0	10.24	7.82	11.05	19.18	0.71	-0.34	0.01
4	Genre_New	Comedy	11.0	93.0	104.0	14.59	6.15	17.42	10.58	0.35	-1.05	0.12
5	Genre_New	Crime	9.0	65.0	74.0	10.38	5.03	12.17	12.16	0.41	-0.89	0.06
6	Genre_New	Drama	32.0	159.0	191.0	26.79	17.88	29.78	16.75	0.60	-0.51	0.06
7	Genre_New	Family	1.0	1.0	2.0	0.28	0.56	0.19	50.00	2.95	1.08	0.00
8	Genre_New	Film-Noir	0.0	1.0	1.0	0.14	0.00	0.19	0.00	0.01	-4.61	0.00
9	Genre_New	Horror	2.0	7.0	9.0	1.26	1.12	1.31	22.22	0.85	-0.16	0.00
10	Genre_New	Mystery	1.0	6.0	7.0	0.98	0.56	1.12	14.29	0.50	-0.69	0.00
11	Genre_New	Western	0.0	4.0	4.0	0.56	0.00	0.75	0.00	0.01	-4.61	0.00

O IV (Information Value) do Gênero do Filme transformado (coluna "Genre_New") em relação ao Faturamento transformado (coluna "Gross_Classification") foi de 0,59 o que é considerado um forte poder de separação entre as variáveis. Assim, o gênero do filme apresenta um alto poder de separação para sabermos se o filme apresentará um alto faturamento.

Quando o Odds da categoria de Gênero de filme é maior que 1, é sinal que temos mais chance de aquele gênero apresentar alto faturamento (faturamento posicionado acima do 3º quartil) do que baixo faturamento.

O gênero "Action", por exemplo, foi responsável por 33,52% do total de filmes com alto faturamento e por 12,55% do total de filmes com baixo faturamento, resultando em um Odds de 2,67. Ou seja, filmes do gênero "Action" possuem mais chance de ter um faturamento posicionado acima do 3º quartil do que um faturamento posicionado abaixo do 3º quartil.

Por outro lado, o gênero "Drama", por exemplo, foi responsável por 17,88% do total de filmes com alto faturamento e por 29,78% do total de filmes com baixo faturamento, resultando em um Odds de 0,60. Ou seja, filmes do gênero "Drama" possuem mais chance de ter um faturamento posicionado abaixo do 3º quartil do que um faturamento posicionado acima do 3º quartil.

In [78]:
```
df_iv_gross.get_bivariada(var_escolhida='Certificate')
```

```
==> "Certificate" tem IV de 0.58
```

Out[78]:

	Gross_Classification	Variavel	Var_Range	#Target_1	#Target_0	Total	%Freq	%Target_1	%Target_0	%Taxa_de_Target_1	Odds	LN(Odds)	IV
0	Certificate	A	43.0	130.0	173.0	24.26	24.02	24.34	24.86	0.99	-0.01	0.00
1	Certificate	Approved	0.0	6.0	6.0	0.84	0.00	1.12	0.00	0.01	-4.61	0.00
2	Certificate	G	1.0	8.0	9.0	1.26	0.56	1.50	11.11	0.37	-0.99	0.01
3	Certificate	GP	0.0	1.0	1.0	0.14	0.00	0.19	0.00	0.01	-4.61	0.00
4	Certificate	PG	0.0	19.0	19.0	2.66	0.00	3.56	0.00	0.01	-4.61	0.00
5	Certificate	PG-13	4.0	34.0	38.0	5.33	2.23	6.37	10.53	0.35	-1.05	0.04
6	Certificate	Passed	0.0	9.0	9.0	1.26	0.00	1.69	0.00	0.01	-4.61	0.00
7	Certificate	R	8.0	123.0	131.0	18.37	4.47	23.03	6.11	0.19	-1.66	0.31
8	Certificate	TV-PG	0.0	1.0	1.0	0.14	0.00	0.19	0.00	0.01	-4.61	0.00
9	Certificate	U	59.0	124.0	183.0	25.67	32.96	23.22	32.24	1.42	0.35	0.03
10	Certificate	U/A	0.0	1.0	1.0	0.14	0.00	0.19	0.00	0.01	-4.61	0.00
11	Certificate	UA	64.0	78.0	142.0	19.92	35.75	14.61	45.07	2.45	0.90	0.19

O IV (Information Value) da Classificação Etária (coluna "Certificate") em relação ao Faturamento transformado (coluna "Gross_Classification") foi de 0,58 o que é considerado um forte poder de separação entre as variáveis. Assim, a classificação etária apresenta um alto poder de separação para sabermos se o filme apresentará um alto faturamento.

Quando o Odds da categoria de Classificação Etária de filme é maior que 1, é sinal que temos mais chance de aquele gênero apresentar alto faturamento (faturamento posicionado acima do 3º quartil) do que baixo faturamento.

A Classificação Etária "U", por exemplo, foi responsável por 32,96% do total de filmes com alto faturamento e por 23,22% do total de filmes com baixo faturamento, resultando em um Odds de 1,42. Ou seja, filmes da Classificação Etária "U" possuem mais chance de ter um faturamento posicionado acima do 3º quartil do que um faturamento posicionado abaixo do 3º quartil.

Por outro lado, a Classificação Etária "R", por exemplo, foi responsável por 4,47% do total de filmes com alto faturamento e por 23,03% do total de filmes com baixo faturamento, resultando em um Odds de 0,19. Ou seja, filmes da classificação etária "R" possuem mais chance de ter um

faturamento posicionado abaixo do 3º quartil do que um faturamento posicionado acima do 3º quartil.

Qual filme recomendar para uma pessoa desconhecida (Questão 2A)

Partindo do pressuposto da Teoria Frequentista, o filme que indicarei para uma pessoa desconhecida levará em consideração a popularidade daquele filme expressa pela quantidade de votos que ele recebeu. Ou seja, um filme com muitos votos reflete a atenção que as pessoas estão depositando sobre ele. E também levarei em consideração a qualidade do filme, expressa pela nota que o filme recebeu.

Assim, irei filtrar a base de dados com os 10 filmes com a maior quantidade de votos.

E, destes 10 filmes, irei filtrar aquele que apresentar a maior nota.

In [82]:
```
# Ordenando o dataframe pelos números de votos em ordem decrescente
df_ordenado = df.sort_values(by='No_of_Votes', ascending=False)

# Selecionando os 10 filmes com os maiores números de votos
top_10_votos = df_ordenado.head(10)

# Encontrando o índice do filme com maior nota nos 10 primeiros
ind_top_10 = top_10_votos['IMDB_Rating'].idxmax()

# Acessando o título do filme com a maior nota entre os 10 primeiros
filme_maior_nota_top_10 = top_10_votos.loc[ind_top_10, 'Series_Title']
maior_nota = top_10_votos.loc[ind_top_10, 'IMDB_Rating']

# Resultado
print(f"Filme com a maior nota entre os 10 filmes com mais votos: {filme_maior_nota_top_10} (Rating: {maior_nota})")
```

```
Filme com a maior nota entre os 10 filmes com mais votos: The Godfather (Rating: 9.2)
```

Assim, o filme que eu recomendaria para uma pessoa desconhecida seria o **The Godfather**, que entre os 10 filmes com mais votos, foi o que apresentou a melhor nota (9,2)

Principais fatores relacionados com a alta expectativa de faturamento de um filme (Questão 2B)

Como estamos falando em **expectativa** de faturamento, não faz muito sentido considerarmos fatores como quantidade de votos, nota do IMDB e média ponderada das críticas, que ocorrem a posteriori do lançamento do filme.

Como apresentado na Análise Exploratória de Dados acima, a variável quantitativa restante, Runtime, apresenta fraca Correlação com o Faturamento (Correlação de 0,17). Então essa variável não está fortemente relacionada com alta (ou baixa) expectativa de faturamento.

Já a variável qualitativa Certificate apresentou alto Information Value (IV) em relação ao Faturamento (IV de 0,58). Assim, filmes de classificação etária como o "U" e "UA" possuem mais alta probabilidade de apresentarem alto faturamento se comparadas com outras categorias de classificação etária.

E a variável qualitativa Gênero (coluna "Genre_New") apresentou alto Information Value (IV) em relação ao Faturamento (IV de 0,59). Assim, filmes do gênero "Animation" e "Action" possuem mais alta probabilidade de apresentarem alto faturamento se comparadas com outros gêneros.

Desenvolvimento do Modelo - Questão 3

Os modelos que serão treinados e avaliados pretendem estimar a nota do IMDB. Como o Target é um valor numérico, trata-se de um problema de regressão. Entretanto, iremos usar a estratégia comum de transformar um problema de regressão em um problema de classificação.

Assim como tratado na Análise Exploratória de Dados (EDA), vamos considerar que notas do IMDB maiores ou iguais a 8,1 serão classificadas como 1 e notas abaixo de 8,1 serão classificadas como 0. Ou seja, o nosso Target será a coluna "IMDB_Classification".

Nós iremos utilizar a estratégia de transformar um problema de regressão em um problema de classificação porque os modelos de classificação apresentam mais métricas de performance.

Definição do *Target* e das *Features*

In [89]:
```
# Target (variável resposta)

y = df['IMDB_Classification']
```

In [90]:
```
# Features (variáveis explicativas)
x_var = [
    'Certificate', 'Runtime', 'Genre_New', 'Meta_score', 'No_of_Votes',
    'Gross'
    ]
x = pd.get_dummies(df[x_var], drop_first=True)
```

In [91]:
```
y.head()
```

Out[91]:
```
0    1
1    1
2    1
3    1
4    1
Name: IMDB_Classification, dtype: int32
```

In [92]:
```
x.head()
```

Out[92]:

	Runtime	Meta_score	No_of_Votes	Gross	Certificate_Approved	Certificate_G	Certificate_GP	Certificate_PG	Certificate_PG-13	Certificate_Passed
0	175	100.0	1620367	134966411.0	False	False	False	False	False	False
1	152	84.0	2303232	534858444.0	False	False	False	False	False	False
2	202	90.0	1129952	57300000.0	False	False	False	False	False	False
3	96	96.0	689845	4360000.0	False	False	False	False	False	False
4	201	94.0	1642758	377845905.0	False	False	False	False	False	False

5 rows × 26 columns

Divisão das bases em Treino e Teste

In [94]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.30,
                                                    random_state=42)
```

In [95]:
```
x_train.shape
```

Out[95]:
```
(499, 26)
```

In [96]:
```
x_test.shape
```

Out[96]:
```
(214, 26)
```

Importando Bibliotecas e Métricas de Desempenho

In [98]:

```
import pandas as pd
import numpy as np
from IPython.display import display
from ydata_profiling import ProfileReport
import sweetviz as sv
# Métricas de Desempenho
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from scipy.stats import ks_2samp
# Função para cálculo do KS
def ks_stat(y, y_pred):
    return ks_2samp(y_pred[y==1], y_pred[y!=1]).statistic
# Função para cálculo do desempenho de modelos
def calcula_desempenho(modelo, x_train, y_train, x_test, y_test):

    # Cálculo dos valores preditos
    ypred_train = modelo.predict(x_train)
    ypred_proba_train = modelo.predict_proba(x_train)[:,1]

    ypred_test = modelo.predict(x_test)
    ypred_proba_test = modelo.predict_proba(x_test)[:,1]

    # Métricas de Desempenho
    acc_train = accuracy_score(y_train, ypred_train)
    acc_test = accuracy_score(y_test, ypred_test)

    roc_train = roc_auc_score(y_train, ypred_proba_train)
    roc_test = roc_auc_score(y_test, ypred_proba_test)

    ks_train = ks_stat(y_train, ypred_proba_train)
    ks_test = ks_stat(y_test, ypred_proba_test)

    prec_train = precision_score(y_train, ypred_train, zero_division=0)
    prec_test = precision_score(y_test, ypred_test, zero_division=0)

    recl_train = recall_score(y_train, ypred_train)
    recl_test = recall_score(y_test, ypred_test)

    f1_train = f1_score(y_train, ypred_train)
    f1_test = f1_score(y_test, ypred_test)
```

```
    df_desemp = pd.DataFrame({'Treino':[acc_train, roc_train, ks_train,
                                        prec_train, recl_train, f1_train],
                              'Teste':[acc_test, roc_test, ks_test,
                                       prec_test, recl_test, f1_test]},
                             index=['Acurácia','AUROC','KS',
                                    'Precision','Recall','F1'])

    df_desemp['Variação'] = round(df_desemp['Teste'] / df_desemp['Treino'] - 1, 2)

    return df_desemp
```

Criando o nosso Modelo Base-Line:

Modelo individual: Regressão Logística

In [101…

```
# Definição do modelo
modelo_rl = LogisticRegression(max_iter=1000)

# Ajuste do Modelo
modelo_rl.fit(x_train, y_train)
```

Out[101]:

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In [102…

```
desemp_rl=calcula_desempenho(modelo_rl, x_train, y_train, x_test, y_test)

desemp_rl
```

Out[102]:

	Treino	Teste	Variação
Acurácia	0.571142	0.560748	-0.02
AUROC	0.748136	0.722093	-0.03
KS	0.469503	0.462649	-0.01
Precision	0.377551	0.370968	-0.02
Recall	0.781690	0.741935	-0.05
F1	0.509174	0.494624	-0.03

In [103…

```
percentual_0s = (df['IMDB_Classification'] == 0).mean() * 100

print(f"Participação percentual de 0s na coluna IMDB_Classification: {percentual_0s:.2f}%")
```

```
Participação percentual de 0s na coluna IMDB_Classification: 71.39%
```

Como podemos verificar acima, que 71,39% das observações da coluna IMDB_Classification são do tipo 0 (Nota abaixo de 8,1).

Ou seja, essa base é desbalanceada. Para bases desbalanceadas, a Acurácia não é uma métrica de desempenho muito boa porque, neste tipo de base, a maioria das instâncias pertence a uma classe majoritária, enquanto uma minoria pertence a uma classe minoritária. Então, se um modelo simplesmente predizer a classe majoritária para todas as instâncias, ele ainda pode obter uma alta Acurácia. Isso ocorre porque a maioria das previsões estará correta apenas devido à predominância da classe majoritária, ignorando completamente a classe minoritária.

Ex: uma base de dados com 1000 instâncias, das quais 950 pertencem à classe majoritária (classe 0) e 50 pertencem à classe minoritária (classe 1). Um modelo que prediz sempre a classe 0 terá uma acurácia de 95% (950/1000), mesmo que ele nunca consiga prever corretamente a classe 1. Essa acurácia de 95% pode parecer muito boa à primeira vista, mas o modelo não tem utilidade prática porque falha completamente em identificar a classe minoritária.

Na base em questão, se montarmos um modelo de classificação que estima todas as instâncias como sendo do tipo 0, já teríamos uma Acurácia de 71,39%. Então, iremos olhar com mais ênfase as outras métricas de desempenho

Módelo Individual - Árvore de Decisão (Decision Tree)

In [106…

```
modelo_dt = DecisionTreeClassifier(min_samples_leaf=5,
                                   max_depth=3,
```

```
                                    random_state=42)
```

In [107…
```
modelo_dt.fit(x_train, y_train)
```

Out[107]:
```
DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, random_state=42)
```

In [108…
```
desemp_dt=calcula_desempenho(modelo_dt, x_train, y_train, x_test, y_test)

desemp_dt
```

Out[108]:

	Treino	Teste	Variação
Acurácia	0.845691	0.775701	-0.08
AUROC	0.820028	0.753608	-0.08
KS	0.503196	0.426995	-0.15
Precision	0.891566	0.694444	-0.22
Recall	0.521127	0.403226	-0.23
F1	0.657778	0.510204	-0.22

Ensemble: *Bagging de Regressão Logística*

Para criar um Bagging mais geral, primeiro precisamos criar um modelo base. Nesse caso, estamos escolhendo uma Regressão Logística. Ele está sendo nomeado como "modelo_base", na célula abaixo. Não estamos usando o fit. Estamos apenas estanciando e criando esse objeto da classe LogisticRegression

In [111…
```
# Inicia o estimador base para o Bagging
modelo_base = LogisticRegression(max_iter=1000)
```

Na célula abaixo, estamos definindo que a quantidade de estimadores que estamos escolhendo é 100. Ou seja, serão criados 100 modelos de Regressão Logística. E cada um desses modelos irá olhar para uma base diferente, seja em termos de observações, seja em termos de features. Assim, cada modelo terá uma visão diferente do conjunto de dados

In [113…
```
num_estimadores = 100
```

O argumento "max_samples" é o nº de amostras que o Bagging vai sortear do X, para fazer o treinamento de cada estimador base.

O default do argumento "max_samples" é 1. Ou seja, sortear 100% sempre. Isso perde um pouco a motivação de usar uma modelo de Bagging.

Assim, estamos considerando que o argumento "max_samples" será de 50%. Ou seja, cada modelo terá 50% da base em termos de observações para fazer o treinamento.

Na documentação do BaggingClassifier, no argumento "max_samples" é informado que há reposição de observações entre os modelos. Se não houvesse, não seria um modelo de bagging, e sim de pasting.

O argumento max_features é o nº de features que serão sorteadas do x para treinar cada estimador base. O default desse argumento é 1. Ou seja, ele vai sorter todas as features. Também vamos considerar 50% para esse argumento. Ou seja, cada modelo irá considerar metade das features que estão disponíveis.

Modelos de Bagging permitem paralelização (processos simultâneos independentes). No argumento n_jobs iremos considerar -1. O -1 irá fazer com que o computador utilize todos os seus cores (independentemente da quantidade de cores que cada computador tiver).

In [115…
```
# Classificador de Bagging
from sklearn.ensemble import BaggingClassifier

modelo_bagging = BaggingClassifier(estimator = modelo_base,
                                   n_estimators = num_estimadores,
                                   max_samples=0.5,
                                   max_features=0.5,
                                   random_state = 42,
                                   n_jobs=-1)
```

In [116…
```
# Ajuste do Modelo
modelo_bagging.fit(x_train,y_train)
```

Out[116]:

▸ BaggingClassifier

▸ estimator: LogisticRegression

▸ LogisticRegression

In [117…
```
desemp_bagging=calcula_desempenho(modelo_bagging, x_train, y_train, x_test, y_test)

desemp_bagging
```

Out[117]:

	Treino	Teste	Variação
Acurácia	0.783567	0.771028	-0.02
AUROC	0.819071	0.772814	-0.06
KS	0.546159	0.529075	-0.03
Precision	0.947368	1.000000	0.06
Recall	0.253521	0.209677	-0.17
F1	0.400000	0.346667	-0.13

Ensemble: *Bagging Random Forest*

O modelo de Bagging que utiliza Árvores de Decisão é o modelo conhecido como Random Forest (Floresta Aleatória).

Iremos utilizar o RandomForestClassifier. Muitos dos seus argumentos são os mesmos argumentos do algoritmo de Árvore de Decisão.

O que há de diferente no algoritmo de Random Forest em relação ao algoritmo de Árvore de Decisão é o fato de, no RandomForestClassifier, nós termos um argumento de número de estimadores, que é o argumento em que informamos quantas Árvores de Decisão serão treinadas para fazer as estimativas.

In [120…
```
from sklearn.ensemble import RandomForestClassifier

modelo_rf = RandomForestClassifier(n_estimators=100,
                                   min_samples_leaf=10,
                                   max_depth=3,
                                   random_state = 42)
```

In [121…
```
modelo_rf.fit(x_train, y_train)
```

Out[121]:
```
RandomForestClassifier(max_depth=3, min_samples_leaf=10, random_state=42)
```

In [122…
```
desemp_rf = calcula_desempenho(modelo_rf, x_train, y_train, x_test, y_test)

desemp_rf
```

Out[122]:

	Treino	Teste	Variação
Acurácia	0.803607	0.789720	-0.02
AUROC	0.870419	0.818230	-0.06
KS	0.592417	0.512309	-0.14
Precision	0.892857	0.814815	-0.09
Recall	0.352113	0.354839	0.01
F1	0.505051	0.494382	-0.02

Ensemble: *Boosting* - **AdaBoost**

In [124…
```
modelo_ada = AdaBoostClassifier(n_estimators=50,
                                learning_rate = 0.6,
                                random_state = 42)
```

In [125…
```
# Ajuste do Modelo
modelo_ada.fit(x_train, y_train)
```

Out[125]:
```
AdaBoostClassifier(learning_rate=0.6, random_state=42)
```

In [126…
```
desemp_ada = calcula_desempenho(modelo_ada,
                                x_train, y_train,x_test, y_test)
desemp_ada
```

Out[126]:

	Treino	Teste	Variação
Acurácia	0.873747	0.822430	-0.06
AUROC	0.950872	0.862320	-0.09
KS	0.772557	0.590620	-0.24
Precision	0.862385	0.750000	-0.13
Recall	0.661972	0.580645	-0.12
F1	0.749004	0.654545	-0.13

Ensemble: *Boosting* - **Gradient Boosting**

O Gradient Boosting também é baseado em Árvore de Decisão. Por isso que muitos argumentos do método GradientBoostingClassifier são argumentos de Árvores de Decisão.

O argumento subsample tem como padrão 1. Ele vai definir quanto que cada modelo terá da base para ser treinado. Neste exemplo usaremos 0.2.

In [129…
```
modelo_gb = GradientBoostingClassifier(n_estimators=100,
                                       learning_rate = 0.4,
                                       subsample=0.2,
                                       min_samples_leaf=10,
                                       max_depth=2,
                                       random_state = 42)
```

In [130…
```
# Ajuste do modelo
modelo_gb.fit(x_train,y_train)
```

Out[130]:
```
GradientBoostingClassifier(learning_rate=0.4, max_depth=2, min_samples_leaf=10,
                           random_state=42, subsample=0.2)
```

In [131…
```
# Métricas de Desempenho
desemp_gb = calcula_desempenho(modelo_gb, x_train, y_train, x_test, y_test)
```

```
desemp_gb
```

Out[131]:

	Treino	Teste	Variação
Acurácia	0.879760	0.789720	-0.10
AUROC	0.943563	0.829160	-0.12
KS	0.737188	0.511036	-0.31
Precision	0.810606	0.660377	-0.19
Recall	0.753521	0.564516	-0.25
F1	0.781022	0.608696	-0.22

Comparação dos Modelos

In [133…

```
desemp_rl
```

Out[133]:

	Treino	Teste	Variação
Acurácia	0.571142	0.560748	-0.02
AUROC	0.748136	0.722093	-0.03
KS	0.469503	0.462649	-0.01
Precision	0.377551	0.370968	-0.02
Recall	0.781690	0.741935	-0.05
F1	0.509174	0.494624	-0.03

In [134…

```
desemp_dt
```

Out[134]:

	Treino	Teste	Variação
Acurácia	0.845691	0.775701	-0.08
AUROC	0.820028	0.753608	-0.08
KS	0.503196	0.426995	-0.15
Precision	0.891566	0.694444	-0.22
Recall	0.521127	0.403226	-0.23
F1	0.657778	0.510204	-0.22

In [135…

```
desemp_bagging
```

Out[135]:

	Treino	Teste	Variação
Acurácia	0.783567	0.771028	-0.02
AUROC	0.819071	0.772814	-0.06
KS	0.546159	0.529075	-0.03
Precision	0.947368	1.000000	0.06
Recall	0.253521	0.209677	-0.17
F1	0.400000	0.346667	-0.13

In [136…

```
desemp_rf
```

Out[136]:

	Treino	Teste	Variação
Acurácia	0.803607	0.789720	-0.02
AUROC	0.870419	0.818230	-0.06
KS	0.592417	0.512309	-0.14
Precision	0.892857	0.814815	-0.09
Recall	0.352113	0.354839	0.01
F1	0.505051	0.494382	-0.02

In [137…

```
desemp_ada
```

Out[137]:

	Treino	Teste	Variação
Acurácia	0.873747	0.822430	-0.06
AUROC	0.950872	0.862320	-0.09
KS	0.772557	0.590620	-0.24
Precision	0.862385	0.750000	-0.13
Recall	0.661972	0.580645	-0.12
F1	0.749004	0.654545	-0.13

In [138…

```
desemp_gb
```

Out[138]:

	Treino	Teste	Variação
Acurácia	0.879760	0.789720	-0.10
AUROC	0.943563	0.829160	-0.12
KS	0.737188	0.511036	-0.31
Precision	0.810606	0.660377	-0.19
Recall	0.753521	0.564516	-0.25
F1	0.781022	0.608696	-0.22

Medida a ser escolhida para avaliação do desempenho dos modelos

Uma vez que estamos na posição de orientar qual o próximo filme que será produzido, uma boa medida de avaliação do desempenho dos modelos poderia ser o Precision. O Precision mediria o percentual da quantidade de filmes que o modelo prevê como boa nota no IMDB que de fato tiveram boa nota no IMDB. Ou seja, quanto mais alto fosse o Precision, menor seria a probabilidade de indicarmos um filme que acreditamos, pelo modelo, que seria bem avaliado e ele, de fato, ser mal avaliado. Ou seja, deixaríamos de propor a produção de um filme, ser gasto dinheiro na sua produção, e ele ser mal avaliado.

Como a produção de um filme envolve muito dinheiro, a medida de perfomance Precision seria mais indicada que a medida Recall. O Recall iria nos mostrar o percentual da quantidade de filmes que de fato foram bem avaliados em relação a quantidade de filmes que o modelo previu que seriam bem avaliados.

Ou seja, quanto mais alto o Precision, menor a probabilidade de indicarmos a produção de um filme que será mal avaliado. E quanto maior o Recall, menor a probabilidade de deixarmos de indicar a produção de um filme que será bem avaliado.

Entretanto, **iremos usar a medida AUROC** para avaliar a performance dos modelos, pois ela é uma medida que sintetizará o desempenho dos acertos do modelo. Quanto mais alto for o AUROC, maiores serão os Verdadeiros Positivos (TP: Modelo previu que o filme será bem avaliado e ele, de fato, é bem avaliado) e Verdadeiros Negativos (TN: Modelo prevê que o filme será mal avaliado e ele, de fato, é mal avaliado) e menores serão os Falsos Positivos (FP: Modelo prevê que o filme será bem avaliado mas ele, de fato, é mal avaliado) e Falsos Negativos (FN: Modelo prevê que o filme será mal avaliado mas ele, de fato, é bem avaliado).

Tunagem de Hiperparâmetros

Ao avaliarmos os 6 modelos elaborados acima (Regressão Logística, Árvore de Decisão, Bagging de Regressão Logística, Random Forest, AdaBoost e Gradient Boosting), verificamos que **os 2 modelos que apresentaram as maiores medidas AUROC na base de Teste foram o AdaBoost e o Gradient Boosting**. E verifamos que não há uma perda muito significativa de desempenho entre a base de Treino e a base de Teste, o que denota que os modelos não estão apresentando overfitting.

Agora iremos realizar a tunagem de Hiperparâmetros desses dois modelos, afim de aumentar ainda mais o desempenho desses dois modelos, tomando a medida AUROC como referência. Ou seja, iremos buscar a configuração de Hiperparâmetros de cada um desses dois modelos que otimize a medida de perfomance AUROC.

Para tunar os Hiperparâmetros iremos utilizar metodologia Bayesian Search, que é mais performática que a metodologia Grid Search.

In [143…
```
# Importando as bibliotecas
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from sklearn.metrics import roc_auc_score
import optuna
roc_auc_scorer = make_scorer(roc_auc_score, needs_proba=True)
```

Tunagem dos Hiperparâmetros do modelo AdaBoost

In [145…
```
# Definindo nossa função objetivo
def objective(trial):

    # Definindo o espaço de busca dos hiperparâmetros
    n_estimators = trial.suggest_int('n_estimators', 10, 500)
    learning_rate = trial.suggest_float('learning_rate', 0.01,0.99)


    # Criando, treinando e retornando o resultado do modelo
    modelo = AdaBoostClassifier(n_estimators = n_estimators, learning_rate = learning_rate, random_state = 42)
    score = cross_val_score(estimator = modelo, X = x_train, y = y_train, n_jobs = -1, cv = 3, scoring = roc_auc_scorer).mean()
    return score

# Criando o estudo do Optuna e buscando os melhores parâmetros
study = optuna.create_study(direction = 'maximize')
study.optimize(objective, n_trials = 100)
```

```
# Melhores hiperpaâmetros encontrados
best_params = study.best_params
print(best_params)
```

```
[I 2024-07-16 15:06:10,490] A new study created in memory with name: no-name-4c815de5-751b-40c3-8dc6-69f3af0806b5
[I 2024-07-16 15:06:11,656] Trial 0 finished with value: 0.8268147684605758 and parameters: {'n_estimators': 186, 'learning_rat
e': 0.602211615621923}. Best is trial 0 with value: 0.8268147684605758.
[I 2024-07-16 15:06:13,344] Trial 1 finished with value: 0.8436227327810557 and parameters: {'n_estimators': 390, 'learning_rat
e': 0.24340146305149452}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:15,583] Trial 2 finished with value: 0.8314150128136361 and parameters: {'n_estimators': 478, 'learning_rat
e': 0.45065502607488556}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:16,051] Trial 3 finished with value: 0.8409197012138189 and parameters: {'n_estimators': 143, 'learning_rat
e': 0.4058050678708481}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:17,224] Trial 4 finished with value: 0.8123199634463716 and parameters: {'n_estimators': 365, 'learning_rat
e': 0.9388137237599552}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:18,056] Trial 5 finished with value: 0.834252140572541 and parameters: {'n_estimators': 257, 'learning_rat
e': 0.49201342207996845}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:18,954] Trial 6 finished with value: 0.8029270218725788 and parameters: {'n_estimators': 252, 'learning_rat
e': 0.852411219504153}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:19,957] Trial 7 finished with value: 0.8294445437749568 and parameters: {'n_estimators': 343, 'learning_rat
e': 0.603331346725953}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:20,332] Trial 8 finished with value: 0.8271102767348074 and parameters: {'n_estimators': 117, 'learning_rat
e': 0.6089399172409856}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:20,676] Trial 9 finished with value: 0.8087788306812086 and parameters: {'n_estimators': 92, 'learning_rat
e': 0.7600300149024075}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:20,756] Trial 10 finished with value: 0.7772854709259591 and parameters: {'n_estimators': 18, 'learning_rat
e': 0.07923851912281166}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:22,095] Trial 11 finished with value: 0.84094577547331 and parameters: {'n_estimators': 455, 'learning_rat
e': 0.22626618495670353}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:23,507] Trial 12 finished with value: 0.8414672606631305 and parameters: {'n_estimators': 486, 'learning_rat
e': 0.18812827935397944}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:24,632] Trial 13 finished with value: 0.8426704014939309 and parameters: {'n_estimators': 397, 'learning_rat
e': 0.24408718262200776}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:25,771] Trial 14 finished with value: 0.839243498817967 and parameters: {'n_estimators': 390, 'learning_rat
e': 0.3161616019452073}. Best is trial 1 with value: 0.8436227327810557.
[I 2024-07-16 15:06:26,652] Trial 15 finished with value: 0.856675382919125 and parameters: {'n_estimators': 302, 'learning_rat
e': 0.0439241706408752}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:27,532] Trial 16 finished with value: 0.8406558545403978 and parameters: {'n_estimators': 294, 'learning_rat
e': 0.023444703001643512}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:28,375] Trial 17 finished with value: 0.8532826251067803 and parameters: {'n_estimators': 309, 'learning_rat
e': 0.0983700403200563}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:29,378] Trial 18 finished with value: 0.8510557591831058 and parameters: {'n_estimators': 307, 'learning_rat
e': 0.11549110672113572}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:30,124] Trial 19 finished with value: 0.8405546615809444 and parameters: {'n_estimators': 183, 'learning_rat
e': 0.35090792363580503}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:31,004] Trial 20 finished with value: 0.8347587261855097 and parameters: {'n_estimators': 232, 'learning_rat
e': 0.02262022920451864}. Best is trial 15 with value: 0.856675382919125.
```

```
[I 2024-07-16 15:06:32,050] Trial 21 finished with value: 0.8556646949559966 and parameters: {'n_estimators': 320, 'learning_rat
e': 0.14774620617053558}. Best is trial 15 with value: 0.856675382919125.
[I 2024-07-16 15:06:33,154] Trial 22 finished with value: 0.8571025041222162 and parameters: {'n_estimators': 311, 'learning_rat
e': 0.14108945971989567}. Best is trial 22 with value: 0.8571025041222162.
[I 2024-07-16 15:06:34,633] Trial 23 finished with value: 0.8491169517452372 and parameters: {'n_estimators': 429, 'learning_rat
e': 0.1575858017190109}. Best is trial 22 with value: 0.8571025041222162.
[I 2024-07-16 15:06:35,632] Trial 24 finished with value: 0.8395365238293898 and parameters: {'n_estimators': 340, 'learning_rat
e': 0.3129288122597207}. Best is trial 22 with value: 0.8571025041222162.
[I 2024-07-16 15:06:36,444] Trial 25 finished with value: 0.8407725678924053 and parameters: {'n_estimators': 277, 'learning_rat
e': 0.024617579701611936}. Best is trial 22 with value: 0.8571025041222162.
[I 2024-07-16 15:06:37,084] Trial 26 finished with value: 0.8585061684248169 and parameters: {'n_estimators': 221, 'learning_rat
e': 0.1523177788537226}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:37,641] Trial 27 finished with value: 0.8527207868963188 and parameters: {'n_estimators': 188, 'learning_rat
e': 0.22056151237352364}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:38,343] Trial 28 finished with value: 0.8502139330909668 and parameters: {'n_estimators': 233, 'learning_rat
e': 0.09422857778933676}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:38,952] Trial 29 finished with value: 0.8201049923515505 and parameters: {'n_estimators': 197, 'learning_rat
e': 0.6922167113132978}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:40,121] Trial 30 finished with value: 0.8346209050996286 and parameters: {'n_estimators': 210, 'learning_rat
e': 0.3872635026472707}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:41,250] Trial 31 finished with value: 0.8536290402685897 and parameters: {'n_estimators': 319, 'learning_rat
e': 0.1660337140894626}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:42,061] Trial 32 finished with value: 0.8429013449351371 and parameters: {'n_estimators': 279, 'learning_rat
e': 0.27797872041085536}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:43,085] Trial 33 finished with value: 0.8511730933508154 and parameters: {'n_estimators': 365, 'learning_rat
e': 0.13867484254661042}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:44,004] Trial 34 finished with value: 0.8556851818741681 and parameters: {'n_estimators': 329, 'learning_rat
e': 0.0817824190258325}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:44,441] Trial 35 finished with value: 0.8528281880127938 and parameters: {'n_estimators': 156, 'learning_rat
e': 0.06858064597955721}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:45,567] Trial 36 finished with value: 0.8338895842024753 and parameters: {'n_estimators': 353, 'learning_rat
e': 0.014938353073305127}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:46,867] Trial 37 finished with value: 0.8312349762600076 and parameters: {'n_estimators': 410, 'learning_rat
e': 0.449902640290804}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:47,841] Trial 38 finished with value: 0.8548290770208792 and parameters: {'n_estimators': 255, 'learning_rat
e': 0.19090126269038757}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:48,815] Trial 39 finished with value: 0.8422705961817352 and parameters: {'n_estimators': 279, 'learning_rat
e': 0.2759438749304487}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:49,759] Trial 40 finished with value: 0.7970627967499055 and parameters: {'n_estimators': 224, 'learning_rat
e': 0.9605525242247829}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:50,886] Trial 41 finished with value: 0.8522837326419931 and parameters: {'n_estimators': 317, 'learning_rat
e': 0.06997683232438105}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:51,837] Trial 42 finished with value: 0.8520757593817668 and parameters: {'n_estimators': 328, 'learning_rat
e': 0.1767487531103771}. Best is trial 26 with value: 0.8585061684248169.
```

```
[I 2024-07-16 15:06:52,850] Trial 43 finished with value: 0.8297226692095278 and parameters: {'n_estimators': 368, 'learning_rat
e': 0.5436267243094254}. Best is trial 26 with value: 0.8585061684248169.
[I 2024-07-16 15:06:53,598] Trial 44 finished with value: 0.8606914396964459 and parameters: {'n_estimators': 258, 'learning_rat
e': 0.11855396042600441}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:54,343] Trial 45 finished with value: 0.8546068249995034 and parameters: {'n_estimators': 256, 'learning_rat
e': 0.057822583338838565}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:54,832] Trial 46 finished with value: 0.8529107565011821 and parameters: {'n_estimators': 165, 'learning_rat
e': 0.12050051541206433}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:55,096] Trial 47 finished with value: 0.8407440103701055 and parameters: {'n_estimators': 93, 'learning_rat
e': 0.2585789660655905}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:55,921] Trial 48 finished with value: 0.8472135308023919 and parameters: {'n_estimators': 289, 'learning_rat
e': 0.22542719630274813}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:56,693] Trial 49 finished with value: 0.8550016637860818 and parameters: {'n_estimators': 265, 'learning_rat
e': 0.05325714262182693}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:57,392] Trial 50 finished with value: 0.8047584281939727 and parameters: {'n_estimators': 234, 'learning_rat
e': 0.838619251694458}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:58,207] Trial 51 finished with value: 0.8528163925144526 and parameters: {'n_estimators': 299, 'learning_rat
e': 0.1362158598594797}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:06:59,339] Trial 52 finished with value: 0.8489213947990543 and parameters: {'n_estimators': 340, 'learning_rat
e': 0.11105548897875946}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:00,550] Trial 53 finished with value: 0.846980104098377 and parameters: {'n_estimators': 383, 'learning_rat
e': 0.1987478857255345}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:01,593] Trial 54 finished with value: 0.8529449013648014 and parameters: {'n_estimators': 330, 'learning_rat
e': 0.14649117788011615}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:02,662] Trial 55 finished with value: 0.8553722907602758 and parameters: {'n_estimators': 305, 'learning_rat
e': 0.09301639046151265}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:03,560] Trial 56 finished with value: 0.8435743091562866 and parameters: {'n_estimators': 244, 'learning_rat
e': 0.31184888211841955}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:04,380] Trial 57 finished with value: 0.8551171355066849 and parameters: {'n_estimators': 211, 'learning_rat
e': 0.05491215680437403}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:05,782] Trial 58 finished with value: 0.8470906092933627 and parameters: {'n_estimators': 355, 'learning_rat
e': 0.20801484112138408}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:05,989] Trial 59 finished with value: 0.8428994824880306 and parameters: {'n_estimators': 37, 'learning_rat
e': 0.17339500389299545}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:07,409] Trial 60 finished with value: 0.8488264099966227 and parameters: {'n_estimators': 444, 'learning_rat
e': 0.11694996585331419}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:08,316] Trial 61 finished with value: 0.8569280549099072 and parameters: {'n_estimators': 300, 'learning_rat
e': 0.09052039295189732}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:09,064] Trial 62 finished with value: 0.8562159792995213 and parameters: {'n_estimators': 267, 'learning_rat
e': 0.14556058836305968}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:09,813] Trial 63 finished with value: 0.8563587669110198 and parameters: {'n_estimators': 268, 'learning_rat
e': 0.045050712710644474}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:10,668] Trial 64 finished with value: 0.8469558922859926 and parameters: {'n_estimators': 269, 'learning_rat
e': 0.0319036806620695}. Best is trial 44 with value: 0.8606914396964459.
```

```
[I 2024-07-16 15:07:11,471] Trial 65 finished with value: 0.8301150247332975 and parameters: {'n_estimators': 293, 'learning_rat
e': 0.015463537846458458}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:12,131] Trial 66 finished with value: 0.8574532649939407 and parameters: {'n_estimators': 241, 'learning_rat
e': 0.10227714822346468}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:12,726] Trial 67 finished with value: 0.8558211405129427 and parameters: {'n_estimators': 213, 'learning_rat
e': 0.08769268573812132}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:13,446] Trial 68 finished with value: 0.8272654806603493 and parameters: {'n_estimators': 250, 'learning_rat
e': 0.6575046609165651}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:13,935] Trial 69 finished with value: 0.845174151220772 and parameters: {'n_estimators': 174, 'learning_rat
e': 0.04699573071710929}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:14,352] Trial 70 finished with value: 0.7844900371496116 and parameters: {'n_estimators': 137, 'learning_rat
e': 0.010829909882255434}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:15,115] Trial 71 finished with value: 0.8569212259371835 and parameters: {'n_estimators': 284, 'learning_rat
e': 0.14937039672839117}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:15,976] Trial 72 finished with value: 0.8550891988000874 and parameters: {'n_estimators': 288, 'learning_rat
e': 0.11034646679570342}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:16,668] Trial 73 finished with value: 0.8493714861831257 and parameters: {'n_estimators': 242, 'learning_rat
e': 0.2500455299358979}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:17,473] Trial 74 finished with value: 0.85495820668693 and parameters: {'n_estimators': 280, 'learning_rat
e': 0.1643075042573176}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:18,145] Trial 75 finished with value: 0.8578220295210283 and parameters: {'n_estimators': 193, 'learning_rat
e': 0.08669729050677479}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:18,841] Trial 76 finished with value: 0.857238462760991 and parameters: {'n_estimators': 192, 'learning_rat
e': 0.0815245224876654}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:19,635] Trial 77 finished with value: 0.8542989004112282 and parameters: {'n_estimators': 207, 'learning_rat
e': 0.19627059313530004}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:20,354] Trial 78 finished with value: 0.8519813953950375 and parameters: {'n_estimators': 193, 'learning_rat
e': 0.12830057802736866}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:20,981] Trial 79 finished with value: 0.8552934471661006 and parameters: {'n_estimators': 137, 'learning_rat
e': 0.08447916801463712}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:21,875] Trial 80 finished with value: 0.851364304587083 and parameters: {'n_estimators': 222, 'learning_rat
e': 0.2280340979868314}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:22,557] Trial 81 finished with value: 0.8577065578004252 and parameters: {'n_estimators': 187, 'learning_rat
e': 0.08087466734955553}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:23,060] Trial 82 finished with value: 0.8567976836124521 and parameters: {'n_estimators': 178, 'learning_rat
e': 0.09649770923293305}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:23,615] Trial 83 finished with value: 0.8583310983968055 and parameters: {'n_estimators': 191, 'learning_rat
e': 0.1686997784297236}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:24,076] Trial 84 finished with value: 0.8552667520909073 and parameters: {'n_estimators': 152, 'learning_rat
e': 0.07371426631544267}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:24,445] Trial 85 finished with value: 0.8466703170629954 and parameters: {'n_estimators': 120, 'learning_rat
e': 0.17968382608868835}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:25,008] Trial 86 finished with value: 0.8529554552317381 and parameters: {'n_estimators': 195, 'learning_rat
e': 0.12385746551921321}. Best is trial 44 with value: 0.8606914396964459.
```

```
[I 2024-07-16 15:07:25,467] Trial 87 finished with value: 0.8564307815324712 and parameters: {'n_estimators': 171, 'learning_rat
e': 0.06726511346615865}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:26,118] Trial 88 finished with value: 0.8331495718854919 and parameters: {'n_estimators': 218, 'learning_rat
e': 0.543461427176281}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:26,668] Trial 89 finished with value: 0.8543212497765063 and parameters: {'n_estimators': 200, 'learning_rat
e': 0.10226165825931509}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:27,317] Trial 90 finished with value: 0.8571074706478335 and parameters: {'n_estimators': 231, 'learning_rat
e': 0.15650046732994505}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:28,028] Trial 91 finished with value: 0.8553902944156387 and parameters: {'n_estimators': 229, 'learning_rat
e': 0.162384647300749}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:28,471] Trial 92 finished with value: 0.852725753421936 and parameters: {'n_estimators': 162, 'learning_rat
e': 0.13232983509311033}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:29,072] Trial 93 finished with value: 0.8534794236843674 and parameters: {'n_estimators': 183, 'learning_rat
e': 0.21096937748715108}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:29,700] Trial 94 finished with value: 0.8499507072332478 and parameters: {'n_estimators': 241, 'learning_rat
e': 0.03698143597238088}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:30,279] Trial 95 finished with value: 0.8558993632914159 and parameters: {'n_estimators': 204, 'learning_rat
e': 0.18158181327814377}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:30,810] Trial 96 finished with value: 0.8496235373582057 and parameters: {'n_estimators': 188, 'learning_rat
e': 0.2877543829303197}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:31,584] Trial 97 finished with value: 0.834196267159346 and parameters: {'n_estimators': 231, 'learning_rat
e': 0.38551126371147304}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:32,293] Trial 98 finished with value: 0.8538512922899656 and parameters: {'n_estimators': 257, 'learning_rat
e': 0.07227385661969374}. Best is trial 44 with value: 0.8606914396964459.
[I 2024-07-16 15:07:32,896] Trial 99 finished with value: 0.8554716212726227 and parameters: {'n_estimators': 218, 'learning_rat
e': 0.10696701144122325}. Best is trial 44 with value: 0.8606914396964459.
{'n_estimators': 258, 'learning_rate': 0.11855396042600441}
```

In [146…

```
# Avaliando o modelo com os melhores parâmetros
modelo_ada_tunado = AdaBoostClassifier(**best_params, random_state = 42)
modelo_ada_tunado.fit(x_train, y_train)
y_pred_proba = modelo_ada_tunado.predict_proba(x_test)[:, 1]
roc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC: {roc:.4f}")
```

```
ROC: 0.8682
```

In [147…

```
desemp_ada_tunado=calcula_desempenho(modelo_ada_tunado, x_train, y_train, x_test, y_test)

desemp_ada_tunado
```

Out[147]:

	Treino	Teste	Variação
Acurácia	0.875752	0.817757	-0.07
AUROC	0.950635	0.868209	-0.09
KS	0.756953	0.604202	-0.20
Precision	0.892157	0.734694	-0.18
Recall	0.640845	0.580645	-0.09
F1	0.745902	0.648649	-0.13

Tunagem dos Hiperparâmetros do modelo Gradient Boosting

In [149…

```
# Definindo nossa função objetivo
def objective(trial):

    # Definindo o espaço de busca dos hiperparâmetros
    n_estimators = trial.suggest_int('n_estimators', 10, 500)
    learning_rate = trial.suggest_float('learning_rate', 0.01,0.99)
    subsample = trial.suggest_float('subsample', 0.1, 1, step = 0.1)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 5, 15)
    max_depth = trial.suggest_int('max_depth', 2, 5)


    # Criando, treinando e retornando o resultado do modelo
    modelo = GradientBoostingClassifier(n_estimators = n_estimators, learning_rate = learning_rate, subsample= subsample, min_sam
    score = cross_val_score(estimator = modelo, X = x_train, y = y_train, n_jobs = -1, cv = 3, scoring = roc_auc_scorer).mean()
    return score

# Criando o estudo do Optuna e buscando os melhores parâmetros
study = optuna.create_study(direction = 'maximize')
study.optimize(objective, n_trials = 100)

# Melhores hiperpaâmetros encontrados
best_params = study.best_params
print(best_params)
```

```
[I 2024-07-16 15:07:34,221] A new study created in memory with name: no-name-ce28a510-9a3d-4d71-b3c5-3158912702a9
[I 2024-07-16 15:07:34,437] Trial 0 finished with value: 0.844395027514552 and parameters: {'n_estimators': 73, 'learning_rate':
0.5905324110271619, 'subsample': 0.8, 'min_samples_leaf': 5, 'max_depth': 2}. Best is trial 0 with value: 0.844395027514552.
[I 2024-07-16 15:07:35,950] Trial 1 finished with value: 0.8523278105568468 and parameters: {'n_estimators': 451, 'learning_rat
e': 0.10321098205889392, 'subsample': 0.7000000000000001, 'min_samples_leaf': 12, 'max_depth': 5}. Best is trial 1 with value:
0.8523278105568468.
[I 2024-07-16 15:07:36,376] Trial 2 finished with value: 0.8462922403003755 and parameters: {'n_estimators': 203, 'learning_rat
e': 0.7607675219980904, 'subsample': 0.8, 'min_samples_leaf': 10, 'max_depth': 2}. Best is trial 1 with value: 0.852327810556846
8.
[I 2024-07-16 15:07:36,527] Trial 3 finished with value: 0.8197560939269325 and parameters: {'n_estimators': 40, 'learning_rat
e': 0.7610042322054561, 'subsample': 0.6, 'min_samples_leaf': 8, 'max_depth': 5}. Best is trial 1 with value: 0.852327810556846
8.
[I 2024-07-16 15:07:36,934] Trial 4 finished with value: 0.8486538232314201 and parameters: {'n_estimators': 187, 'learning_rat
e': 0.44695625602886413, 'subsample': 0.8, 'min_samples_leaf': 10, 'max_depth': 2}. Best is trial 1 with value: 0.85232781055684
68.
[I 2024-07-16 15:07:37,309] Trial 5 finished with value: 0.8409954407294832 and parameters: {'n_estimators': 98, 'learning_rat
e': 0.301068781829219, 'subsample': 0.8, 'min_samples_leaf': 5, 'max_depth': 4}. Best is trial 1 with value: 0.8523278105568468.
[I 2024-07-16 15:07:38,198] Trial 6 finished with value: 0.8405981186800963 and parameters: {'n_estimators': 289, 'learning_rat
e': 0.56428861902183, 'subsample': 0.8, 'min_samples_leaf': 9, 'max_depth': 4}. Best is trial 1 with value: 0.8523278105568468.
[I 2024-07-16 15:07:38,353] Trial 7 finished with value: 0.8258922363271551 and parameters: {'n_estimators': 28, 'learning_rat
e': 0.8769474994897937, 'subsample': 1.0, 'min_samples_leaf': 9, 'max_depth': 5}. Best is trial 1 with value: 0.852327810556846
8.
[I 2024-07-16 15:07:38,679] Trial 8 finished with value: 0.5580735840435466 and parameters: {'n_estimators': 228, 'learning_rat
e': 0.910116423122033, 'subsample': 0.30000000000000004, 'min_samples_leaf': 6, 'max_depth': 2}. Best is trial 1 with value: 0.8
523278105568468.
[I 2024-07-16 15:07:39,288] Trial 9 finished with value: 0.8446160379045237 and parameters: {'n_estimators': 419, 'learning_rat
e': 0.35727537666213205, 'subsample': 0.5, 'min_samples_leaf': 7, 'max_depth': 2}. Best is trial 1 with value: 0.852327810556846
8.
[I 2024-07-16 15:07:39,656] Trial 10 finished with value: 0.8123143761050521 and parameters: {'n_estimators': 431, 'learning_rat
e': 0.07516558590021982, 'subsample': 0.1, 'min_samples_leaf': 14, 'max_depth': 5}. Best is trial 1 with value: 0.85232781055684
68.
[I 2024-07-16 15:07:40,317] Trial 11 finished with value: 0.8663892862109382 and parameters: {'n_estimators': 335, 'learning_rat
e': 0.02535209436965534, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 11 with value: 0.8663892862109
382.
[I 2024-07-16 15:07:40,923] Trial 12 finished with value: 0.8652730595784414 and parameters: {'n_estimators': 332, 'learning_rat
e': 0.045416548189273875, 'subsample': 0.5, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 11 with value: 0.866389286210
9382.
[I 2024-07-16 15:07:41,428] Trial 13 finished with value: 0.8505237201263484 and parameters: {'n_estimators': 332, 'learning_rat
e': 0.1851489786137997, 'subsample': 0.5, 'min_samples_leaf': 15, 'max_depth': 3}. Best is trial 11 with value: 0.86638928621093
82.
[I 2024-07-16 15:07:41,935] Trial 14 finished with value: 0.8763595863877466 and parameters: {'n_estimators': 355, 'learning_rat
e': 0.021816346912467165, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 14 with valu
e: 0.8763595863877466.
```

```
[I 2024-07-16 15:07:42,421] Trial 15 finished with value: 0.8476791425790173 and parameters: {'n_estimators': 375, 'learning_rat
e': 0.22277002666024498, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:43,179] Trial 16 finished with value: 0.8519987782346982 and parameters: {'n_estimators': 495, 'learning_rat
e': 0.20411080661813408, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 4}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:43,506] Trial 17 finished with value: 0.8585881160975027 and parameters: {'n_estimators': 295, 'learning_rat
e': 0.015074320275986803, 'subsample': 0.2, 'min_samples_leaf': 15, 'max_depth': 3}. Best is trial 14 with value: 0.876359586387
7466.
[I 2024-07-16 15:07:43,807] Trial 18 finished with value: 0.8313144406698849 and parameters: {'n_estimators': 163, 'learning_rat
e': 0.3809225663416619, 'subsample': 0.4, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 14 with value: 0.87635958638774
66.
[I 2024-07-16 15:07:44,756] Trial 19 finished with value: 0.8370768520174027 and parameters: {'n_estimators': 369, 'learning_rat
e': 0.13984409624527872, 'subsample': 1.0, 'min_samples_leaf': 11, 'max_depth': 4}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:45,045] Trial 20 finished with value: 0.8012862059717504 and parameters: {'n_estimators': 278, 'learning_rat
e': 0.26193124840316495, 'subsample': 0.1, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:45,639] Trial 21 finished with value: 0.8689147644873553 and parameters: {'n_estimators': 341, 'learning_rat
e': 0.03800407932755415, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:46,406] Trial 22 finished with value: 0.8706455986649978 and parameters: {'n_estimators': 371, 'learning_rat
e': 0.01799756605037503, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:46,967] Trial 23 finished with value: 0.8541778413493057 and parameters: {'n_estimators': 376, 'learning_rat
e': 0.129017241280269, 'subsample': 0.4, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 14 with value: 0.876359586387746
6.
[I 2024-07-16 15:07:47,854] Trial 24 finished with value: 0.8510563799988081 and parameters: {'n_estimators': 401, 'learning_rat
e': 0.12484987668163255, 'subsample': 0.7000000000000001, 'min_samples_leaf': 14, 'max_depth': 4}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:48,590] Trial 25 finished with value: 0.8713831277191728 and parameters: {'n_estimators': 465, 'learning_rat
e': 0.019298046812405465, 'subsample': 0.4, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 14 with value: 0.876359586387
7466.
[I 2024-07-16 15:07:49,418] Trial 26 finished with value: 0.8476431352682917 and parameters: {'n_estimators': 484, 'learning_rat
e': 0.30131109981496706, 'subsample': 0.4, 'min_samples_leaf': 11, 'max_depth': 4}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:50,155] Trial 27 finished with value: 0.8471750402288576 and parameters: {'n_estimators': 450, 'learning_rat
e': 0.16977455694652935, 'subsample': 0.2, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:50,861] Trial 28 finished with value: 0.5280081004032819 and parameters: {'n_estimators': 462, 'learning_rat
e': 0.48000746712086784, 'subsample': 0.2, 'min_samples_leaf': 9, 'max_depth': 2}. Best is trial 14 with value: 0.87635958638774
66.
[I 2024-07-16 15:07:51,547] Trial 29 finished with value: 0.8372978624073744 and parameters: {'n_estimators': 401, 'learning_rat
e': 0.6151517886097901, 'subsample': 0.4, 'min_samples_leaf': 12, 'max_depth': 2}. Best is trial 14 with value: 0.87635958638774
```

```
66.
[I 2024-07-16 15:07:52,342] Trial 30 finished with value: 0.8479926545086119 and parameters: {'n_estimators': 307, 'learning_rat
e': 0.989048112671089, 'subsample': 0.7000000000000001, 'min_samples_leaf': 10, 'max_depth': 3}. Best is trial 14 with value: 0.
8763595863877466.
[I 2024-07-16 15:07:52,974] Trial 31 finished with value: 0.8730779545860897 and parameters: {'n_estimators': 253, 'learning_rat
e': 0.015742146167218172, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 14 with value: 0.876359586387
7466.
[I 2024-07-16 15:07:53,554] Trial 32 finished with value: 0.8586042573057592 and parameters: {'n_estimators': 257, 'learning_rat
e': 0.09679211998178447, 'subsample': 0.5, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:54,253] Trial 33 finished with value: 0.8677178318135764 and parameters: {'n_estimators': 251, 'learning_rat
e': 0.012303499949170031, 'subsample': 0.7000000000000001, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:54,772] Trial 34 finished with value: 0.870319049605658 and parameters: {'n_estimators': 142, 'learning_rat
e': 0.10046929145246403, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:55,576] Trial 35 finished with value: 0.856853557025647 and parameters: {'n_estimators': 215, 'learning_rat
e': 0.09400408524788516, 'subsample': 0.9, 'min_samples_leaf': 10, 'max_depth': 3}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:56,789] Trial 36 finished with value: 0.8486339571289508 and parameters: {'n_estimators': 452, 'learning_rat
e': 0.2379833141259567, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 4}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:57,654] Trial 37 finished with value: 0.8298058585136182 and parameters: {'n_estimators': 356, 'learning_rat
e': 0.6910921058489367, 'subsample': 0.6, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 14 with value: 0.87635958638774
66.
[I 2024-07-16 15:07:58,397] Trial 38 finished with value: 0.8657250134096192 and parameters: {'n_estimators': 401, 'learning_rat
e': 0.07714880926276627, 'subsample': 0.4, 'min_samples_leaf': 15, 'max_depth': 4}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:58,571] Trial 39 finished with value: 0.8622546536345034 and parameters: {'n_estimators': 89, 'learning_rat
e': 0.17140234296581142, 'subsample': 0.5, 'min_samples_leaf': 10, 'max_depth': 2}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:59,133] Trial 40 finished with value: 0.8412959155293324 and parameters: {'n_estimators': 311, 'learning_rat
e': 0.28079732645731614, 'subsample': 0.7000000000000001, 'min_samples_leaf': 12, 'max_depth': 2}. Best is trial 14 with value:
0.8763595863877466.
[I 2024-07-16 15:07:59,416] Trial 41 finished with value: 0.87503724894213 and parameters: {'n_estimators': 146, 'learning_rat
e': 0.05871598562461487, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 14 with value: 0.8763595863877
466.
[I 2024-07-16 15:07:59,645] Trial 42 finished with value: 0.8773677910880663 and parameters: {'n_estimators': 114, 'learning_rat
e': 0.0661938683771798, 'subsample': 0.7000000000000001, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 42 with value:
0.8773677910880663.
[I 2024-07-16 15:07:59,825] Trial 43 finished with value: 0.8633696386355961 and parameters: {'n_estimators': 62, 'learning_rat
e': 0.06866852442161842, 'subsample': 0.9, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 42 with value: 0.8773677910880
663.
[I 2024-07-16 15:08:00,069] Trial 44 finished with value: 0.8653699068279795 and parameters: {'n_estimators': 126, 'learning_rat
```

```
e': 0.15754708240317453, 'subsample': 0.7000000000000001, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 42 with value:
0.8773677910880663.
[I 2024-07-16 15:08:00,309] Trial 45 finished with value: 0.8705077775791167 and parameters: {'n_estimators': 111, 'learning_rat
e': 0.06502957300205651, 'subsample': 0.8, 'min_samples_leaf': 12, 'max_depth': 2}. Best is trial 42 with value: 0.8773677910880
663.
[I 2024-07-16 15:08:00,611] Trial 46 finished with value: 0.828533186324175 and parameters: {'n_estimators': 175, 'learning_rat
e': 0.35458142832009404, 'subsample': 0.5, 'min_samples_leaf': 8, 'max_depth': 2}. Best is trial 42 with value: 0.87736779108806
63.
[I 2024-07-16 15:08:00,747] Trial 47 finished with value: 0.8565667401712459 and parameters: {'n_estimators': 53, 'learning_rat
e': 0.061191030126898624, 'subsample': 0.8, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 42 with value: 0.877367791088
0663.
[I 2024-07-16 15:08:00,837] Trial 48 finished with value: 0.8559893815682301 and parameters: {'n_estimators': 25, 'learning_rat
e': 0.2031553863848652, 'subsample': 0.30000000000000004, 'min_samples_leaf': 11, 'max_depth': 2}. Best is trial 42 with value:
0.8773677910880663.
[I 2024-07-16 15:08:01,205] Trial 49 finished with value: 0.8679785744084868 and parameters: {'n_estimators': 192, 'learning_rat
e': 0.12515411627045162, 'subsample': 0.7000000000000001, 'min_samples_leaf': 5, 'max_depth': 2}. Best is trial 42 with value:
0.8773677910880663.
[I 2024-07-16 15:08:01,764] Trial 50 finished with value: 0.8501164650257266 and parameters: {'n_estimators': 241, 'learning_rat
e': 0.5568303707656983, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 5}. Best is trial 42 with value: 0.87736779108806
63.
[I 2024-07-16 15:08:02,085] Trial 51 finished with value: 0.8622124381667561 and parameters: {'n_estimators': 155, 'learning_rat
e': 0.012289059610775664, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 42 with value: 0.877367791088
0663.
[I 2024-07-16 15:08:02,289] Trial 52 finished with value: 0.8659286409599302 and parameters: {'n_estimators': 91, 'learning_rat
e': 0.039933518855011556, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 42 with value: 0.877367791088
0663.
[I 2024-07-16 15:08:02,678] Trial 53 finished with value: 0.8717022269900867 and parameters: {'n_estimators': 208, 'learning_rat
e': 0.04566665982733785, 'subsample': 0.5, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 42 with value: 0.8773677910880
663.
[I 2024-07-16 15:08:03,048] Trial 54 finished with value: 0.8627438564078114 and parameters: {'n_estimators': 205, 'learning_rat
e': 0.1113358052136771, 'subsample': 0.5, 'min_samples_leaf': 15, 'max_depth': 3}. Best is trial 42 with value: 0.87736779108806
63.
[I 2024-07-16 15:08:03,394] Trial 55 finished with value: 0.8666388541232095 and parameters: {'n_estimators': 220, 'learning_rat
e': 0.05367198735367938, 'subsample': 0.4, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 42 with value: 0.8773677910880
663.
[I 2024-07-16 15:08:03,632] Trial 56 finished with value: 0.8679922323539344 and parameters: {'n_estimators': 131, 'learning_rat
e': 0.1479352882565595, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 42 with value:
0.8773677910880663.
[I 2024-07-16 15:08:04,197] Trial 57 finished with value: 0.8560713292409163 and parameters: {'n_estimators': 270, 'learning_rat
e': 0.08320556951541808, 'subsample': 0.5, 'min_samples_leaf': 15, 'max_depth': 4}. Best is trial 42 with value: 0.8773677910880
663.
[I 2024-07-16 15:08:04,530] Trial 58 finished with value: 0.8626345928442299 and parameters: {'n_estimators': 182, 'learning_rat
e': 0.19944226330920875, 'subsample': 0.4, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 42 with value: 0.8773677910880
663.
```

```
[I 2024-07-16 15:08:04,884] Trial 59 finished with value: 0.8827962035878181 and parameters: {'n_estimators': 234, 'learning_rat
e': 0.04028597459008615, 'subsample': 0.2, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:05,201] Trial 60 finished with value: 0.8373475276635478 and parameters: {'n_estimators': 232, 'learning_rat
e': 0.23986742818774218, 'subsample': 0.2, 'min_samples_leaf': 6, 'max_depth': 2}. Best is trial 59 with value: 0.88279620358781
81.
[I 2024-07-16 15:08:05,528] Trial 61 finished with value: 0.8705984166716331 and parameters: {'n_estimators': 159, 'learning_rat
e': 0.0485523589631083, 'subsample': 0.2, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value: 0.88279620358781
81.
[I 2024-07-16 15:08:05,868] Trial 62 finished with value: 0.8190198065041621 and parameters: {'n_estimators': 265, 'learning_rat
e': 0.01149006689165348, 'subsample': 0.1, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:06,430] Trial 63 finished with value: 0.8659050499632478 and parameters: {'n_estimators': 294, 'learning_rat
e': 0.1204959305335269, 'subsample': 0.30000000000000004, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:06,959] Trial 64 finished with value: 0.8690550688360449 and parameters: {'n_estimators': 199, 'learning_rat
e': 0.04103033604106692, 'subsample': 0.5, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:07,572] Trial 65 finished with value: 0.8552108786777123 and parameters: {'n_estimators': 245, 'learning_rat
e': 0.08328678190029276, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:08,230] Trial 66 finished with value: 0.8505448278602222 and parameters: {'n_estimators': 317, 'learning_rat
e': 0.14981748638611425, 'subsample': 0.4, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:08,490] Trial 67 finished with value: 0.8664526094125593 and parameters: {'n_estimators': 117, 'learning_rat
e': 0.038928927532914825, 'subsample': 0.2, 'min_samples_leaf': 11, 'max_depth': 4}. Best is trial 59 with value: 0.882796203587
8181.
[I 2024-07-16 15:08:08,990] Trial 68 finished with value: 0.8599042453860978 and parameters: {'n_estimators': 282, 'learning_rat
e': 0.11213902256865343, 'subsample': 0.30000000000000004, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:09,411] Trial 69 finished with value: 0.8688328168146692 and parameters: {'n_estimators': 145, 'learning_rat
e': 0.08032956606041357, 'subsample': 0.5, 'min_samples_leaf': 14, 'max_depth': 4}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:09,976] Trial 70 finished with value: 0.8190129775314382 and parameters: {'n_estimators': 475, 'learning_rat
e': 0.1829440605171604, 'subsample': 0.1, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 59 with value: 0.88279620358781
81.
[I 2024-07-16 15:08:10,742] Trial 71 finished with value: 0.8698919284025667 and parameters: {'n_estimators': 351, 'learning_rat
e': 0.013662031979636296, 'subsample': 0.7000000000000001, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:11,125] Trial 72 finished with value: 0.8288982259570495 and parameters: {'n_estimators': 169, 'learning_rat
e': 0.8140838514644927, 'subsample': 0.6, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value: 0.88279620358781
81.
[I 2024-07-16 15:08:11,880] Trial 73 finished with value: 0.8631747025051154 and parameters: {'n_estimators': 441, 'learning_rat
e': 0.04725803255615338, 'subsample': 0.6, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
```

```
181.
[I 2024-07-16 15:08:12,670] Trial 74 finished with value: 0.8484315712100443 and parameters: {'n_estimators': 431, 'learning_rat
e': 0.09944114450987829, 'subsample': 0.6, 'min_samples_leaf': 12, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:13,529] Trial 75 finished with value: 0.8694014839978546 and parameters: {'n_estimators': 418, 'learning_rat
e': 0.010733692382577866, 'subsample': 0.7000000000000001, 'min_samples_leaf': 14, 'max_depth': 3}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:14,368] Trial 76 finished with value: 0.8451164153604704 and parameters: {'n_estimators': 389, 'learning_rat
e': 0.13627269798935968, 'subsample': 0.7000000000000001, 'min_samples_leaf': 11, 'max_depth': 3}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:14,789] Trial 77 finished with value: 0.825250312891114 and parameters: {'n_estimators': 213, 'learning_rat
e': 0.43403020970150585, 'subsample': 0.4, 'min_samples_leaf': 13, 'max_depth': 3}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:15,160] Trial 78 finished with value: 0.8683647217752348 and parameters: {'n_estimators': 228, 'learning_rat
e': 0.06646183187389511, 'subsample': 0.5, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:15,600] Trial 79 finished with value: 0.872411198521962 and parameters: {'n_estimators': 323, 'learning_rat
e': 0.03587443515905809, 'subsample': 0.30000000000000004, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:16,010] Trial 80 finished with value: 0.871587996900888 and parameters: {'n_estimators': 327, 'learning_rat
e': 0.03619900042907129, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:16,462] Trial 81 finished with value: 0.8751614120825635 and parameters: {'n_estimators': 326, 'learning_rat
e': 0.04085211361289099, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:16,842] Trial 82 finished with value: 0.8453548085901027 and parameters: {'n_estimators': 324, 'learning_rat
e': 0.0966172474904747, 'subsample': 0.2, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value: 0.88279620358781
81.
[I 2024-07-16 15:08:17,399] Trial 83 finished with value: 0.8745654290084829 and parameters: {'n_estimators': 350, 'learning_rat
e': 0.04248340810099158, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:17,813] Trial 84 finished with value: 0.8713595367224904 and parameters: {'n_estimators': 347, 'learning_rat
e': 0.06172203306917122, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:18,197] Trial 85 finished with value: 0.869062518624471 and parameters: {'n_estimators': 300, 'learning_rat
e': 0.035915343532801486, 'subsample': 0.2, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value: 0.882796203587
8181.
[I 2024-07-16 15:08:18,641] Trial 86 finished with value: 0.8534514869777698 and parameters: {'n_estimators': 362, 'learning_rat
e': 0.12322525500214208, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:19,143] Trial 87 finished with value: 0.8613234300812525 and parameters: {'n_estimators': 381, 'learning_rat
e': 0.07903691394468533, 'subsample': 0.2, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:19,507] Trial 88 finished with value: 0.8043611061445856 and parameters: {'n_estimators': 341, 'learning_rat
```

```
e': 0.15827026055603516, 'subsample': 0.1, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:19,913] Trial 89 finished with value: 0.8816601108528518 and parameters: {'n_estimators': 261, 'learning_rat
e': 0.05606464804575051, 'subsample': 0.30000000000000004, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:19,997] Trial 90 finished with value: 0.8520372688082324 and parameters: {'n_estimators': 11, 'learning_rat
e': 0.6959515354157059, 'subsample': 0.30000000000000004, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:20,348] Trial 91 finished with value: 0.8641692492599877 and parameters: {'n_estimators': 281, 'learning_rat
e': 0.05681249211853894, 'subsample': 0.2, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:20,501] Trial 92 finished with value: 0.8603710987941277 and parameters: {'n_estimators': 76, 'learning_rat
e': 0.03108524415467831, 'subsample': 0.30000000000000004, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:20,823] Trial 93 finished with value: 0.8666537537000615 and parameters: {'n_estimators': 256, 'learning_rat
e': 0.1094027625230779, 'subsample': 0.30000000000000004, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value:
0.8827962035878181.
[I 2024-07-16 15:08:21,283] Trial 94 finished with value: 0.8622819695253988 and parameters: {'n_estimators': 307, 'learning_rat
e': 0.08463735721691222, 'subsample': 0.4, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:21,875] Trial 95 finished with value: 0.8719617279535927 and parameters: {'n_estimators': 268, 'learning_rat
e': 0.058664490163403034, 'subsample': 0.8, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value: 0.882796203587
8181.
[I 2024-07-16 15:08:22,483] Trial 96 finished with value: 0.8681561276993066 and parameters: {'n_estimators': 270, 'learning_rat
e': 0.06310432084284466, 'subsample': 0.9, 'min_samples_leaf': 14, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:23,092] Trial 97 finished with value: 0.8617617259669825 and parameters: {'n_estimators': 291, 'learning_rat
e': 0.13807302487984469, 'subsample': 0.8, 'min_samples_leaf': 13, 'max_depth': 2}. Best is trial 59 with value: 0.8827962035878
181.
[I 2024-07-16 15:08:23,864] Trial 98 finished with value: 0.8735994397759104 and parameters: {'n_estimators': 336, 'learning_rat
e': 0.028193862422144944, 'subsample': 0.8, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value: 0.882796203587
8181.
[I 2024-07-16 15:08:24,666] Trial 99 finished with value: 0.8701179053181557 and parameters: {'n_estimators': 361, 'learning_rat
e': 0.025771343920638883, 'subsample': 0.9, 'min_samples_leaf': 15, 'max_depth': 2}. Best is trial 59 with value: 0.882796203587
8181.
{'n_estimators': 234, 'learning_rate': 0.04028597459008615, 'subsample': 0.2, 'min_samples_leaf': 12, 'max_depth': 3}
```

In [150…

```python
# Avaliando o modelo com os melhores parâmetros
modelo_gb_tunado = GradientBoostingClassifier(**best_params, random_state = 42)
modelo_gb_tunado.fit(x_train, y_train)
y_pred_proba = modelo_gb_tunado.predict_proba(x_test)[:, 1]
roc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC: {roc:.4f}")
```

```
ROC: 0.8853
```

In [151…
```
desemp_gb_tunado=calcula_desempenho(modelo_gb_tunado, x_train, y_train, x_test, y_test)

desemp_gb_tunado
```

Out[151]:

	Treino	Teste	Variação
Acurácia	0.905812	0.859813	-0.05
AUROC	0.953426	0.885293	-0.07
KS	0.783525	0.631791	-0.19
Precision	0.880000	0.796296	-0.10
Recall	0.774648	0.693548	-0.10
F1	0.823970	0.741379	-0.10

Comparando os dois modelos tunados

In [153…
```
desemp_ada_tunado
```

Out[153]:

	Treino	Teste	Variação
Acurácia	0.875752	0.817757	-0.07
AUROC	0.950635	0.868209	-0.09
KS	0.756953	0.604202	-0.20
Precision	0.892157	0.734694	-0.18
Recall	0.640845	0.580645	-0.09
F1	0.745902	0.648649	-0.13

In [154…
```
desemp_gb_tunado
```

Out[154]:

	Treino	Teste	Variação
Acurácia	0.905812	0.859813	-0.05
AUROC	0.953426	0.885293	-0.07
KS	0.783525	0.631791	-0.19
Precision	0.880000	0.796296	-0.10
Recall	0.774648	0.693548	-0.10
F1	0.823970	0.741379	-0.10

Após a tunagem dos Hiperparâmetos, o modelo com a **melhor performance** na base de teste, considerando a medida AUROC como referência, foi o modelo **Gradient Boosting**. Ele apresentou uma performance melhor que a do modelo AdaBoost na base de teste e também melhor capacidade de generalização.

In [245…

```
import pickle
```

In [275…

```
filename = 'imdb.pkl'
pickle.dump(modelo_gb_tunado,open(filename,'wb'))
```