


1) EDA

Importando Bibliotecas:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import warnings as wr
6
7 wr.filterwarnings('ignore')
8
```

Carregamento e análise preliminar do dataset:

```
1 df = pd.read_csv('desafio_indicium_imdb.csv')
2
3 # verificando se dataset foi carregado corretamente
4 df.head(5)
```



Unnamed: 0		Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Ratin
0	1	The Godfather	1972	A	175 min	Crime, Drama	9.
1	2	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.
2	3	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.
3	4	12 Angry Men	1957	U	96 min	Crime, Drama	9.
4	5	The Lord of the Rings: The Return of the King	2003	U	201 min	Action, Adventure, Drama	8.

Próximas etapas:

Gerar código com df

 Ver gráficos recomendados

```
1 # Obtendo mais informações do dataset
2
3 print(df.shape)
4 print('\n')
5 print(df.info())
6 print('\n')
7 print(df.columns)
```

→ (999, 16)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  -
 0   Unnamed: 0            999 non-null   int64
 1   Series_Title          999 non-null   object
 2   Released_Year        999 non-null   object
 3   Certificate           898 non-null   object
 4   Runtime              999 non-null   object
 5   Genre                999 non-null   object
 6   IMDB_Rating          999 non-null   float64
 7   Overview             999 non-null   object
 8   Meta_score           842 non-null   float64
 9   Director             999 non-null   object
10   Star1                999 non-null   object
11   Star2                999 non-null   object
12   Star3                999 non-null   object
13   Star4                999 non-null   object
14   No_of_Votes          999 non-null   int64
15   Gross                830 non-null   object
dtypes: float64(2), int64(2), object(12)
memory usage: 125.0+ KB
None
```

```
Index(['Unnamed: 0', 'Series_Title', 'Released_Year', 'Certificate', 'Runtime',
      'Genre', 'IMDB_Rating', 'Overview', 'Meta_score', 'Director', 'Star1',
      'Star2', 'Star3', 'Star4', 'No_of_Votes', 'Gross'],
      dtype='object')
```

Removendo coluna "Unnamed:0" pois aparentemente é uma coluna que não traz nenhuma informação relevante:

```
1 df = df.drop('Unnamed: 0', axis=1)
```

```
1 df.head()
```

	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
0	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
1	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havoc...	84.0	Christopher Nolan	Christian Bale
2	The Godfather: Part II	1974	A	202 min	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Al Pacino
3	12 Angry Men	1957	U	96 min	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarriage...	96.0	Sidney Lumet	Henry Fonda
4	The Lord of the Rings: The Return of the King	2003	U	201 min	Action, Adventure, Drama	8.9	Gandalf and Aragorn lead the World of Men against...	94.0	Peter Jackson	Elijah Wood

Próximas etapas:

Gerar código com df

Ver gráficos recomendados

Obtendo estatísticas (média, desvio, quartis, etc) de colunas com variáveis numéricas:

```
1 df.describe()
```

	IMDB_Rating	Meta_score	No_of_Votes
count	999.000000	842.000000	9.990000e+02
mean	7.947948	77.969121	2.716214e+05
std	0.272290	12.383257	3.209126e+05
min	7.600000	28.000000	2.508800e+04
25%	7.700000	70.000000	5.547150e+04
50%	7.900000	79.000000	1.383560e+05
75%	8.100000	87.000000	3.731675e+05
max	9.200000	100.000000	2.303232e+06

Verificando valores nulos:

```
1 df.isnull().sum()
```

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

```

É verificado que as colunas referentes a faixa etária, média das críticas e faturamento possuem lacunas. Aqui pode ser feita a escolha de remover apenas as colunas caso elas não sejam relevantes para o nosso objetivo, ou remover as linhas onde ocorrem as lacunas. Optei por remover as linhas, pois acredito que todas as colunas serão utilizadas:

```
1 df.dropna(inplace=True)
```

```
1 df.info()
2 df.isnull().sum()
```

```

↳ <class 'pandas.core.frame.DataFrame'>
Index: 713 entries, 0 to 996
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  --
 0   Series_Title    713 non-null   object  
 1   Released_Year   713 non-null   object  
 2   Certificate      713 non-null   object  
 3   Runtime         713 non-null   object  
 4   Genre           713 non-null   object  
 5   IMDB_Rating     713 non-null   float64  
 6   Overview        713 non-null   object  
 7   Meta_score      713 non-null   float64  
 8   Director        713 non-null   object  
 9   Star1           713 non-null   object  
10  Star2           713 non-null   object  
11  Star3           713 non-null   object  
12  Star4           713 non-null   object  
13  No_of_Votes     713 non-null   int64  
14  Gross           713 non-null   object  
dtypes: float64(2), int64(1), object(12)
memory usage: 89.1+ KB
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

```

Por último verificamos a existência de linhas duplicadas:

```
1 df.duplicated().sum()
```

```
↳ 0
```

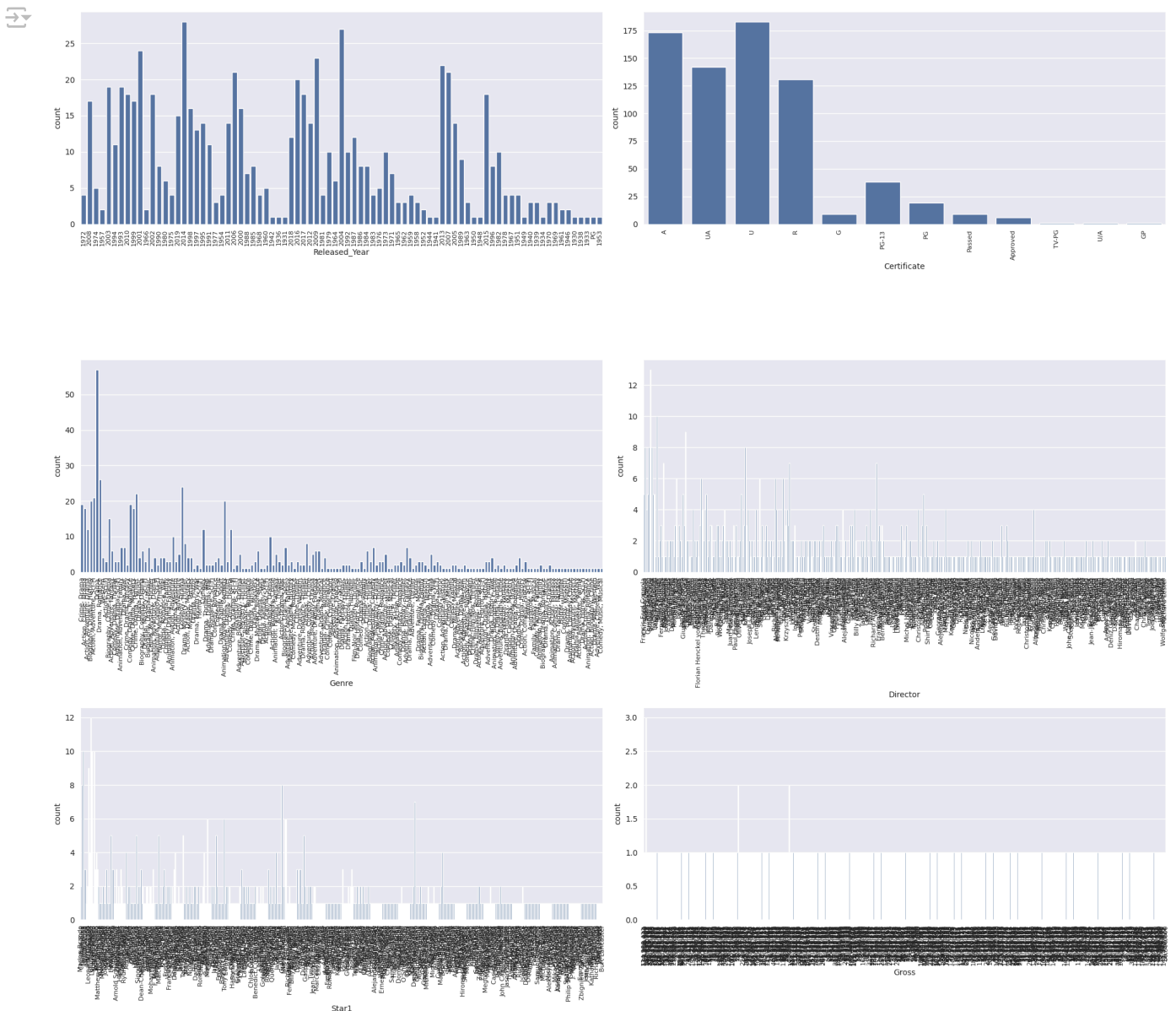
▼ Análise Univariada:

Colunas Categóricas:

```

1
2
3 sns.set_style("darkgrid")
4
5
6 CategoricalColumns = ['Released_Year', 'Certificate', 'Genre', 'Director', 'Star1', 'Gross' ]
7
8
9 plt.figure(figsize=(20, len(CategoricalColumns) * 3))
10 for i in range(len(CategoricalColumns)):
11     plt.subplot(3, 2, i+1)
12     sns.countplot(x=df[CategoricalColumns[i]])
13     plt.xlabel(CategoricalColumns[i])
14     plt.xticks(rotation=90)
15     plt.tick_params(axis='x', which='major', labelsize=8)
16
17
18 plt.tight_layout()
19 plt.show()

```

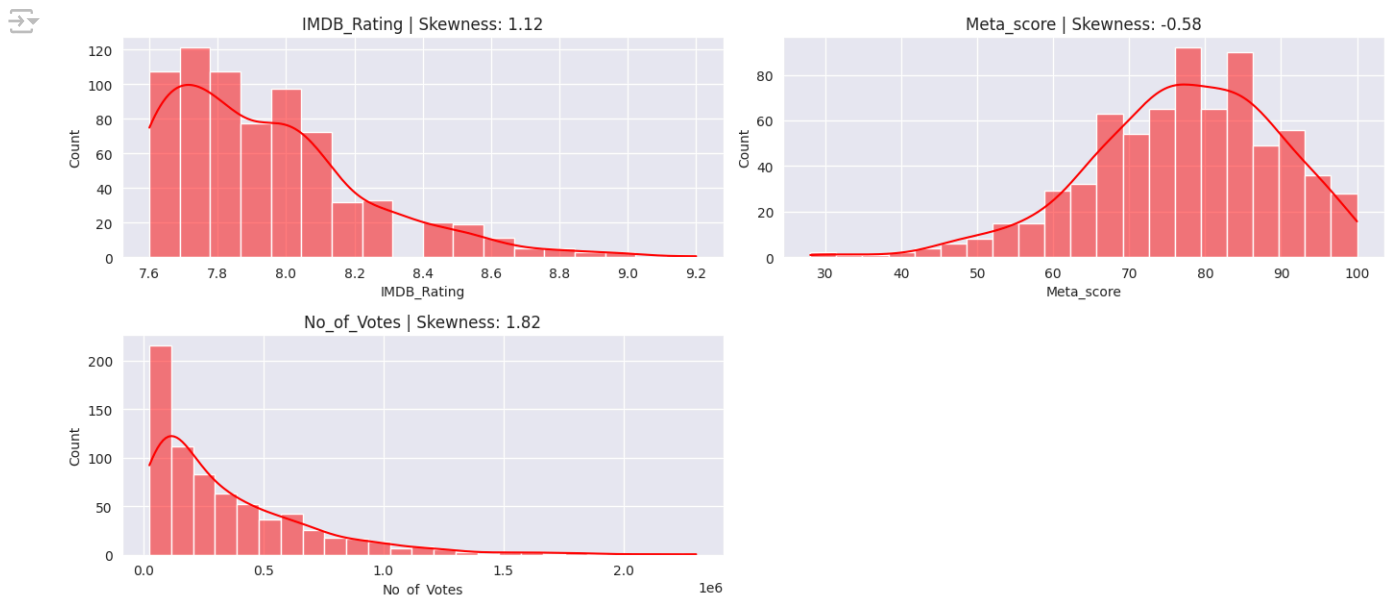


Colunas numéricas:

```

1 sns.set_style("darkgrid")
2
3
4 numericalColumns = df.select_dtypes(include=["int64", "float64"]).columns
5
6
7 plt.figure(figsize=(14, len(numericalColumns) * 3))
8 for idx, feature in enumerate(numericalColumns, 1):
9     plt.subplot(len(numericalColumns), 2, idx)
10    sns.histplot(df[feature], kde=True, color='red')
11    plt.title(f"{feature} | Skewness: {round(df[feature].skew(), 2)}")
12
13
14 plt.tight_layout()
15 plt.show()

```



Conclusões análise univariada:

A análise para as colunas categóricas não foi muito boa devido ao grande número de gêneros, diretores e atores que tornaram os gráficos bastante ilegíveis. No entanto graças a eles foi possível notar a presença de uma linha preenchida incorretamente, pois no ano de lançamento um dos valores apontados pelo gráfico é o "PG", algo que deveria estar na coluna "Certificate". Outra coisa interessante de se analisar é a variedade de gêneros, visto que podem haver combinações (drama, ação, mistério podem se referir a um mesmo filme).

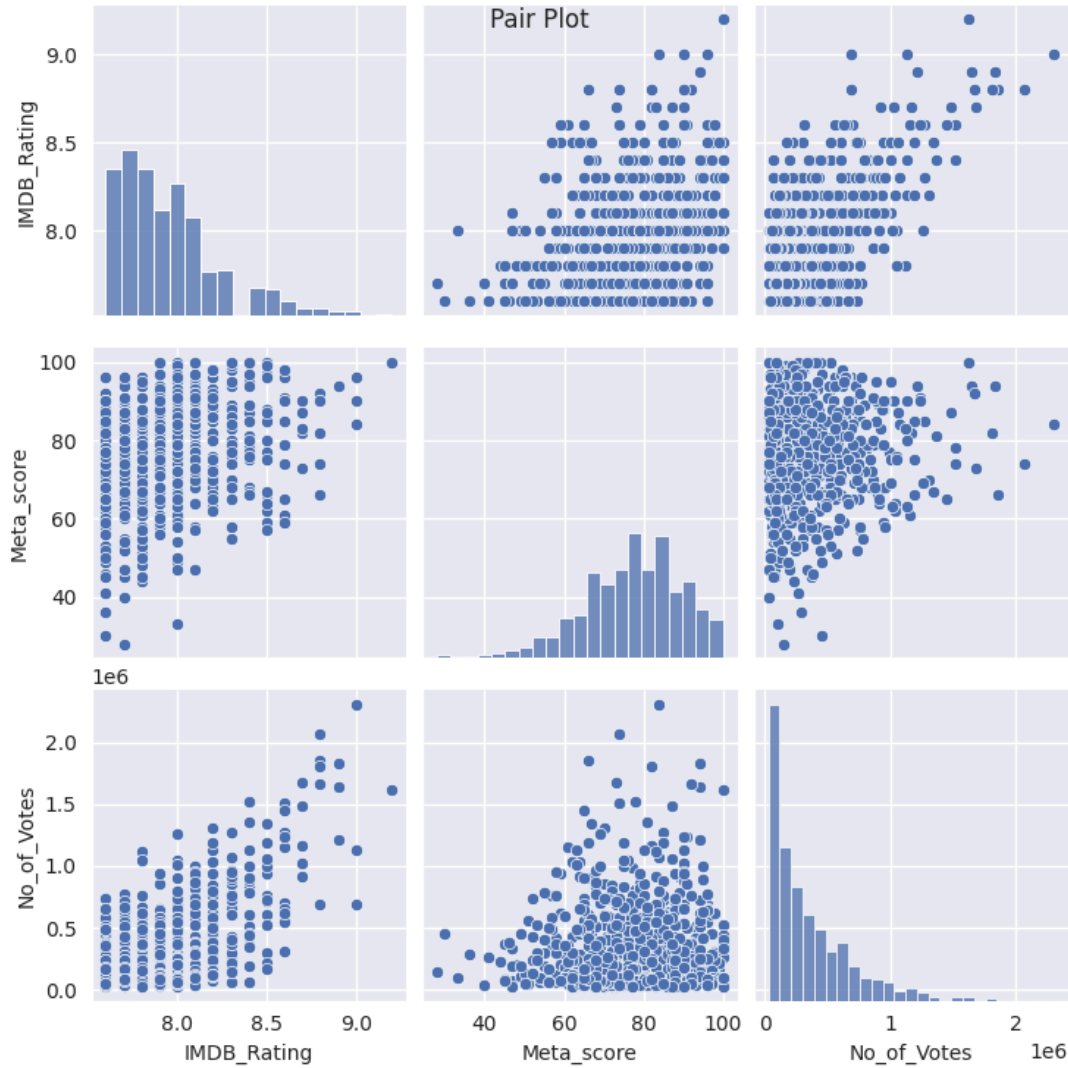
✓ Análise Bivariada:

```

1 sns.set_palette("deep")
2
3 plt.figure(figsize=(10, 6))
4
5 sns.pairplot(df)
6
7 plt.suptitle('Pair Plot')
8 plt.show()

```

 <Figure size 1000x600 with 0 Axes>



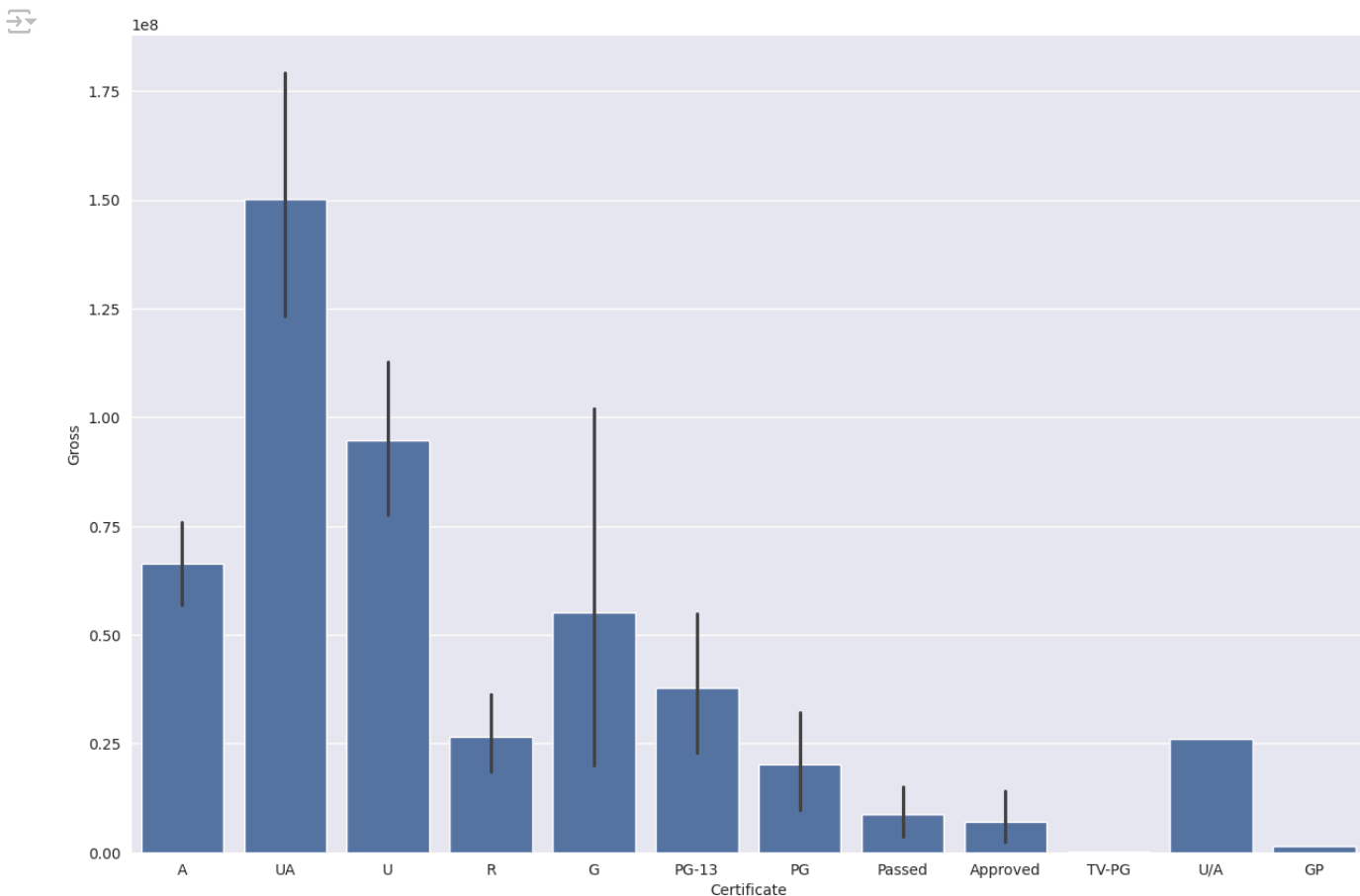
✎ Outros gráficos de interesse:

Achei interessante transformar a coluna do faturamento para uma coluna numérica de modo a obter gráficos que pudessem dar mais *insights*.

```
1 df['Gross'] = df['Gross'].str.replace(',', '').astype(float)
2
```

✎ Faixa etária x Faturamento:

```
1 plt.figure(figsize=(15, 10))
2
3 sns.barplot(data=df, x='Certificate', y='Gross')
4
5 plt.xlabel('Certificate')
6 plt.ylabel('Gross')
7
8 plt.show()
```



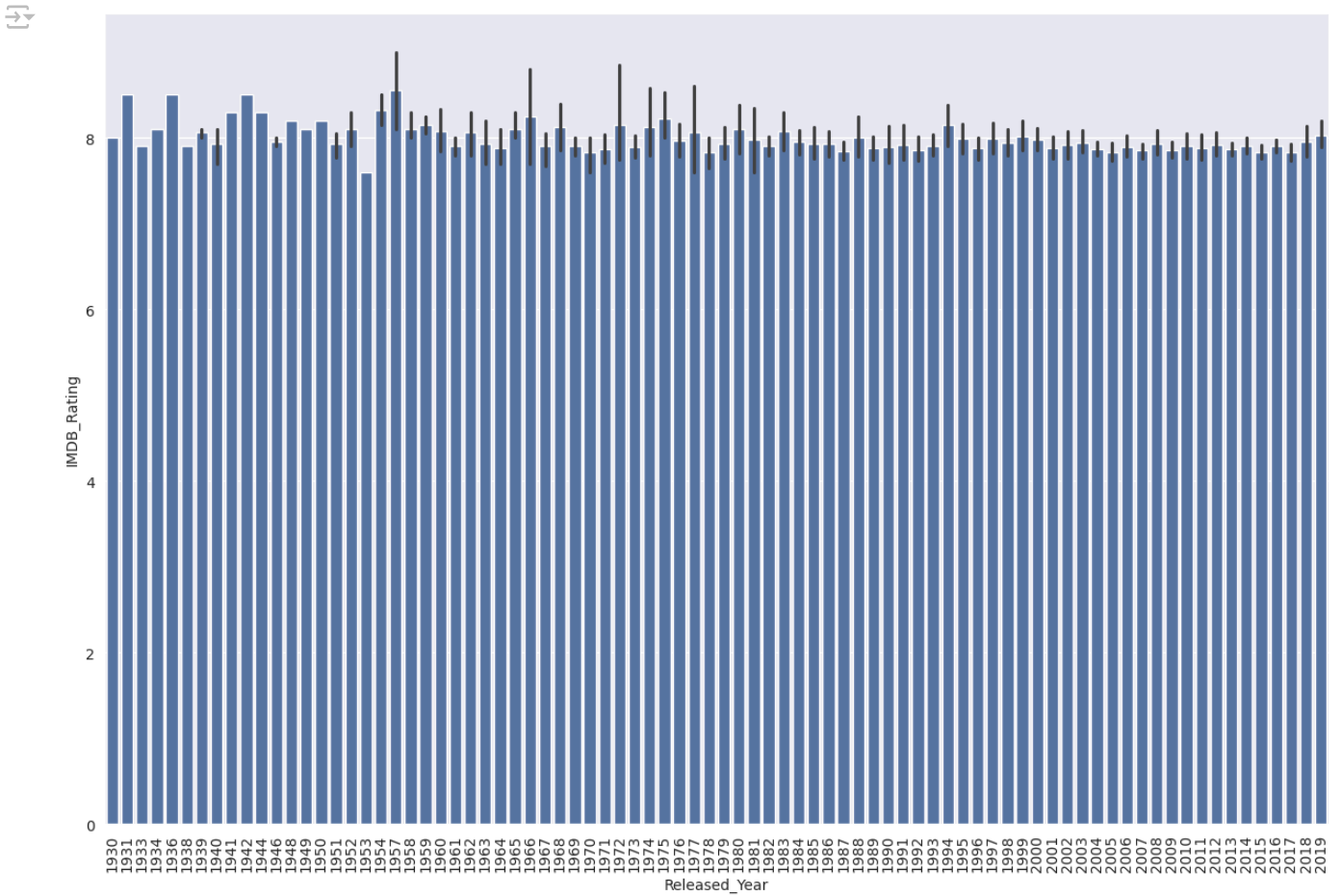
✎ Correção da linha com valor incorreto no ano:

Aqui eu pensei em remover a linha por completo, mas optei por inserir o ano correto após uma breve pesquisa.

```
1 df.Released_Year[df.Released_Year == 'PG'] = '1995'
```

✎ Ano de lançamento x Nota IMDB:

```
1 plt.figure(figsize=(15, 10))
2 yearDf = df.sort_values(by='Released_Year')
3
4 sns.barplot(data=yearDf, x='Released_Year', y='IMDB_Rating')
5
6 plt.xlabel('Released_Year')
7 plt.ylabel('IMDB_Rating')
8
9 plt.xticks(rotation=90)
10
11 plt.show()
```

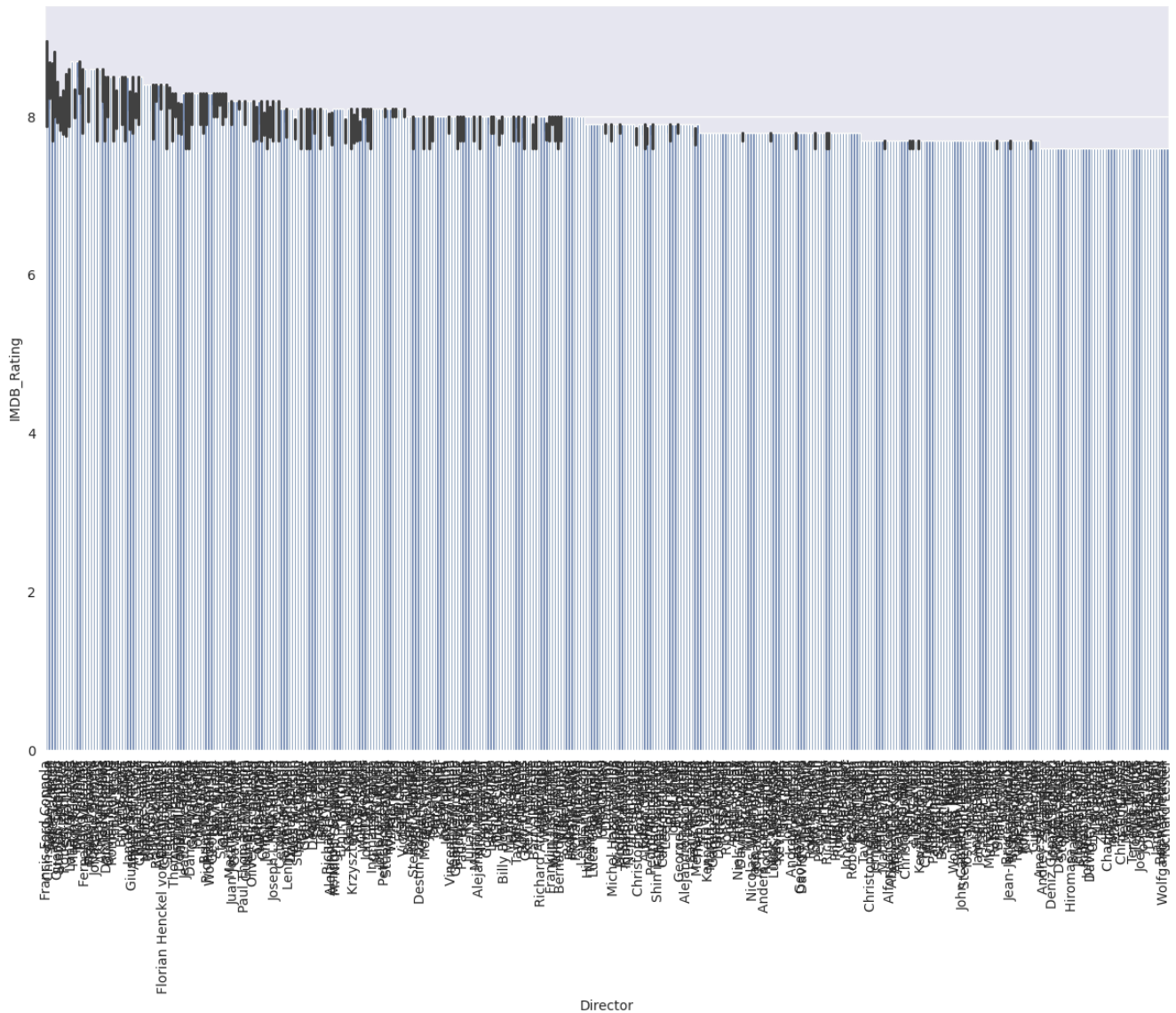



▼ Diretor x Nota IMDB:

```

1 plt.figure(figsize=(15, 10))
2
3 sns.barplot(data=df, x='Director', y='IMDB_Rating')
4
5 plt.xlabel('Director')
6 plt.ylabel('IMDB_Rating')
7
8 plt.xticks(rotation=90)
9
10 plt.show()

```



Conclusões:

Uma das coisas que mais me deu problemas foi que as colunas categóricas possuem muitos valores distintos de modo que os gráficos ficam ilegíveis quando os plotamos. Apesar de não ter implementado, penso que aqui poderia ser realizada uma redução do número de valores diferentes, talvez por meio de agrupamento (exemplo: gêneros poderiam ser reduzidos a apenas um principal ao invés de vários). Agora para os filmes, atores e diretores em si, achei mais complicado de resolver.

2)

- Qual filme você recomendaria para uma pessoa que você não conhece? **Resposta: Um filme de um dos gêneros com média de notas mais altas, e com um alto faturamento**
- Quais são os principais fatores que estão relacionados com alta expectativa de faturamento de um filme? **Resposta: Nota da crítica, faixa etária do filme**
- Quais insights podem ser tirados com a coluna Overview? É possível inferir o gênero do filme a partir dessa coluna? **Resposta: Essa coluna me parece mais oferecer insights pra depois de decidir qual gênero assistir, mais como um critério de desempate. Esse dataset em particular torna as coisas um pouco mais complicadas pois é possível que um filme tenha mais de um gênero, mas se fosse apenas um único gênero seria possível descobrir qual é a partir da coluna Overview.**

3) Explique como você faria a previsão da nota do imdb a partir dos dados. Quais variáveis e/ou suas transformações você utilizou e por quê? Qual tipo de problema estamos resolvendo (regressão, classificação)? Qual modelo melhor se aproxima dos dados e quais seus prós e contras? Qual medida de performance do modelo foi escolhida e por quê?

Resposta: As alterações feitas no dataset foram explicadas durante o processo de EDA. O que precisa ser feito agora é verificar quais colunas vão realmente auxiliar na previsão da nota. No meu entendimento, as colunas Overview, Release_Year, Runtime, e possivelmente Star3 e Star4 podem ser removidas, pois não auxiliarão na previsão. O problema a ser resolvido é um de regressão visto que o que estamos tentando prever é o valor da nota do IMDB que é numérico, e não uma categoria. Para colunas categóricas como Director e Star1 vou transformá-las utilizando um encoding de frequência, pois quero analisar a influência de atores e diretores conhecidos na alteração da nota IMDB

Encoding de Frequência:

```
1 frequencyEncoding = df['Director'].value_counts().to_dict()
2 df['Director_Encoded'] = df['Director'].map(frequencyEncoding)
3
4 frequencyEncoding = df['Star1'].value_counts().to_dict()
5 df['Star1_Encoded'] = df['Star1'].map(frequencyEncoding)
```

Encoding por label:

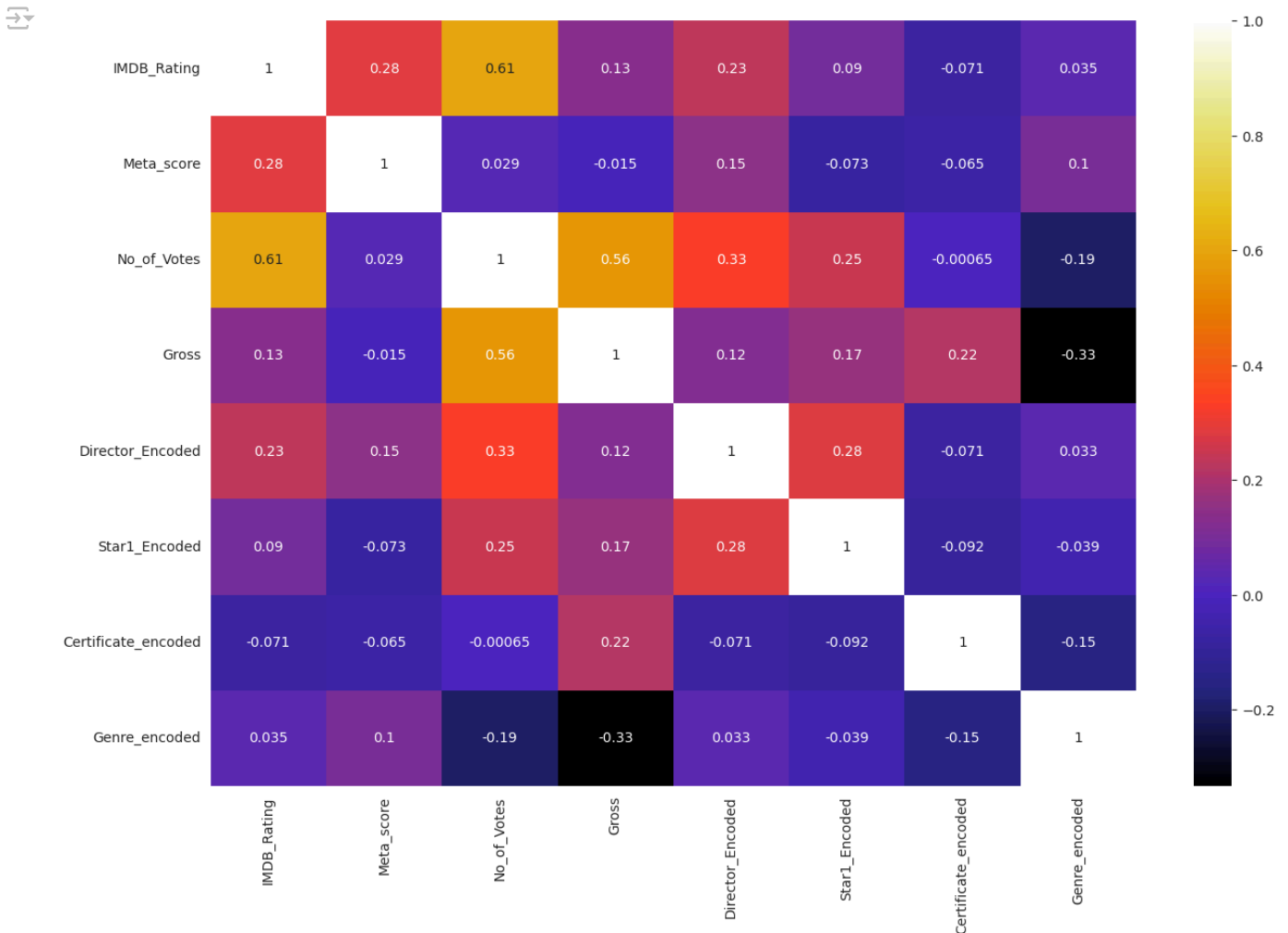
Aqui houve uma tentativa de transformar as colunas de faixa etária e gênero em colunas numéricas.

```
1 from sklearn.preprocessing import LabelEncoder
2
3 label_encoder = LabelEncoder()
4 df['Certificate_encoded'] = label_encoder.fit_transform(df['Certificate'])
5
6 label_encoder = LabelEncoder()
7 df['Genre_encoded'] = label_encoder.fit_transform(df['Genre'])
8
```

Mapa de correlação:

Mas ao plotar o mapa de correlação foi observado que essas novas colunas não tem uma relação tão boa com a coluna de nota do IMDB, portanto ao treinar a máquina essas colunas não serão utilizadas. Devemos lembrar que o label encoder pode adicionar uma certa ordinalidade à colunas onde isso não havia antes, o que pode acarretar em erros da análise de correlação.

```
1 numeric_df = df.select_dtypes(include=['number'])
2 plt.figure(figsize=(15, 10))
3 sns.heatmap(numeric_df.corr(), cmap='CMRmap', annot=True)
4 plt.show()
```



Modelo de Regressão Linear Multivariada:

```

1 #Importando Bibliotecas:
2
3 from sklearn.model_selection import train_test_split
4 from sklearn.linear_model import LinearRegression
5 from sklearn import metrics

1 model = LinearRegression()

1 y = df['IMDB_Rating']
2 X = df[['No_of_Votes', 'Gross', 'Meta_score', 'Director_Encoded', 'Star1_Encoded']]

1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

1 X_train.shape
(499, 5)

1 X_test.shape
(214, 5)

1 model.fit(X_train, y_train)

```

LinearRegression
LinearRegression()

✓ Análise de parâmetros do modelo:

```
1 print("R quadrado = {}".format(model.score(X_train,y_train)))
2 print('Intercept do Modelo :', model.intercept_)
3 print('Coeficientes do Modelo : ', model.coef_)
```

```
R quadrado = 0.5007712955304284
Intercept do Modelo : 7.318078454348875
Coeficientes do Modelo : [ 6.64584016e-07 -8.07589879e-10  5.88899155e-03 -3.02883600e-03
-1.35845121e-03]
```

```
1 lm = model.predict(X_test)
```

```
1 print("R quadrado de teste = {}".format(metrics.r2_score(y_test,lm).round(2)))
```

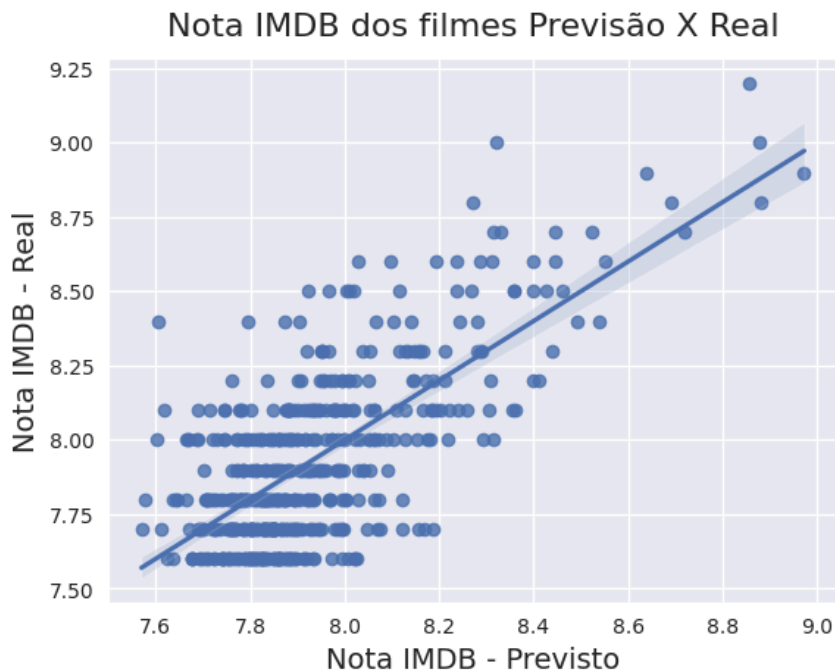
```
R quadrado de teste = 0.51
```

Podemos ver a partir destes parâmetros que o modelo ficou bastante aquém do esperado. Isso pode ter acontecido por conta das colunas que eu selecionei como sendo importantes para a decisão da nota, ou pelo próprio encoding que pode ter afetado de alguma maneira inesperada, ou pelo número reduzido de linhas visto que as que possuíam valores nulos foram removidas.

```
1 y_predict_train = model.predict(X_train)
```

```
1 ax = sns.regplot(x = y_predict_train,y=y_train)
2 ax.set_title('Nota IMDB dos filmes Previsão X Real', fontsize=16, y=1.02)
3 ax.set_xlabel("Nota IMDB - Previsto", fontsize=14)
4 ax.set_ylabel("Nota IMDB - Real", fontsize=14)
5 ax
```

```
<Axes: title={'center': 'Nota IMDB dos filmes Previsão X Real'}, xlabel='Nota IMDB - Previsto', ylabel='Nota IMDB - Real'>
```



✓ Salvando o modelo no formato.pkl:

```
1 import pickle
```

```
#####
```

```
4         pickle.dump(model, f)
```