

## ✓ Desafio Cientista de Dados

### Análise exploratória de dados e Machine Learning

Relatório feito por: Ian Périgo

[LinkedIn](#) | [GitHub](#) | [WhatsApp](#)

[Repositório do código](#)

#### Objetivo:

Fazer uma análise de um banco de dados cinematográfico para orientar qual tipo de filme deverá ser o próximo a ser produzido pelo estúdio.

#### Entregas:

1- Análise exploratória dos dados, demonstrar principais características e apresentar algumas hipóteses

2- Reponder as seguintes perguntas:

A- Qual filme seria recomendado para uma pessoa que não conheço

B- Quais são os principais fatores que estão relacionados com alta expectativa de faturamento de um filme ?

C- Quais insights podem ser tirados com a coluna Overview? é possível inferir o gênero do filme a partir dela?

3- Fazer a previsão da nota do IMDB a partir dos dados

A- Quais variáveis e transformações foram utilizadas

B- Qual tipo de problema está sendo resolvido

C- Qual modelo melhor se aproximou dos dados, prós e contras

D- Qual medida de performance foi escolhida

4- Com o modelo de Machine Learning treinado, avaliar qual será a nota do IMDB para um filme com essas características:

```
{'Series_Title': 'The Shawshank Redemption', 'Released_Year': '1994', 'Certificate': 'A', 'Runtime': '142 min', 'Genre': 'Drama', 'Overview': 'Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.', 'Meta_score': 80.0, 'Director': 'Frank Darabont', 'Star1': 'Tim Robbins', 'Star2': 'Morgan Freeman', 'Star3': 'Bob Gunton', 'Star4': 'William Sadler', 'No_of_Votes': 2343110, 'Gross': '28,341,469'}
```

#### Instruções de entrega

1- Salvar o modelo serializado em .pkl

2- Entrega através de um repositório público com os seguintes arquivos:

A- README explicando a instalação e execução do projeto

B- Arquivos de requisitos, pacotes e bibliotecas utilizados e as respectivas versões

C- Relatório das análises estatísticas e exploratória em PDF, Jupyter Notebook.

D- Códigos utilizados

E- Arquivo .pkl do modelo

```
df = pd.read_csv('./drive/MyDrive/imdb.csv')
df.info()
```

```
<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
```

df.head()

	Unnamed: 0	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
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
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
2	3	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	4	12 Angry Men	1957	U	96 min	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	Henry Fonda
4	5	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 agai...	94.0	Peter Jackson	Elijah Wood

Próximas etapas: [Gerar código com df](#) [Ver gráficos recomendados](#)

Primeiras observações:

Nosso dataset contém mais features tipo object.

Há alguns valores nulos em Gross, Meta\_score e Certificate.

As features Gross, Runtime e Released\_Year são objects porém podem ser convertidas para int e float com mais facilidade.

As demais features precisarão de outras abordagens.

Mostrar código

```
[ '1972' '2008' '1974' '1957' '2003' '1994' '1993' '2010' '1999' '2001'
  '1966' '2002' '1990' '1980' '1975' '2020' '2019' '2014' '1998' '1997'
  '1995' '1991' '1977' '1962' '1954' '1946' '2011' '2006' '2000' '1988'
  '1985' '1968' '1960' '1942' '1936' '1931' '2018' '2017' '2016' '2012'
  '2009' '2007' '1984' '1981' '1979' '1971' '1963' '1964' '1950' '1940'
  '2013' '2005' '2004' '1992' '1987' '1986' '1983' '1976' '1973' '1965'
  '1959' '1958' '1952' '1948' '1944' '1941' '1927' '1921' '2015' '1996'
  '1989' '1978' '1961' '1955' '1953' '1925' '1924' '1982' '1967' '1951'
  '1949' '1939' '1937' '1934' '1928' '1926' '1920' '1970' '1969' '1956'
  '1947' '1945' '1930' '1938' '1935' '1933' '1932' '1922' '1943' 'PG' ]
```

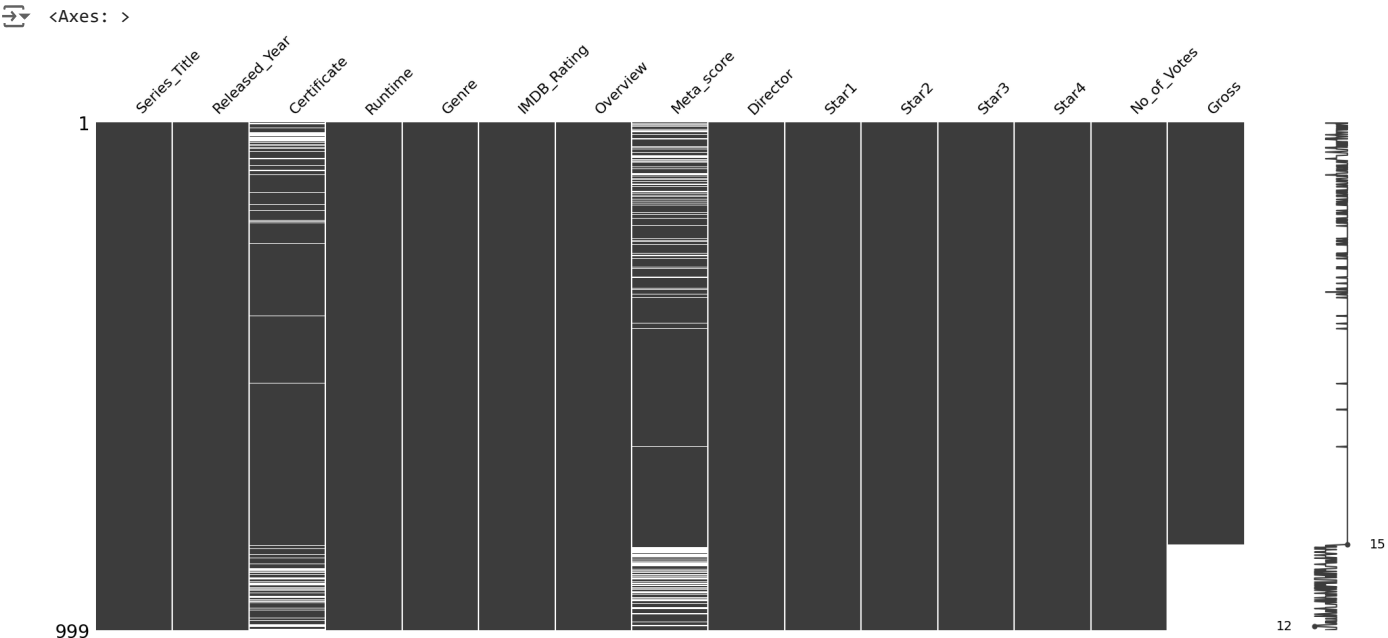
#Vamos analisar as porcentagens de valores nulos

```
df_null_percent = df.isnull().sum() / len(df) * 100
df_null_percent
```

```
Series_Title      0.000000
Released_Year     0.000000
Certificate       10.110110
Runtime           0.000000
Genre             0.000000
IMDB_Rating       0.000000
```

Overview 0.000000  
Meta\_score 15.715716  
Director 0.000000  
Star1 0.000000  
Star2 0.000000  
Star3 0.000000  
Star4 0.000000  
No\_of\_Votes 0.000000  
Gross 16.916917  
dtype: float64

Mostrar código



Mostrar código

Número de linhas com valores faltantes em Gross, Meta\_score e Certificate: 286  
Porcentagem de linhas com valores faltantes em Gross, Meta\_score e Certificate: 28.628628628628626

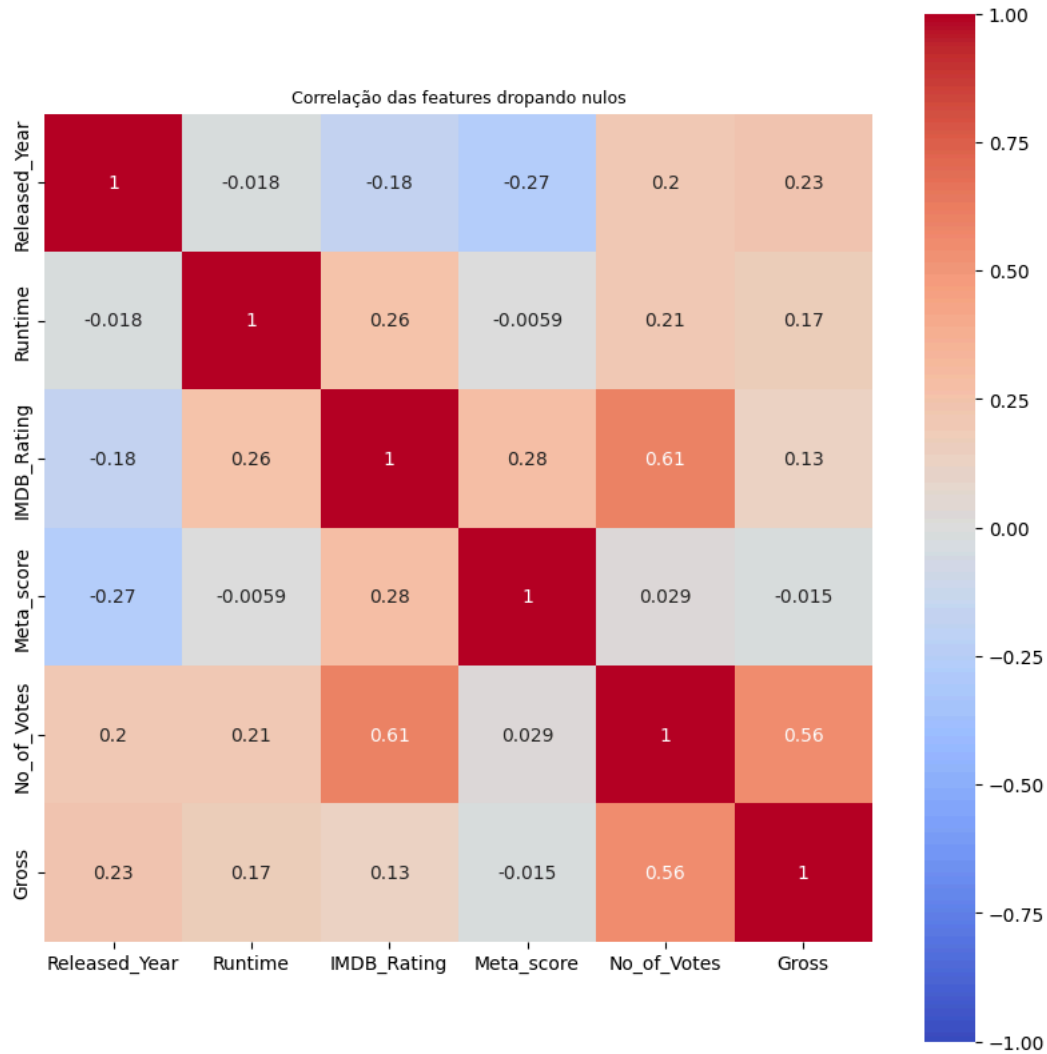
Notamos uma forte correlação dos dados faltantes na feature Gross, certificate e Meta\_score

Temos 28% de valores de ao menos um valor faltante. Vamos verificar qual será a melhor abordagem, dropar ou fazer input data.


Vamos dividir o data set em numéricos e categoricos para começar algumas manipulações dos dados para ter fazer uma análise exploratória

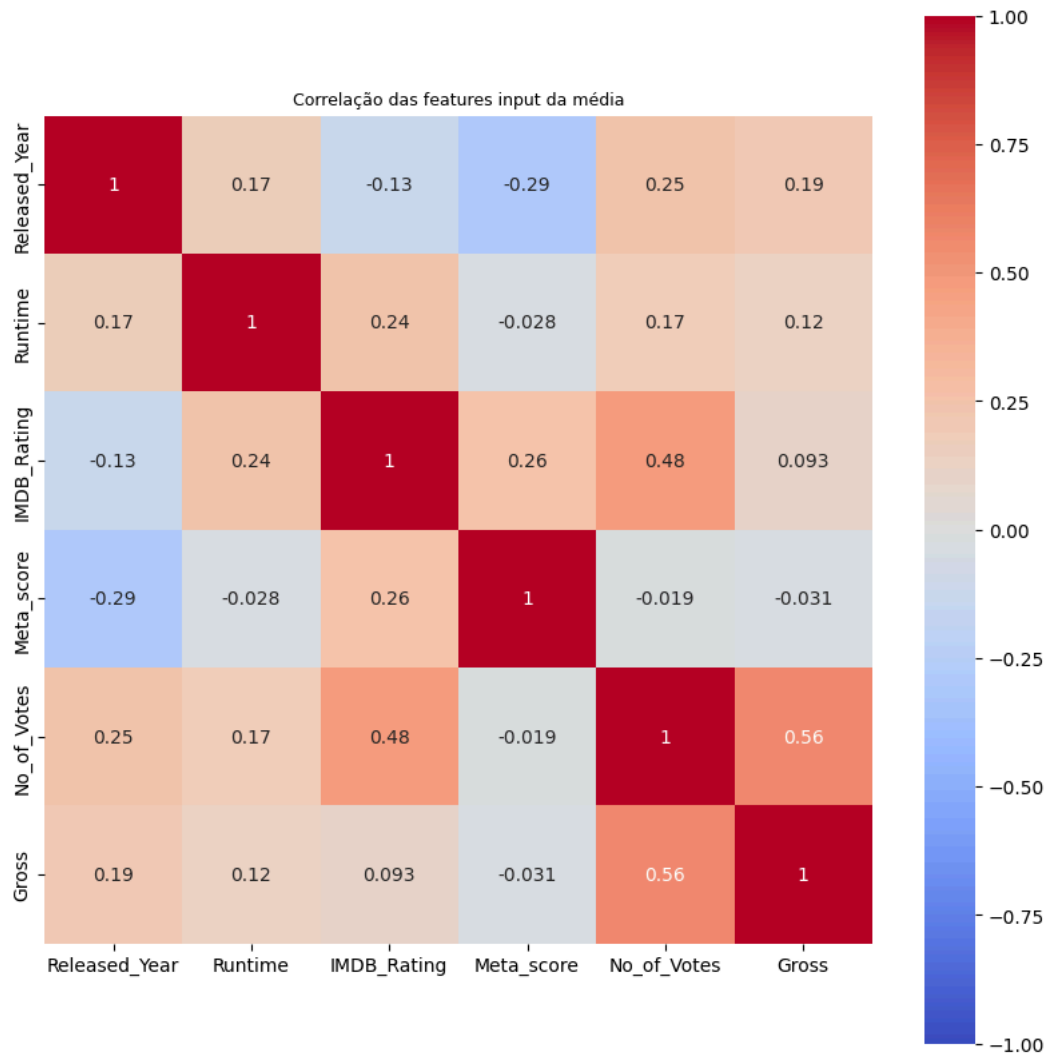
Mostrar código

```
<Axes: title={'center': 'Correlação das features dropando nulos'}>
```



[Mostrar código](#)

 <Axes: title={'center': 'Correlação das features input da média'}>



Notamos que os número de votos e o ano de lançamento apresentam uma maior correlação com o faturamento do filme

Dropando os nulos:

tivemos uma maior correlação do IMDB\_Rating com Gross, comparado aos outros métodos de replace.

Tivemos Número de votos, Meta Score e Runtime com maior correlação com IMDB\_Rating que será nosso alvo.

Métodos de Replace:

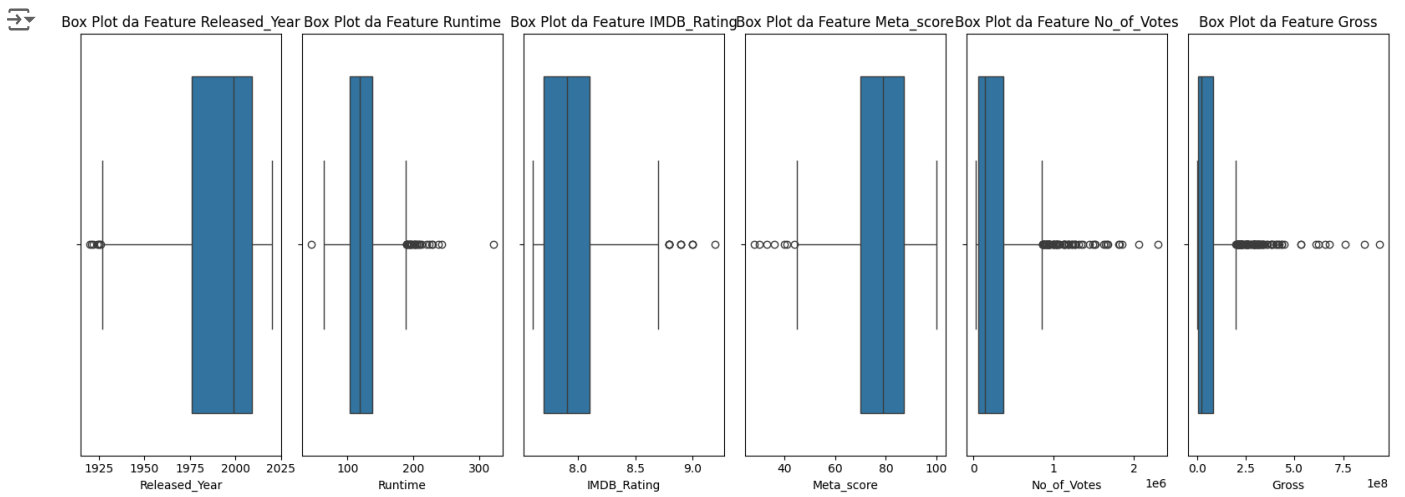
Obtivemos melhores correlações pela mediana comprado a média.

porém dropando os nulos obtivemos correlações mais alta.

Vamos analisar como estão a distribuição gerando um boxplot.

Com isso também queremos analisar a presença de outliers e investigar se precisaremos dropar

[Mostrar código](#)



## Considerações

### Released\_Year:

A maioria dos filmes foi lançada entre aproximadamente 1970 e 2015. Existem alguns outliers, especialmente filmes lançados antes de 1970.

### Runtime:

A maioria dos filmes tem uma duração entre aproximadamente 100 e 200 minutos. Existem outliers para filmes com durações muito curtas e muito longas.

### IMDB\_Rating:

As avaliações IMDb estão concentradas entre 7.5 e 8.5. Existem alguns outliers com avaliações acima de 8.5.

### Meta\_score:

As pontuações Meta estão concentradas entre 50 e 80. Existem vários outliers com pontuações abaixo de 50 e acima de 80.

No\_of\_Votes:

A maioria dos filmes tem até 1 milhão de votos. Existem muitos outliers com mais de 1 milhão de votos.

### Gross:

A maioria dos filmes tem uma receita bruta de até 250 milhões. Existem outliers com receitas brutas significativamente maiores, chegando a até 750 milhões.

As variáveis No\_of\_Votes e Gross têm uma escala significativamente diferente das outras variáveis, o que pode exigir normalização ou padronização

[Mostrar código](#)

```

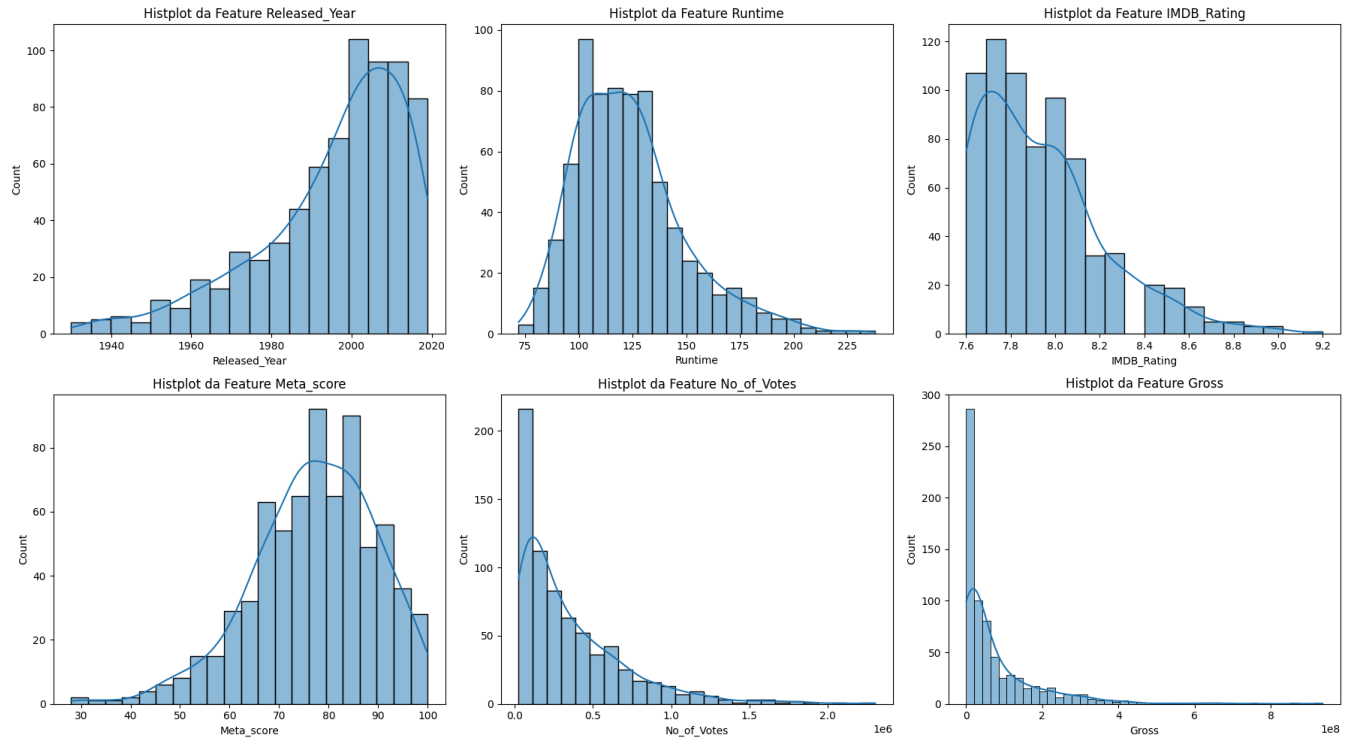
Skewness:
Released_Year    -1.144282
Runtime          1.009530
IMDB_Rating      1.114826
Meta_score       -0.583372
No_of_Votes      1.819175
Gross            2.916618
dtype: float64

```

```

Kurtosis:
Released_Year     0.917191
Runtime           1.355109
IMDB_Rating       1.252506
Meta_score        0.476433
No_of_Votes       4.202182
Gross             12.108811
dtype: float64

```

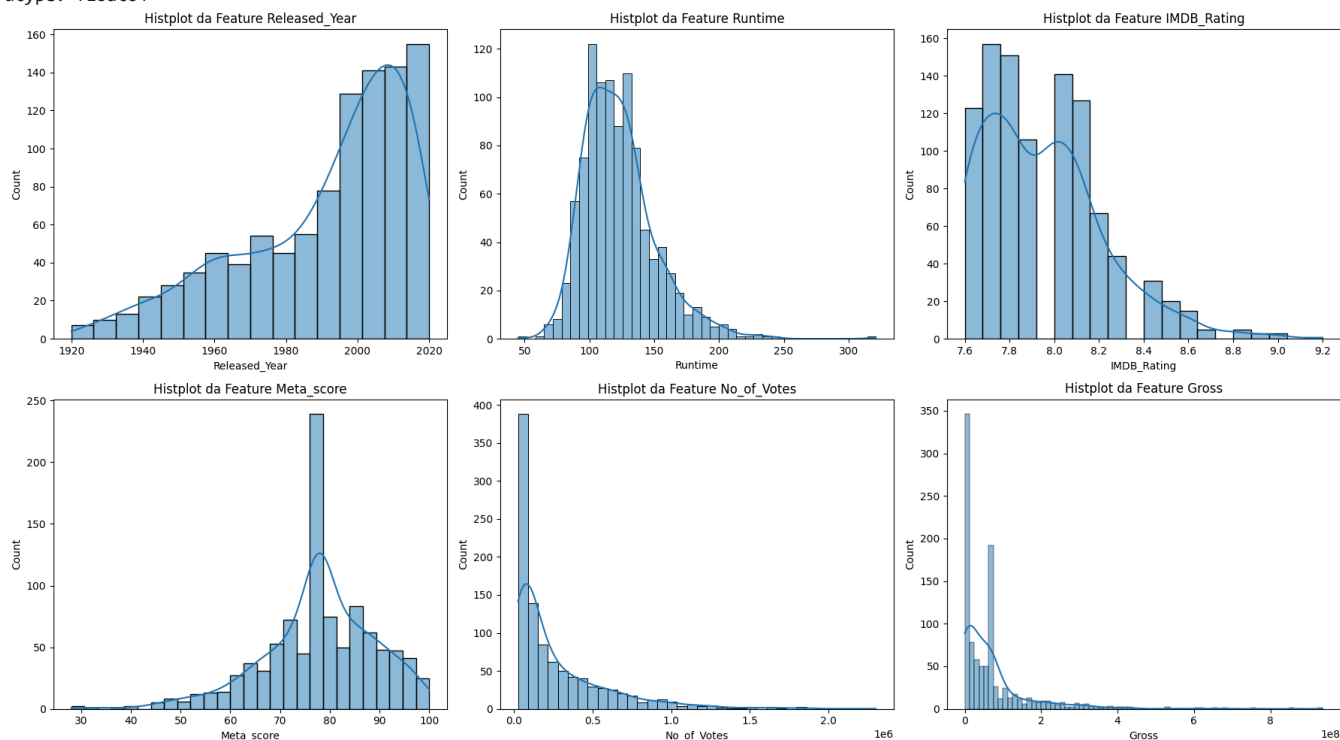


Vamos analisar a caudose e curtose com os valores nulos inputados pela mediana

[Mostrar código](#)

Skewness:  
 Released\_Year -0.938045  
 Runtime 1.208060  
 IMDB\_Rating 0.945271  
 Meta\_score -0.657077  
 No\_of\_Votes 2.191055  
 Gross 3.425225  
 dtype: float64

Kurtosis:  
 Released\_Year -0.027235  
 Runtime 3.405769  
 IMDB\_Rating 1.047107  
 Meta\_score 1.042189  
 No\_of\_Votes 6.005129  
 Gross 17.224666  
 dtype: float64



**Released\_Year:** Apresenta assimetria negativa moderada e kurtosis próxima de zero, indicando uma distribuição levemente distorcida para a esquerda, com caudas normais.

**Runtime:** Alta assimetria positiva e kurtosis acima de 3 indicam uma distribuição fortemente distorcida para a direita, com algumas caudas pesadas.

**IMDB\_Rating:** Assimetria positiva moderada e kurtosis acima de 1, sugerindo uma leve distorção para a direita e algumas caudas mais pesadas.

**Meta\_score:** Moderada assimetria negativa e kurtosis levemente acima de 1, indicando uma distribuição levemente distorcida para a esquerda com algumas caudas mais pesadas.

**No\_of\_Votes:** Alta assimetria positiva e kurtosis muito elevada indicam uma distribuição muito distorcida para a direita com muitas caudas pesadas.

**Gross:** Muito alta assimetria positiva e kurtosis extremamente elevada indicam uma distribuição extremamente distorcida para a direita com muitas caudas pesadas.

#### Comparação Entre Métodos de Imputação

Os valores de skewness e kurtosis são idênticos para ambas as técnicas de imputação (média e mediana), sugerindo que a escolha entre média e mediana não afetou a distribuição das variáveis. No entanto, é importante considerar as correlações com a variável alvo para determinar qual método é mais apropriado para a modelagem.

#### Escolha do Método de Imputação para Modelo de Regressão

**Resistência a Outliers:** A mediana não é influenciada por valores extremos, o que é especialmente relevante para variáveis como Gross e No\_of\_Votes, que apresentam alta assimetria positiva e kurtosis elevada.

Vamos verificar como será o desempenho de ML com sem imputação, média e mediana na regressão.



[Mostrar código](#)

```
↗ Fitting 3 folds for each of 50 candidates, totalling 150 fits
Melhores parâmetros encontrados: {'subsample': 0.5, 'reg_lambda': 1.0, 'reg_alpha': 0.01, 'n_estimators': 100, 'max_depth': 9, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree': 1.0}
Mean Squared Error on teste: 0.033854938232578895
R² Score on teste: 0.5270660509905415
Mean Absolute Error teste: 0.1472445547997535
MAPE: 1.85%
Accuracy: 98.15 %
```

[Mostrar código](#)

```
↗ Fitting 3 folds for each of 50 candidates, totalling 150 fits
Melhores parâmetros encontrados: {'subsample': 0.5, 'reg_lambda': 1.0, 'reg_alpha': 0.01, 'n_estimators': 100, 'max_depth': 9, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree': 1.0}
Mean Squared Error on teste: 0.03235306012841006
R² Score on teste: 0.5279940165454902
Mean Absolute Error teste: 0.1447999739646912
MAPE: 1.82%
Accuracy: 98.18 %
```

[Mostrar código](#)

```
↗ Fitting 3 folds for each of 50 candidates, totalling 150 fits
Melhores parâmetros encontrados: {'subsample': 0.7, 'reg_lambda': 0.1, 'reg_alpha': 0.1, 'n_estimators': 200, 'max_depth': 3, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree': 1.0}
Mean Squared Error on teste: 0.031893310213657246
R² Score on teste: 0.5347014102138088
Mean Absolute Error teste: 0.14077048015594484
MAPE: 1.77%
Accuracy: 98.23 %
```

Nosso melhor modelo até agora foi :

Com imputação pela média

Melhores parâmetros encontrados: {'subsample': 0.7, 'reg\_lambda': 0.1, 'reg\_alpha': 0.1, 'n\_estimators': 200, 'max\_depth': 3, 'learning\_rate': 0.1, 'gamma': 0.1, 'colsample\_bytree': 1.0}

Mean Squared Error on teste: 0.031893310213657246

R² Score on teste: 0.5347014102138088

Mean Absolute Error teste: 0.14077048015594484

MAPE: 1.77%

Accuracy: 98.23 %

[Mostrar código](#)

```
0 organized crime dynastys aging patriarch trans...
1 menace known joker wreaks havoc chaos people g...
2 early life career vito corleone new york city ...
3 jury holdout attempts prevent miscarriage just...
4 gandalf aragorn lead world men saurons army dr...
...
994 young new york socialite becomes interested yo...
995 sprawling epic covering life texas cattle ranc...
996 hawaii private cruelly punished boxing units t...
997 several survivors torpedoed merchant ship worl...
998 man london tries help counterespionage agent a...
Name: Overview, Length: 999, dtype: object
Palavras mais frequentes para o gênero 'Crime':
young: 16
murder: 15
crime: 13
family: 11
man: 11
police: 11
son: 8
life: 8
lives: 7
years: 7
Palavras mais frequentes para o gênero 'Action':
must: 19
young: 15
man: 14
world: 12
former: 12
war: 12
find: 11
help: 10
battle: 10
officer: 10
Palavras mais frequentes para o gênero 'Biography':
story: 22
life: 17
man: 10
war: 7
world: 6
becomes: 6
american: 6
new: 6
ii: 5
first: 5
Palavras mais frequentes para o gênero 'Drama':
life: 42
man: 41
young: 40
woman: 34
love: 32
new: 27
war: 26
world: 23
find: 20
family: 17
Palavras mais frequentes para o gênero 'Western':
joins: 3
bounty: 2
hunting: 1
scam: 1
men: 1
uneasy: 1
alliance: 1
third: 1
race: 1
find: 1
Palavras mais frequentes para o gênero 'Comedy':
young: 24
man: 17
life: 17
love: 16
friends: 14
new: 12
get: 11
finds: 11
girl: 10
family: 9
Palavras mais frequentes para o gênero 'Adventure':
world: 10
story: 8
war: 7
find: 7
man: 7
young: 7
group: 6
friends: 6
journey: 6
american: 6
Palavras mais frequentes para o gênero 'Animation':
```

young: 23  
 girl: 14  
 world: 13  
 new: 12  
 boy: 9  
 must: 9  
 find: 6  
 home: 6  
 life: 6  
 save: 6  
 Palavras mais frequentes para o gênero 'Horror':  
 young: 3  
 becomes: 3  
 run: 2  
 mother: 2  
 mysterious: 2  
 life: 2  
 soon: 2  
 old: 2  
 children: 2  
 convinced: 2  
 Palavras mais frequentes para o gênero 'Mystery':  
 man: 3  
 saskia: 3  
 wives: 2  
 murderer: 2  
 murder: 2  
 detective: 2  
 missing: 2  
 ever: 2  
 world: 2  
 young: 2  
 Palavras mais frequentes para o gênero 'Film-Noir':  
 pulp: 1  
 novelist: 1  
 holly: 1  
 martins: 1  
 travels: 1  
 shadowy: 1  
 postwar: 1  
 vienna: 1  
 find: 1  
 investigating: 1  
 Palavras mais frequentes para o gênero 'Fantasy':  
 hypnotist: 1  
 dr: 1  
 caligari: 1  
 uses: 1  
 somnambulist: 1  
 cesare: 1  
 commit: 1  
 murders: 1  
 vampire: 1  
 count: 1  
 Palavras mais frequentes para o gênero 'Family':  
 troubled: 1  
 child: 1  
 summons: 1  
 courage: 1  
 help: 1  
 friendly: 1  
 alien: 1  
 escape: 1  
 earth: 1  
 return: 1  
 Palavras mais frequentes para o gênero 'Thriller':  
 recently: 1  
 blinded: 1  
 woman: 1  
 terrorized: 1  
 trio: 1  
 thugs: 1  
 search: 1  
 heroinstuffed: 1  
 doll: 1  
 believe: 1

Word Cloud para o gênero Crime







Word Cloud para o gênero Horror

### Word Cloud para o gênero Mystery

### Word Cloud para o gênero Film-Noir

### Word Cloud para o gênero Fantasy

Word cloud para o gênero fantasia



A word cloud for the fantasy genre. The most prominent word is 'caligari' in large green letters. Other visible words include 'commit.', 'vampire', 'murders', 'expresses', 'estate', 'real', 'somnambulist', 'hutters', 'new', 'orlok', and 'estate'.



hypnotist  
 interest uses agent  
 cesare wife residence count

Word Cloud para o gênero Family

home chocolate send boy courage friendly mysterious tickets wonkas summons golden tour  
 child  
 poor seeks return alien earth world escape hopeful five  
 factory

Word Cloud para o gênero Thriller

woman trio thugs recently terrorized search  
 doll believe apartment  
 blinded  
 hero stuffed

Palavras mais frequentes para o gênero 'Crime': young: 16 murder: 15 crime: 13 family: 11 man: 11

Palavras mais frequentes para o gênero 'Action': must: 19 young: 15 man: 14 world: 12 former: 12

Palavras mais frequentes para o gênero 'Biography': story: 22 life: 17 man: 10 war: 7 world: 6

Palavras mais frequentes para o gênero 'Drama': life: 42 man: 41 young: 40 woman: 34 love: 32

Palavras mais frequentes para o gênero 'Western': joins: 3 bounty: 2 hunting: 1 scam: 1 men: 1

Palavras mais frequentes para o gênero 'Comedy': young: 24 man: 17 life: 17 love: 16 friends: 14

Palavras mais frequentes para o gênero 'Adventure': world: 10 story: 8 war: 7 find: 7 man: 7

Palavras mais frequentes para o gênero 'Animation': young: 23 girl: 14 world: 13 new: 12 boy: 9

Palavras mais frequentes para o gênero 'Horror': young: 3 becomes: 3 run: 2 mother: 2 mysterious: 2

Palavras mais frequentes para o gênero 'Mystery': man: 3 saskia: 3 wives: 2 murderer: 2 murder: 2

Palavras mais frequentes para o gênero 'Film-Noir': pulp: 1 novelist: 1 holly: 1 martins: 1 travels: 1

Palavras mais frequentes para o gênero 'Fantasy': hypnotist: 1 dr: 1 caligari: 1 uses: 1 somnambulist: 1

Palavras mais frequentes para o gênero 'Family': troubled: 1 child: 1 summons: 1 courage: 1 help: 1

Palavras mais frequentes para o gênero 'Thriller': recently: 1 blinded: 1 woman: 1 terrorized: 1 trio: 1

## Mostrar código

```

995         drama, western
996         drama, romance, war
997             drama, war
998 crime, mystery, thriller

                                Overview                                Director \
0  an organized crime dynasty's aging patriarch t...  francis ford coppola
1  when the menace known as the joker wreaks havo...  christopher nolan
2  the early life and career of vito corleone in ...  francis ford coppola
3  a jury holdout attempts to prevent a miscarria...  sidney lumet
4  gandalf and aragorn lead the world of men agai...  peter jackson
..
994 a young new york socialite becomes interested ...  blake edwards
995 sprawling epic covering the life of a texas ca...  george stevens
996 in hawaii in 1941, a private is cruelly punish...  fred zinnemann
997 several survivors of a torpedoed merchant ship...  alfred hitchcock
998 a man in london tries to help a counter-espion...  alfred hitchcock

                                Star1                                Star2                                Star3                                Star4 \
0  marlon brando                                al pacino                                james caan                                diane keaton
1  christian bale                                heath ledger                                aaron eckhart                                michael caine
2  al pacino                                robert de niro                                robert duvall                                diane keaton
3  henry fonda                                lee j. cobb                                martin balsam                                john fiedler
4  elijah wood                                viggo mortensen                                ian mckellen                                orlando bloom
..
994 audrey hepburn                                george peppard                                patricia neal                                buddy ebsen
995 elizabeth taylor                                rock hudson                                james dean                                carroll baker
996 burt lancaster                                montgomery clift                                deborah kerr                                donna reed
997 tallulah bankhead                                john hodiak                                walter slezak                                william bendix
998 robert donat                                madeleine carroll                                lucie mannheim                                godfrey tearle

                                action ... horror music musical mystery romance sci-fi sport \
0  0 ... 0 0 0 0 0 0 0
1  1 ... 0 0 0 0 0 0 0
2  0 ... 0 0 0 0 0 0 0
3  0 ... 0 0 0 0 0 0 0
4  1 ... 0 0 0 0 0 0 0
..
994 0 ... 0 0 0 0 1 0 0
995 0 ... 0 0 0 0 0 0 0
996 0 ... 0 0 0 0 1 0 0
997 0 ... 0 0 0 0 0 0 0
998 0 ... 0 0 0 1 0 0 0

                                thriller war western
0  0 0 0
1  0 0 0
2  0 0 0
3  0 0 0
4  0 0 0
..
994 0 0 0
995 0 0 1
996 0 1 0
997 0 1 0
998 1 0 0

[999 rows x 30 columns]
```

Vamos continuar a exploração e tratamentos das features categoricas

vamos verificar a coluna certificate que contém a classificação indicativa dos filmes

Foi realizado uma pesquisa sobre os tipos de Certificates, alguns a maioria deles em nossa features são do padrão Grã-bretanha e dos EUA. Porém temos como agrupar para diminuir uma melhor padronização e diminuir a quantidades para gerar melhores insights.

```
categorics_one_hot['Certificate'].value_counts()
```

```

Certificate
u          335
a          196
ua         175
r          146
pg-13       43
pg          37
passed      34
g           12
approved    11
tv-pg        3
gp           2
tv-14        1
16           1
tv-ma        1
unrated      1
```

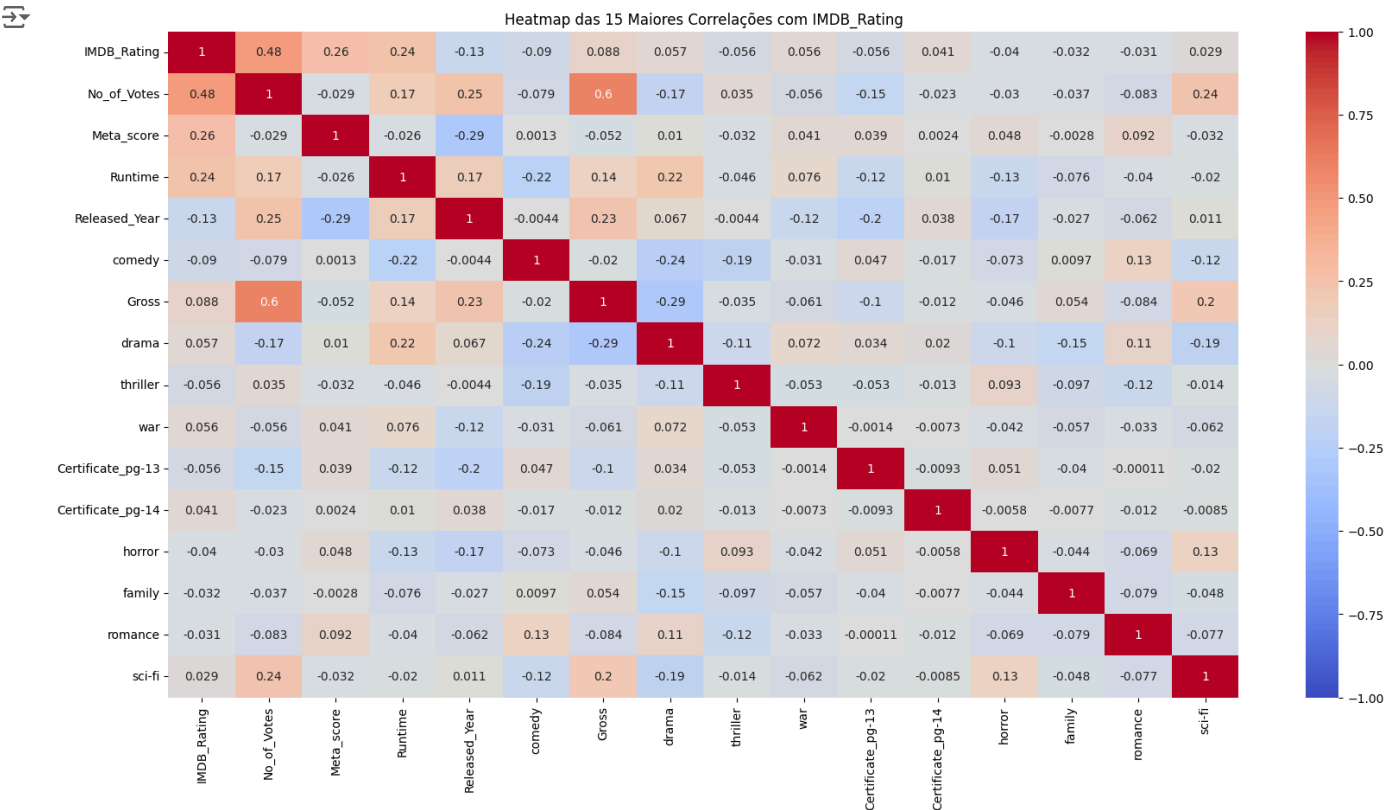


u/a 1  
Name: count, dtype: int64

Mostrar código

Certificate\_Agrupado  
adult 343  
pg-12 176  
pg-13 80  
pg 39  
general audience 23  
pg-14 1  
pg-16 1  
unrated 1  
Name: count, dtype: int64

Mostrar código



Notamos que os gêneros que apresentaram maior correlação positiva com a nota do IMDB foi drama e war, já comedy e thriler apresentaram uma correlação mais fortes que as demais porém negativa. Isso indica que os filmes com as maiores notas do IMDB são de drama e comédia. As classificações indicativa que apresenta maior correlação com a nota do IMDB é PG-13 e PG-14

Mostrar código

Fitting 3 folds for each of 50 candidates, totalling 150 fits  
Melhores parâmetros encontrados: {'subsample': 0.7, 'reg\_lambda': 0.1, 'reg\_alpha': 0.1, 'n\_estimators': 200, 'max\_depth': 3, 'learning\_rate': 0.1}  
Mean Squared Error on teste: 0.03320164268439449  
R² Score on teste: 0.5156138570709294  
Mean Absolute Error teste: 0.14425987148284913  
MAPE: 1.81%  
Accuracy: 98.19 %

No modelo com as features numéricas nulas inputadas com a média se demonstrou mais sensível quando fizemos o treino com as features de Gênero e Classificação indicativa, diminuindo o R² para 0.49 já o modelo com mediana ficou em 0.51, se mostrando mais estável. Porém não tivemos melhoras comparado aos primeiros modelos testados.

Mostrar código

```

Top 20 Diretores:
Director
Steven Spielberg      13
Martin Scorsese       10
Alfred Hitchcock      9
David Fincher         8
Clint Eastwood        8
Quentin Tarantino     8
Christopher Nolan     8
Woody Allen           7
Rob Reiner            7
Hayao Miyazaki        7
Stanley Kubrick       6
Richard Linklater     6
Wes Anderson         6
Ridley Scott          6
Joel Coen             6
James Cameron         5
Alfonso Cuarón        5
Denis Villeneuve      5
Francis Ford Coppola  5
Ron Howard            5
Name: count, dtype: int64

```

```

Top 20 Atores:
Robert De Niro      16
Tom Hanks           14
Al Pacino           13
Brad Pitt           12
Matt Damon          11
Christian Bale      11
Leonardo DiCaprio  11
Clint Eastwood      11
Johnny Depp         9
Denzel Washington  9
Scarlett Johansson  9
Ethan Hawke         9
Harrison Ford       8
Ian McKellen        7
Jake Gyllenhaal     7
Robert Downey Jr.   7
Emma Watson         7
Edward Norton       7
Russell Crowe       7
Bruce Willis        7
Name: count, dtype: int64

```

Temos a lista dos diretores e atores com maior frequência nos filmes do nosso data set. Steven Spielberg e Robert De Niro lideram o ranking

[Mostrar código](#)

```

Feature Correlation
0      No_of_Votes    0.479308
1      Meta_score    0.261010
2      Runtime        0.242751
3  Director_Christopher Nolan    0.194498
4  Director_Francis Ford Coppola    0.135251
5      Released_Year   -0.133355
6      Star_Harrison Ford    0.120726
7  Director_Stanley Kubrick    0.105733
8  Director_Martin Scorsese    0.096967
9      Star_Ian McKellen    0.096225
10     comedy          -0.090209
11  Director_Quentin Tarantino    0.088451
12      Gross          0.088139
13  Star_Denzel Washington   -0.087735
14  Star_Robert De Niro      0.086457
Fitting 3 folds for each of 50 candidates, totalling 150 fits
Melhores parâmetros encontrados: {'subsample': 0.7, 'reg_lambda': 0.1, 'reg_alpha': 0.1, 'n_estimators': 200, 'max_depth': 3, 'lear
Mean Squared Error no teste: 0.03640133981120897
R² Score no teste: 0.4852678939010877
Mean Absolute Error no teste: 0.15243037986755367
MAPE: 1.92%
Accuracy: 98.08%

```

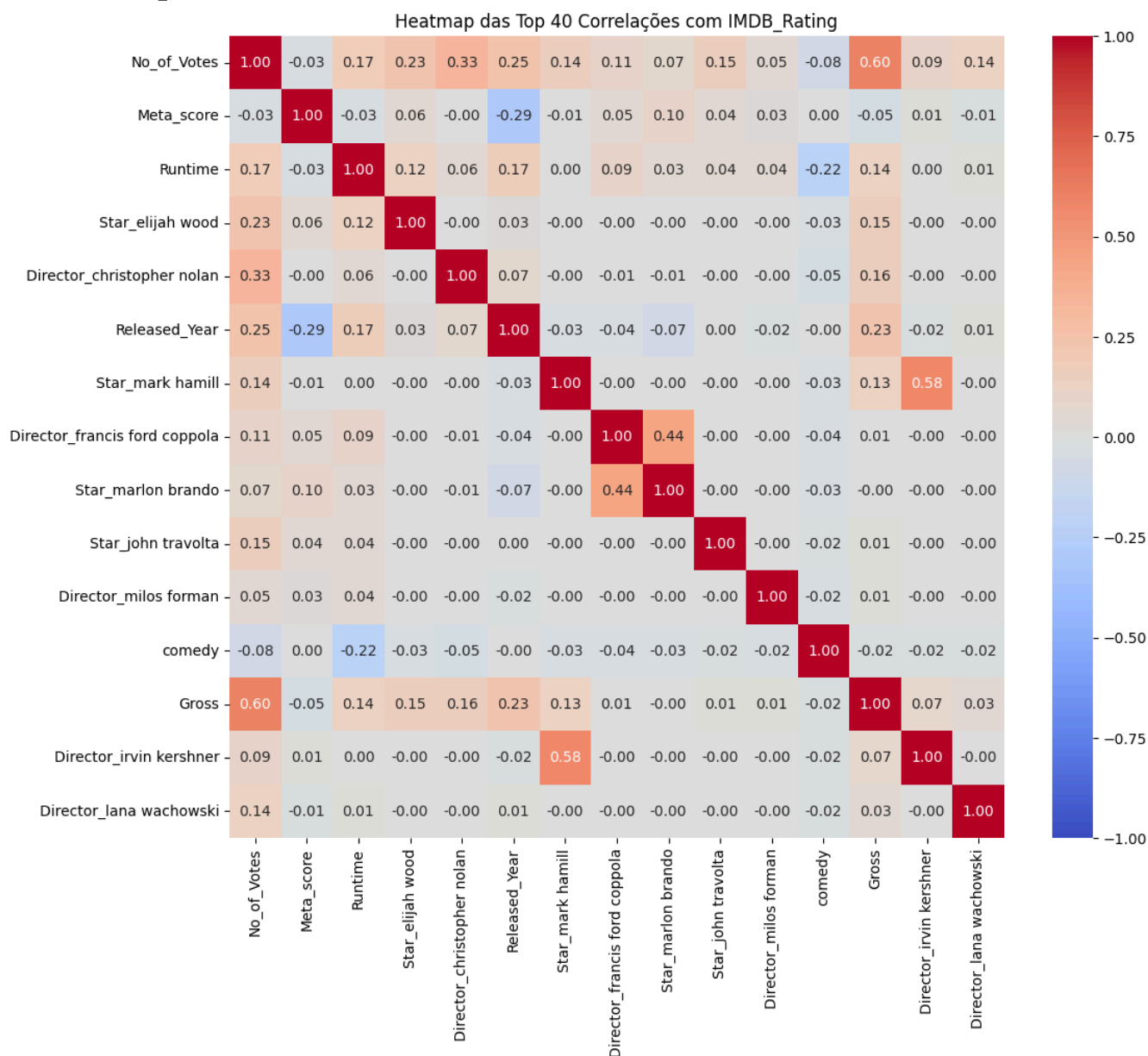
Os diretores Christopher Nolan e Francis Ford Coppola lideram o ranking o primeiro ator que aparece com maior correlação é Harrison Ford.

Vamos verificar agora se os atores e diretores filtrados por estarem nos filmes com maior nota do IMDB terão maior correlação.

[Mostrar código](#)



	Feature	Correlation
0	No_of_Votes	0.479308
1	Meta_score	0.261010
2	Runtime	0.242751
3	Star_elijah wood	0.171824
4	Director_christopher nolan	0.169873
5	Released_Year	-0.133355
6	Star_mark hamill	0.118048
7	Director_francis ford coppola	0.117806
8	Star_marlon brando	0.111140
9	Star_john travolta	0.110734
10	Director_milos forman	0.090852
11	comedy	-0.090209
12	Gross	0.088139
13	Director_irvin kershner	0.087472
14	Director_lana wachowski	0.087472



Fitting 3 folds for each of 50 candidates, totalling 150 fits  
Melhores parâmetros encontrados: {'subsample': 0.5, 'reg\_lambda': 1.0, 'reg\_alpha': 0.01, 'n\_estimators': 100, 'max\_depth': 9, 'lea  
Mean Squared Error no teste: 0.0320988156782857  
R<sup>2</sup> Score no teste: 0.5317032453245453  
Mean Absolute Error no teste: 0.14338677501678468  
MAPE: 1.80%  
Accuracy: 98.2%

Nosso melhor modelo de regressão foi treinado com os atores com as maiores notas do IMDB. Fizemos a seleção das features com maior valor absoluto da correlação com nosso alvo.

A métricas são :

Mean Squared Error no teste: 0.0320988156782857

R<sup>2</sup> Score no teste: 0.5317032453245453

Mean Absolute Error no teste: 0.14338677501678468

MAPE: 1.80%

Accuracy: 98.2%

A acurácia está alta, o que significa que uma boa parte das previsões no conjunto de teste estão corretas. No entanto, queremos aumentar o R<sup>2</sup> score para garantir que nosso modelo capture melhor a variação nos dados e, assim, possa fazer previsões mais precisas para uma variedade maior de conjuntos de dados.

Vamos importar dois data sets, o primeiro de filmes ganhadores do oscar e o segundo de diretores/diretoras, atores e atrizes ganhadores do oscar.

Nosso intuito será encontrar features que tenham correlação maior com a nota do IMDB, parece intuitivo que filmes que ganharam o oscar e atores ganhadores do oscar estejam em filmes com maiores notas. Vamos verificar por meio de correlações, heatmaps e treinando alguns modelos.

```
oscar_movies = pd.read_csv('./drive/MyDrive/Colab Notebooks/oscars/oscar_movies.csv')
oscar_actors = pd.read_csv('./drive/MyDrive/Colab Notebooks/oscars/oscar_actors.csv')
df_numerical_features = df.copy()
df_numerical_features = df_numerical_features.select_dtypes(include=['number'])
df_median = df_numerical_features.median()
df_numerical_features = df_numerical_features.fillna(df_median)
```

[Mostrar código](#)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1360 entries, 0 to 1359
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Unnamed: 0   1360 non-null  int64
1   Film         1359 non-null  object
2   Year         1360 non-null  int64
3   Award        1360 non-null  int64
4   Nomination   1360 non-null  int64
dtypes: int64(4), object(1)
memory usage: 53.2+ KB
Unnamed: 0      0
Film             1
Year             0
Award            0
Nomination       0
dtype: int64
```

oscar\_actors.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10889 entries, 0 to 10888
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  -
0   year_film    10889 non-null  int64
1   year_ceremony 10889 non-null  int64
2   ceremony     10889 non-null  int64
3   category     10889 non-null  object
4   name         10884 non-null  object
5   film         10570 non-null  object
6   winner       10877 non-null  object
7   winner_desloc 645 non-null    object
8   Unnamed: 8   5 non-null     object
dtypes: int64(3), object(6)
memory usage: 765.8+ KB
```

[Mostrar código](#)

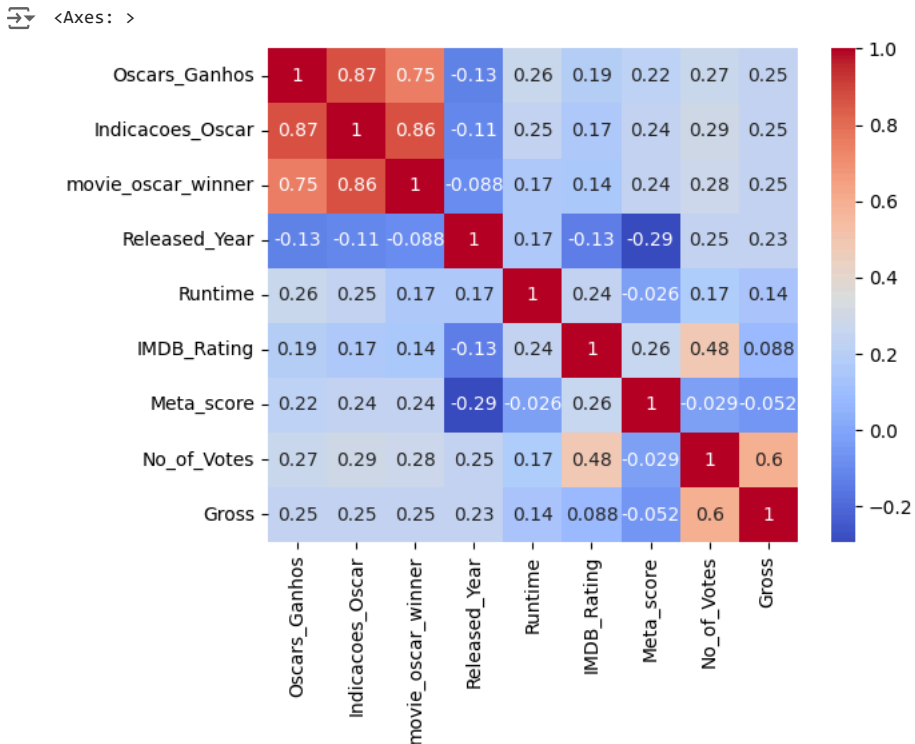
```
Proporção de filmes indicados ao Oscar: 27.23%
Proporção de filmes com indicações que ganharam algum Oscar: 27.03%
Proporção de filmes indicados que não ganharam Oscar: 0.20%
Proporção de filmes que não foram indicados ao Oscar: 72.77%
Filmes que foram indicados mas não ganharam Oscar:
   Series_Title  Indicacoes_Oscar  Oscars_Ganhos
100      toy story                 3              0
425  planet of the apes             2              0
```

```
df_oscar = pd.concat([df_new_categoricals, df_new_numerical], axis=1)
df_oscar = df_oscar.select_dtypes(include=['number'])
df_oscar = df_oscar.dropna()
df_oscar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Oscars_Ganhos         999 non-null    int64
1   Indicacoes_Oscar      999 non-null    int64
2   movie_oscar_winner    999 non-null    int64
3   Released_Year         999 non-null    int64
4   Runtime               999 non-null    int64
5   IMDB_Rating           999 non-null    float64
6   Meta_score            999 non-null    float64
7   No_of_Votes           999 non-null    int64
8   Gross                 999 non-null    float64
dtypes: float64(3), int64(6)
memory usage: 70.4 KB
```

Fizemos alguns tratamentos necessário no data set e concatenação com nosso data set do IMDB, vamos investigar algumas correlações

```
#Vamos calcular a correlação com gross
correlation = df_oscar.corr()
correlation['IMDB_Rating'].sort_values(ascending=False)
#Vamos gerar um heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm')
```



Nossas features da quantidade de oscars ganhos, indicações ao oscar e se o filme foi ganhador do oscar tiveram correlação mais fortes com a nota do IMDB do que muitas de nossas features do data set original.

As correlações com o faturamento foram até maior do que a nota do IMDB, indicado que os filmes ganhadores de oscar e a quantidade de oscars influencia sim o faturamento do filme.

Oscars Ganhos e indicações oscars apresentam uma forte multicolinearidade.

```
# Dataset com top 10 de diretores e atores mais frequentes
X = df_oscar.drop(['IMDB_Rating'], axis=1)
y = df_oscar['IMDB_Rating']
```

```
params = {
    'objective': 'reg:squarederror',
    'max_depth': 8,
    'learning_rate': 0.02,
    'n_estimators': 600,
    'subsample': 0.5,
```

```

'colsample_bytree': 0.8,
'reg_alpha': 0.01,
'reg_lambda': 1.0,
'gamma': 0.03,
'random_state': 42
}

# Dividir os dados normalizados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=43)

# Criar o modelo XGBRegressor
regressor = xgb.XGBRegressor(**params)

# Treinar o modelo
regressor.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = regressor.predict(X_test)

# Avaliar o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
r2 = r2_score(y_test, y_pred)
print(f'R² Score: {r2}')
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')

errors = abs(y_pred - y_test)

# Calculando o MAPE
mape = 100 * (errors / y_test)
mean_mape = np.mean(mape)
print(f'MAPE: {mean_mape:.2f}%')

# Calculando a acurácia
accuracy = 100 - mean_mape
print('Accuracy:', round(accuracy, 2), '%')
#modelo6

```

```

➡ Mean Squared Error: 0.0323459618200779
R² Score: 0.5280975753430779
Mean Absolute Error: 0.14009504222869876
MAPE: 1.76%
Accuracy: 98.24 %

```

Nosso modelo que está apresentando bom desempenho tem as features dos filmes que ganharam o oscar, os top 10 diretores e atores com maiores notas do IMDB e as demais features ( gross, ano de lançamento, meta score, quantidade de avaliações)

Agora vamos verificar nosso data set com atores e diretores ganhadores do oscar.

```

oscar_actors.info()
oscar_actors.isnull().sum()

➡ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 10889 entries, 0 to 10888
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  -
 0   year_film       10889 non-null  int64
 1   year_ceremony   10889 non-null  int64
 2   ceremony        10889 non-null  int64
 3   category        10889 non-null  object
 4   name            10884 non-null  object
 5   film            10570 non-null  object
 6   winner          10877 non-null  object
 7   winner_desloc   645 non-null    object
 8   Unnamed: 8      5 non-null      object
dtypes: int64(3), object(6)
memory usage: 765.8+ KB
year_film      0
year_ceremony  0
ceremony        0
category        0
name            5
film            319
winner          12
winner_desloc   10244
Unnamed: 8      10884
dtype: int64

```

Foi verificado que muitos dos valores nulos estão deslocados uma coluna para direita, vamos relocar.

oscar\_actors.head(20)

	category	name	film	winner	
0	actor	richard barthelmess	the noose	False	
1	actor	emil jannings	the last command	True	
2	actress	louise dresser	a ship comes in	False	
3	actress	janet gaynor	7th heaven	True	
4	actress	gloria swanson	sadie thompson	False	
5	art direction	rochus gliese	sunrise	False	
6	art direction	william cameron menzies	the dove	True	
7	art direction	harry oliver	7th heaven	False	
8	cinematography	george barnes	the devil dancer	False	
9	cinematography	charles rosher	sunrise	True	
10	cinematography	karl struss	sunrise	True	
11	directing (comedy picture)	lewis milestone	two arabian knights	True	
12	directing (comedy picture)	ted wilde	speedy	False	
13	directing (dramatic picture)	frank borzage	7th heaven	True	
14	directing (dramatic picture)	herbert brenon	sorrell and son	False	
15	directing (dramatic picture)	king vidor	the crowd	False	
17	engineering effects	roy pomeroy	wings	True	
19	outstanding picture	the caddo company	the racket	False	
20	outstanding picture	fox	7th heaven	False	
21	outstanding picture	paramount famous lasky	wings	True	

Próximas etapas: [Gerar código com oscar\\_actors](#) [Ver gráficos recomendados](#)

```
allowed_categories = ['actor', 'actress', 'directing']
```

```
# Filtrar o DataFrame para manter apenas os registros com 'category' permitido
oscar_actors_filtered = oscar_actors[oscar_actors['category'].str.lower().str.split().str[0].isin(allowed_categories)]
# Filtrar oscar_actors para obter apenas os vencedores
oscar_winners = oscar_actors_filtered[oscar_actors_filtered['winner']]
```

```
df_new_categories = df_new_categories.applymap(lambda x: x.strip().lower() if isinstance(x, str) else x)
oscar_winners['name'] = oscar_winners['name'].str.strip().str.lower()
```

```
# Obter conjunto de nomes únicos em oscar_winners para comparação eficiente
oscar_winner_names = set(oscar_winners['name'])
```

```
# Função para verificar se algum nome está em oscar_winners
def check_previous_winner(row):
    for col in ['Director', 'Star1', 'Star2', 'Star3', 'Star4']:
        if row[col] in oscar_winner_names:
            return True
    return False
```

```
# Aplicar a função a cada linha de df_new_categorical
df_new_categories['previous_oscar_winner'] = df_new_categories.apply(check_previous_winner, axis=1)
```

```
<ipython-input-67-040bef2cb750>:9: 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
oscar_winners['name'] = oscar_winners['name'].str.strip().str.lower()
```

```
df_new_categories
df_new_categories = df_new_categories.drop(['Series_Title', 'Film_Normalizado'], axis=1)
```

```
df_new_categories.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 999 entries, 0 to 998
Data columns (total 13 columns):
#   Column                               Non-Null Count  Dtype  
---  -
0   Certificate                           999 non-null    object  
1   Genre                                999 non-null    object  
2   Overview                              999 non-null    object  
3   Director                             999 non-null    object  
4   Star1                                999 non-null    object  
5   Star2                                999 non-null    object  
6   Star3                                999 non-null    object  
7   Star4                                999 non-null    object  
8   Series_Title_Normalizado             999 non-null    object  
9   Oscars_Ganhos                        999 non-null    int64  
10  Indicacoes_Oscar                     999 non-null    int64  
11  movie_oscar_winner                   999 non-null    int64  
12  previous_oscar_winner                 999 non-null    bool    
dtypes: bool(1), int64(3), object(9)
memory usage: 94.8+ KB

```

[Mostrar código](#)

```

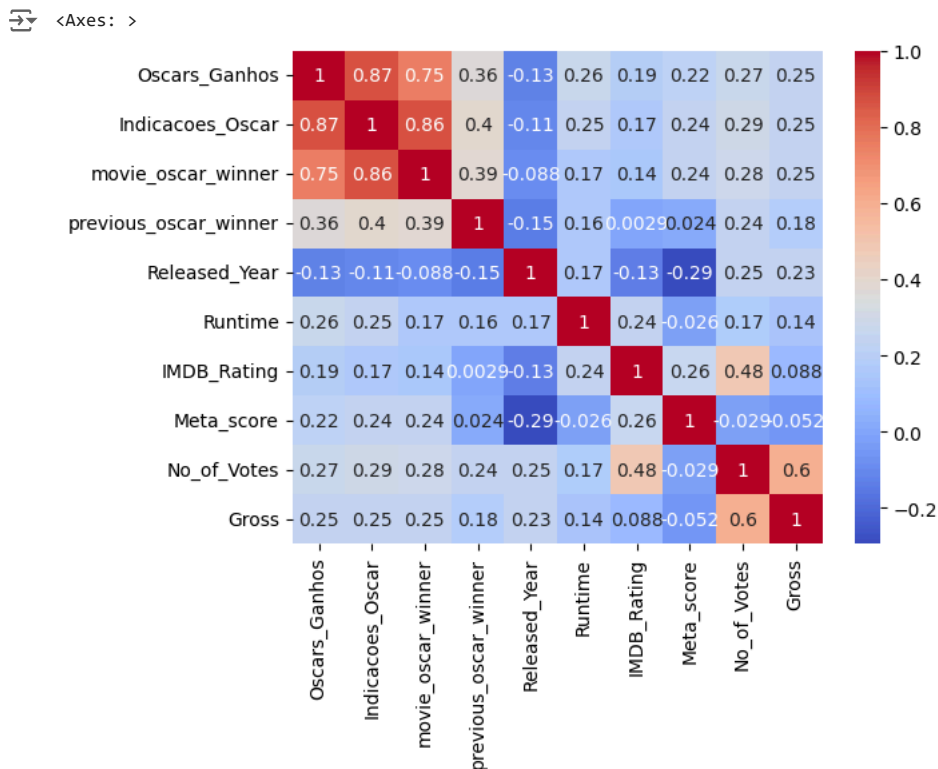
Porcentagem de ganhadores do Oscar no nosso dataset: 48.25%
Porcentagem de não ganhadores do Oscar no nosso dataset: 51.75%

```

```

#Vamos calcular a correlação com gross
correlation = df_categorics_num_oscar.corr()
correlation['IMDB_Rating'].sort_values(ascending=False)
#Vamos gerar um heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm')

```



[Mostrar código](#)

```

Mean Squared Error: 0.03207678685276501
R² Score: 0.532024628755138
Mean Absolute Error: 0.13978295660018922
MAPE: 1.75%
Accuracy: 98.25 %

```

Nosso modelo de regressão teve um desempenho levemente maior com o dataset original com as features numéricas concatenado com os datasets de filmes, atores e diretores ganhadores do oscar.

Vamos concatenar o dataset com as features de gênero e classificação para verificar o desempenho.

[Mostrar código](#)



```
Feature Correlation
0      No_of_Votes 0.479308
1      Meta_score  0.261010
2      Runtime     0.242751
3      Oscars_Ganhos 0.187774
4      Star_elijah wood 0.171824
..      ...
65     adventure   0.007430
66     history     0.005035
67     previous_oscar_winner 0.002864
68     action      0.001294
69     musical     -0.000430

[70 rows x 2 columns]
Fitting 3 folds for each of 50 candidates, totalling 150 fits
Melhores parâmetros encontrados: {'subsample': 0.7, 'reg_lambda': 10.0, 'reg_alpha': 0, 'n_estimators': 200, 'max_depth': 3, 'learn
Mean Squared Error no teste: 0.03273865078069816
R² Score no teste: 0.522368548836354
Mean Absolute Error no teste: 0.1412326216697693
MAPE: 1.77%
Accuracy: 98.23%
```

Mostrar código

```
Feature Correlation
0      No_of_Votes 0.479308
1      Meta_score  0.261010
2      Runtime     0.242751
3      Oscars_Ganhos 0.187774
4      Star_elijah wood 0.171824
..      ...
65     adventure   0.007430
66     history     0.005035
67     previous_oscar_winner 0.002864
68     action      0.001294
69     musical     -0.000430

[70 rows x 2 columns]
Model: RandomForestRegressor
Mean Squared Error: 0.033538299999999965
R² Score: 0.510702288684235
Mean Absolute Error: 0.14457999999999996
MAPE: 1.81%
Accuracy: 98.19%

Model: GradientBoostingRegressor
Mean Squared Error: 0.03231566047629319
R² Score: 0.528539648380878
Mean Absolute Error: 0.14372255338905357
MAPE: 1.80%
Accuracy: 98.2%

Model: AdaBoostRegressor
Mean Squared Error: 0.04289630050488991
R² Score: 0.3741763398575375
Mean Absolute Error: 0.1706391976289605
MAPE: 2.16%
Accuracy: 97.84%
```

Nosso modelo com Gradiente XGBoost apresentou melhores resultados comparado ao AdaBoost, GradienteBoost e RandomForest

Porém nosso modelo com a combinação do dataset com onehot de diretores , atores , genre e certificate não apresentou melhores resultados do que o modelo com as features numéricas do dataset original combinado com os ganhadores do oscars.

Vamos retomar com nosso modelo com melhor desempenho e verificar o desempenho dele com o input do filme a ser predito.

Também vamos serializar o modelo.

```
{'Series_Title': 'The Shawshank Redemption', 'Released_Year': '1994', 'Certificate': 'A', 'Runtime': '142 min', 'Genre': 'Drama', 'Overview': 'Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.', 'Meta_score': 80.0, 'Director': 'Frank Darabont', 'Star1': 'Tim Robbins', 'Star2': 'Morgan Freeman', 'Star3': 'Bob Gunton', 'Star4': 'William Sadler', 'No_of_Votes': 2343110, 'Gross': '28,341,469'}
```

Mostrar código

```
Mean Squared Error: 0.03210785932190307
R² Score: 0.5315713055982045
Mean Absolute Error: 0.14130054235458378
```

MAPE: 1.77%  
Accuracy: 98.23 %  
{'Director\_Won\_Oscar': 1, 'Star1\_Won\_Oscar': 1, 'Star2\_Won\_Oscar': 1, 'Star3\_Won\_Oscar': 1, 'Star4\_Won\_Oscar': 1}  
Predicted IMDB Rating: 7.823731899261475

Fizemos um treino no nosso modelo com filme, atores e diretores que ganharam o oscar para verificar se a função de checar se o input do dicionário estava retornando corretamente o valor boelano

Mostrar código

➦ Mean Squared Error: 0.03210785932190307  
R² Score: 0.5315713055982045  
Mean Absolute Error: 0.14130054235458378  
MAPE: 1.77%  
Accuracy: 98.23 %  
{'Director\_Won\_Oscar': 0, 'Star1\_Won\_Oscar': 1, 'Star2\_Won\_Oscar': 1, 'Star3\_Won\_Oscar': 0, 'Star4\_Won\_Oscar': 0}  
Processed Input:  
Oscars\_Ganhos   Indicacoes\_Oscar   movie\_oscar\_winner   previous\_oscar\_winner   \  
0   0   0   0   True  
  
Released\_Year   Runtime   Meta\_score   No\_of\_Votes   Gross  
0   1994   142   80.0   2343110   28341469.0  
Predicted IMDB Rating: 8.738204956054688

Mostrar código

➦

	Feature	Correlation
0	No_of_Votes	0.479308
1	Meta_score	0.261010
2	Runtime	0.242751
3	Star_elijah wood	0.171824
4	Director_christopher nolan	0.169873
5	Released_Year	-0.133355
6	Star_mark hamill	0.118048
7	Director_francis ford coppola	0.117806
8	Star_marlon brando	0.111140
9	Star_john travolta	0.110734
10	Director_milos forman	0.090852
11	comedy	-0.090209
12	Gross	0.088139
13	Director_irvin kershner	0.087472
14	Director_lana wachowski	0.087472

Mean Squared Error no teste: 0.03339910788627613  
R² Score no teste: 0.5127329933615227  
Mean Absolute Error no teste: 0.14445956707000734  
MAPE: 1.81%  
Accuracy: 98.19%  
Predicted IMDB Rating: 8.63430404663086

```

# Separar as features e o alvo
X = df_mean.drop('IMDB_Rating', axis=1)
y = df_mean['IMDB_Rating']

# Melhores parâmetros avaliados com grid search
params = {
    'objective': 'reg:squarederror',
    'subsample': 0.7,
    'reg_lambda': 0.1,
    'reg_alpha': 0.1,
    'n_estimators': 200,
    'max_depth': 3,
    'learning_rate': 0.1,
    'gamma': 0.1,
    'colsample_bytree': 1.0
}

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=43)

# Normalizar os dados
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Criar o modelo XGBRegressor
regressor = xgb.XGBRegressor(**params)

# Treinar o modelo
regressor.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = regressor.predict(X_test)

# Avaliar o desempenho do modelo nos dados de teste
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error on test: {mse}')
r2 = r2_score(y_test, y_pred)
print(f'R² Score on test: {r2}')
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error on test: {mae}')

errors = abs(y_pred - y_test)

# Calculando o MAPE
mape = 100 * (errors / y_test)
mean_mape = np.mean(mape)
print(f'MAPE: {mean_mape:.2f}%')

# Calculando a acurácia
accuracy = 100 - mean_mape
print('Accuracy:', round(accuracy, 2), '%')

# Função para fazer a previsão de um novo input
def preprocess_input(input_data):
    input_df = pd.DataFrame([input_data])
    # Realizar as mesmas transformações aplicadas ao conjunto de treinamento
    input_df['Gross'] = input_df['Gross'].str.replace(',', '').astype(float)
    input_df['Runtime'] = input_df['Runtime'].str.replace(' min', '').astype(int)
    input_df['Meta_score'] = input_df['Meta_score'].astype(float)
    # Assegurar que todas as colunas necessárias estão presentes
    for column in X.columns:
        if column not in input_df:
            input_df[column] = 0
    input_df = input_df[X.columns]
    return scaler.transform(input_df)

# Novo input
new_input = {
    'Series_Title': 'The Shawshank Redemption',
    'Released_Year': '1994',
    'Certificate': 'A',
    'Runtime': '142 min',
    'Genre': 'Drama',
    'Overview': 'Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.',
    'Meta_score': 80.0,
    'Director': 'Frank Darabont',
    'Star1': 'Tim Robbins',
    'Star2': 'Morgan Freeman',
    'Star3': 'Bob Gunton',
    'Star4': 'William Sadler',

```

```
'No_of_Votes': 2343110,  
'Gross': '28,341,469'
```

```
}
```

```
# Preprocessar o novo input  
processed_input = preprocess_input(new_input)  
  
# Fazer a previsão  
prediction = regressor.predict(processed_input)  
print(f'Predicted IMDB Rating: {prediction[0]}')
```

```
➡ Mean Squared Error on test: 0.03237910359808707  
R² Score on test: 0.5276140625791984  
Mean Absolute Error on test: 0.14328102159500122  
MAPE: 1.80%  
Accuracy: 98.2 %  
Predicted IMDB Rating: 8.761387825012207
```

Obtivemos melhores resultados com nosso modelo de regressão treinado com o dataset de ganhadores do oscar. Tivemos um R² Score um pouco maior e um MSE e MAPE um pouco menor. A diferença foi pouca, mas vamos fazer alguns testes com filmes mais recentes que não estão no dataset e verificar qual modelo de aproximou mais do nosso alvo.

```
##### Modelo Usado #####
X = df_categorics_num_oscar.drop(['IMDB_Rating'], axis=1)
y = df_categorics_num_oscar['IMDB_Rating']

# Melhores parâmetros avaliados com grid search
params = {
    'objective': 'reg:squarederror',
    'subsample': 0.5,
    'reg_lambda': 1.0,
    'reg_alpha': 0.01,
    'n_estimators': 100,
    'max_depth': 9,
    'learning_rate': 0.05,
    'gamma': 0.1,
    'colsample_bytree': 1.0
}

# Dividir os dados em conjunto de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=43)

# Normalizar os dados
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Criar o modelo XGBRegressor
regressor = xgb.XGBRegressor(**params)

# Treinar o modelo
regressor.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = regressor.predict(X_test)

# Avaliar o desempenho do modelo
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
r2 = r2_score(y_test, y_pred)
print(f'R² Score: {r2}')
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')

errors = abs(y_pred - y_test)

# Calculando o MAPE
mape = 100 * (errors / y_test)
mean_mape = np.mean(mape)
print(f'MAPE: {mean_mape:.2f}%')

# Calculando a acurácia
accuracy = 100 - mean_mape
print('Accuracy:', round(accuracy, 2), '%')

# Lista fictícia de títulos de filmes que ganharam o Oscar (exemplo)
oscar_winner_titles = []

# Função para verificar se o diretor ou atores são vencedores do Oscar
def check_oscar_winners(director, stars):
    oscar_status = {
        'Director_Won_Oscar': 0,
        'Star1_Won_Oscar': 0,
        'Star2_Won_Oscar': 0,
        'Star3_Won_Oscar': 0,
        'Star4_Won_Oscar': 0
    }

    if director.lower() in oscar_winner_names:
        oscar_status['Director_Won_Oscar'] = 1
    for i, star in enumerate(stars):
        if star.lower() in oscar_winner_names:
            oscar_status[f'Star{i+1}_Won_Oscar'] = 1

    return oscar_status

# Função para fazer a previsão de um novo input
def preprocess_input(input_data):
    input_df = pd.DataFrame([input_data])

    # Converter para minúsculas
    input_df['Series_Title'] = input_df['Series_Title'].str.lower()
    input_df['Director'] = input_df['Director'].str.lower()
    input_df['Star1'] = input_df['Star1'].str.lower()
```

```

input_df['Star2'] = input_df['Star2'].str.lower()
input_df['Star3'] = input_df['Star3'].str.lower()
input_df['Star4'] = input_df['Star4'].str.lower()

# Extrair o valor numérico de Runtime e converter para int
input_df['Runtime'] = input_df['Runtime'].str.replace(' min', '').astype(int)

# Remover vírgulas de Gross e converter para float
input_df['Gross'] = input_df['Gross'].str.replace(',', '').astype(float)

# Verificar se o diretor e estrelas são vencedores do Oscar
oscar_status = check_oscar_winners(input_df['Director'].values[0],
                                   [input_df['Star1'].values[0], input_df['Star2'].values[0],
                                    input_df['Star3'].values[0], input_df['Star4'].values[0]])

print(oscar_status)
for key, value in oscar_status.items():
    input_df[key] = value

# Verificar se o título do filme está na lista de filmes que ganharam o Oscar
input_df['movie_oscar_winner'] = (input_df['Series_Title'].isin(oscar_winner_titles)).astype(int)
# Verificar se o Series_Title está na lista de títulos de filmes vencedores do Oscar
input_df['movie_oscar_winner'] = input_df['Series_Title'].isin(df_new_categorical_oscars.loc[df_new_categorical_oscars['movie_oscar_winner'] == 1]['Series_Title'])

# Adicionar a coluna previous_oscar_winner com base nos resultados da verificação
input_df['previous_oscar_winner'] = any(list(oscar_status.values()))

# Assegurar que todas as colunas necessárias estão presentes e preencher com 0 se não estiverem
for column in X.columns:
    if column not in input_df:
        input_df[column] = 0

input_df = input_df[X.columns]
print("Processed Input:")
print(input_df)
return scaler.transform(input_df)

# Novo input teste oscar
new_input = {
    'Series_Title': 'Guardiões da Galáxia Vol. 3',
    'Released_Year': '2023',
    'Certificate': 'PG-13',
    'Runtime': '150 min',
    'Genre': 'Action, Adventure, Comedy',
    'Overview': 'The Guardians of the Galaxy embark on one last mission to protect one of their own.',
    'Meta_score': 64.0,
    'Director': 'James Gunn',
    'Star1': 'Chris Pratt',
    'Star2': 'Zoe Saldana',
    'Star3': 'Dave Bautista',
    'Star4': 'Vin Diesel',
    'No_of_Votes': 388000,
    'Gross': '1500000000'
}

# Preprocessar o novo input
processed_input = preprocess_input(new_input)

# Fazer a previsão
prediction = regressor.predict(processed_input)
print(f'Predicted IMDB Rating: {prediction[0]}')

➡ Mean Squared Error: 0.03210785932190307
R² Score: 0.5315713055982045
Mean Absolute Error: 0.14130054235458378
MAPE: 1.77%
Accuracy: 98.23 %
{'Director_Won_Oscar': 0, 'Star1_Won_Oscar': 0, 'Star2_Won_Oscar': 0, 'Star3_Won_Oscar': 0, 'Star4_Won_Oscar': 0}
Processed Input:
   Oscars_Ganhos  Indicacoes_Oscar  movie_oscar_winner  previous_oscar_winner \
0              0              0              0              0              False

   Released_Year  Runtime  Meta_score  No_of_Votes  Gross
0          2023      150         64.0      388000  1.500000e+09
Predicted IMDB Rating: 7.836304664611816

```