Eduardo Darrazão - 1906399

Marcelo Guimarães da Costa - 1937570

Leandro Batista de Almeida - Professor

Big Data e Aplicações

06 de dezembro de 2023

Projeto 2: Desenvolvimento de um modelo de machine learning

Este projeto é a parte 2 de um projeto final de duas partes, sendo o objetivo final uma apresentação para a classe que envolveria todos os processos desde a seleção de um *dataset* até a utilização de modelos de *AutoML* sobre o *dataset* escolhido.

Nesta parte do projeto, os processos incluem:

1 - Geração de dataset para ML

- Feature engineering (criação de possíveis campos de interesse)
- Adaptação dos campos à necessidade dos potenciais algoritmos (se necessário)

2 - Seleção e execução de algoritmos

- Análise de algoritmos que possam resolver o problema estipulado
- Testar ao menos dois algoritmos disponíveis
- Treinamento dos modelos

3 - Comparação de modelos

- Avaliar a acurácia e outras métricas de cada modelo testado
- Comparar e justificar as conclusões obtidas

1 – GERAÇÃO DE DATASET PARA ML

Definimos que nosso objetivo é utilizar o *dataset* para a previsão de *averageRating* de um filme. Para tal, precisamos utilizar os dados disponíveis tanto como *features* para os modelos quanto processá-los para criar informações adicionais sobre as informações disponíveis.

A primeira *feature* criada é apenas uma booleana que é o resultado da comparação da igualdade entre os campos *primaryTitle* e *originalTitle*, que tem por objetivo indicar se o título mais popular é o original. Esta feature foi chamada de *popularIsOriginal*.

Depois trabalhamos com a criação de *'dummies'* da coluna multivariada *genres*, que se trata de transformar uma lista contendo todos os gêneros em várias colunas, onde cada uma representa uma das categorias, e cada entrada recebe o valor 1 caso tenha aquela categoria, 0 caso contrário. Segue um pedaço do código utilizado, e exemplos de saída.

```
# Dividir a coluna 'genres' por vírgulas e expandir em colunas
genres_split = title_basics_filtered.withColumn('genres', split('genres', ','))

# Usar a função explode() para criar múltiplas linhas para cada gênero
genres_exploded = genres_split.withColumn('genre', explode('genres'))

# Criar dummies para cada gênero usando pivot()
dummies = genres_exploded.groupBy('tconst').pivot('genre').agg(lit(1)).fillna(0)
```

```
df[df['tconst']=='tt0001184'][['genres','Action', 'Adult',
      'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Documentary',
      'Drama', 'Family', 'Fantasy', 'Film-Noir']]
```

✓ 0.8s Python

	genres	Action	Adult	Adventure	Animation	Biography	Comedy	Crime	Documentary	Drama	Family	Fantasy	Film-Noir
0	Adventure,Drama	0	0	1	0	0	0	0	0	1	0	0	0

Outra feature criada foi utilizando os dados disponíveis na tabela title.akas.csv, que possui informações referentes a tradução dos títulos para outros idiomas. Criamos uma *feature*

numérica que contém a quantidade de traduções para cada título, que pode ser interessante visto que, quanto mais traduzido, possivelmente é mais popular e portanto, mais bem avaliado.

Suponhamos que informações quantitativas como *averageRating* e *averageNumberOfVotes* dos atores participantes de um filme seriam relevantes para a popularidade do mesmo. Para calcular estes valores, selecionamos a média de *averageRating* e *numberOfVotes* de todos os filmes que cada ator participava, e fizemos uma média destes valores para os atores presentes em cada filme. De forma análoga, fizemos o mesmo para produtores (*producers)* e equipe (*crew*).

Por fim, com o dataset já limpo e organizado, inserimos eles no Orange. A primeira etapa aqui foi utilizar a ferramenta de discretização dos dados para discretizar nossa variável *target* (rating dos filmes), em dez 'baldes': 0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-9, 9-10, assim possivelmente melhorando o desempenho dos modelos subsequentes.

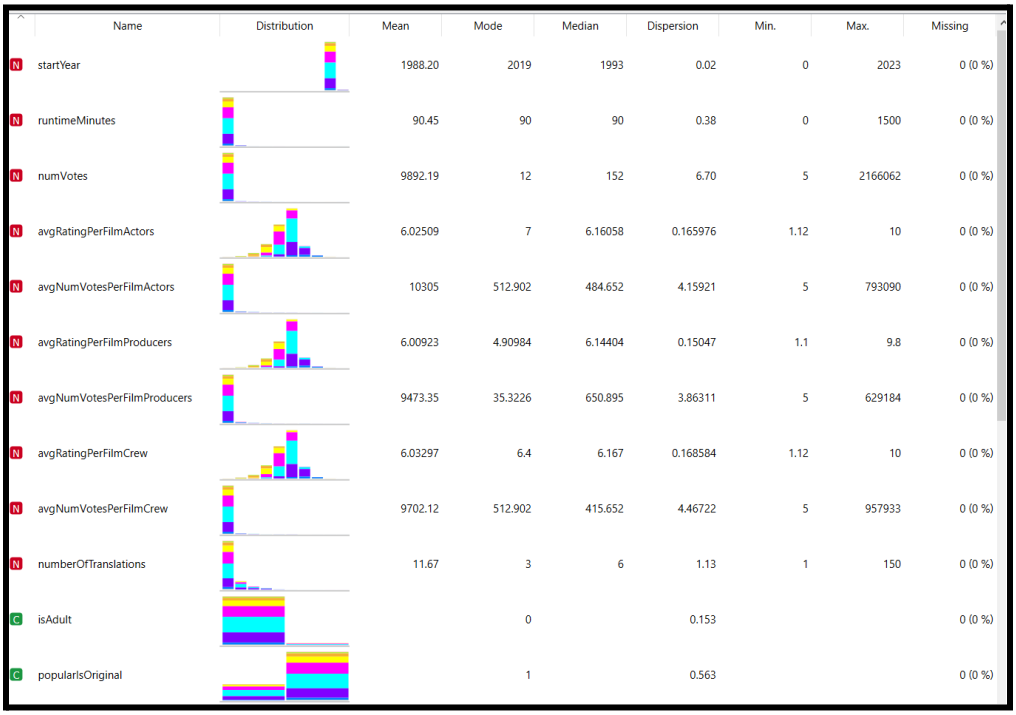
IMAGENS

Com o Orange também é possível ter vários *insights* sobre nossos dados, como alguns a seguir:

- Ranking das features baseado na relevância para a variável objetivo (rating):

		#	Gain ratio	Gini	χ^2
1	N avgRatingPerFilmProducers		0.244	0.095	10518.143
2	N avgRatingPerFilmActors		0.234	0.093	10218.334
3	N avgRatingPerFilmCrew		0.230	0.093	10087.660
4	C Talk-Show	2	0.196	0.000	25.015
5	C Reality-TV	2	0.191	0.000	49.192
6	C Horror	2	0.147	0.012	2871.940
7	C Documentary	2	0.141	0.015	2567.843
8	C News	2	0.063	0.000	52.998
9	C Biography	2	0.062	0.003	414.015
10	C Film-Noir	2	0.059	0.001	76.308
11	C Sci-Fi	2	0.052	0.002	531.487
12	C History	2	0.037	0.001	222.673
13	C isAdult	2	0.035	0.002	223.854
14	C Drama	2	0.034	0.006	611.022
15	C Adult	2	0.034	0.001	214.768
16	C Thriller	2	0.033	0.003	496.641
17	C Action	2	0.032	0.003	529.098
18	N numVotes		0.026	0.008	472.635
19	N runtimeMinutes		0.024	0.008	725.572
20	N startYear		0.023	0.006	1061.440
21	N numberOfTranslations		0.021	0.007	604.786

- Estatísticas das features, com a coloração baseado nos ‘baldes’ da variável objetivo:



	Name	Distribution	Mean	Mode	Median	Dispersion	Min.	Max.	Missing
N	startYear		1988.20	2019	1993	0.02	0	2023	0 (0 %)
N	runtimeMinutes		90.45	90	90	0.38	0	1500	0 (0 %)
N	numVotes		9892.19	12	152	6.70	5	2166062	0 (0 %)
N	avgRatingPerFilmActors		6.02509	7	6.16058	0.165976	1.12	10	0 (0 %)
N	avgNumVotesPerFilmActors		10305	512.902	484.652	4.15921	5	793090	0 (0 %)
N	avgRatingPerFilmProducers		6.00923	4.90984	6.14404	0.15047	1.1	9.8	0 (0 %)
N	avgNumVotesPerFilmProducers		9473.35	35.3226	650.895	3.86311	5	629184	0 (0 %)
N	avgRatingPerFilmCrew		6.03297	6.4	6.167	0.168584	1.12	10	0 (0 %)
N	avgNumVotesPerFilmCrew		9702.12	512.902	415.652	4.46722	5	957933	0 (0 %)
N	numberOfTranslations		11.67	3	6	1.13	1	150	0 (0 %)
C	isAdult			0		0.153			0 (0 %)
C	popularIsOriginal			1		0.563			0 (0 %)

- Correlações entre as features:

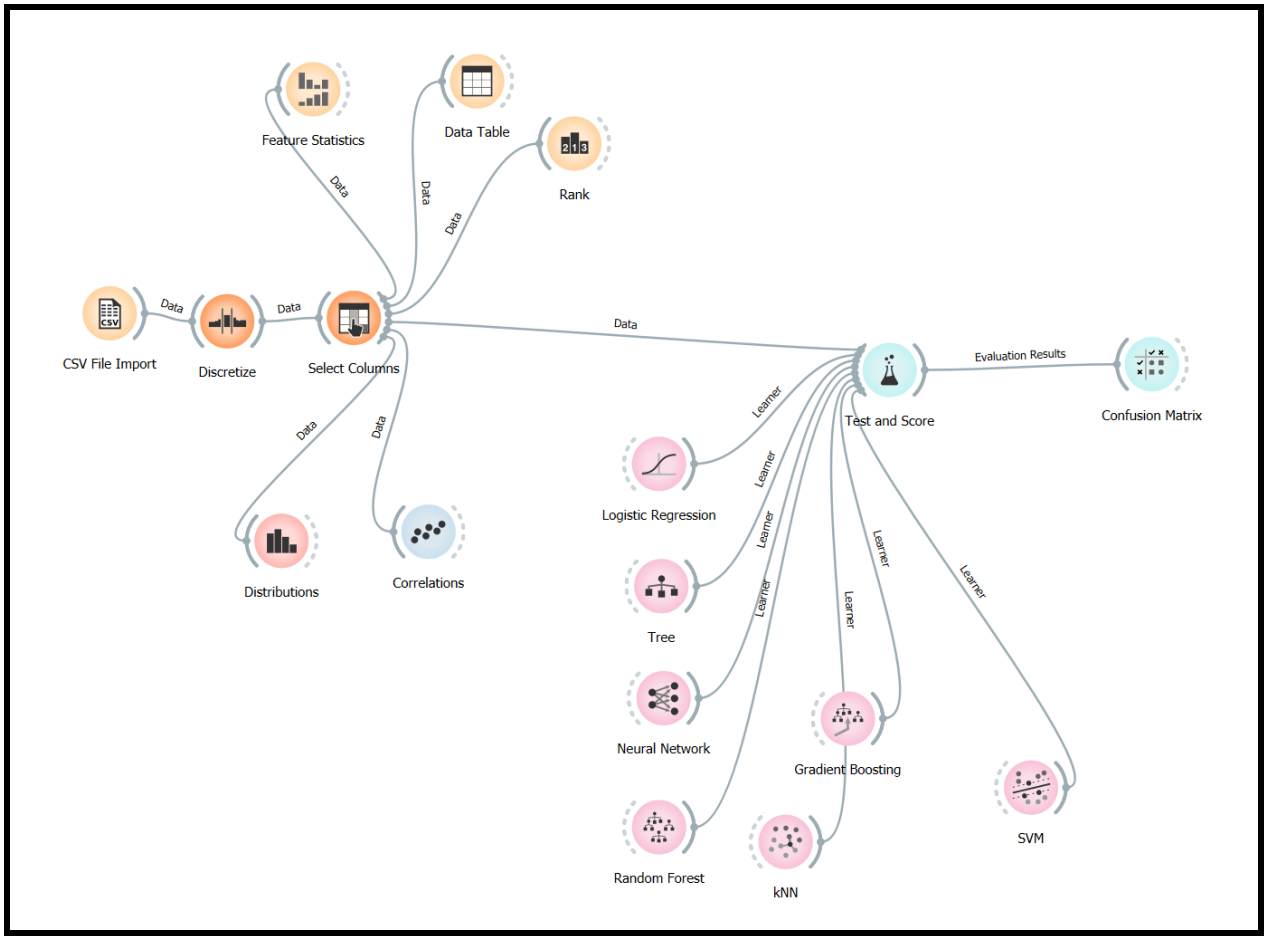
13	+0.521	numVotes	numberOfTranslations
14	+0.279	numberOfTranslations	runtimeMinutes
15	+0.145	avgNumVotesPerFilmProducers	runtimeMinutes
16	+0.145	numVotes	runtimeMinutes
17	+0.144	avgNumVotesPerFilmActors	avgRatingPerFilmActors
18	+0.142	avgNumVotesPerFilmActors	runtimeMinutes
19	+0.139	avgNumVotesPerFilmCrew	runtimeMinutes
20	+0.139	avgNumVotesPerFilmCrew	avgRatingPerFilmCrew
21	+0.130	avgNumVotesPerFilmCrew	avgRatingPerFilmActors
22	+0.130	avgNumVotesPerFilmProducers	avgRatingPerFilmActors
23	+0.129	avgNumVotesPerFilmProducers	avgRatingPerFilmProducers
24	+0.126	avgNumVotesPerFilmActors	avgRatingPerFilmCrew
25	+0.124	avgRatingPerFilmActors	numberOfTranslations
26	+0.122	avgNumVotesPerFilmProducers	avgRatingPerFilmCrew
27	+0.121	avgNumVotesPerFilmActors	avgRatingPerFilmProducers
28	+0.118	avgRatingPerFilmCrew	numberOfTranslations
29	+0.118	avgNumVotesPerFilmCrew	avgRatingPerFilmProducers
30	+0.117	avgRatingPerFilmCrew	runtimeMinutes
31	+0.116	avgRatingPerFilmActors	runtimeMinutes
32	+0.113	avgRatingPerFilmProducers	runtimeMinutes
33	+0.109	avgRatingPerFilmProducers	numberOfTranslations
34	+0.096	avgRatingPerFilmActors	numVotes
35	+0.094	avgRatingPerFilmCrew	numVotes

2 – SELEÇÃO E EXECUÇÃO DE ALGORITMOS

Nesta etapa optamos por utilizar a ferramenta Orange Data Mining[1], que é uma ferramenta de código aberto para aprendizado de máquina e visualização de dados. Com os dados tratados e trabalhados nas seções anteriores, podemos importá-los para a ferramenta e utilizá-los para criar modelos de classificação para os filmes.

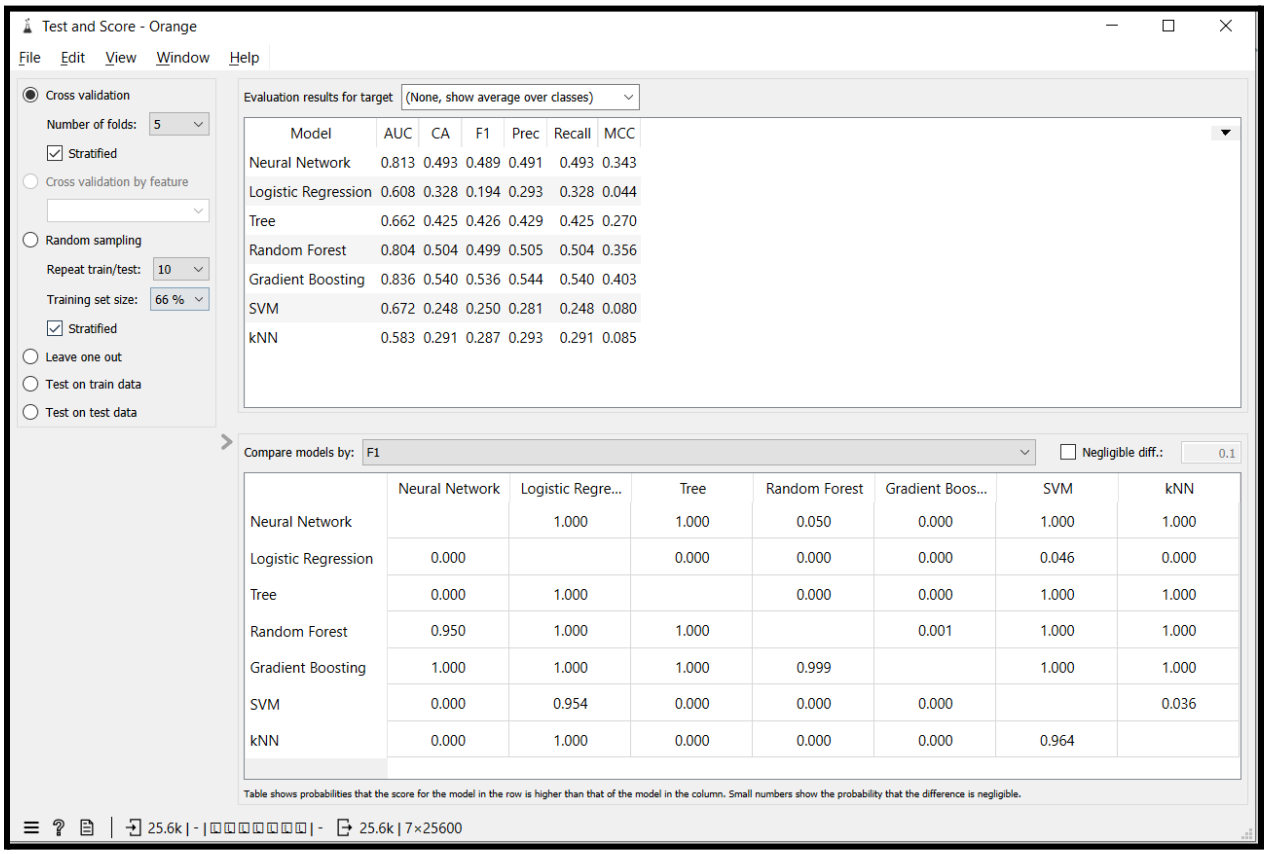
Como o problema escolhido foi o de classificação, selecionamos algoritmos conhecidos para essa tarefa, cada um com suas particularidades. Escolhemos dois algoritmos de aproximação de curvas, sendo regressão logística e *Support Vector Machine* (SVM), três baseados em árvores, sendo árvore de decisão comum, floresta aleatória e *gradient boosting*. Foi utilizado também um algoritmo de agrupamento, o k-Nearest Neighbour (kNN) e uma rede neural artificial. A escolha dos algoritmos se deu por dois motivos, primeiro que é os disponíveis na ferramenta e que são adequados para a tarefa, e segundo que é possível comparar os algoritmos com abordagens similares com algoritmos com diferentes abordagens. Segue uma imagem do *workflow* criado:

[1] https://orangedatamining.com



3 – COMPARAÇÃO DE MODELOS

Com os modelos treinados, foi possível contratar os respectivos resultados. Vale citar que todos os modelos foram treinados com validação cruzada de 5 ‘dobras’. A primeira observação é que nenhum dos modelos obteve resultados muito bons, sugerindo que os dados e features disponíveis não são suficientes para compreender o que leva um filme ser ou não bem avaliado. Como visto no ranking das features, as que conseguem descrever melhor o problema são as médias dos filmes dos atores e equipe no geral, dando a entender que pessoas famosas e que fazem filmes de sucesso, tendem a repetir o feito.



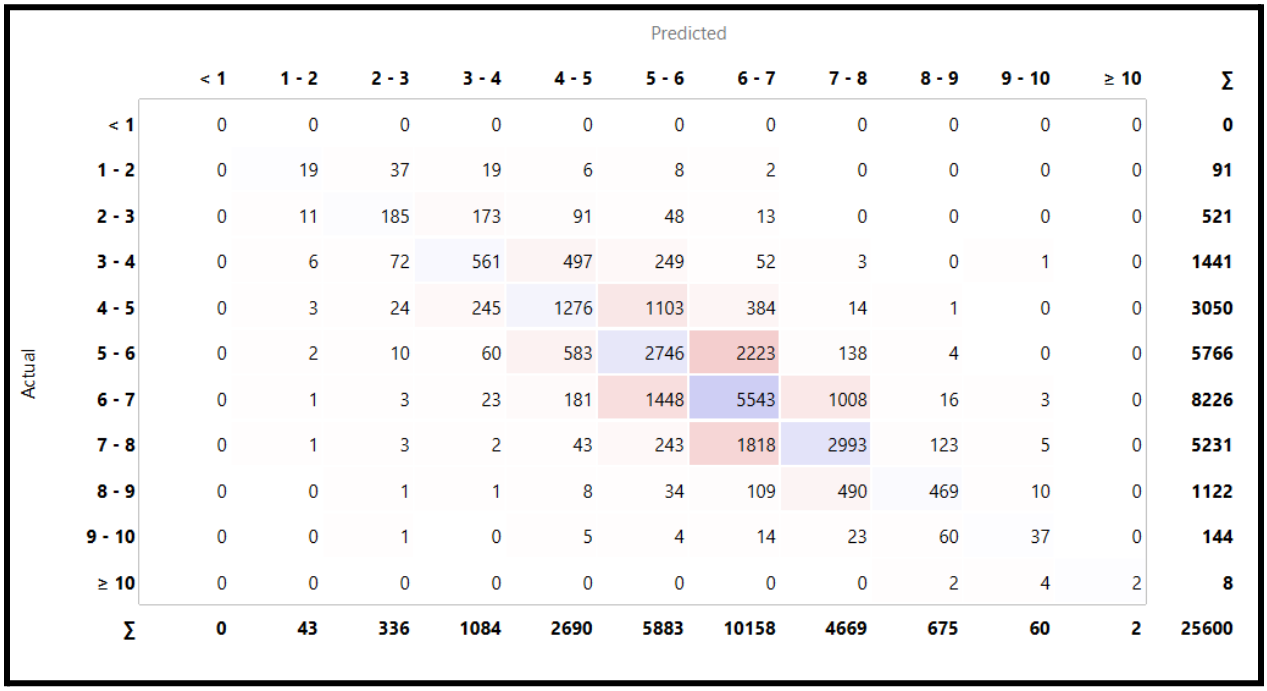
A métrica principal comparada dos modelos foi o F1-Score, que é uma métrica que leva em consideração tanto a acurácia quanto o *recall*. Sobre o desempenho dos modelos em si, o melhor de todos foi o *Gradient Boosting*, que é baseado em árvores. É interessante ver que os três modelos baseados em árvores estiveram entre os melhores resultados, e o desempenho deles segue o nível de complexidade. O modelo mais básico de árvore, teve o menor dos resultados entre os três, enquanto o mais complexo, teve o melhor.

Juntamente com os modelos de árvores, a rede neural também obteve resultados similares. Os algoritmos baseados em aproximações de curva (regressão logística e SVM) e o baseado em agrupamento (kNN) tiveram resultados bem ruins comparados aos demais, obtendo menos que a metade do F1-Score. Podemos inferir então que, os dados não seguem uma

tendência similar (portanto trazendo péssimos resultados nas aproximações de curva) e não tendem a agrupar entre si.

Por fim, temos a matriz de confusão do melhor modelo, onde no eixo X temos os previstos e no Y o valor real.

- Gradient Boosting:



Actual \ Predicted	< 1	1 - 2	2 - 3	3 - 4	4 - 5	5 - 6	6 - 7	7 - 8	8 - 9	9 - 10	≥ 10	Σ
< 1	0	0	0	0	0	0	0	0	0	0	0	**0**
1 - 2	0	19	37	19	6	8	2	0	0	0	0	**91**
2 - 3	0	11	185	173	91	48	13	0	0	0	0	**521**
3 - 4	0	6	72	561	497	249	52	3	0	1	0	**1441**
4 - 5	0	3	24	245	1276	1103	384	14	1	0	0	**3050**
5 - 6	0	2	10	60	583	2746	2223	138	4	0	0	**5766**
6 - 7	0	1	3	23	181	1448	5543	1008	16	3	0	**8226**
7 - 8	0	1	3	2	43	243	1818	2993	123	5	0	**5231**
8 - 9	0	0	1	1	8	34	109	490	469	10	0	**1122**
9 - 10	0	0	1	0	5	4	14	23	60	37	0	**144**
≥ 10	0	0	0	0	0	0	0	0	2	4	2	**8**
Σ	**0**	**43**	**336**	**1084**	**2690**	**5883**	**10158**	**4669**	**675**	**60**	**2**	**25600**

Todo o código está disponível em: https://github.com/eduponto21/IMDB-BigData-Spark .