INF1038: Relatório do T1
Grupo 2: Mariana Barreto, Thiago Levis, Paulo de Tarso

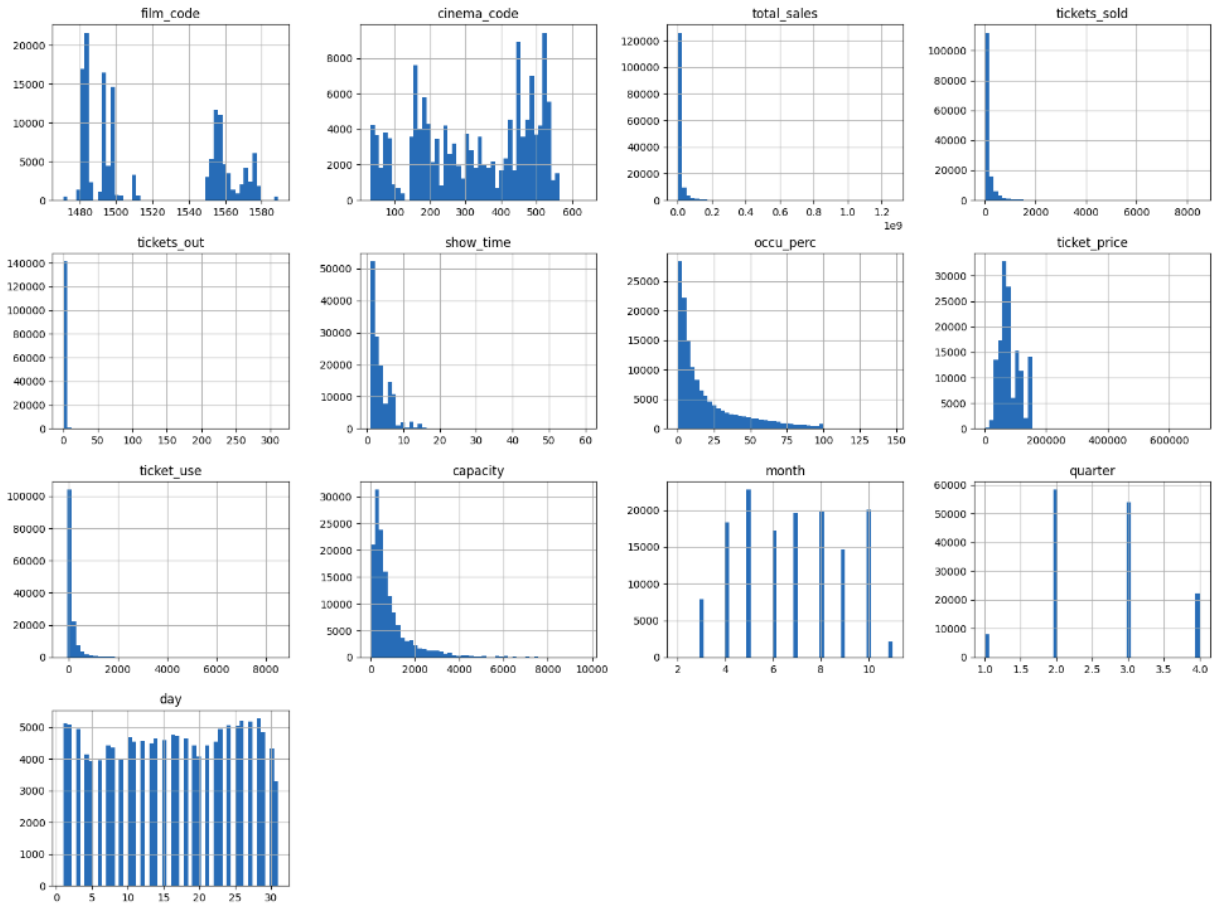
1. Regressão (Cinema Ticket Dataset)

Para a análise de regressão, estamos utilizando um dataset sobre venda de ingressos de cinema. Na tabela abaixo, mostramos todas as colunas com algumas informações sobre cada. No total, são 142418 linhas. O objetivo é tentar prever o preço total de vendas com base em outras variáveis.

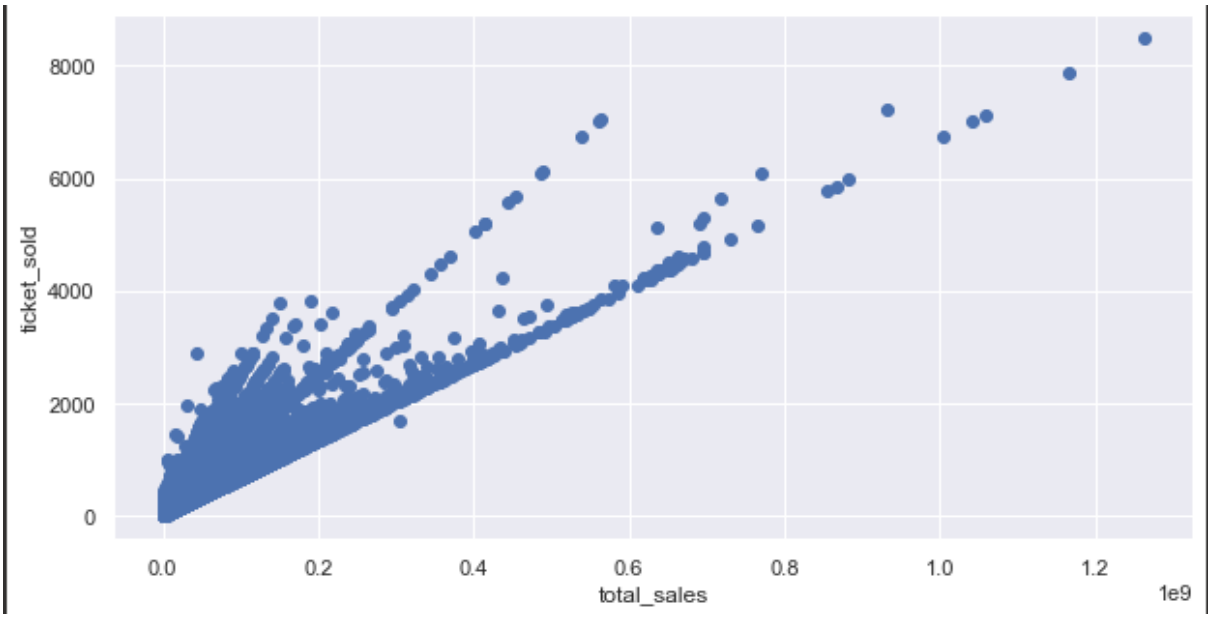
	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use
count	142524.000000	142524.000000	142524.000000	142524.000000	142524.000000	142524.000000	142399.000000	142524.000000	142524.000000
mean	1518.985111	320.378427	12347275.407531	140.137570	0.237413	3.932103	19.965986	81234.599886	139.900157
std	36.184450	159.701229	30654858.289367	279.758733	2.923206	3.056276	22.653445	33236.599278	279.564935
min	1471.000000	32.000000	20000.000000	1.000000	0.000000	1.000000	0.000000	483.870968	-219.000000
25%	1485.000000	181.000000	1260000.000000	18.000000	0.000000	2.000000	3.750000	60000.000000	18.000000
50%	1498.000000	324.000000	3720000.000000	50.000000	0.000000	3.000000	10.350000	79454.235185	50.000000
75%	1556.000000	474.000000	11100000.000000	143.000000	0.000000	5.000000	28.210000	100000.000000	143.000000
max	1589.000000	637.000000	1262819994.000000	8499.000000	311.000000	60.000000	147.500000	700000.000000	8499.000000

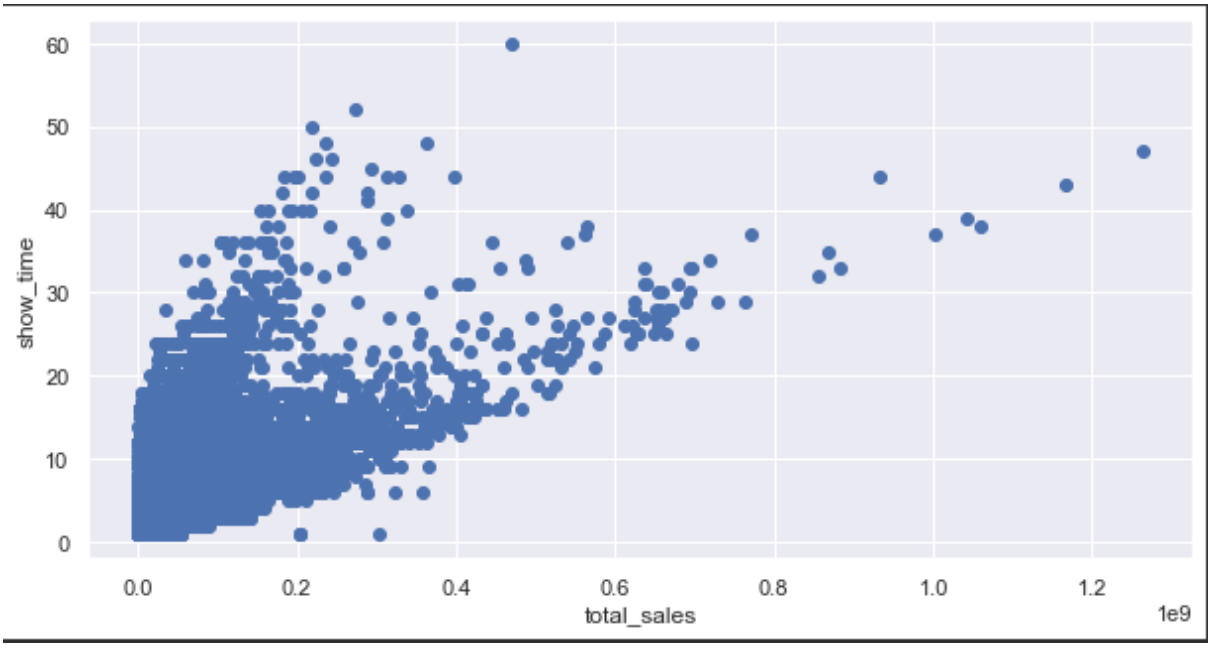
capacity	month	quarter	day
142399.000000	142524.000000	142524.000000	142524.000000
854.723605	6.776852	2.634721	16.112585
953.118103	2.195843	0.809692	8.949471
-2.000000	2.000000	1.000000	1.000000
276.994486	5.000000	2.000000	8.000000
525.714286	7.000000	3.000000	16.000000
1038.961039	9.000000	3.000000	24.000000
9692.097160	11.000000	4.000000	31.000000

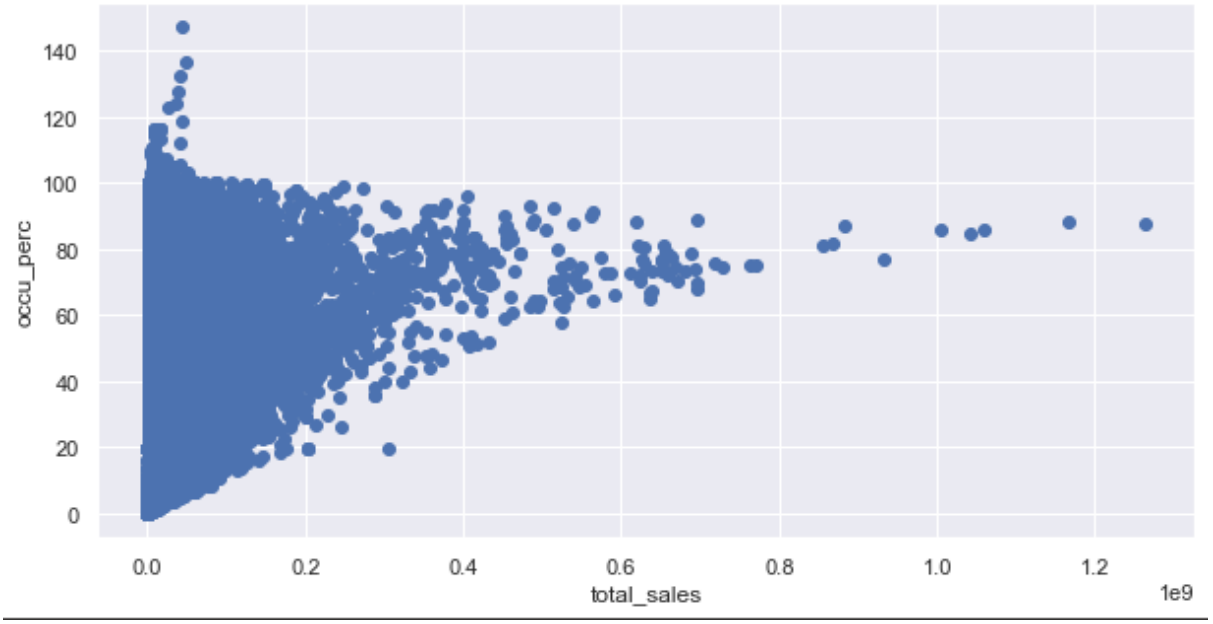
Aqui podemos observar que existem dados faltando em “occu_perc” e em “capacity”. Além disso, temos valores zerados que precisam ser investigados e existem valores negativos que precisam ser removidos em “ticket_use” e “capacity”. Procurando por células vazias, nós percebemos que eram 0.013% do dataset. Para tratar os dados, resolvemos preencher as células vazias com o valor médio de cada coluna. Além disso, encontramos 106 tuplas duplicadas e removemos elas. Com o histograma abaixo, destacamos a grande variação de escala e distribuição dos dados.

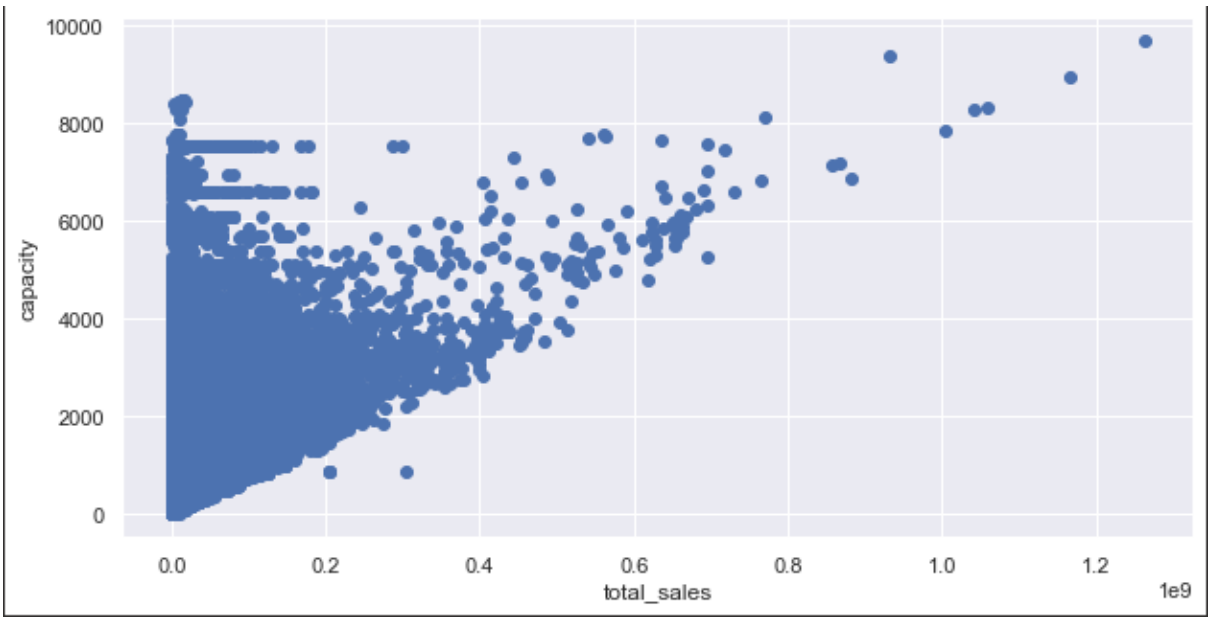


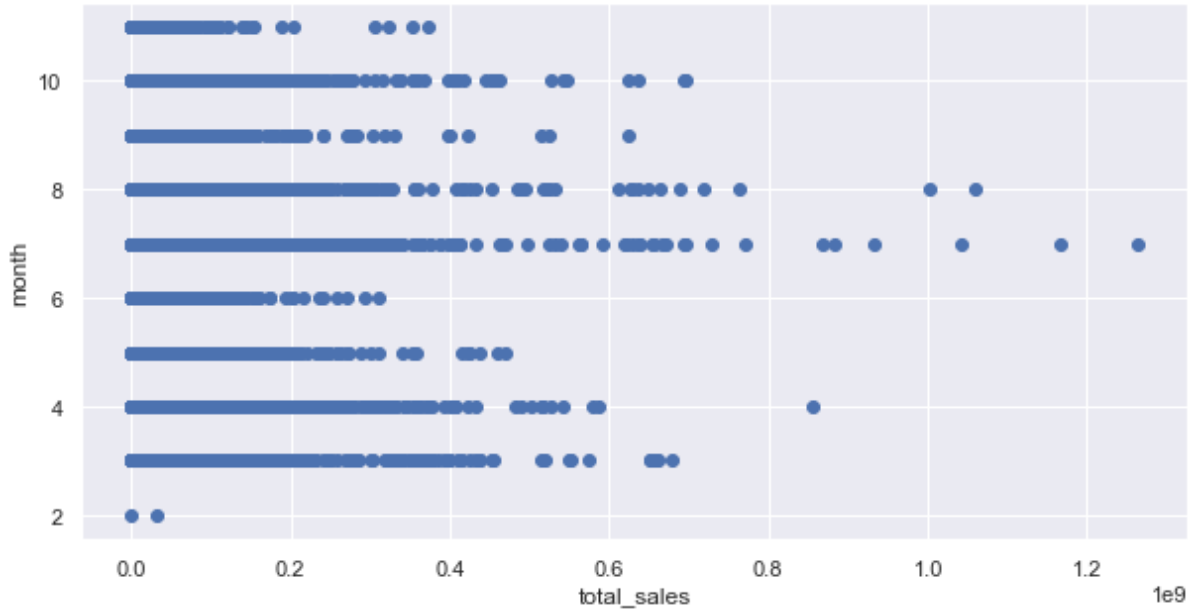
Para conhecer melhor a relação do total sales com as features traçamos alguns gráficos para conhecer sua associação com outras variáveis.




show_time
total_sales
1e9
0
10
20
30
40
50
60
0.0
0.2
0.4
0.6
0.8
1.0
1.2

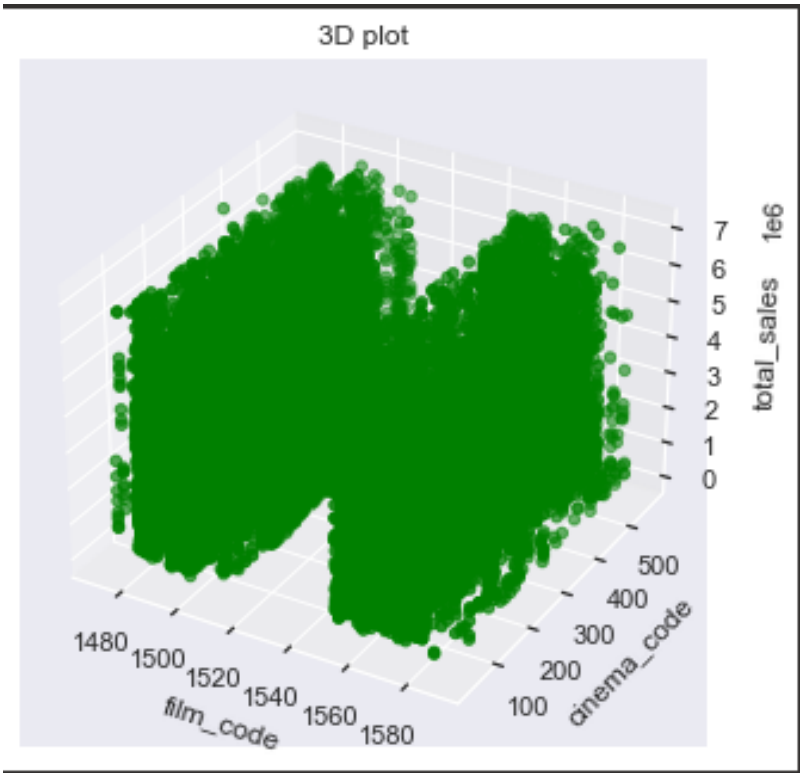

occu_perc
total_sales
1e9
0
20
40
60
80
100
120
140
0.0
0.2
0.4
0.6
0.8
1.0
1.2

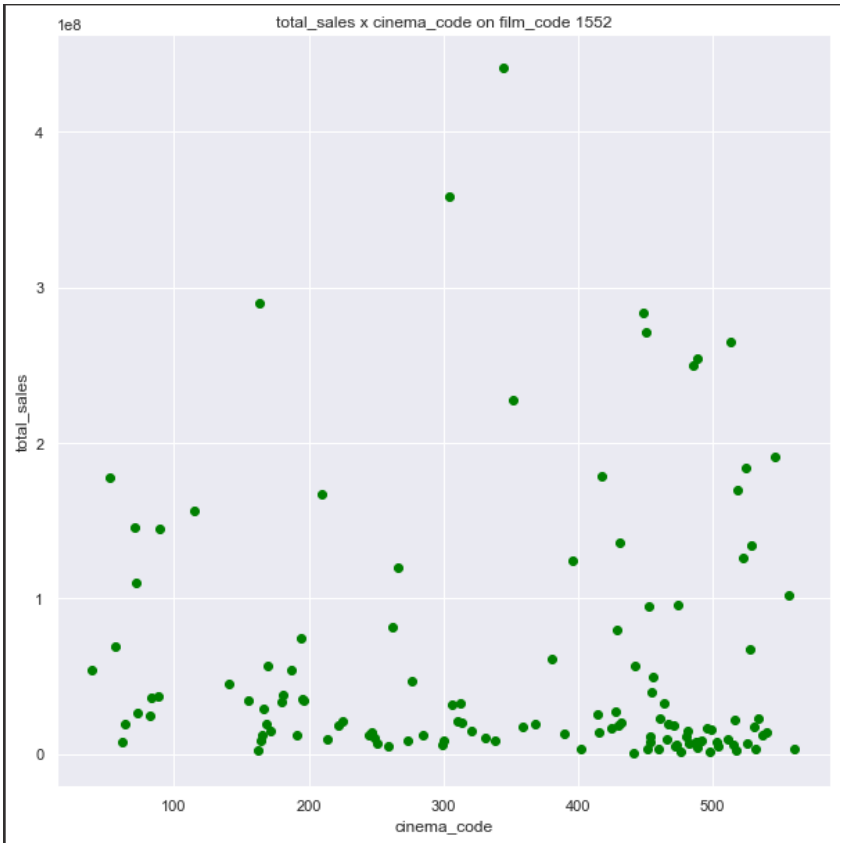




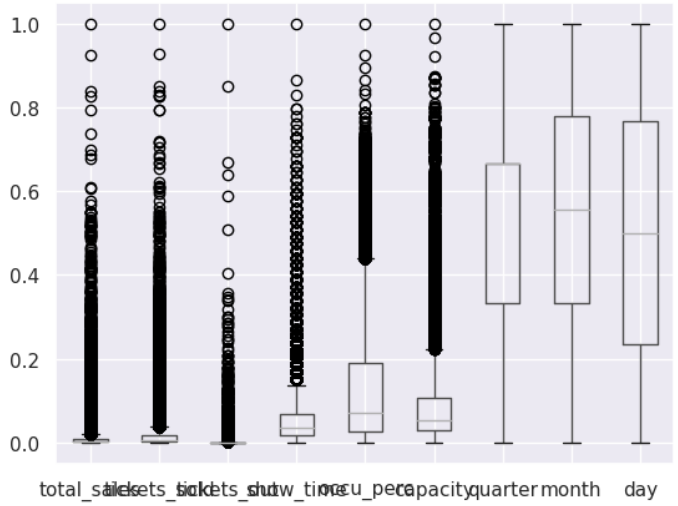
Podemos observar que em algumas features ele relembra uma certa relação com linearidade.

Tentamos ainda analisar se existia algum cinema com maior sucesso que os outros, logo geramos um gráfico 3D que demonstrasse o total de vendas de um certo filme, que foi passado em um cinema, porém com a utilização de outliers e a dificuldade de fazer um gráfico interativo ficou complicado de fazer uma análise em cima dele. Portanto foi feito um gráfico 2D em cima de um único filme, onde conseguimos perceber que certos cinemas apresentam maior sucesso em relação a outros.

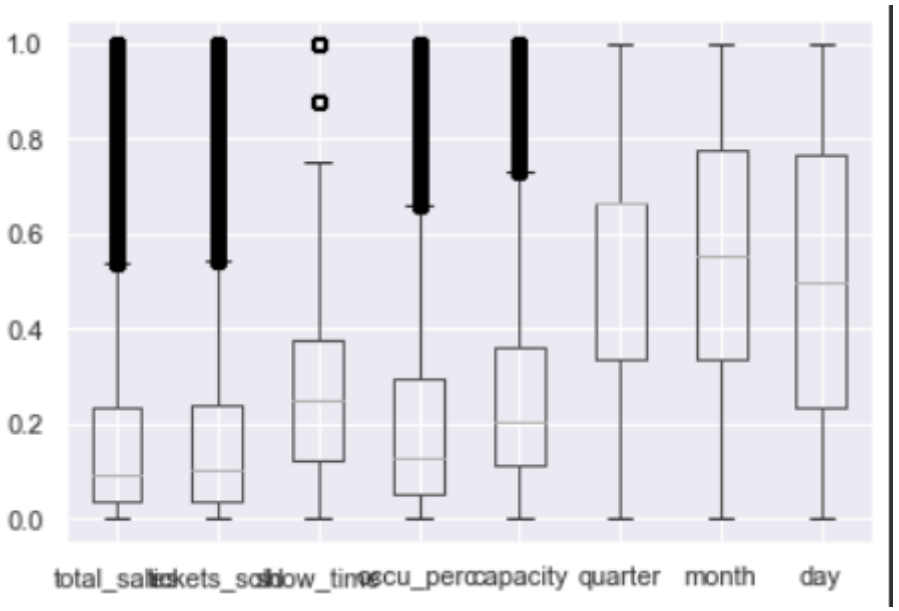




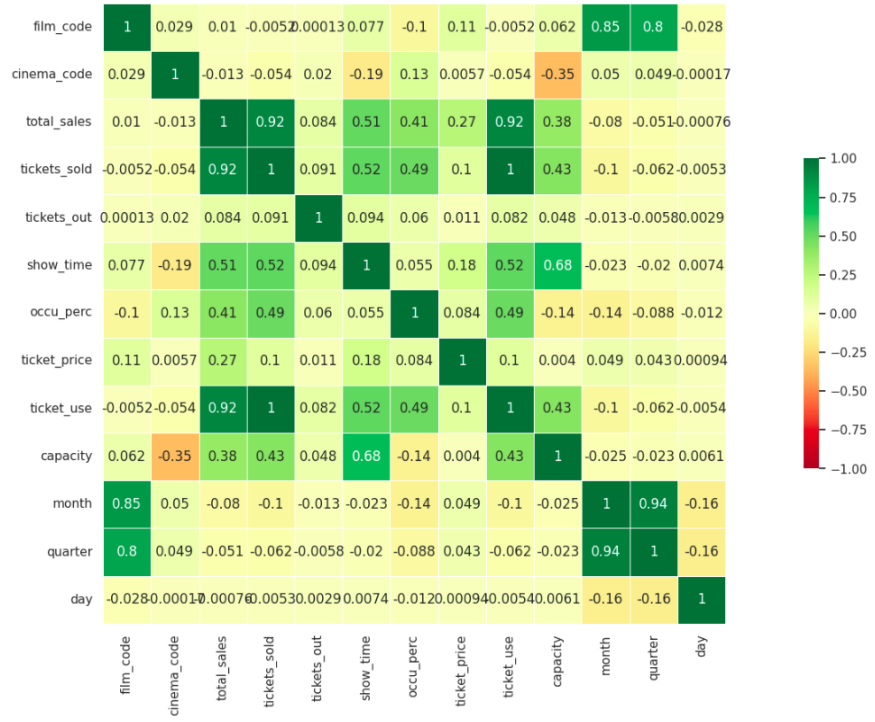
Devido a grande variação de escalas, se viu necessário normalizar os dados e fizemos um boxplot de cada coluna. Dá para perceber uma grande quantidade de outliers na maioria das categorias. Especialmente no total_sales, que é nosso target. Esses valores extremos certamente podem afetar as análises. Também vale destacar que a grande maioria dos valores do “ticket_out” são 0, então vamos removê-lo.



Após remover outliers, conforme indicado na figura abaixo, ficamos com um total de 55128 linhas.



Em seguida, fizemos uma matriz de correlação para descobrir a conexão entre as variáveis.



Como resultado, percebemos que “ticket_use” e “tickets_sold” possuem relação linear exata. “tickets_sold” e “total_sales” ou “total_sales” e “ticket_use” possuem também forte correlação (92%). Para determinar as colunas que usamos como features, fizemos testes com todas menos “ticket_use” e “tickets_sold”, que descartamos pela multicolinearidade. A combinação atual foi a que deu melhores resultados para a regressão polinomial de grau 2.

Utilizando o modelo Gaussiano, observamos que 68%[1] do “total_sales” fica de 0.05 a 0.21. Antes de normalizar os dados, estava de -18301689.16 a 42973469.84.

Para fazer a análise do modelo utilizamos as seguintes métricas como parâmetros principais:

Mean squared error, Mean absolute error, Root mean squared error, e R2 score.

O MSE mensura a quantidade de erros de um modelo e representa a distância Euclidiana da linha de regressão com os pontos dos dados. Quando seu valor é 0, significa que o modelo não tem erros. O MAE e o RMSE também são melhores quando menores e representam a magnitude média de erros das previsões, desconsiderando a orientação dos erros. Contudo, o RMSE é mais interessante de usar quando desejamos controlar erros maiores e o MAE quando não faz muita diferença o nível do erro. MAE usa distância de Manhattan. O R2 score, nosso coeficiente de determinação, é um valor que fica entre 0 e 1, e quanto mais próximo do 1, melhor a performance do modelo. Significa a proporção de variação do target que nossas features, as variáveis independentes, podem justificar. Outro dado que surge na regressão é o Intercept, o valor médio da variável dependente, o target, quando as independentes são 0.

Regressão linear

Para treinamento, utilizamos 70% dos dados e o resto para teste. Ao aplicar regressão linear tivemos os seguintes resultados:

Score = 0.614.

MSE = 0.015

MEA = 0.085

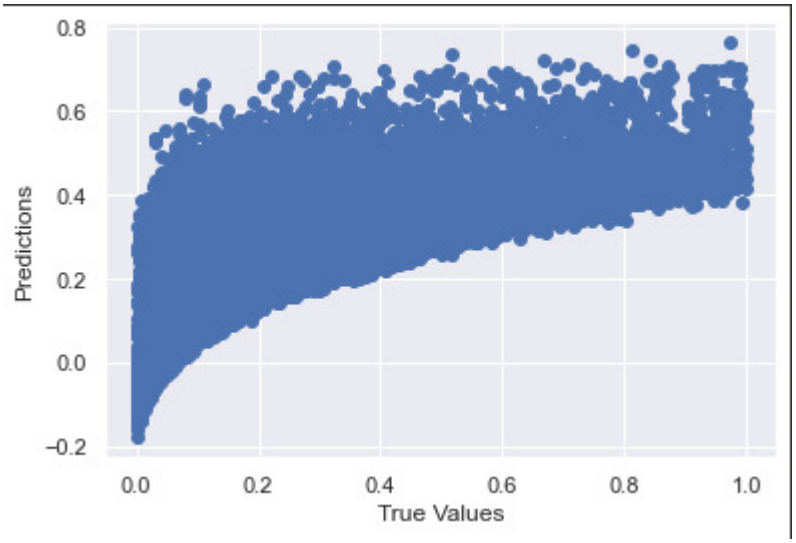
[1] https://www.inf.ufsc.br/~andre.zibetti/probabilidade/figures/normal.PNG

RMSE = 0.125

R2 score = 0.614

Intercept = -0.206

Foi feito ainda um cross_val_score de NMSE onde tivemos como média = -0.015, e desvio padrão = 0.0007. De uma forma geral podemos observar que as métricas desse modelo foram ruins, e que ele erra bastante como demonstra o gráfico abaixo.



Lasso

O Lasso (ou least absolute shrinkage and selection operator) Regression é usado para restringir a regressão linear usando norma e potencialmente obter mais coeficientes nulos, Na sua aplicação, usando CV de valor 7, o score deu -0.020, o que significa que esse modelo não deu muito certo.

Ridge

O Ridge Regression (ou regularização de Tikhonov) é um método para aperfeiçoar o modelo, analisando os dados com alta correlação. Ao aplicarmos, conseguimos

lr mean score: 0.600

rg mean score: 0.597

rg alpha: 0.00025

O valor do CV foi 7 também. Os valores de score estão melhores que o Lasso, provavelmente devido ao fato dele lidar melhor com dados com alta relação linear. O alpha é a penalidade de retração adicional que vai ser implementada na equação e podemos ver que seu valor está baixo.

ElasticNet

Usando dois fatores de penalidade, cada um proporcional a norma utilizada no Lasso e no Ridge, o ElasticNet junta algumas qualidades dos dois e evita a exclusão seletiva de variáveis com alta relação linear. Os resultados nesse caso foram:

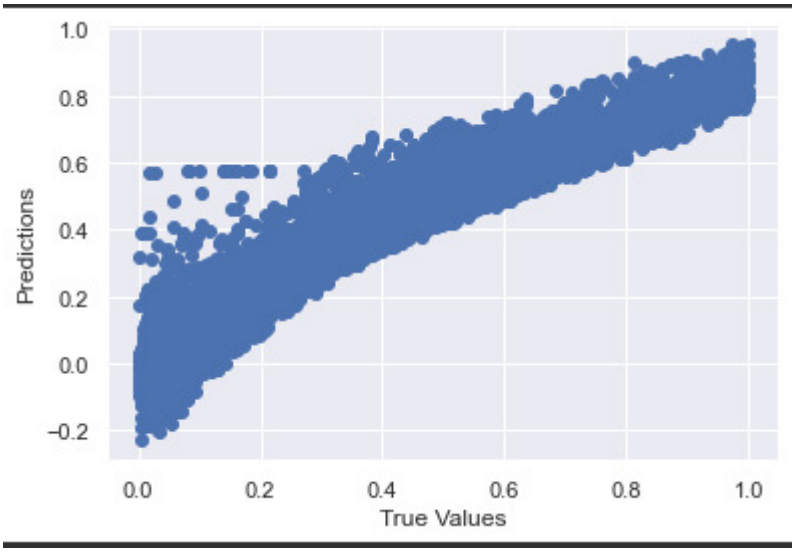
ElasticNet mean score: -0.024
encv.alpha_: 0.001
encv.l1_ratio: 0.1

O valor do CV foi 10. Os valores do encv alpha e do beta (l1_ratio) foram baixos, o que contribuiu pro valor negativo do mean score. É curioso notar como esse score é parecido com o do Lasso, o que deve indicar maior influência do que o Ridge.

Polinomial 2 Grau

Ao utilizar grau 2, obtivemos os seguintes resultados:

score: 0.95
MSE: 0.001891
MAE: 0.023715
RMSE: 0.043483
R2 Score: 0.952761



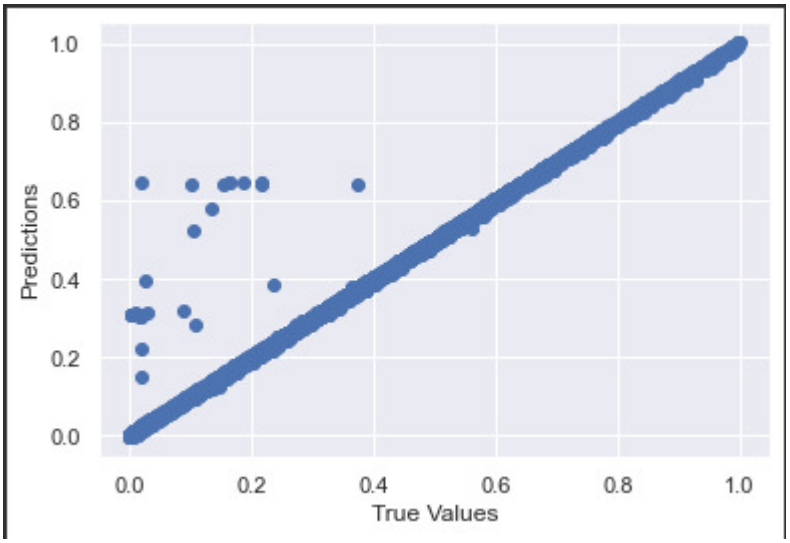
O score está parecendo adequado, com um valor alto, mas não exagerado a ponto de ser overfitting. Com o gráfico, vemos outliers, mas não estão tão distantes. O MSE, MAE e RMSE estão baixos, o que demonstra boa precisão das previsões. E o R2 Score está próximo de 1, indicando boa performance do modelo.

Polinomial 3 Grau

Ao utilizar grau 3, chegamos a

score: 0.997
MSE: 0.000116

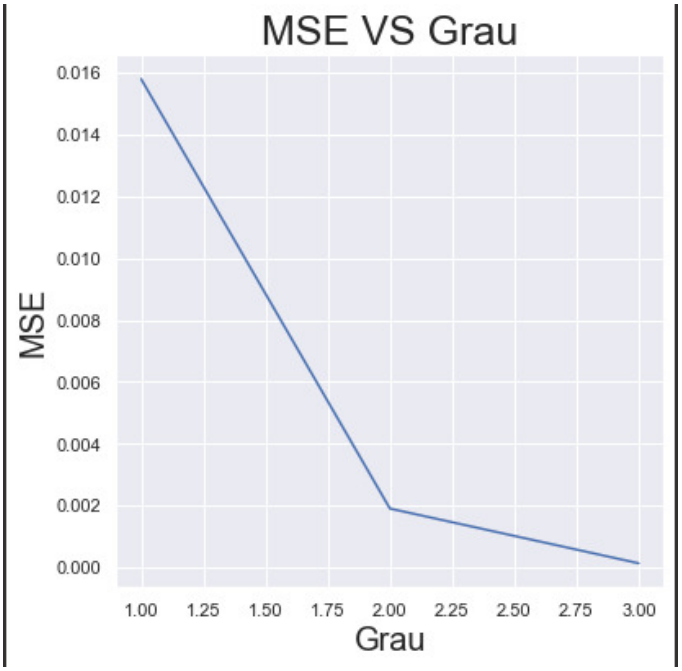
MAE: 0.001814
RMSE: 0.010755
R2 Score: 0.997117



Ao observar as informações acima podemos observar que é um caso de overfitting, onde o modelo apresenta grande ajuste para valores do modelo apresentado, porém se mostra ineficaz para valores novos.

Conclusão

De uma forma geral podemos observar que o modelo que apresentou as melhores métricas foi o polinomial de 2 grau, as métricas achadas no 3 grau, assim como a visualização do gráfico demonstram um caso de overfitting. E os outros modelos não chegaram aos valores ideais, abaixo está uma relação do MSE com o grau do modelo



2. Classificação (Youtube Video Dataset)

Para a análise de classificação, estamos utilizando um dataset sobre vídeos no YouTube. Na tabela abaixo, mostramos 5 tuplas com todas as colunas. No total, são 11211 linhas. O objetivo é tentar prever a categoria dos vídeos com base no seu título e descrição.

	Title	Videourl	Category	Description
0	Madagascar Street Food!!! Super RARE Malagasy ...	/watch?v=EwBA1fOQ96c	Food	🎥GIANT ALIEN SNAIL IN JAPAN! » https://youtu.b...
1	42 Foods You Need To Eat Before You Die	/watch?v=0SPwwpruGIA	Food	This is the ultimate must-try food bucket list...
2	Gordon Ramsay's Top 5 Indian Dishes	/watch?v=upfu5nQB2ks	Food	We found 5 of the best and most interesting In...
3	How To Use Chopsticks - In About A Minute 🍜	/watch?v=xFRzzSF_6gk	Food	You're most likely sitting in a restaurant wit...
4	Trying Indian Food 1st Time!	/watch?v=K79bXtaRwcM	Food	HELP SUPPORT SINSTV!! Shop Our Sponsors!\r\nLa...

Pré-processamento de dados

A primeira questão foi tratar o dado de uma forma geral, retirando colunas que não são interessantes para a análise e retirando linhas com nulo. A coluna nesse caso que não foi de interesse é a URL do vídeo. Isso porque a abordagem utilizada será em cima de texto e a URL é única para cada vídeo. Os valores nulos também não são interessantes, pois podem gerar ruído desnecessário. Isso reduziu um pouco o tamanho do dataframe, perdendo uma coluna e X linhas.

A segunda questão foi como tratar o texto, tanto da Descrição quanto do Título. Existem diversas técnicas para processar o texto, como remoção de stopwords, lemmatization e stemming. Um detalhe importante que foi observado é que nem todas as descrições e títulos estão em inglês. Palavras em idiomas diferentes, como hindi, foram observadas, porém o NLTK não possui stopwords para grande parte delas. Antes de passar pelo processo de remoção, todas as letras maiúsculas foram substituídas por minúsculas, as URLs foram removidas[2] assim como os e-mails[3]. Há também uma discussão entre lemmatization e stemming, normalmente indicando qual das duas utilizar. No entanto, foi observado que as duas podem ser utilizadas em conjunto e que isso pode fazer com que o número de features diminua. Para o trabalho, era almejado que o máximo de técnicas para preparar o dado fossem utilizadas. Isso porque nesse caso é interessante reduzir o número de features, para que palavras similares que poderiam ser generalizadas como uma só não fossem tratadas como features diferentes.

Escolha do modelo de feature extraction

[2] https://regex101.com/r/hG9t0Q/1
[3] https://stackoverflow.com/questions/42407785/regex-extract-email-from-strings

Essa foi uma dúvida que o grupo se encontrou, porque existem várias alternativas e é difícil mudar de ideia depois de executar todo o restante do código. De início, foi considerado utilizar o Bag of Words. Houve uma tentativa de fazer o Bag of Words do zero, dizendo para cada linha quais palavras do vocabulário estavam presentes no combinado do Título + Descrição. No entanto, isso foi logo descartado porque o custo computacional de montar tudo do zero era alto. Foi pensada então em utilizar o CountVectorizer, versão do BoW disponibilizada pelo SKLearn. Por mais que ela tenha sido melhor em termos de desempenho, durante a pesquisa surgiu a ideia do TF-IDF, que possui a mesma ideia do Bag of Words mas ao invés de colocar zeros ou uns, é feito um cálculo que penaliza palavras muito frequentes ou palavras muito raras. Ele assume que essas palavras não são estatisticamente importantes para encontrar um padrão.[4]

Foram consideradas outras abordagens também, como BERT ou Word2Vec. Mas os dois casos não foram considerados porque não só seria um maior custo computacional como também seria um problema para juntar as informações de Título e Descrição, já que nesse caso a posição das palavras no texto importa. Para o contexto de classificar as categorias, foi entendido que as duas colunas têm textos muito similares e que a sua junção poderia ser encarada como uma possibilidade de detectar quais palavras são menos importantes.

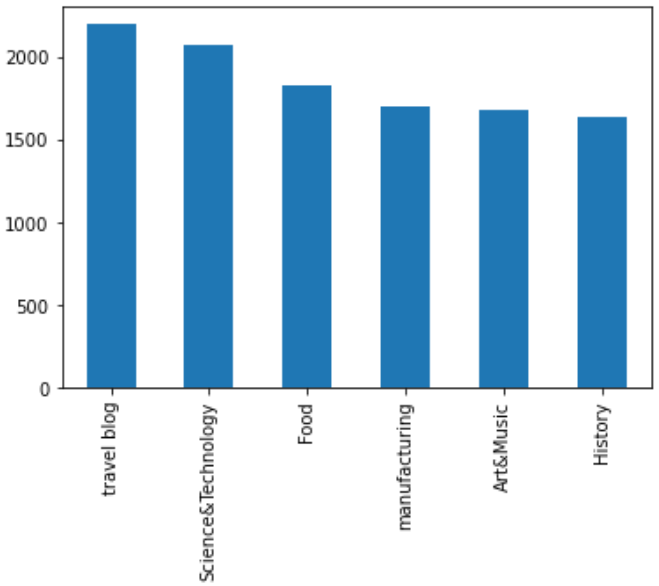
Separação entre dados de treinamento e de teste

Nessa etapa, uma das ideias era usar K-Fold Cross Validation pois esse método traz uma segurança maior em relação às métricas. No entanto, como se trata de um problema de classificação, foi utilizado o StratifiedKFold, que é uma versão do K-Fold com o intuito de garantir que a porcentagem de cada classe nos dados de treinamento e de teste estejam equilibrados. Não só isso como o shuffle foi deixado com o True para que as amostras por classe sejam embaralhadas antes de serem separadas nos lotes[5]. Isso foi feito apenas para garantir que existisse um equilíbrio na distribuição de classes, mas pela frequência apresentada para cada classe no dataset, não foi observado um grande desequilíbrio entre elas.

---

4 https://www.linkedin.com/pulse/count-vectorizers-vs-tfidf-natural-language-processing-sheel-saket#:~:text=TF%2DIDF%20is%20better%20than,by%20reducing%20the%20input%20dimensions

5 https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html



Um detalhe interessante que ocorreu durante o trabalho foi a detecção de um possível Data Leakage. Isso porque o vectorizer do TF-IDF era feito antes da separação entre dados de treinamento e de teste. Dessa forma, os dados de treinamento compartilhavam do mesmo vocabulário que os de teste. Por isso, a função cross_validate que era utilizada antes para verificar as métricas dos modelos teve que ser resubstituída por uma análoga (cross_val_tfidf) que faz um TF-IDF para os dados de treinamento dentro de um fold e outro para os de teste[6].

Avaliação dos diferentes modelos (*model selection*)

Para verificar qual dos modelos obteve melhor performance, foram considerados: Regressão Logística, Gradiente Estocástico Descendente, Perceptron, SVC (Support Vector Machine), Árvore de Decisão, Random Forest, Naive Bayes e KNN. O ideal seria fazer ajuste de hiperparâmetros em todos eles, porém nem todos indicavam que os hiperparâmetros iriam ajudar a levar o modelo a ser escolhido e o processo em alguns casos demoraria muito para pouco benefício. Por isso, apenas alguns tiveram seus hiperparâmetros ajustados.

Em ordem de melhor acurácia, obtivemos o seguinte resultado: Regressão Logística (~ 97.5%), SVM (~97.4%), Gradiente Descendente Estocástico (~ 97.3%), Perceptron (~ 96.1%), Random Forest (~95.6%), KNN (~ 91.7%), Árvore de Decisão (~ 91.3%), Naive Bayes (~ 90.0%).

No caso dos modelos lineares, é preciso levar em consideração que ela assume que os dados podem ser separados linearmente. Dessa forma, a dimensionalidade deve ser a mais alta possível e, como temos um caso de classificação de texto, ter esse alto número de features faz com que os dados

---

6 https://stackoverflow.com/questions/46010617/do-i-use-the-same-tfidf-vocabulary-in-k-fold-cross-validation

possam ser separados em uma dimensão maior[7]. Como o resultado da Regressão Logística já havia sido bom sem mexer nos hiperparâmetros, foi feito o tuning deles para que nós conseguíssemos obter uma acurácia ainda maior. Isso foi obtido com o solver linlinear, C = 100 e a penalidade para L2. Por causa da questão de não aplicar o TF-IDF para o dataset inteiro, o ajuste foi feito sem utilizar o GridSearchCV, mas sim a função criada (cross_val_tfidf) para cada combinação dos hiperparâmetros selecionados para o tuning.

De qualquer forma o GridSearchCV com o X levando em consideração o TF-IDF de todo o dataset não deveria ser um grande problema nesse caso e por isso ele foi considerado para o tuning do SVM. Acabou que nesse caso o próprio parâmetro default acabou sendo o escolhido.

Escolha do modelo e matriz de confusão

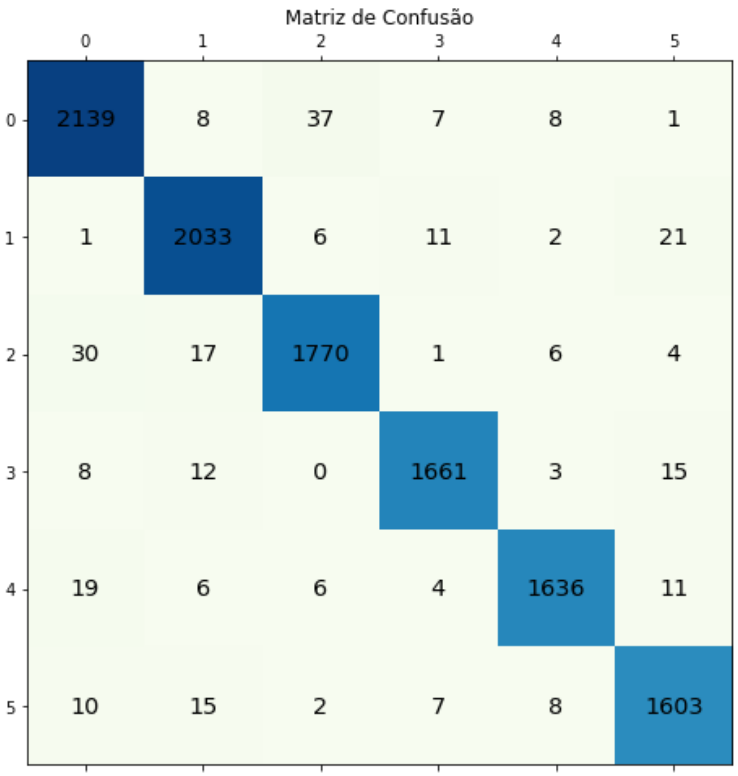
De fato, o modelo escolhido seria o de Regressão Logística, que possui tanto acurácia quanto log loss bastante satisfatórios. Para trabalhos futuros, seria interessante averiguar o tuning de hiperparâmetros tanto para o Random Forest quanto para o SVM. Os dois foram os modelos que mais estavam demorando e por mais que tenham dado bons resultados com a configuração *default*, o tuning ia ser bastante custoso em ambos os casos. Levando em consideração o bom resultado da Regressão Logística, seria difícil dar um resultado que valesse a pena passar por processo.

Pela acurácia dos dados de teste, não existe uma diferença para a acurácia dos dados de treinamento (100%) que sugeririam um overfitting. A curva de aprendizado foi gerada, mas é importante deixar claro que ela faz uso do X e Y completos, com o X já em formato TF-IDF. Por isso, há uma possibilidade do gráfico incluir esse "data leakage".

Para enxergar melhor o resultado do modelo, foi plotada a Matriz de Confusão (cujas funções mostradas em sala de aula foram utilizadas para considerar a matriz a partir da Stratified K-Fold Cross Validation). Note que o resultado é bom, pois as diagonais estão com a maior concentração. Isso indica que o modelo faz mais previsão correta do que incorreta, além de não parecer ter nenhum problema específico com alguma classe.

---

7 https://medium.com/@akshayc123/logistic-regression-87f7fbb4aaf6#:~:text=Logistic%20Regression%20(LR)%20is%20a,well%20on%20linearly%20separable%20classes



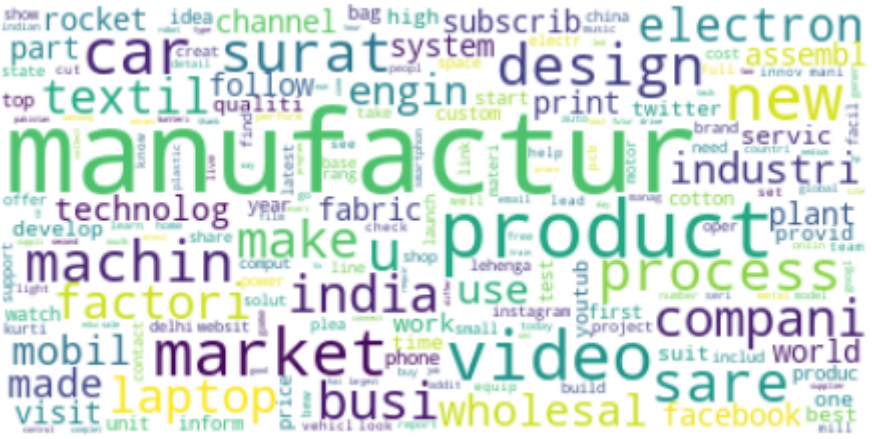
Análise do Dataset e Próximos Passos

Com o intuito de analisar o dataset, foi também vista a nuvem de palavras total e para cada classe. Isso foi feito em um notebook independente (*analises_classificacao.ipynb*). Essa análise foi feita para ver o que faria um bom conjunto de título e descrição para promover seu vídeo como pertencente a uma determinada classe. Um detalhe a se notar é que as palavras que dão nome às categorias, como 'travel', 'music', 'art', são as palavras mais frequentes de suas categorias.

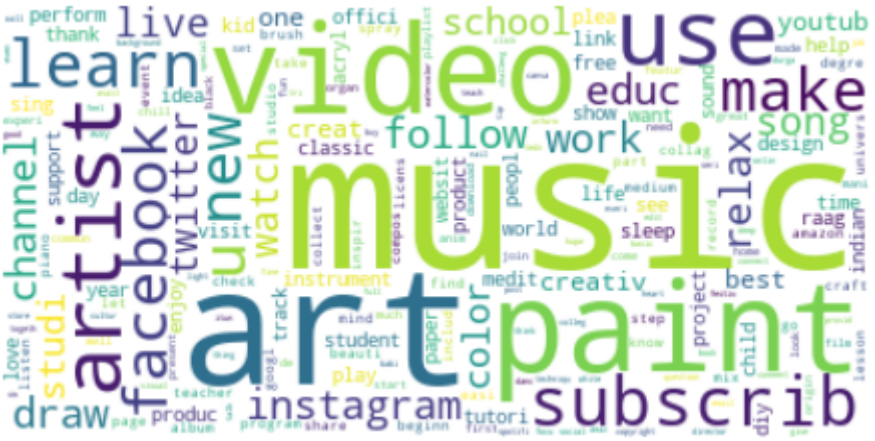
Nuvem de palavra da classe 'travel & blog'

Nuvem de palavra da classe 'science & technology'

Nuvem de palavra da classe 'food'



Nuvem de palavra da classe 'manufacturing'



Nuvem de palavra da classe 'art & music'

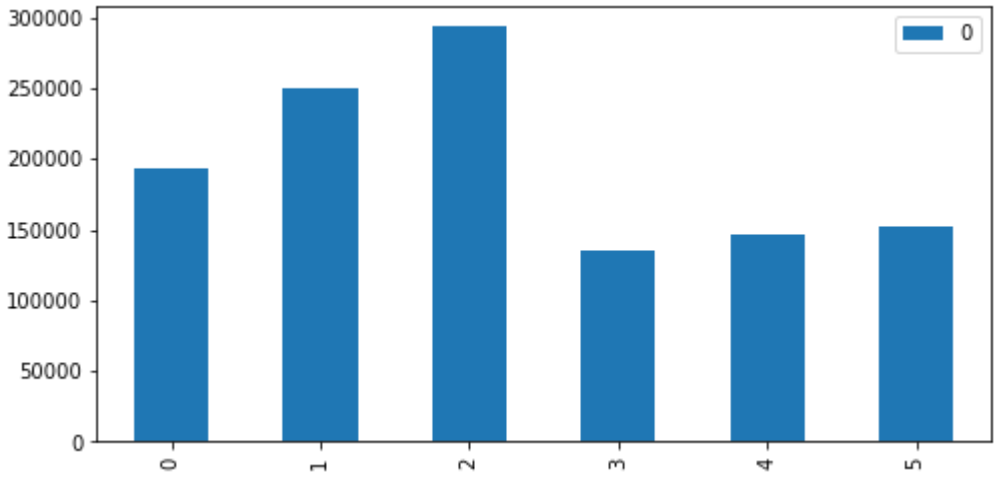
Nuvem da palavra da classe 'history'

Nuvem de palavras do dataset

O resultado encontrado foi bastante interessante e uma análise nossa é que existem palavras específicas ao dataset como "video", "subscrib", "channel" ou "follow". Isso pode ser observado porque não só "video" parece ser a segunda palavra mais frequente como também ela aparece como uma das palavras mais frequentes de todas as classes. O mesmo é visto para as demais, porém em menor frequência. Chegamos a fazer um teste retirando essas palavras, considerando que

elas poderiam ser “stopwords” do dataset, porém a performance dos modelos não chegaram a melhorar.

Outra informação interessante é que parece estar correta a suposição de que havia várias palavras em hindi, já que “hindi”, “india”, “goa” e “surat” aparecem como frequentes em algumas das classes. Mas nenhuma palavra em outra língua parece aparecer nas nuvens, então a remoção dos stopwords não seria necessária.



Categoria	**Média de Palavras por Documento**	**Quantidade de Linhas**
travel blog	87,80	2200
Science&Technology	120,70	2074
food	160,44	1828
manufacturing	79,87	1699
art & music	87,23	1682
history	92,83	1645
Total	105,31	11128

Um dos gráficos que também foram gerados na análise foi o que exibe a quantidade de palavras utilizadas em cada classe (somando título e descrição). Essa informação é auxiliada pela tabela que informa a média de palavras por documento de cada classe.

O que ambas as informações mostram é que a classe ‘food’, por exemplo, utiliza muito mais palavras do que as demais por documento, mesmo tendo menos vídeos no dataset associados a ela. Considerando a nuvem de palavras da categoria, é possível assumir que a palavra ‘food’ parece ser utilizada mais de uma vez por documento (olhando as palavras mais utilizadas de cada classe e sua

frequência, essa observação é comprovada já que essa palavra é utilizada 12448 vezes para 1828 linhas da categoria). Ou seja, uma dica para alguém fosse realizar um upload de um vídeo para o Youtube na categoria de comida, seria interessante inserir a palavra ‘food’ e inseri-la até mais de uma vez.

Para trabalhos futuros, seria interessante tentar novas abordagens além do TF-IDF, como também verificar se teria como generalizar ainda mais as palavras. Seria interessante ver a correlação entre algumas delas, além de verificar se a análise da distribuição das palavras por classe pode ser utilizada para melhorar os resultados.