EEL891 - Trabalho de Regressão

João Pedro Brandão e Silva[1]

[1]Escola Politécnica – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68536 – 21941-909 – Rio de Janeiro – RJ – Brasil

jp.brs@poli.ufrj.br

Resumo. *Este relatório tem por objetivo descrever o passo-a-passo de um trabalho de ciência de dados com foco em Regressão Linear da matéria EEL891.*

1. Datasets

Apesar de existir apenas um Dataset no Kaggle, o uso deste foi feito de duas formas diferentes. O primeiro foi renomeando as váriaveis qualitativas para quantitativas, enquanto o segundo foi transformando cada tipo de variável qualitativa em uma coluna do Dataset, ou seja, foram criados dummies para todas as variáveis qualitativas.

1.1. Dataset com transformação de variáveis qualitativas para quantitativas

1.1.1. Estudo e transformação dos dados

Aqui houve uma substituição dos valores de classificação como corte, clareza e cor, para respectivamente, números que cresciam de acordo com a grandeza da qualidade de cada uma dessas variáveis. Por exemplo, dessa forma já que existia 7 possíveis cores diferentes, onde cada variável de classificação era menos valiosa do que outra, a cor 'J' que era a menos valiosa foi substituída por 1 enquanto a cor 'D' que era a mais valiosa foi substituída por 7. Aplicando isso também à clareza e corte, seguindo o mesmo princípio utilizado anteriormente, foi obtido o dataset ficou da seguinte forma:

```
map_clarity = {"I1":1, "SI2":2, "SI1":2, "VS2":3, "VS1":3, "VVS2":4, "VVS1":4, "IF":5}
train = train.replace({"clarity":map_clarity})
train.head()
```

	id	carat	cut	color	clarity	x	y	z	depth	table	price
0	20000	0.35	3	4	4	4.44	4.48	2.80	62.8	58.0	798
1	20001	0.70	5	5	2	5.66	5.69	3.55	62.6	56.0	2089
2	20002	0.32	5	5	4	4.42	4.38	2.70	61.4	56.0	990
3	20003	0.30	5	3	4	4.32	4.35	2.67	61.7	54.2	631
4	20004	0.33	4	2	4	4.41	4.47	2.76	62.2	59.0	579

Figure 1. Alteração das variáveis qualitativas por quantitativas

E com esse dataset, foi possível obter a matriz de correlação das variáveis, que foi melhorada visualmente pela biblioteca seaborn. A partir da matriz de correlação pode-se ter uma idéia sobre quais variáveis, futuramente, podem ser melhor estudadas para saber se a existência delas no mesmo dataset pode ser benéfica ou maléfica para a criação do modelo preditivo.

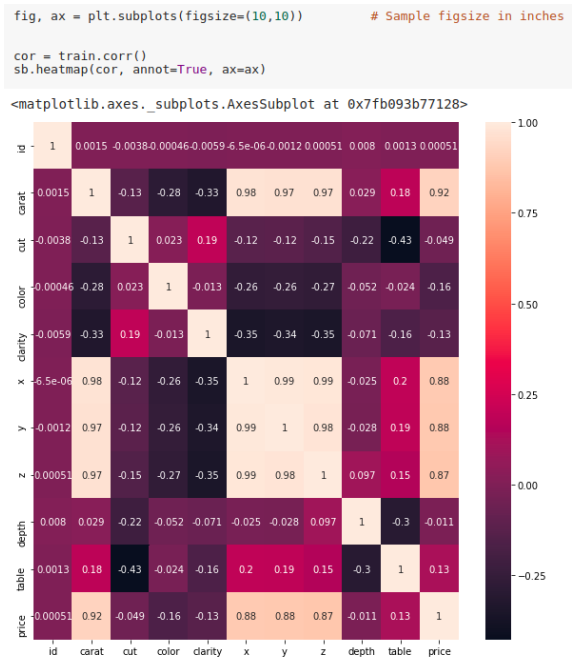

Figure 2. Matriz de correlação do Dataset

1.1.2. Remoção de outliers e dados suspeitos

A partir desse ponto ocorreu a remoção de amostras que, a partir do estudo de uma variável, demonstrou-se estranhamente diferente e portanto foi considerada um outlier. É válido dizer que, por mais que tenha acontecido a remoção dos outliers no dataset, ainda havia uma cópia intacta do mesmo para fins de comparação. Primeiramente isso foi feito com valores que poderiam apenas prejudicar o aprendizado do modelo, e foram visualizados a partir de um boxplot como a imagem abaixo mostra.

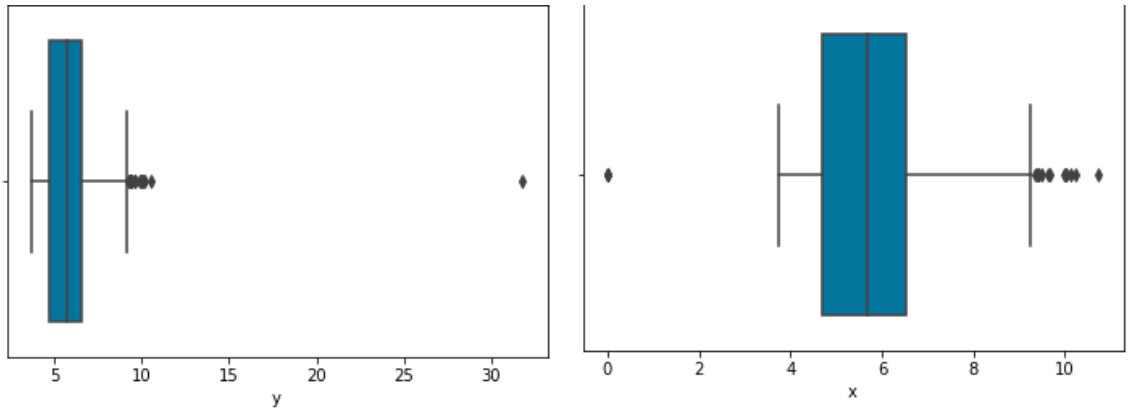

Figure 3. Boxplot das variaveis 'x' e 'y'

Nesses dois Boxplots em específico conseguimos ver os dois tipos de amostras que foram removidas do dataset, a primeira seria, olhando para y, uma amostra muito distoante em relação às outras. Já o segundo tipo de amostra que foi removida foram amostras que, foram removidas justamente por estarem em baixa ocorrência, e que ap-

resentavam alguma das variáveis de medidas(x,y ou z) com o valor de 0.0 milímetros. O que é muito suspeito, visto que existiam algumas amostras com os 3 valores de 0.0 e 0.00 milímetros, isso acaba sendo um número extremamente baixo para um tamanho de diamante medido(contando obviamente que o número foi representado de acordo com a incerteza da medição e portanto pode possuir por exemplo tamanho z de 0.0003mm).

1.1.3. Resultados Preliminares

Primeiramente foram escolhidos diversos modelos(como pode-se ver na tabela abaixo) para passarem por uma análise preliminar utilizando-se validação cruzada com 10 grupamentos. Sendo assim, a métrica utilizada para a avaliação preliminar não foi a métrica da competição e sim uma métrica mais simples da própria biblioteca de validação cruzada para apenas filtrar os algoritmos de regressão que melhor se adaptaram ao caso. Isso foi feito porque futuramente haverá uma otimização de hiperparâmetros feita em relação a cada um desses algoritmos escolhidos, porém um número muito grande de modelos a serem otimizados acabaria tornando inviável o processo de otimização pela grande quantidade de tempo necessário.

Table 1. Resultados Preliminares

Modelo de Regressão	Resultado Cross Validate
RandomForestRegressor	0.9717104691704993
GradientBoostingRegressor	0.9699682283314569
DecisionTreeRegressor	0.9518510710947605
KNeighborsRegressor	0.947098046967308
MultiLayerPerceptronRegression	0.9195778544137543
LinearRegrion	0.9039596630523906
Lars	0.9039596630523903
PassiveAggressiveRegressor	0.8222602642349324

1.1.4. Otimização de Hiperparâmetros

A otimização de hiperparâmetros foi feita utilizando a biblioteca RandomizedSearchCV, que é uma biblioteca parecida com a GridSearchCV, que tem como príncipio a partir de uma Grid de hiperparâmetros ela após fazer todas as permutações possíveis dos Hiperparâmetros consegue dizer qual a combinação deles é a que otimiza o Modelo Preditivo. Porém o RnadomizedSearchCV, utiliza um mecanismo semi aleatório de escolha dos Hiperparâmetros, a partir de um conjunto de hiperparâmetros passado como parâmetro da função. A partir dessa escolha aleatória, a função consegue identificar com quais variáveis ocorrem a maior variação do resultado e consegue otimizar o Modelo Preditivo com maior variabilidade de permutações dentro de um número máximo de iterações da função. Dessa forma, apesar de ainda existir uma quantidade de tempo grande gasta com a otimização dos hiperparâmetros, esse tempo ainda é muito menor do que caso usássemos a função GridSearchCV. O algoritmo de otimização fez 500 iterações para cada modelo abaixo

```
espaco_de_parametros_tree = {
    "max_depth" : range(1,30),
    "min_samples_split" : range(2,256),
    "min_samples_leaf" : range(1,256),
}
```

```
busca = RandomizedSearchCV(DecisionTreeRegressor(),
                    espaco_de_parametros_tree,
                    n_iter = 500,
                    cv = KFold(n_splits = 5))

busca.fit(X, Y)
#dessa maneira, diferentemente do gridsearch que vai rodar em cima de todo o espaco de
```

```
RandomizedSearchCV(cv=KFold(n_splits=5, random_state=None, shuffle=False),
                   error_score='raise-deprecating',
                   estimator=DecisionTreeRegressor(criterion='mse',
                                                   max_depth=None,
                                                   max_features=None,
                                                   max_leaf_nodes=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   presort=False,
                                                   random_state=None,
                                                   splitter='best'),
                   iid='warn', n_iter=500, n_jobs=None,
                   param_distributions={'max_depth': range(1, 30),
                                        'min_samples_leaf': range(1, 256),
                                        'min_samples_split': range(2, 256)},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring=None, verbose=0)
```

Figure 4. Otimização de hiperparâmetros

Table 2. Comparação Após Otimização de Hiperparâmetros

Modelo de Regressão	Resultado anterior	Resultado Pós Otimização
RandomForestRegressor	0.9717104691704993	0.9741701749699716
GradientBoostingRegressor	0.9699682283314569	0.9712121313457821
DecisionTreeRegressor	0.9518510710947605	0.9680635845519397

1.1.5. Transformação logarítmica das variáveis

Nessa seção ocorreu a transformação logarítmica das variáveis, pois, apesar de tanto as variáveis 'x','y','z' e 'quilate' possuírem uma grande correlação com a variável preço, pode-se se notar que o comportamente de correlação era muito mais evidente quando ao invés de utilizarmos as variáveis citadas, utilizássemos o log dos valores das variáveis em seu lugar, assim como ao invés da variável 'preço', utilizar o log do preço. Aqui é válido citar que quando o valor das variáveis que possuiam amostras com valor zero foram somados em 1 antes da transformação. Sendo assim a variação dos dados continuou intacta enquanto foi resolvido o problema de fazer a tranformação em log de valores 0.

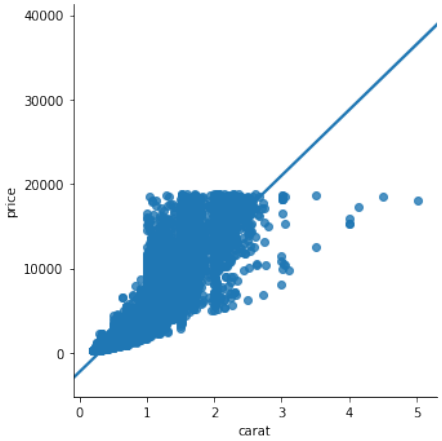

Figure 5. Correlação antes da transformação em logarítmos

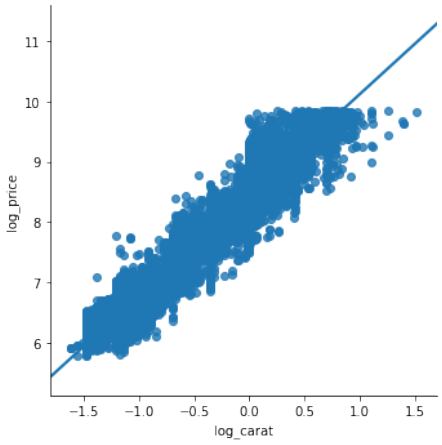

Figure 6. Correlação depois da transformação em logarítmos

1.1.6. Comparação do modelo treinado com outliers e sem

Após a otimização do modelo de Random Forest(escolhido por ter apresentado melhor resultado) e o respectivo treino do mesmo utilizando o dataset que não havia outliers, foi feito uma comparação entre este modelo e um outro que, apesar de também ser um Random Forest, foi treinado e testado com validação cruzada em cima do dataset que não tinha sido modificado, ou seja, possuia todas as variáveis julgadas por mim como outliers. Surpreendentemente(pelo menos para mim) o modelo treinado e otimizado utilizado o dataset completo acabou possuindo um erro médio quadrático menor do que o modelo cujo treino e predição foi feita com o dataset sem outliers.

1.1.7. Tratamento melhorado dos dados

Visto que há considerável perda de informação com a remoção dos Outliers e considerando que a possibilidade de existência de atributos dimensionais nulos é muito pequena, os dados considerados outliers que possuíam um valor muito distante da média(como y=32.5 na figura 3) foram multiplicados por 0.1, visto que a ordem de

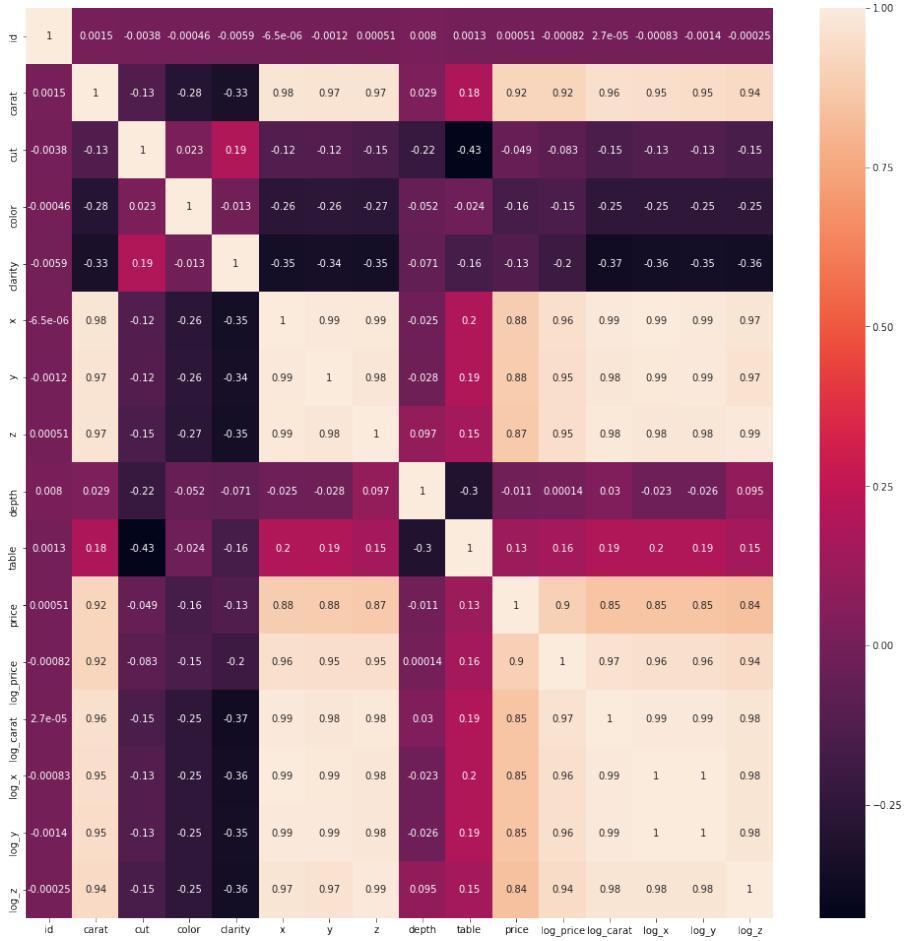

Figure 7. Matriz de correlação do dataset com Logs

grandeza provavelmente estava errada; enquanto os atributos dimensionais nulos foram estudados e substituídos de maneira mais inteligente. Por exemplo, visto que existia um número de dados que possuíam z=0, mas possuiam tanto depth quanto x e y, sabendo-se que depth=z*200/(X+Y) pode-se obter o valor exato de z a partir dessa conta e portanto não haveria desperdício de informações. Já os dados que possuíam tanto x,y e z nulos, foram modificados a partir de um estudo observando-se diamantes que possuíam as características quantitativas próximas e qualitativas iguais, o que não foi difícil de encontrar pois o dataset possui um grande número de dados. Além do que, todas as substituições obedeciam a regra do cálculo de depth, tornando os dados mais próximos da realidade. Utilizando esse tipo de tratamento de dados foi notada uma melhora considerável em relação ao Erro Médio Quadrático do algorítmo de RandomForestRegressor, mostrando que o Dataset resultante não só possui dados com maior qualidade em relação ao Dataset com outliers removidos como possui qualidade maior também em relação aos dados que não possuíram nenhum tratamento(dataset puro).

1.1.8. Conclusões da exploração dos dados do primeiro dataset

O dataset que teve suas variáveis qualitativas alteradas para quantitativas possui, após ter seu melhor modelo otimizado e treinado com o melhor conjunto de dados, um desempenho na competição de 0.11504. Isso é, ele foi otimizado com RandomizedSearchCV, treinado com o Dataset que possuía todas as amostras e todas as variáveis, porém houve uma transformação dos valores das variáveis em seus respectivos valores logarítmicos, e o modelo utilizado foi o RandomForestRegressor, que buscava otmizar o erro médio quadrático.

1.2. Dataset com dummies/One-Hot Enconding

Todos os passos de Limpeza de Dados acima citados foram feitos exatamente na mesma ordem para criar esse dataset, porém dessa vez utilizando colunas dummies ao invés de variáveis quantitativas. Dessa maneira, com a mesma transformação de variáveis em logaritmos, com a mesma remoção dos outliers e com os mesmos testes submetidos tanto às predições com outliers e sem outliers, com logarítmos e sem logarítmos, e sendo otimizados com o RandomizedSearchCV.

Porém, ao comparar o desempenho desses dois Datasets, com o mesmo processo de limpeza e utilizando o mesmo algoritmo, foi visto que o Dataset que utiliza Dummies melhorou o desempenho do Algoritmo consideravelmente. Sendo assim, as seguintes melhorias foram feitas no processo de escolha de atributos e hiperparâmetro utilizando o RandomForestRegressor treinado com esse dataset.

cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_Very Good
0	0	0	0	1
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	0	1	0

Figure 8. Dataset após a aplicação da função getDummies da biblioteca Pandas

1.2.1. Transformação de Variáveis com Log e Raíz Cúbica

Além de transformação logarítmica de variáveis também existe outros tipos de transformação, como normalização, transformação por raiz, etc. Um tipo interessante para esse dataset foi a transformação utilizando raiz cúbica de atributos que estavam ligados de certa maneira com o volume, como quilate, x,y e z. Porém, o melhor conjunto de transformações aplicadas a esses atributos foi a Raíz Cúbica de Quilates, a Raíz Cúbica de X, o log de y e Z sem nenhuma transformação. Essa combinação de atributos forneceram os melhores resultados quando aplicados aos algoritmos preditivos.

cubic_root_carat	cubic_root_x	log_y	z
0.704730	1.643593	1.499623	2.80
0.887904	1.782128	1.738710	3.55
0.683990	1.641122	1.477049	2.70
0.669433	1.628651	1.470176	2.67
0.691042	1.639883	1.497388	2.76

Figure 9. Dataset após a aplicação da função getDummies da biblioteca Pandas

1.2.2. Escolha dos atributos

Em relação a escolha dos atributos, a principio todos os atributos resultavam numa melhora de desempenho do algoritmo. Porém, com a limpeza sendo devidamente feita, a variável depth se tornou completamente irrelevante, como ela era uma combinação das variáveis x,y e z, a existência dela no dataset resultava em apenas uma redundância de informações, o que acabava até prejudicando o desempenho do algoritmo. O atributo depth foi o único que quando removido do dataset acarretou num aumento de desempenho do algoritmo preditivo e portanto ele não foi utilizado nas predições.

1.2.3. Otimização de Hiperparâmetros

Em relação à otimização de hiperparâmetros, dado que o algoritmo RandomForestRegressor possui sua própria maneira de otimizar o tamanho das árvores geradas a partir do método de otimização do Erro Médio Quadrático, como a competição levava em conta justamente o erro médio quadrático percentual, não houve nenhuma alteração na otimização do algoritmo com a função GridSearchCV. Porém, o algoritmo possui hiperparâmetros que podem ser ajustados, pois recebem valores padrões na inicialização do algoritmo que podem ser mudados para explorar uma melhora no desempenho. Por exemplo, o número de árvores geradas pela floresta têm como valor padrão 100, porém mudar esse valor pode acarretar numa melhora ou piora do desempenho do algoritmo. Da mesma maneira em que o número máximo de variáveis pode seguir diferentes critérios, por exemplo ”auto” coloca como número de variáveis sendo o máximo, enquanto o ’sqrt’ e o ’log2’ colocam como número máximo de variáveis do algoritmo ou a raiz quadrada ou o log do número de variáveis do dataset.

```
no_of_test=[10,25,50,100,125,137,150,167,180,200]

params_dict={'n_estimators':no_of_test,'n_jobs':[-1],'max_features':["auto",'sqrt','log2']}
clf_rf=GridSearchCV(estimator=RandomForestRegressor(),param_grid=params_dict)
```

Figure 10. Mapa de parâmetros explorados

Após a devida otimização dos hiperparâmetros o melhor conjunto de parâmetros encontrado quando houve a busca foram os da figura 11. Garantindo uma pontuação de 0.09337 de Erro Médio Quadrático Percentual na competição.

```
clf_rf.best_params_
```

```
{'max_features': 'auto', 'n_estimators': 150, 'n_jobs': -1}
```

Figure 11. Melhores Parâmetros Encontrados

1.3. Informações Finais

Esse relatório pertence ao usuário do Kaggle jpbrs, identificado na competição ”DEL/UFRJ - EEL891 - 2019-1 - Trabalho 1” como João Pedro Brandão. O código será enviado tanto por e-mail para o professor Heraldo quanto estará em https://github.com/jpbrs/UFRJ-Projects