# Regressão Linear

Rodrigo Rusa
208592
rodrigorusa@gmail.com

Thales Eduardo Nazatto
074388
tenazatto@gmail.com

## I. Introdução

Regressão Linear em Aprendizado de Máquina é uma técnica de aprendizado supervisionado que consiste na regressão de um modelo matemático que relaciona variáveis de entrada $X_i (i = 1, 2, ..., n)$ com uma variável de saída $Y$. Esse modelo é definido como:

$$h(\theta) = \theta_0 + \theta_1 * x_1 + \theta_2 * x_2 + ... + \theta_n * x_n \quad (1)$$

Encontrar esse modelo matemático envolve encontrar os valores de $\theta_i$ para um conjunto de entrada $X_i$ de forma que os valores de saída sejam próximos dos valores desejáveis $Y_i$. Para isso é definido uma função de custo:

$$J(\theta_0, \theta_1, ..., \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad (2)$$

Essa função representa o erro do modelo $h(\theta)$. Assim, para encontrar o melhor modelo precisamos encontrar os valores de $\theta_j$ que minimize a função custo $J$. Para isso podemos usar duas técnicas.

- **Gradiente Descendente:** Essa técnica consiste em inicializar os valores de $\theta_j$ e iterativamente atualizá-los até encontrar o minímo da função custo $J$, esse mínimo pode ser local ou global dependendo do tipo da função $J$. A atualização dos parâmetros $\theta_j$ a cada iteração é feita segundo a fórmula:

  $$\theta_j := \theta_j - \alpha * \frac{\partial}{\partial \theta_j} * J(\theta_0, \theta_1, ..., \theta_n) \quad (3)$$

  Onde $\alpha$ representa o *learning rate*, ou seja, a taxa de velocidade da descida do gradiente. Substituindo $J$ por (2) e aplicando a derivada, temos:

  $$\theta_j := \theta_j - \alpha * \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) * {x_j}^{(i)} \quad (4)$$

  Esse algoritmo pode operar em três modos distintos:
  - **Batch:** Nesse modo todo o conjunto de treinamento é utilizado a cada passo da descida do gradiente
  - **Stochastic:** Nesse modo apenas um exemplo de treinamento é utilizado
  - **Mini Batch:** Nesse modo um sub conjunto menor do conjunto de treinamento é utilizado
- **Equação Normal** Essa técnica consiste em encontrar os valores de $\theta_j$ analiticamente através de cálculo matricial, segundo a fórmula:

  $$\theta = (X^T X)^{-1} X^T y \quad (5)$$

  Onde $(X^T X)^{-1}$ representa a inversa da matriz $X^T X$. A derivação dessa fórmula pode ser encontrada em [1]

Essa técnica de aprendizado é muito utilizada para casos de predição de valores, por exemplo, de casas e diamantes. Neste trabalho exploramos as técnicas de Regressão Linear e alternativas para encontrar o melhor modelo matemático que é capaz de predizer preços de diamantes, utilizando a base de dados *diamonds* disponibilizado em [2].

Essa base de dados possui um total de 53,940 exemplos, 45,849 exemplos de treinamento e 8,091 de teste. Cada exemplo possui valores para cada um dos 9 atributos que caracterizam um diamante e seu respectivo valor em doláres. Os atributos que caracterizam os diamantes são:

- **Carat:** Peso do diamante (0.2–5.01)
- **Cut:** Qualidade do corte (Fair, Good, Very Good, Premium, Ideal)
- **Color:** Cor do diamante, de J (pior) a D (melhor)
- **Clarity:** Nível de clareza do diamante (I1 (pior), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (melhor))
- **X:** Comprimento do diamante em *mm* (0–10.74)
- **Y:** Largura do diamante em *mm* (0–58.9)
- **Z:** Profundidade do diamante em *mm* (0–31.8)
- **Depth:** Porcentagem total de profundidade $= \frac{z}{mean(x,y)} = 2 * \frac{z}{(x+y)}$ (43-79)
- **Table:** Largura do topo do diamante relativo ao ponto de maior largura (43-95)
- **Price:** Preço em dólares (US$)

## II. Metodologia

A fim de encontrar o modelo matemático que melhor modela o conjunto de dados diamantes foram implementadas as duas técnicas de descida de gradiente estudadas, a iterativa e suas três variantes (*Batch*, *Stochastic* e *Mini Batch*) e a analítica, Equação Normal, como descritas anteriormente. Como validação da corretude da implementação do algoritmo Gradiente Descendente Estocástico foi utilizada a função *SGDRegressor* da biblioteca *Scikit Learn* [3] em *Python* que implementa essa técnica.

Além disso, foi aplicado uma etapa de pré-processamento dos dados antes de submetê-los aos algoritmos de Regressão Linear. Nessa etapa aplicou-se uma técnica para codificar variáveis categóricas em numéricas, pois a Regressão Linear não suporta valores não numéricos. Existem diferentes maneiras de codificar variáveis categóricas, porém, como as variáveis categóricas da base diamantes (*Cut*, *Color*, *Clarity*) possuem características qualitativas, ou seja, possuem uma escala de

qualidade, optou-se por atribuir valores diferentes para cada categoria atribuindo valor 1 a categoria de pior qualidade e valores maiores que 1 crescentes as escalas qualitativamente melhores. A categoria *Cut* que representa qualidade do corte do diamante, por exemplo, foi codificado da seguinte forma:

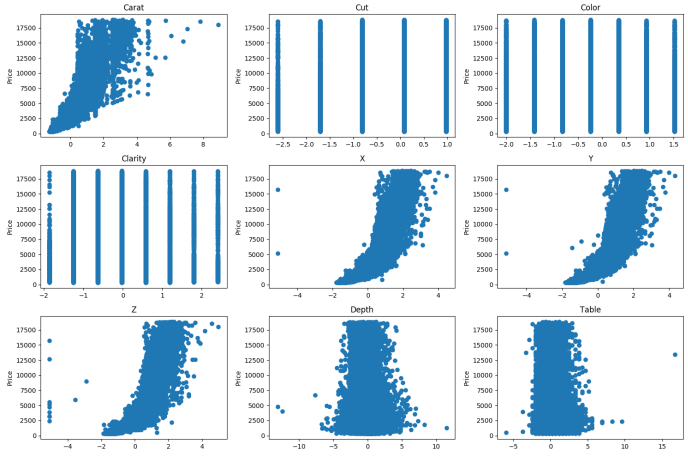
- Fair: 1
- Good: 2
- Very Good: 3
- Premium: 4
- Ideal: 5

Nessa mesma etapa de pré-processamento foi aplicado também uma normalização dos dados, chamada *feature scaling*, que faz com que os dados fiquem em similar escala o que resulta em uma descida de gradiente mais rápida, pois a função custo $J$ terá um formato simétrico. Para essa normalização subtraiu-se a média e dividiu-se pelo desvio padrão para cada exemplo, como a seguir:

$$x_i = \frac{x_i - \mu}{\sigma} \tag{6}$$

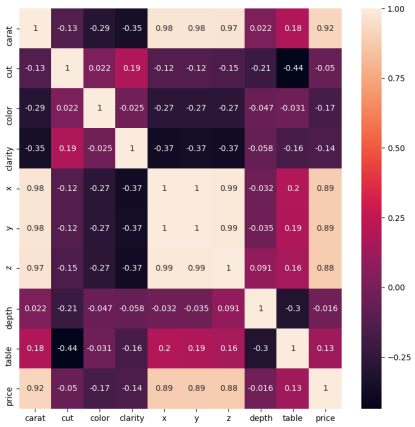
A fim de encontrar informações adidionais para extrair o melhor modelo matemático que representa a base de dados diamantes foi realizado uma análise dos dados de treinamento, visualizando a distribuição dos dados de cada característica em relação ao preço do diamante como ilustrado na Figura 1 e a correlação entre elas, Figura 2.

Figure 1. Distribuição dos dados para cada característica




Assim, com esse framework desenvolvido realizou-se alguns experimentos variando os algoritmos utilizados e seus parâmetros, como *learning rate*, número de iterações e experimentos com diferentes modelos de hipótese, atribuindo graus $j >= 1$ as variáveis ${X_i}^j$ com o objetivo de encontrar o melhor modelo sem causar *Overfitting*. Esses experimentos foram conduzidos usando apenas o conjunto de treinamento que foi divido em treinamento e validação, sendo 80% para validação e 20% para validação. O conjunto de teste foi usado apenas uma vez para testar o melhor modelo encontrado.

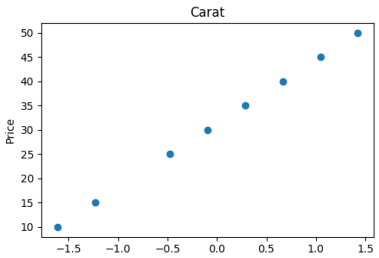
Figure 2. Correlação entre as variáveis




## III. Experimentos e Discussão

A fim de validar a corretude da implementação do algoritmo Estocástico de Descida do Gradiente foi realizado um experimento com dados produzidos sintéticamente utilizando apenas uma variável, como ilustra a Figura 3.

Figure 3. Dados produzidos sintéticamente




Esses dados foram submetidos ao algoritmo implementado e a função *SGDRegressor* da biblioteca *Scikit Learn* utilizando os mesmos parâmetros, 1.000 iterações, *learning rate* de 0.01 e tolerância de $10^{-6}$. O resultado comparando ambos é ilustrado na Tabela a seguir:

| Algoritmo | $\theta_0$ | $\theta_1$ | EQM |
|---|---|---|---|
| Implementado | 31.2470 | 13.1680 | 0.00 |
| *Scikit Learn* | 31.2475 | 13.1672 | 0.00 |

Como podemos observar, a implementação do Gradiente Descendente Estocástico apresentou resultado similar ao da biblioteca *Scikit Learn*, validando assim a implementação.

Para encontrar o modelo matemático que melhor se ajusta aos dados de diamantes foram realizados experimentos de modelos mais simples a modelos mais complexos, evitando assim a ocorrência de *Overfitting*.

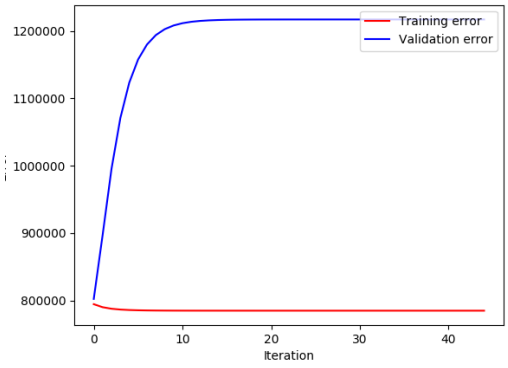
- **Modelo Linear:**

Nesse modelo o objetivo é encontrar uma reta que melhor modela os preços dos diamantes segundo as características

de entrada. Inicialmente foi realizado o experimento com os algoritmos baseados em Gradiente Descendente, *Stochastic*, *Batch* e *Mini-Batch* e posteriormente com o método analítico de Equação Normal.

Para o algoritmo *Stochastic* baseado em Gradiente Descendente, com 1.000 iterações, *learning rate* de 0.01 e tolerância de $10^{-6}$ o algoritmo alcançou convergência com 46 iterações. A Figura 4 ilustra a curva de erro dos conjuntos de treinamento e validação a cada iteração do algoritmo:

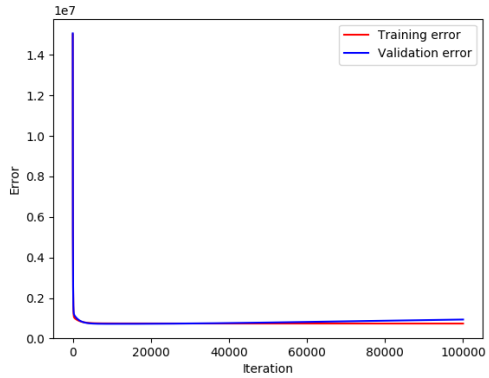
Figure 4. Curva de Erro do algoritmo *Stochastic*




Podemos observar que o conjunto de validação apresentou um erro maior que o conjunto de treinamento caracterizando um *Overfitting* do modelo, ou seja, o modelo não é capaz de generalizar para o conjunto de validação.

Para o algoritmo *Batch*, com 100.000 iterações, *learning rate* de 0.01 e tolerância de $10^{-6}$ não foi alcançado a convergência como ilustra a curva de erro:

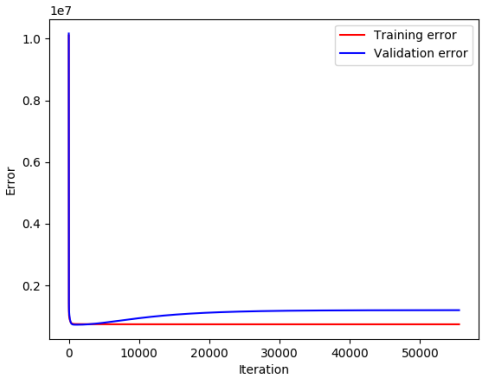
Figure 5. Curva de Erro do algoritmo *Batch*




De modo a alcançar convergência foi realizado um experimento aumentando o *learning rate* para 0.1 e mantendo o número de iterações e o algoritmo *Batch* convergiu após 55.561 iterações como ilustra a Figura 6:

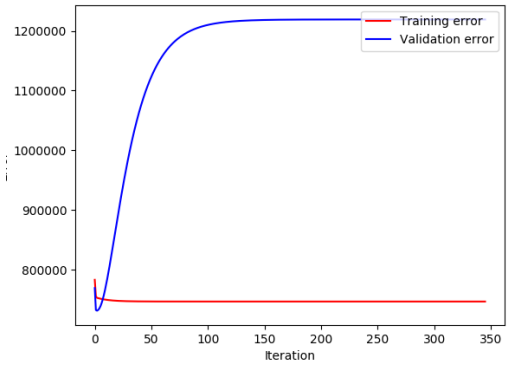
Com base nos dados, podemos concluir que o aumento do *learning rate* acelerou a convergência do algoritmo baseado em Gradiente Descendente diminuindo o número de iterações necessárias em aproximadamente 50%.

Figure 6. Curva de Erro do algoritmo *Batch* com *learning rate* de 0.1




Para o algoritmo *Mini-Batch*, com 1.000 iterações, *learning rate* de 0.01, tolerância de $10^{-6}$ e *batch* de tamanho 10 o algoritmo alcançou convergência com 347 iterações, mais iterações que o algoritmo *Stochastic* que convergiu com 46, porém apresentou menor erro. Em relação ao algoritmo *Batch*, o *Mini-Batch* convergiu com menos iterações, mas com erro maior. Assim, podemos concluir que o algoritmo *Mini-Batch* é um algoritmo intermediário aos outros dois, *Stochastic* e *Batch*.

Figure 7. Curva de Erro do algoritmo *Mini-Batch*




Por fim, foi realizado o experimento com o metódo da Equação Normal que nos fornece o melhor modelo analíticamente, sem a necessidade de iterações. Os resultados obtidos nos experimentos estão ilustrados na Tabela abaixo:

| | Treinamento | | Validação | | |
|---|---|---|---|---|---|
| Algoritmo | EQM | $R^2$ | EQM | $R^2$ | Iteração |
| *Stochastic* GD | 784.783,49 | 0.9000 | 1.217.109,27 | 0.8549 | 46 |
| *Batch* GD | 738.408,39 | 0.8976 | 1.196.529,37 | 0.8460 | 55561 |
| *Mini-Batch* | 747.059,58 | 0.9029 | 1.219.047,92 | 0.8525 | 347 |
| Equação Normal | 738.408,38 | 0.8976 | 1.197.328,51 | 0.8459 | - |

Podemos observar que os métodos baseados em Gradiente Descendente, por serem iterativos, levam mais tempo para convergirem comparado com ao método analítico de Equação Normal. Assim, para casos em que a base de dados possuem

um número pequeno de exemplos, como o a base de dados diamantes, e sabendo que a matriz $(X^T X)^{-1}$ é inversível, o uso da Equação Normal é mais eficiente.

- **Modelo Polinomial:**

Diante dos resultados alcançados com a modelagem linear e com o objetivo de encontrar o melhor modelo para estimação de preços de diamantes, foi realizado um experimento com um modelo não linear, polinomial, levando em consideração a análise da distribuição de cada característica em relação ao preço do diamante ilustrado na Figura 1. Nela podemos observar que as características *Carat*, *X*, *Y* e *Z* possuem uma distribuição próxima de grau 2. Dessa forma foi realizado um experimento atribuindo grau 2 a essas características mencionadas. Para esse experimento foi utilizado a Equação Normal, pela rápida convergência. Os resultados obtidos são ilustrados na Tabela a seguir:

| | Treinamento | | Validação | |
|---|---|---|---|---|
| Algoritmo | EQM | $R^2$ | EQM | $R^2$ |
| Equação Normal | 4.926.909,35 | -0.6287 | 3.643.581.276,45 | 0.0013 |

Como podemos observar, o modelo polinomial apresentou um erro quadrático médio (*EQM*) muito grande para os conjuntos de treinamento e validação. Assim, podemos concluir que um modelo linear, não polinomial, se ajuste melhor a base de dados diamantes.

- **Agrupamento/Remoção de características:**

Após o insucesso do modelo polinomial, foi notado que os valores de *X*, *Y* e *Z* possuíam uma correlação extremamente alta entre seus dados, e devido a este fato foi realizado um novo experimento com modelos lineares, mas desta vez com características agrupadas ou removidas, para verificar se uma simplificação do modelo linear seria mais efetiva. Nele, avaliou-se os seguintes tipos de modelos com as seguintes características:

- **Modelo 1:** *X*, *Y* e *Z* agrupadas em um produto das mesmas, denominado *Volume*, e todas as características anteriores foram preservadas.
- **Modelo 2:** *Cut*, *Color* e *Clarity* agrupadas em um produto das mesmas, denominado *Nominals*, e todas as características anteriores foram preservadas.
- **Modelo 3:** Modelo com apenas as características que possuem alta correlação com *Price* (*Carat*, *X*, *Y* e *Z*).
- **Modelo 4:** *X*, *Y* e *Z* agrupadas em um produto das mesmas, denominado *Volume*, e todas as outras características presentes são removidas com exceção de *Carat*.

Para o experimento foram realizados os mesmos procedimentos do modelo polinomial, utilizando a Equação Normal para aferir os dados. Os resultados obtidos são ilustrados na Tabela a seguir.

Neste experimento, podemos observar que o agrupamento de características com correlação alta pode ajudar a melhorar o modelo linear, uma vez que os modelos onde tiveram os valores de *X*, *Y* e *Z* agrupados tiveram os valores de *EQM* e $R^2$ *score* muito próximos entre os conjuntos de Treinamento e de Validação, evitando o *Overfitting* e garantindo uma confiabilidade maior na predição dos dados no conjunto de teste. Também é possível observar que, no conjunto de Treinamento, o $R^2$ *score* é próximo a 1 em todas as ocasiões, certificando uma boa distribuição em seus dados.

| | Treinamento | | Validação | |
|---|---|---|---|---|
| Modelo | EQM | $R^2$ | EQM | $R^2$ |
| 1 | 762124.48 | 0.8940 | 748206.54 | 0.8959 |
| 2 | 850672.61 | 0.8802 | 1839764.42 | 0.7761 |
| 3 | 1138312.81 | 0.8329 | 2916333.93 | 0.6689 |
| 4 | 1186587.72 | 0.8246 | 1206977.71 | 0.8224 |

- **Testes e predições do modelo encontrado:**

Após os experimentos citados, o Modelo 1 do experimento de agrupamento foi o escolhido por apresentar valores de *EQM* mais baixos e próximos entre conjunto de Treinamento e conjunto de Validação, assim como valores de $R^2$ *score* mais próximos de 1. Finalizando nossos experimentos, aplicamos as predições desse modelo em nosso conjunto de teste e verificamos a confiabilidade delas através do $R^2$ *score*. A Tabela abaixo mostra os resultados finais em cada algoritmo, considerando os resultados conseguidos anteriormente através dos experimentos no modelo linear:

| Algoritmo | $R^2$ |
|---|---|
| *Stochastic* GD | 0.8953 |
| *Batch* GD | 0.9051 |
| *Mini-Batch* | 0.9039 |
| Equação Normal | 0.9051 |

Além de apresentar um $R^2$ *score* em média de 0.9, o que garante uma altíssima confiabilidade em sua regressão, o modelo escolhido também se provou ser muito mais otimizado para os algoritmos em questão: No *Batch* GD ele levou 2351 iterações para convergir até a tolerância, no *Mini-Batch* ele levou 15 iterações e no *Stochastic* GD ele levou 3 iterações.

## IV. Conclusões

Nesse trabalho foi explorado as técnicas de Regressão Linear com o objetivo de encontrar o melhor modelo matemático capaz de predizer preços de diamantes. Foram implementadas as técnicas de Gradiente Descendente *Stochastic*, *Batch* e *Mini-Batch* e o metódo analítico de Equação Normal. Concluímos que a Equação Normal foi o algoritmo que apresentou a menor função de custo e o melhor desempenho, uma vez que o conjunto de dados em si é considerado pequeno. Vários modelos, lineares, polinomiais, com agrupamento e remoção de características foram investigados e ao final o melhor modelo encontrado apresentou resultados com boa confibialidade, em média 0.9. Entretanto, o valor da função custo $J$ foi considerada alta. Acreditamos que isso é um indicativo que predizer preços de diamantes requer um modelo mais complexo e mais dados a serem treinados.

## References


[1] Deriving the Normal Equation: https://ayearofai.com/rohan-3-deriving-the-normal-equation-using-matrix-calculus-1a1b16f65dda
[2] Diamonds dataset: https://www.kaggle.com/shivam2503/diamonds
[3] Scikit Learn: http://scikit-learn.org