

Lista X de Machine Learning

Pedro Barbosa Bahia

IMPA, Verão 2026

Sumário

1	Exercício 1a	2
2	Exercício 1b	3
3	Exercício 1c	4
4	Exercício 1d	5
5	Exercício 2	6
5.1	Parte d	6
5.1.1	i) Geração dos Dados Heteroscedásticos	6
5.1.2	ii) Estimadores de Mínimos Quadrados	6
5.1.3	iii) Cálculo de p-valores para Mínimos Quadrados Ordinários	8
5.1.4	iv) Estatística Z para o Estimador Generalizado	9
5.1.5	v) P-valores para o Estimador Generalizado	9
6	Exercício 3a	11
7	Exercício 3b	12
8	Exercício 3c	13
9	Exercício 3d	14
10	Exercício 3e	15
11	Exercício 4a	16
12	Exercício 4b	17

1 Exercício 1a

Falso. A proximidade entre $\varepsilon_{\text{treino}}$ e $\varepsilon_{\text{teste}}$ pode dar informações sobre o ajuste do modelo aos dados de treino.

Caso $\varepsilon_{\text{treino}}$ seja próximo ao $\varepsilon_{\text{teste}}$, o modelo pode estar *subajustado*, de modo que aumentar a complexidade poderia melhorar sua performance ainda mais.

De maneira análoga, caso o $\varepsilon_{\text{treino}}$ seja menor que o $\varepsilon_{\text{teste}}$, o modelo estará *sobreajustado*, com redução de complexidade podendo resultar em melhoras.

2 Exercício 1b

Verdadeiro. A distribuição t surge do fato de $\mathcal{N}(0, 1)$ dividido por $\sqrt{\frac{K}{N}}$ ter distribuição t com N graus de liberdade.

No nosso caso, $\mathcal{N}(0, 1)$ é a distribuição de $\frac{\hat{\beta}_1 - \beta_1}{\text{Var}(\hat{\beta}_1)}$. Isso é normal, pois $\hat{\beta}_1$ é normal.

Isso, por sua vez, vem do fato de $\hat{\beta}$ ser resultante de uma combinação linear de gaussianas, no caso, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$.

3 Exercício 1c

Falso. Considerando a classe $k = 0$ como as transações fraudulentas, o objetivo do modelo pode ser interpretado como:

$$\sum_{y_i \in k=0} \mathbf{1}_{[y_i \neq \hat{y}_i]} = 0$$

Não há restrições entretanto em relação às transações legítimas, ou seja, para:

$$\sum_{y_i \in k=1} \mathbf{1}_{[y_i \neq \hat{y}_i]}$$

Dado um modelo de acurácia $(1 - \varepsilon)$, têm-se que

$$1 - \frac{1}{n} \left[\sum_{y_i \in k=0} \mathbf{1}_{[y_i \neq \hat{y}_i]} + \sum_{y_i \in k=1} \mathbf{1}_{[y_i \neq \hat{y}_i]} \right] = 1 - \varepsilon$$

Dado uma acurácia

$$1 - \epsilon$$

, há infinitos valores de $\sum_{y_i \in k=0} \mathbf{1}_{[y_i \neq \hat{y}_i]}$ e $\sum_{y_i \in k=1} \mathbf{1}_{[y_i \neq \hat{y}_i]}$ que resolvem essa equação e, portanto, o valor da acurácia não é informativo para o erro individual das classes. Assim, apenas com a acurácias dos Modelos 1 e 2 não é possível determinar qual modelo tem menor erro em transações fraudulentas.

4 Exercício 1d

Falso

$$L_{ridge}(\beta) = (Y - Yhat)^T(Y - \hat{Y}) + \lambda\beta^T\beta$$

$$L_{ridge}(\beta) = (Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$$

$$L_{linear}(\beta) = (Y - Yhat)^T(Y - \hat{Y})$$

Caso $\lambda = 0$, temos que $L_{ridge}(\beta) = L_{linear}(\beta)$. Logo, a performance de ambos os modelos será a mesma. Então para o case de $\lambda = 0$, a afirmação é falsa.

5 Exercício 2

5.1 Parte d

5.1.1 i) Geração dos Dados Heteroscedásticos

Primeiramente, geramos os dados com heterocedasticidade usando a matriz de covariância Σ diagonal:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 n = 50
5 Sigma = np.diag([10 ** ((i - 20) / 5) for i in range(1, n + 1)])
6 np.random.seed(0)
7 X = np.array([np.ones(n), np.random.normal(0, 1, n)]).T
8 beta = np.array([1, 0.25])
9 epsilon = np.random.multivariate_normal(np.zeros(n), Sigma)
10 y = X @ beta + epsilon
```

A Figura 1 mostra os dados gerados com heterocedasticidade:

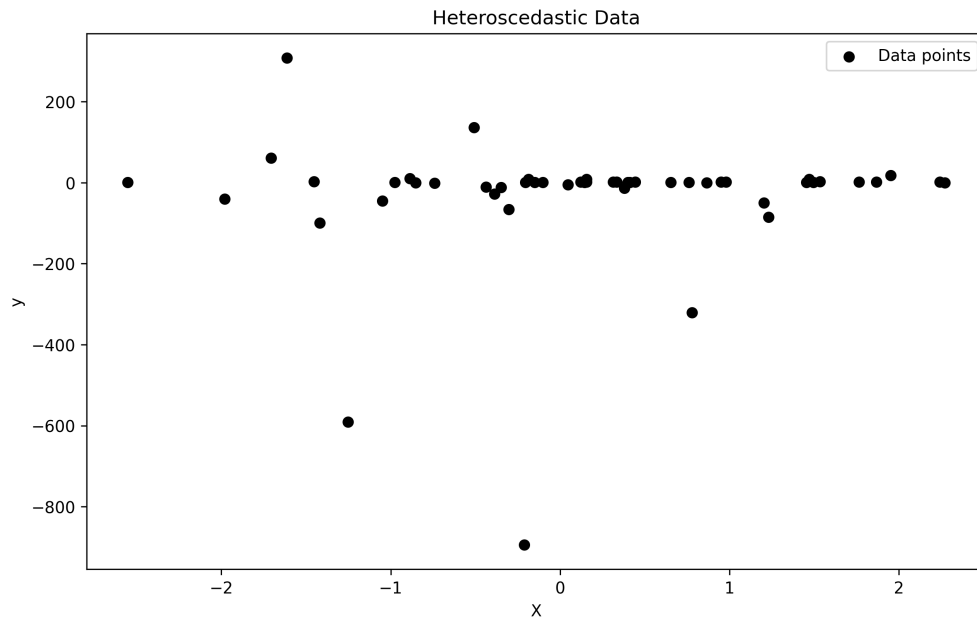


Figura 1: Dados heteroscedásticos gerados

A Figura 2 mostra os elementos diagonais da matriz Σ , evidenciando a heterocedasticidade:
As Figuras 3 e 4 mostram a distribuição e valores dos erros:

5.1.2 ii) Estimadores de Mínimos Quadrados

Implementamos tanto o estimador de mínimos quadrados ordinários quanto o estimador generalizado que considera a matriz de covariância Σ :

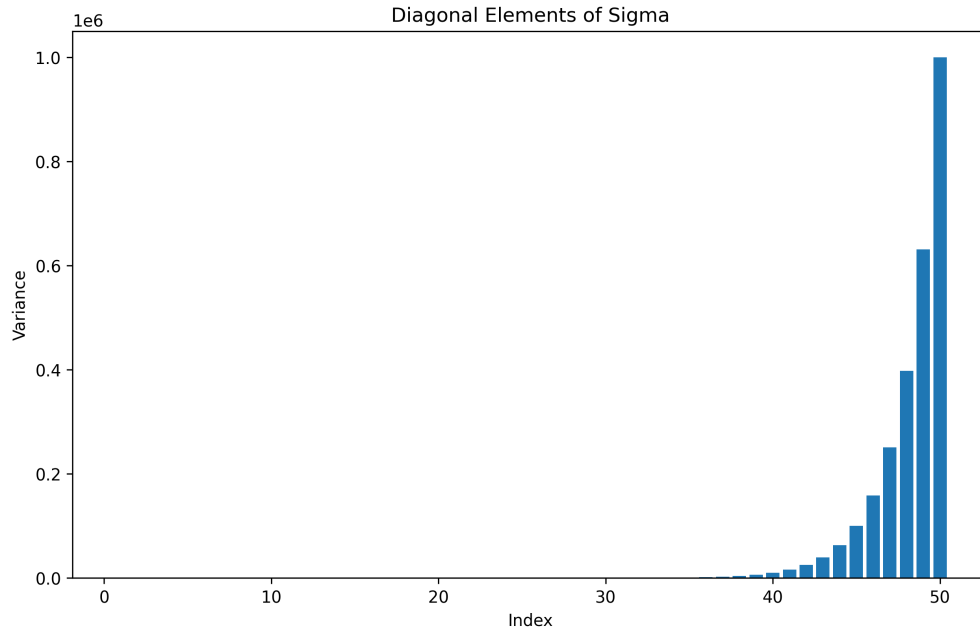


Figura 2: Elementos diagonais da matriz Σ

```

1 def beta_ordinary(X: np.ndarray, Y: np.ndarray) -> np.ndarray:
2     """
3     Compute the ordinary least squares estimator.
4     """
5     beta = np.linalg.inv(X.T @ X) @ X.T @ Y
6     return beta
7
8 def beta_sigma(X: np.ndarray, Y: np.ndarray, Sigma: np.ndarray) ->
9     np.ndarray:
10    """
11    Compute the generalized least squares estimator considering
12    the covariance matrix Sigma.
13    """
14    Sigma_1 = np.linalg.inv(Sigma)
15    beta = np.linalg.inv(X.T @ Sigma_1 @ X) @ X.T @ Sigma_1 @ Y
16    return beta
17
18 beta_hat_ordinary = beta_ordinary(X, y)
19 beta_hat_sigma = beta_sigma(X, y, Sigma)
20
21 print("True Beta:", beta)
22 print("Ordinary Beta:", beta_hat_ordinary)
23 print("Beta Sigma:", beta_hat_sigma)

```

Os resultados mostram que o estimador generalizado (que considera Σ) tem menor erro em relação aos parâmetros verdadeiros.

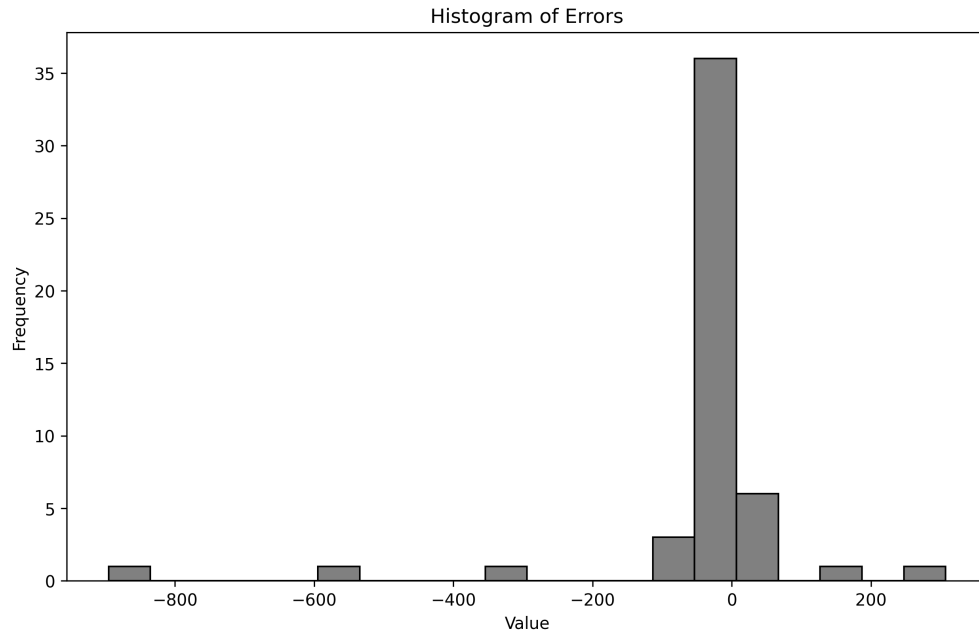


Figura 3: Histograma dos erros

5.1.3 iii) Cálculo de p-valores para Mínimos Quadrados Ordinários

```

1 def p_value_ordinary_least_square(X: np.ndarray, Y: np.ndarray,
2                                   beta_ordinary_hat: np.ndarray, j: int)
3                                   -> float:
4
5     """
6     Compute the p-value for the j-th coefficient of the ordinary
7     least squares estimator.
8     """
9
10    Y_hat = X @ beta_ordinary_hat
11    n, p = X.shape
12    dof = n - p
13    errors = Y - Y_hat
14    beta_j = beta_ordinary_hat[j]
15
16    # Z statistic
17    x_j_var = (np.linalg.inv(X.T @ X))[j, j]
18    Z = beta_j / np.sqrt(x_j_var)
19
20    # Estimate of sigma^2
21    sigma2_hat = (1 / dof) * (errors.T @ errors)
22
23    # t statistic and p-value
24    t_statistics = Z / np.sqrt(sigma2_hat)
25    t_statistics = np.abs(t_statistics)
26    p_value = 2 * (1 - scipy.stats.t.cdf(t_statistics, dof))

```

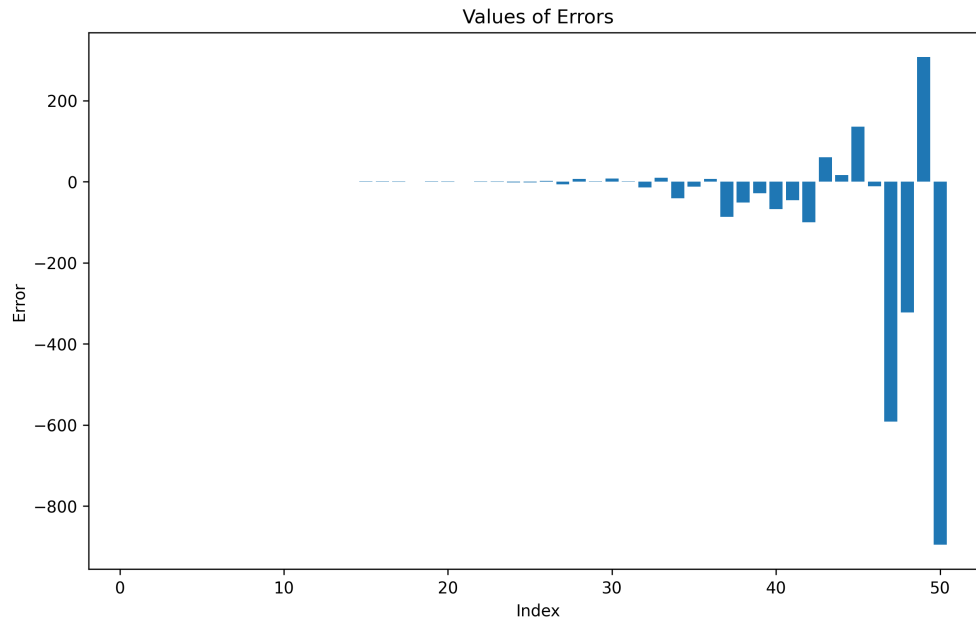



Figura 4: Valores dos erros por índice

```
25 return p_value
```

5.1.4 iv) Estatística Z para o Estimador Generalizado

```
1 def calculate_Z_sigma(X: np.ndarray, Sigma: np.ndarray,
2                       Beta_sigma: np.ndarray, j: int) -> float:
3     """
4     Compute the Z statistic for the j-th coefficient of the
5     generalized least squares estimator.
6     """
7     Sigma_inv = np.linalg.inv(Sigma)
8     den = np.linalg.inv(X.T @ Sigma_inv @ X)
9     den = den[j, j]
10    Z = Beta_sigma[j] / (np.sqrt(den))
11    return Z
```

5.1.5 v) P-valores para o Estimador Generalizado

```
1 def p_value_generalized_least_square(X: np.ndarray, Y: np.ndarray,
2                                       Sigma: np.ndarray,
3                                       beta_ordinary_hat: np.ndarray, j:
4                                           int) -> float:
5     """
6     Compute the p-value for the j-th coefficient of the
7     generalized least squares estimator.
```

```

7      """
8      Y_hat = X @ beta_ordinary_hat
9      n, p = X.shape
10     dof = n - p
11     errors = Y - Y_hat
12
13     # Z statistic
14     Z_sigma = calculate_Z_sigma(X, Sigma, beta_hat_sigma, j)
15
16     # Estimate of sigma^2
17     inverse_Sigma = np.linalg.inv(Sigma)
18     sigma2_hat = (1 / dof) * (errors.T @ inverse_Sigma @ errors)
19
20     # t statistic and p-value
21     t_statistics = Z_sigma / np.sqrt(sigma2_hat)
22     t_statistics = np.abs(t_statistics)
23     p_value = 2 * (1 - scipy.stats.t.cdf(t_statistics, dof))
24
25     return p_value

```

Esta implementação permite comparar os dois estimadores e calcular a significância estatística dos coeficientes em ambos os casos.

6 Exercício 3a

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

7 Exercício 3b

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

8 Exercício 3c

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

9 Exercício 3d

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

10 Exercício 3e

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

11 Exercício 4a

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.

12 Exercício 4b

Insira sua solução aqui, incluindo qualquer conta, código ou figura relevante para a sua solução.