

Universidade Federal do Rio de Janeiro  
Instituto Alberto Luiz Coimbra de  
Pós-Graduação e Pesquisa de Engenharia



Departamento de Engenharia Elétrica  
COE782 - Introdução ao Aprendizado de Máquina  
Prof. Dr. Markus Vinícius Santos Lima

*Lista 3 de exercícios*

Luiz Henrique Souza Caldas  
email: lhscaldas@cos.ufrj.br

11 de junho de 2024

# 1 Exercício 1

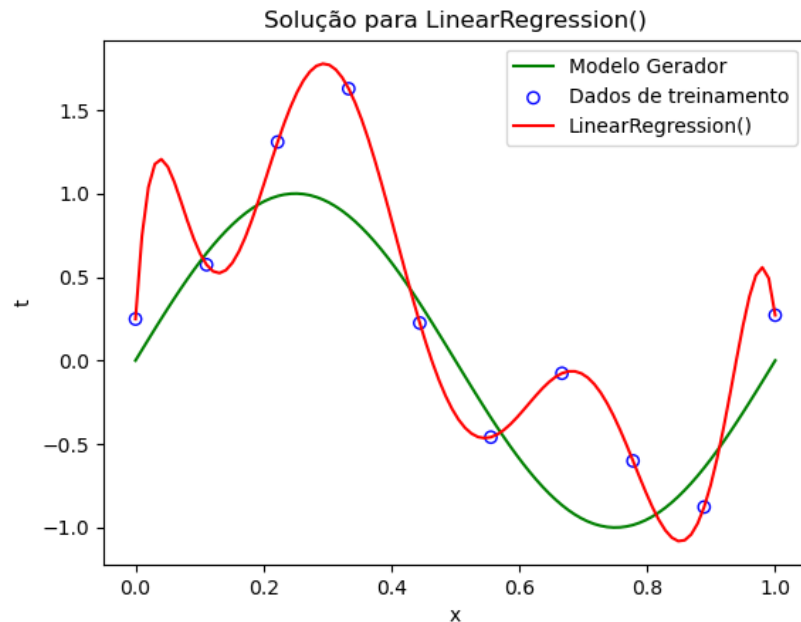
Considere o experimento computacional denominado “Polynomial Curve Fitting”, usado diversas vezes no livro texto (veja páginas 4 e 5 do livro, bem como Apêndice A), considerando a ordem do modelo sendo  $M = 9$  e o tamanho da amostra sendo  $N = 10$ . Faça:

- (a) Calcule a solução de mínimos quadrados (LS)  $\mathbf{w}_{LS}$ ;
- (b) Calcule a solução via regressão ridge (escolha um fator de regularização razoável)  $\mathbf{w}_{ridge}$ ;
- (c) Calcule a solução via regressão lasso (escolha um fator de regularização razoável)  $\mathbf{w}_{lasso}$ ;
- (d) Monte uma tabela exibindo os 10 coeficientes  $\mathbf{w}$  para as 3 soluções obtidas nos itens acima e comente/compare os resultados;
- (e) Plote uma figura contendo o processo gerador em verde (a senoide), e suas estimativas  $y_{LS}$ ,  $y_{ridge}$ , e  $y_{lasso}$  em preto, azul e vermelho, respectivamente;
- (f) Repita todos os itens anteriores para  $N = 20$  e  $N = 50$ .

## 1.1 Resposta do item (a)

Para responder esse item foi utilizada a classe `LinearRegression()` da biblioteca `sklearn` para linguagem `Python`, que realiza uma regressão linear utilizando mínimos quadrados. Foi escolhido um polinômio de ordem 9 como modelo.

**Figura 1:** Solução para mínimos quadrados (LS)

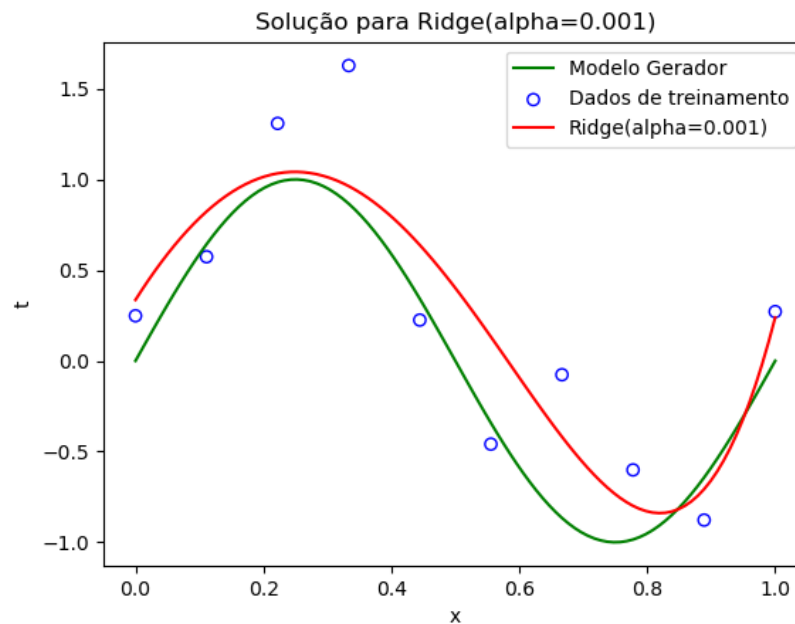


A regressão linear por mínimos quadrados não introduz nenhuma regularização e por isso a curva vermelha passa exatamente pelos dados de treinamento, mostrando que eles foram decorados overfitting.

## 1.2 Resposta do item (b)

Para responder esse item foi utilizada a classe *Ridge()* da biblioteca *sklearn* para linguagem *Python*, que realiza uma regressão linear utilizando mínimos quadrados e regularização de norma  $L_2$  (Ridge). Foi escolhido um polinômio de ordem 9 como modelo e o fator de regularização  $\lambda$ , que na classe *Ridge()* é chamado de *alpha*, que melhor adaptou a curva ao modelo gerador foi  $1^{-0.5}$ .

**Figura 2:** Solução para Ridge com  $\lambda = 1^{-0.5}$

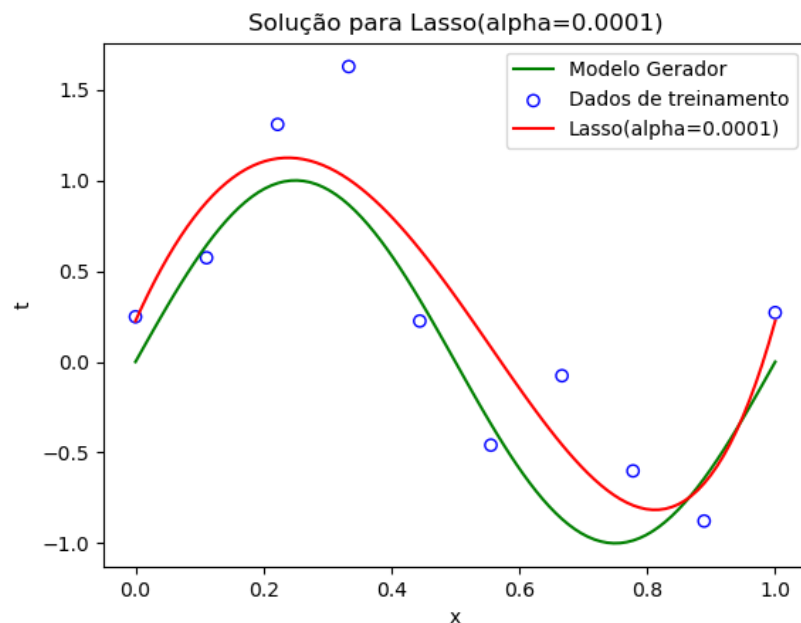


A regressão Ridge introduz uma penalização nos coeficientes através do fator de regularização, ajudando a evitar overfitting.

## 1.3 Resposta do item (c)

Para responder esse item foi utilizada a classe *Lasso()* da biblioteca *sklearn* para linguagem *Python*, que realiza uma regressão linear utilizando mínimos quadrados e regularização de norma  $L_1$  (Lasso). Foi escolhido um polinômio de ordem 9 como modelo e o fator de regularização  $\lambda$ , que na classe *Lasso()* é chamado de *alpha*, que melhor adaptou a curva ao modelo gerador foi  $1^{-0.5}$ .

**Figura 3:** Solução para Lasso com  $\lambda = 10^{-0.6}$



A regressão Lasso, assim como a Ridge, introduz uma penalização nos coeficientes através do fator de regularização, ajudando a evitar overfitting.

## 1.4 Resposta do item (d)

A tabela abaixo exibindo os coeficientes para as três soluções permite comparar diretamente o impacto da regularização nos coeficientes.

**Tabela 1:** Coeficientes para  $N = 10$

Modelo	LS	Ridge	Lasso
$w_0$	0.00000	0.00000	0.00000
$w_1$	61.35162	5.57635	7.87333
$w_2$	-1305.39950	-10.39014	-18.23029
$w_3$	10864.20021	-3.32097	2.98646
$w_4$	-43958.48661	2.24588	5.01574
$w_5$	96013.72998	3.46376	2.15551
$w_6$	-116795.01934	2.37258	0.11133
$w_7$	75859.90079	0.81213	0.00000
$w_8$	-22103.99856	-0.27544	-0.00000
$w_9$	1363.74433	-0.58096	0.08759

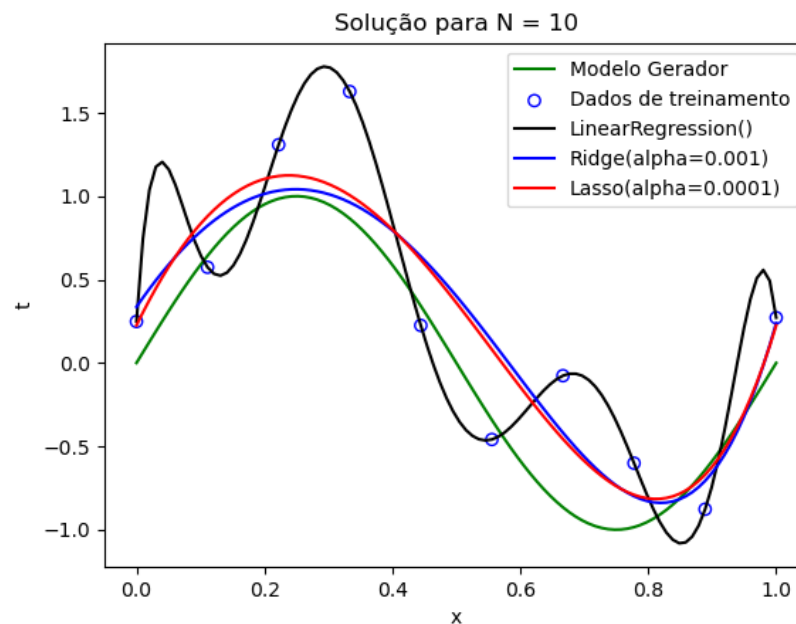
A regressão por mínimos quadrados sem regularização tende a produzir coeficientes maiores, um dos sinais que indicam overfitting ou pelo menos uma alta sensibilidade aos dados de treinamento. Enquanto que os coeficientes produzidos pela Ridge e pela Lasso são menores.

Além disso, é possível observar a presença de alguns coeficientes praticamente nulos para a Lasso. Esse resultado era esperado, uma vez que a Lasso, por utilizar a norma  $L_1$ , tende a produzir uma solução mais esparsa, selecionando atributos mais importantes.

Enquanto isso, a regressão Ridge, apesar de não selecionar atributos como a Lasso, tende a penalizar ainda mais os coeficientes grandes, consequentemente fazendo com que os maiores coeficientes fiquem menores que os produzidos pela Lasso.

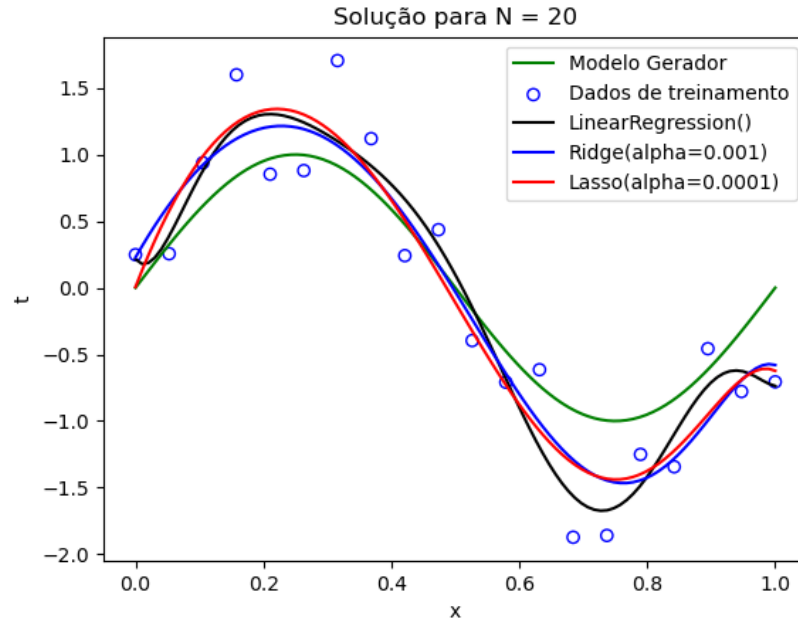
## 1.5 Resposta do item (e)

**Figura 4:** Comparação entre as soluções para  $N = 10$



## 1.6 Resposta do item (f)

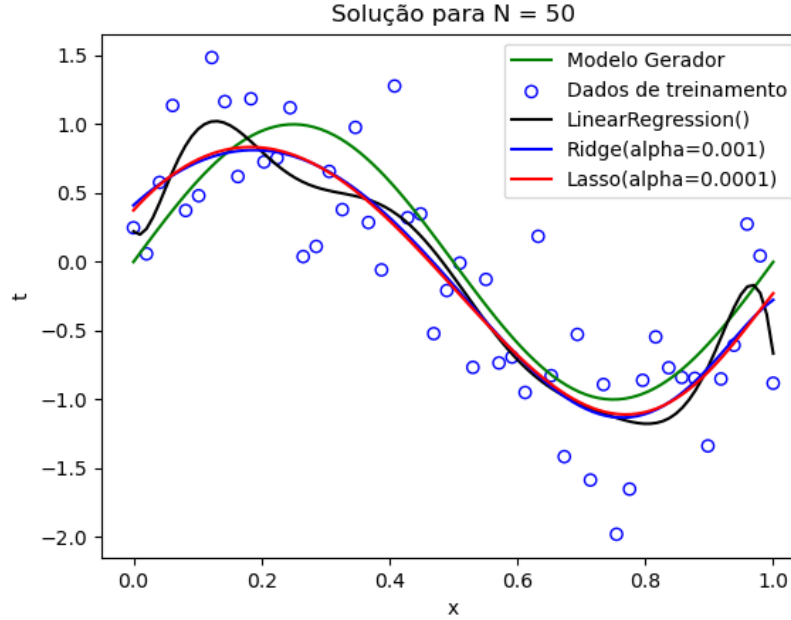
**Figura 5:** Comparação entre as soluções para  $N = 20$



**Tabela 2:** Coeficientes para  $N = 20$

Modelo	LS	Ridge	Lasso
$w_0$	0.00000	0.00000	0.00000
$w_1$	-6.01368	8.39435	12.41611
$w_2$	266.50406	-16.21245	-29.59271
$w_3$	-2042.90037	-8.39927	0.72172
$w_4$	7178.62334	2.98152	10.38544
$w_5$	-13245.77187	9.30126	8.97546
$w_6$	11993.83639	9.76174	4.54743
$w_7$	-3017.56404	5.62434	0.00000
$w_8$	-2424.08815	-1.60172	-1.35976
$w_9$	1296.42241	-10.65716	-6.72254

**Figura 6:** Comparação entre as soluções para  $N = 50$



**Tabela 3:** Coeficientes para  $N = 50$

Modelo	LS	Ridge	Lasso
$w_0$	0.00000	0.00000	0.00000
$w_1$	-6.39374	4.21035	5.11672
$w_2$	455.99223	-10.53780	-14.48266
$w_3$	-5270.52166	-3.22874	0.35582
$w_4$	27669.16750	1.83804	4.73458
$w_5$	-80097.04531	4.61815	4.18727
$w_6$	135776.45582	5.20904	2.22228
$w_7$	-134351.92251	3.52870	0.00000
$w_8$	71926.93850	-0.30771	-0.00000
$w_9$	-16103.55817	-6.01761	-2.73785

Pelas figuras, podemos observar que a solução para mínimos quadrados começa a sofrer menos com o overfitting à medida que o tamanho da amostra aumenta. Isso se deve ao fato de que o ruído nos dados de treinamento possui uma distribuição gaussiana com média zero. Pela lei dos grandes números, quanto maior a quantidade de dados de treinamento, mais a média da amostra se aproxima da média da distribuição original, que é zero, reduzindo assim o efeito do ruído.

Consequentemente, com mais dados, a solução de mínimos quadrados consegue capturar melhor o padrão subjacente dos dados, e o impacto do ruído diminui. Esse comportamento explica por que a solução de mínimos quadrados apresenta um ajuste mais estável e menos propenso ao overfitting quando o tamanho da amostra aumenta.

No entanto, é importante notar que, mesmo com um aumento no tamanho da amostra, métodos de regularização como a regressão ridge e lasso continuam a fornecer soluções mais robustas e generalizáveis, especialmente em situações onde o ruído pode não ser perfeitamente gaussiano ou quando a complexidade do modelo ainda é alta em relação ao tamanho da amostra.



## 2 Exercício 2

Data Source:

- (info) <https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.info.txt>
- (database) <https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data>

“The data for this example come from a study by Stamey et al. (1989). They examined the correlation between the level of prostate-specific antigen (lpsa) and a number of clinical measures in men who were about to receive a radical prostatectomy. The variables are log cancer volume (lcavol), log prostate weight (lweight), age, log of the amount of benign prostatic hyperplasia (lbph), seminal vesicle invasion (svi), log of capsular penetration (lcp), Gleason score (gleason), and percent of Gleason scores 4 or 5 (pgg45).”

Considere a variável (lpsa) como 'target' e as variáveis (lcavol), (lweight), (age), (lbph), (svi), (lcp), (gleason) e (pgg45) como 'entradas'. Siga o roteiro abaixo:

- Padronize os atributos de entrada para que eles tenham média 0 e variância 1;
- Divida o dataset em dois conjuntos, treinamento e teste, conforme indicado nos índices da última coluna (T = treinamento, F = teste);
- Encontre o modelo linear de regressão ótimo no critério de mínimos quadrados (solução LS);
- Implemente modelos lineares regularizados pelos métodos 'Ridge' e 'Lasso' que minimizam a função objetivo  $L(w) = \frac{1}{2N}RSS(w) + \lambda||w||^q$ . Apresente resultados para  $\lambda = 0.25$ ;
- Aplicando as regressões 'Ridge' e 'Lasso' e utilizando k-fold cross-validation, é possível selecionar um valor para  $\lambda$  que resulta em um modelo com melhor capacidade de generalização. Isso é feito selecionando o  $\lambda$  relativo à menor estimativa do erro de predição quadrático médio (usualmente chamado de validation score) ao longo dos k-folds. Também é possível selecionar um valor de  $\lambda$  que seleciona o modelo mais simples dentro de uma tolerância da estimativa do erro de predição quadrático médio. Isso é particularmente útil quando se deseja encontrar soluções esparsas (no caso do Lasso) ou de menor norma L2 (no caso do Ridge). Para tal, um critério comumente adotado é a 'Regra de 1 desvio padrão', onde escolhe-se o maior  $\lambda$  cujo validation score seja igual ou pouco menor do que o 'score mínimo' + '1 desvio padrão do score mínimo'.
  - Monte as curvas de validation score de k-fold cross-validation em função de  $\lambda$  para os modelos regularizados por 'ridge' e 'lasso' (Sugestão: use  $k = 10$ , e procure  $\lambda$  em um intervalo  $[0, 0.5]$ );
  - Calcule o desvio padrão do 'score' mínimo em cada respectiva curva e desenhe-o como barra de erro em torno daquele ponto;
  - Determine o  $\lambda$  que resulta no modelo mais simples de acordo com a 'Regra de 1 desvio padrão';
  - Treine o modelo final 'ridge' e 'lasso' utilizando todos os dados (de treinamento) e o respectivo  $\lambda$  encontrado e apresente os resultados;

- (f) Utilizando o conjunto de teste construído no item (b), calcule a estimativa do erro de predição quadrático médio do conjunto de teste para cada modelo (mínimos quadrados, 'ridge' e 'lasso'). Disserte sobre os resultados obtidos.
- (g) (Bônus) Estime o desvio padrão dos coeficientes do modelo obtido pelo método de bootstrap dos resíduos;

Dica: Veja os slides 9 e 10 de <http://www.est.ufmg.br/~cristianocs/MetComput/Aula8.pdf>

## 2.1 Resposta do item (a)

Para padronizar os dados de entrada com média zero e variância um, foram selecionadas as colunas 'lcvol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason' e 'pgg45' como features e depois utilizada a classe *StandardScaler* da biblioteca *sklearn* para *python*, que realiza a padronização dos dados pelo método *z-score*.

## 2.2 Resposta do item (b)

O dataset foi dividido utilizando a última coluna (T = treinamento, F = teste), como pedido no enunciado.

## 2.3 Resposta do item (c)

Para implementar o método dos mínimos quadrados foi utilizada a classe *LinearRegression* da biblioteca *sklearn* para *python*. Pelo critério de mínimos quadrados (LS) foram obtidos os seguintes resultados ao se calcular o erro quadrático médio:

- MSE - Treinamento (LS): 0.4391997680583344
- MSE - Teste (LS): 0.5212740055076001

## 2.4 Resposta do item (d)

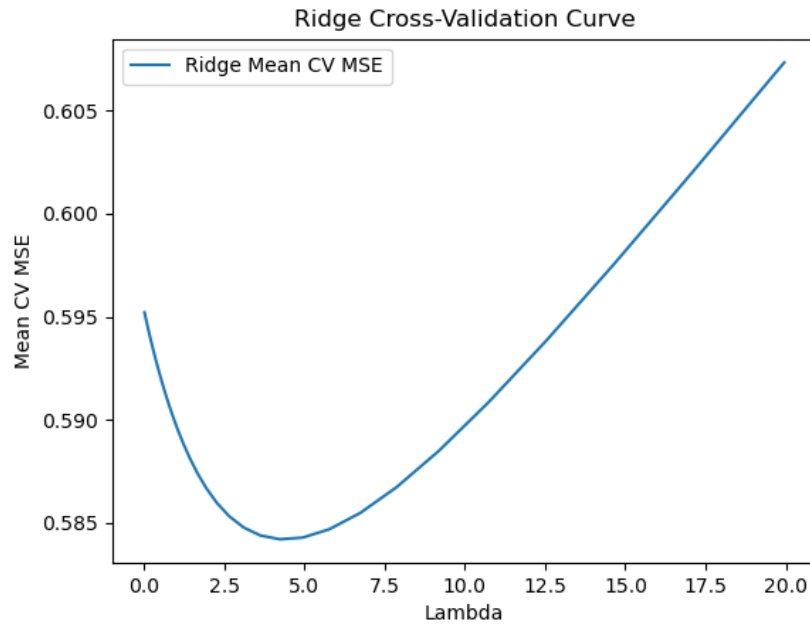
Para implementar o método dos mínimos quadrados foram utilizadas as classe *Ridge* e *Lasso* da biblioteca *sklearn* para *python*, ambos utilizando  $\lambda = 0.25$ . Foram obtidos os seguintes resultados para cada método ao se calcular o erro quadrático médio:

- MSE - Treinamento (Ridge): 0.4392308191154076
- MSE - Teste (Ridge): 0.5189192261819305
- MSE - Treinamento (Lasso): 0.6207140544187021
- MSE - Teste (Lasso): 0.5031909828714028

## 2.5 Resposta do item (e)

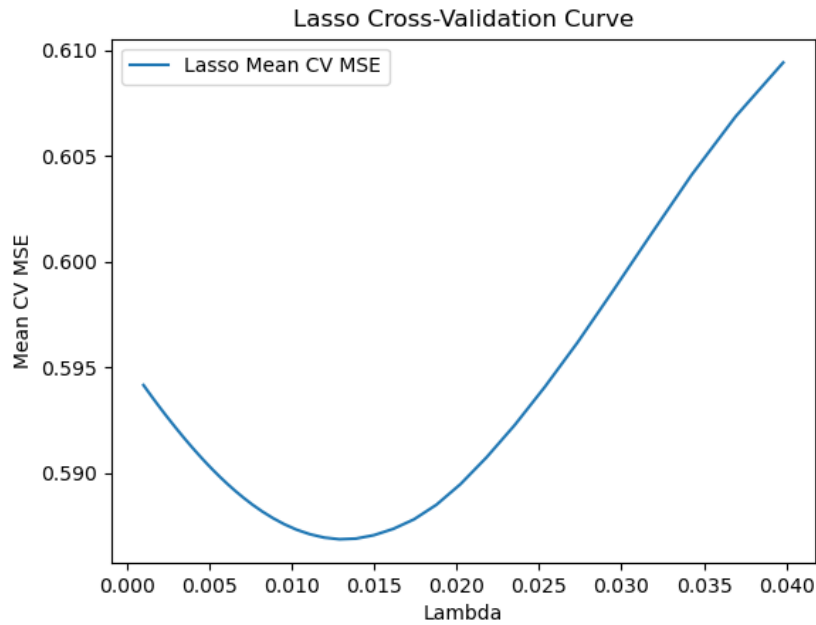
Para implementar o k-fold cross-validation foi utilizada a classe *KFold* e o método *cross\_val\_score* da biblioteca *sklearn* para *python*, utilizando  $k = 10$ . As curvas abaixo mostram os resultados para os dois métodos:

**Figura 7:** Seleção do  $\lambda$  para o Ridge



- Best lambda (Ridge): 284.8035868435802

**Figura 8:** Seleção do  $\lambda$  para o Lasso



- Best lambda (Lasso): 0.4750810162102798

## 2.6 Resposta do item (f)

Utilizando os valores de  $\lambda$  obtidos no item (e) para treinar os modelos e avaliando eles no conjunto de teste, foram obtidos os seguintes resultados:

- MSE - Teste (Final Ridge): 0.6757602963351064
- MSE - Teste (Final Lasso): 0.6427793406214352

## 2.7 Resposta do item (g) (bônus)

Estimando-se o desvio padrão dos modelos utilizando o método de bootstrap foram obtidos os seguintes resultados:

**Tabela 4:** Desvio padrão dos coeficientes estimado pelo método de bootstrap

Coeficiente	LS	Ridge	Lasso
$\sigma(w_{lcavol})$	1.13557895	0.02041017	0.00122013
$\sigma(w_{lweight})$	0.61004347	0.02223836	0.
$\sigma(w_{age})$	1.4497133	0.02089434	0.
$\sigma(w_{lbph})$	0.80596082	0.02063686	0.
$\sigma(w_{svi})$	1.20836005	0.02081217	0.
$\sigma(w_{lcp})$	2.91104245	0.01875707	0.
$\sigma(w_{gleason})$	1.33500253	0.01904607	0.
$\sigma(w_{pgg45})$	1.21153016	0.019532	0.

# Códigos

## Código 1: Exercício 1

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from sklearn.linear_model import LinearRegression, Ridge, Lasso
4 from sklearn.preprocessing import PolynomialFeatures
5 import pandas as pd
6
7 def fitar_curva(x, t, model):
8     X = np.linspace(0,1,100)
9     # regressão
10    poly = PolynomialFeatures(degree=9)
11    A = poly.fit_transform(x.reshape(-1, 1)) # calculo da matriz A
12    model.fit(A, t) # treinamento
13    A = poly.fit_transform(X.reshape(-1, 1))
14    y=model.predict(A) # predição
15
16    return y, model.coef_
17
18 def plotar_curva(x,y,modelo):
19     # função geradora
20     X = np.linspace(0,1,100)
21     modelo_gerador = np.sin(2*np.pi*X)
22
23     # resultados
24     plt.figure()
25     plt.plot(X,modelo_gerador,color='green', label='Modelo Gerador')
26     plt.scatter(x, t, facecolors='none', edgecolors="blue", label = 'Dados de
27         treinamento')
28     plt.plot(X,y,color='red', label = modelo)
29     plt.title(f'Solução para {modelo}')
30     plt.xlabel('x')
31     plt.ylabel('t')
32     plt.legend()
33     plt.show()
34
35 def plotar_tudo(x,y_list,modelos,N):
36     # função geradora
37     X = np.linspace(0,1,100)
38     modelo_gerador = np.sin(2*np.pi*X)
39
40     # resultados
41     plt.figure()
42     plt.plot(X,modelo_gerador,color='green', label='Modelo Gerador')
43     plt.scatter(x, t, facecolors='none', edgecolors="blue", label = 'Dados de
44         treinamento')
45     color_list = ["k", "b", "r"]
46     for i, y in enumerate(y_list):
47         plt.plot(X,y,color=color_list[i],label=modelos[i])
48         # plt.plot(X,y,label=modelos[i])
49     plt.title(f'Solução para N = {N}')
50     plt.xlabel('x')
```

```

49     plt.ylabel('t')
50     plt.legend()
51     plt.show()
52
53 def listar_coefs(coef_list):
54     model_names = ['LS', 'Ridge', 'Lasso']
55
56     coef_dict = {'Modelo': model_names}
57     for i in range(10):
58         coef_dict[f'$w_{i}$'] = [f'{coef[i]:.5f}' for coef in coef_list]
59
60     coef_table = pd.DataFrame(coef_dict)
61     coef_table = coef_table.set_index('Modelo').transpose()
62     print(coef_table.to_latex(index=True))
63
64
65
66 if __name__=="__main__":
67     # amostra
68     np.random.seed(42) # congelando a seed para gerar os mesmos dados de
69                       # treinamento para todos os itens
69     N = 10 # tamanho da amostra
70     x = np.linspace(0,1,N)
71     t = np.sin(2*np.pi*x) + np.random.normal(0, 0.5, size=N)
72     # curve fitting
73     lambda_ridge = 1e-3
74     lambda_lasso = 1e-4
75     modelos = [LinearRegression(), Ridge(alpha=lambda_ridge), Lasso(alpha=
76               lambda_lasso)]
76     # modelos = [Ridge(alpha=1e-1),Ridge(alpha=1e-2),Ridge(alpha=1e-3),Ridge(
77               alpha=1e-4),Ridge(alpha=1e-5),Ridge(alpha=1e-6)]
77     # modelos = [Lasso(alpha=1e-2),Lasso(alpha=1e-3),Lasso(alpha=1e-4),Lasso(
78               alpha=1e-5),Lasso(alpha=1e-6),Lasso(alpha=1e-7)]
78     y_list = list()
79     w_list = list()
80     for model in modelos:
81         y, w = fitar_curva(x, t, model)
82         y_list.append(y)
83         w_list.append(w)
84         plotar_curva(x,y,model)
85
86     listar_coefs(w_list)
87     plotar_tudo(x,y_list,modelos,N)

```

## Código 2: Exercício 2

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.linear_model import LinearRegression, Ridge, Lasso
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.model_selection import train_test_split, cross_val_score, KFold
6 from sklearn.metrics import mean_squared_error
7 import matplotlib.pyplot as plt
8

```

```

9 # Carregar o dataset
10 # data = pd.read_csv('prostate.data', delimiter='\t')
11 url = 'https://hastie.su.domains/ElemStatLearn/datasets/prostate.data'
12 data = pd.read_csv(url, delimiter='\t')
13
14 # (a) Padronização dos atributos de entrada para que eles tenham média 0 e variância 1
15 features = ['lcavol', 'lweight', 'age', 'lbph', 'svi', 'lcp', 'gleason', 'pgg45']
16 X = data[features]
17 y = data['lpsa']
18 scaler = StandardScaler()
19 X_scaled = scaler.fit_transform(X)
20
21 # (b) Divisão o dataset em dois conjuntos, treinamento e teste, conforme indicado nos índices da última coluna
22 train_indices = data['train'] == 'T'
23 test_indices = data['train'] == 'F'
24 X_train, X_test = X_scaled[train_indices], X_scaled[test_indices]
25 y_train, y_test = y[train_indices], y[test_indices]
26
27 # (c) Encontre o modelo linear de regressão ótimo no critério de mínimos quadrados (solução LS)
28 linear_model = LinearRegression()
29 linear_model.fit(X_train, y_train)
30 y_pred_train_linear = linear_model.predict(X_train)
31 y_pred_test_linear = linear_model.predict(X_test)
32 mse_train_linear = mean_squared_error(y_train, y_pred_train_linear)
33 mse_test_linear = mean_squared_error(y_test, y_pred_test_linear)
34 print(f'MSE - Treinamento (LS): {mse_train_linear}')
35 print(f'MSE - Teste (LS): {mse_test_linear}')
36
37 # (d) Implementação modelos lineares regularizados pelos métodos Ridge e Lasso com lambda = 0.25
38 lambda_val = 0.25
39 ridge_model = Ridge(alpha=lambda_val)
40 ridge_model.fit(X_train, y_train)
41 y_pred_train_ridge = ridge_model.predict(X_train)
42 y_pred_test_ridge = ridge_model.predict(X_test)
43 lasso_model = Lasso(alpha=lambda_val)
44 lasso_model.fit(X_train, y_train)
45 y_pred_train_lasso = lasso_model.predict(X_train)
46 y_pred_test_lasso = lasso_model.predict(X_test)
47 mse_train_ridge = mean_squared_error(y_train, y_pred_train_ridge)
48 mse_test_ridge = mean_squared_error(y_test, y_pred_test_ridge)
49 mse_train_lasso = mean_squared_error(y_train, y_pred_train_lasso)
50 mse_test_lasso = mean_squared_error(y_test, y_pred_test_lasso)
51 print(f'MSE - Treinamento (Ridge): {mse_train_ridge}')
52 print(f'MSE - Teste (Ridge): {mse_test_ridge}')
53 print(f'MSE - Treinamento (Lasso): {mse_train_lasso}')
54 print(f'MSE - Teste (Lasso): {mse_test_lasso}')
55
56 # (e) Aplicação do k-fold cross-validation para selecionar o melhor valor de lambda

```



```

57 def plot_cv_curve(model, X, y, lambdas, model_name):
58     # k-fold cross-validation
59     mean_scores = list()
60     std_scores = list()
61     for alpha in lambdas:
62         model.alpha = alpha
63         scores_one_alpha = list()
64         for i in range(1):
65             kf = KFold(n_splits=10, shuffle=True, random_state=i)
66             scores = cross_val_score(model, X, y, cv=kf, scoring='
                neg_mean_squared_error')
67             scores_one_alpha.append(-scores)
68             scores_one_alpha = np.array(scores_one_alpha)
69             mean_scores.append(scores_one_alpha.mean())
70             std_scores.append(scores_one_alpha.std())
71     mean_scores = np.array(mean_scores)
72     std_scores = np.array(std_scores)
73     # Regra de 1 desvio padrão
74     min_score = np.min(mean_scores)
75     min_score_index = np.where(mean_scores == min_score)
76     min_score_std = std_scores[min_score_index]
77     min_score_lambda = lambdas[min_score_index]
78     lambda_1se_index = np.where(mean_scores <= min_score + min_score_std)
79     [0][-1]
80     lambda_1se = lambdas[lambda_1se_index]
81     lambda_1se_score = mean_scores[lambda_1se_index]
82     # plot
83     plt.figure()
84     plt.plot(lambdas, mean_scores, label=f'MSE ({model_name})')
85     plt.scatter(min_score_lambda, min_score, c='r', label='Menor score', zorder
            =3)
86     plt.scatter(lambda_1se, lambda_1se_score, c='g', label='Melhor lambda',
            zorder=3)
87     plt.xlabel('Lambda')
88     plt.ylabel('CV MSE')
89     plt.title(f'Curva de Validação Cruzada do {model_name}')
90     plt.legend()
91     plt.show()
92     return lambda_1se
93
94 best_lambda_ridge = plot_cv_curve(Ridge(), X_train, y_train, np.logspace
    (-2,1.3,50), 'Ridge')
95 best_lambda_lasso = plot_cv_curve(Lasso(), X_train, y_train, np.logspace
    (-3,-1.4,50), 'Lasso')
96
97 print(f'Best lambda (Ridge): {best_lambda_ridge}')
98 print(f'Best lambda (Lasso): {best_lambda_lasso}')
99
100 # (f) Calculando o MSE para o conjunto de teste
101 final_ridge_model = Ridge(alpha=best_lambda_ridge)
102 final_ridge_model.fit(X_train, y_train)
103 final_lasso_model = Lasso(alpha=best_lambda_lasso)
104 final_lasso_model.fit(X_train, y_train)
105 y_pred_test_final_ridge = final_ridge_model.predict(X_test)
106 y_pred_test_final_lasso = final_lasso_model.predict(X_test)

```

```

105 mse_test_final_ridge = mean_squared_error(y_test, y_pred_test_final_ridge)
106 mse_test_final_lasso = mean_squared_error(y_test, y_pred_test_final_lasso)
107 print(f'MSE - Teste (Final Ridge): {mse_test_final_ridge}')
108 print(f'MSE - Teste (Final Lasso): {mse_test_final_lasso}')
109
110 # (g) (Bônus) Estimando o desvio padrão dos coeficientes do modelo pelo método
    de bootstrap dos resíduos
111 def bootstrap_residuals(model, X_train, y_train, n_bootstrap=1000):
112     residuals = y_train - model.predict(X_train)
113     coefs = []
114     for _ in range(n_bootstrap):
115         sampled_residuals = np.random.choice(residuals, size=len(residuals),
            replace=True)
116         y_bootstrap = model.predict(X_train) + sampled_residuals
117         model.fit(X_train, y_bootstrap)
118         coefs.append(model.coef_)
119     coefs = np.array(coefs)
120     return np.std(coefs, axis=0)
121
122 bootstrap_std_linear = bootstrap_residuals(linear_model, X_train, y_train)
123 bootstrap_std_ridge = bootstrap_residuals(final_ridge_model, X_train, y_train)
124 bootstrap_std_lasso = bootstrap_residuals(final_lasso_model, X_train, y_train)
125 print(f'Desvio padrão dos coeficientes (LS): {bootstrap_std_linear}')
126 print(f'Desvio padrão dos coeficientes (Ridge): {bootstrap_std_ridge}')
127 print(f'Desvio padrão dos coeficientes (Lasso): {bootstrap_std_lasso}')

```