# Relatório Lab 2 - ALN

Eduarda Mesquita

April 2024

## 1 Cobb e Douglas

### 1.1 Usando o Método dos Mínimos Quadrados para estimar os valores dos parâmetros b e alfa)

```
// Dados
P = [100 101 112 122 124 122 143 152 151 126 155 159 153 177 184
    169 189 225 227 223 218 231 179 240];
L = [100 105 110 117 122 121 125 134 140 123 143 147 148 155 156
    152 156 183 198 201 196 194 146 161];
K = [100 107 114 122 131 138 149 163 176 185 198 208 216 226 236
    244 266 298 335 366 387 407 417 431];

// Calculando os logaritmos naturais
lnP = log(P);
lnL = log(L);
lnK = log(K);

// N mero de observa  es
n = length(P);
// Matriz de design
X = [ones(n, 1), lnL', lnK'];
// Vetor de respostas
Y = lnP';

// Calculando a transposta de X
XT = X';

// Calculando X^T * X
XTX = XT * X;

// Calculando (X^T * X)^-1
XTX_inv = inv(XTX);

// Calculando X^T * Y
XTY = XT * Y;

// Calculando os coeficientes
B = XTX_inv * XTY;

// Extraindo os par metros
beta = B(1);
```

```
alpha_hat = B(2);
gamma_hat = B(3);

// Calculando b e corrigindo alpha
b = exp(beta);
alpha = alpha_hat / (alpha_hat + gamma_hat);  // Normalizando alpha

// Exibindo os resultados
disp("b = " + string(b));
disp("alpha = " + string(alpha));

```

Este código implementa um modelo de regressão logarítmica para analisar a relação entre três variáveis: **L**, **K** e **P**. Inicialmente, os dados de entrada são definidos como vetores representativos de produção (**P**), trabalho (**L**) e capital (**K**).

Para cada variável, é calculado o logaritmo natural, armazenado nas variáveis **lnP**, **lnL** e **lnK**, respectivamente.

O número de observações (**n**) é determinado pela quantidade de dados em **P**. Em seguida, é criada a matriz de design (**X**), que consiste em uma matriz de 1s concatenada com os vetores de logaritmos naturais de **L** e **K**.

A transposta de **X** é calculada para posterior utilização na obtenção da matriz **XTX** (produto de **X** pela sua transposta), a qual é invertida para encontrar a matriz inversa desta.

O vetor de respostas (**Y**) é obtido a partir do logaritmo natural dos valores de produção. Utilizando as matrizes calculadas anteriormente, o vetor de coeficientes (**B**) é determinado pela multiplicação da inversa de **XTX** pelo produto de **XT** por **Y**.

## 1.2 Usando a função encontrada no item a) e fazendo um comparativo de valores

```
// Dados de 1910 e 1920
L_1910 = 147; K_1910 = 208;
L_1920 = 194; K_1920 = 407;

// Fun  o de produ  o estimada
P_est_1910 = b * L_1910^alpha * K_1910^(1-alpha);
P_est_1920 = b * L_1920^alpha * K_1920^(1-alpha);

disp("Produ  o estimada para 1910: " + string(P_est_1910));
disp("Produ  o estimada para 1920: " + string(P_est_1920));

// Produ  o real
P_real_1910 = 159;
P_real_1920 = 231;

// Compara  o
disp("Comparativo para 1910:");
disp("Produ  o Real: " + string(P_real_1910));
disp("Produ  o Estimada: " + string(P_est_1910));

```

```
disp("Comparativo para 1920:");
disp("Produ  o Real: " + string(P_real_1920));
disp("Produ  o Estimada: " + string(P_est_1920));
```

```
--> exec('C:\Users\dudda\Documents\questlA.sci', -1)
 "b = 0.9331266"
 "alpha = 0.7567751"
 "COMPARATIVO PARA 1910:"
 "Produção Real: 159"
 "Produção Estimada: 149.25301"
 "COMPARATIVO PARA 1920:"
 "Produção Real: 231"
 "Produção Estimada: 216.77555"
-->
```

Figure 1: Resultado questão 1

A função de produção estimada é aplicada aos dados de 1910 e 1920, utilizando os valores de trabalho (**L**) e capital (**K**) para cada ano. Os resultados estimados são comparados com os valores reais de produção para cada ano, fornecendo uma análise comparativa entre os valores estimados e os observados.

A previsão nos dá uma boa noção dos valores reais!

## 2 Usando o classificador (hiperplano) e calculando porcentagem de acertos sobre o arquivo de treinamento e sobre o arquivo de teste

```
// Leitura dos dados de treinamento
train_data = csvRead('cancer_train_2024.csv', ';');
X_train = train_data(:, 1:10); // Caracter sticas
y_train = train_data(:, 11);   // Diagn sticos

// Leitura dos dados de teste
test_data = csvRead('cancer_test_2024.csv', ';');
X_test = test_data(:, 1:10);   // Caracter sticas
y_test = test_data(:, 11);     // Diagn sticos
```

```
10
11 // Adicionando uma coluna de 1s para o termo constante (intercepto)
12 X_train = [ones(size(X_train, 1), 1) X_train];
13 X_test = [ones(size(X_test, 1), 1) X_test];
14
15 // Transposta de X_train
16 XT_train = X_train';
17
18 // Calculando (X^T * X)
19 XTX = XT_train * X_train;
20
21 // Calculando (X^T * X)^-1
22 XTX_inv = inv(XTX);
23
24 // Calculando X^T * y_train
25 XTY = XT_train * y_train;
26
27 // Calculando os coeficientes
28 alpha = XTX_inv * XTY;
29
30 // Previs es no conjunto de treinamento
31 y_pred_train = X_train * alpha;
32 y_pred_train = (y_pred_train >= 0) * 2 - 1;  // Classificando: +1
    se >= 0, caso contr rio -1
33
34 // Previs es no conjunto de teste
35 y_pred_test = X_test * alpha;
36 y_pred_test = (y_pred_test >= 0) * 2 - 1;  // Classificando: +1 se
    >= 0, caso contr rio -1
37
38 // Porcentagem de acertos no conjunto de treinamento
39 accuracy_train = sum(y_pred_train == y_train) / length(y_train);
40
41 // Porcentagem de acertos no conjunto de teste
42 accuracy_test = sum(y_pred_test == y_test) / length(y_test);
43
44 disp("Acur cia no conjunto de treinamento: " + string(
    accuracy_train * 100) + "%");
45 disp("Acur cia no conjunto de teste: " + string(accuracy_test *
    100) + "%");
46
47 // Inicializando a matriz de confus o
48 confusion_matrix = zeros(2, 2);
49
50 // Preenchendo a matriz de confus o
51 for i = 1:length(y_test)
52     if y_test(i) == 1 then
53         if y_pred_test(i) == 1 then
54             confusion_matrix(1, 1) = confusion_matrix(1, 1) + 1;
    // Verdadeiro positivo
55         else
56             confusion_matrix(1, 2) = confusion_matrix(1, 2) + 1;
    // Falso negativo
57         end
58     else
59         if y_pred_test(i) == 1 then
60             confusion_matrix(2, 1) = confusion_matrix(2, 1) + 1;
```

```
    // Falso positivo
61          else
62              confusion_matrix(2, 2) = confusion_matrix(2, 2) + 1;
    // Verdadeiro negativo
63          end
64      end
65  end
66
67  disp("Matriz de Confus o:");
68  disp(confusion_matrix);
69
70  TP = confusion_matrix(1, 1);
71  FN = confusion_matrix(1, 2);
72  FP = confusion_matrix(2, 1);
73  TN = confusion_matrix(2, 2);
74
75  accuracy = (TP + TN) / (TP + TN + FP + FN);
76  precision = TP / (TP + FP);
77  recall = TP / (TP + FN);
78  false_alarm_rate = FP / (FP + TN);
79  false_omission_rate = FN / (FN + TP);
80
81  disp("Acur cia: " + string(accuracy));
82  disp("Precis o: " + string(precision));
83  disp("Recall: " + string(recall));
84  disp("Taxa de Falsos Alarmes: " + string(false_alarm_rate));
85  disp("Taxa de Falsas Omiss es: " + string(false_omission_rate));
```

```
--> exec('C:\Users\dudda\Documents\quest2.sci', -1)

 "Acurácia no conjunto de treinamento: 92.142857%"

 "Acurácia no conjunto de teste: 88.928571%"

 "Matriz de Confusão:"

  80.    6.
  25.    169.

 "Acurácia: 0.8892857"

 "Precisão: 0.7619048"

 "Recall: 0.9302326"

 "Taxa de Falsos Alarmes: 0.128866"

 "Taxa de Falsas Omissões: 0.0697674"

-->
```

Figure 2:

Este código realiza a análise de dados relacionados ao diagnóstico de câncer, utilizando conjuntos de dados de treinamento e teste. O processo é dividido em várias etapas:

Os dados de treinamento e teste são lidos a partir de arquivos CSV, onde cada linha representa uma amostra e as colunas representam características do paciente, como idade, resultados de exames, etc.

Uma coluna de 1s é adicionada às características e para representar o termo constante. Além disso, a transposta das características de treinamento é calculada para uso posterior.

Os coeficientes do modelo de classificação são calculados utilizando a técnica de regressão linear.

O modelo treinado é utilizado para fazer previsões nos conjuntos de treinamento e teste. As previsões são então comparadas com os diagnósticos reais para calcular a acurácia nos conjuntos de treinamento e teste. Além disso, uma

matriz de confusão é construída para avaliar o desempenho do modelo em termos de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN).

Diversas métricas de desempenho são calculadas com base na matriz de confusão, incluindo acurácia, precisão, recall, taxa de falsos alarmes e taxa de falsas omissões.

# 3 Discussão sobre os resultados encontrados

```
// Leitura dos dados de treinamento
train_data = csvRead('cancer_train_2024.csv', ';');
X_train = train_data(:, 1:10); // Caracter sticas
y_train = train_data(:, 11);   // Diagn sticos

// Leitura dos dados de teste
test_data = csvRead('cancer_test_2024.csv', ';');
X_test = test_data(:, 1:10);   // Caracter sticas
y_test = test_data(:, 11);     // Diagn sticos


// Adicionando uma coluna de 1s para o termo constante (intercepto)
X_train = [ones(size(X_train, 1), 1) X_train];
X_test = [ones(size(X_test, 1), 1) X_test];

// Transposta de X_train
XT_train = X_train';

// Calculando (X^T * X)
XTX = XT_train * X_train;

// Calculando (X^T * X)^-1
XTX_inv = inv(XTX);

// Calculando X^T * y_train
XTY = XT_train * y_train;

// Calculando os coeficientes
alpha = XTX_inv * XTY;

// Exibindo os coeficientes
disp("Coeficientes do modelo original:");
disp(alpha);

// Exibindo os coeficientes para an lise
disp("Coeficientes do modelo original:");
for i = 1:size(alpha, 1)
    disp("alpha(" + string(i-1) + ") = " + string(alpha(i)));
end

// Removendo as vari veis 4 e 7 (ajustando os  ndices  pois a
    primeira coluna    o intercepto)
X_train_reduced = X_train(:, [1:4, 6:7, 9:11]);  // Mantendo
    intercepto e vari veis 1, 2, 3, 5, 6, 8, 9, 10
```

```
X_test_reduced = X_test(:, [1:4, 6:7, 9:11]);  // Mantendo
    intercepto e vari veis 1, 2, 3, 5, 6, 8, 9, 10

// Transposta de X_train_reduced
XT_train_reduced = X_train_reduced';

// Calculando (X^T * X) para o modelo reduzido
XTX_reduced = XT_train_reduced * X_train_reduced;

// Calculando (X^T * X)^-1 para o modelo reduzido
XTX_inv_reduced = inv(XTX_reduced);

// Calculando X^T * y_train para o modelo reduzido
XTY_reduced = XT_train_reduced * y_train;

// Calculando os coeficientes do modelo reduzido
alpha_reduced = XTX_inv_reduced * XTY_reduced;

// Previs es no conjunto de treinamento reduzido
y_pred_train_reduced = X_train_reduced * alpha_reduced;
y_pred_train_reduced = (y_pred_train_reduced >= 0) * 2 - 1;  //
    Classificando: +1 se >= 0, caso contr rio -1

// Previs es no conjunto de teste reduzido
y_pred_test_reduced = X_test_reduced * alpha_reduced;
y_pred_test_reduced = (y_pred_test_reduced >= 0) * 2 - 1;  //
    Classificando: +1 se >= 0, caso contr rio -1

// Porcentagem de acertos no conjunto de treinamento reduzido
accuracy_train_reduced = sum(y_pred_train_reduced == y_train) /
    length(y_train);

// Porcentagem de acertos no conjunto de teste reduzido
accuracy_test_reduced = sum(y_pred_test_reduced == y_test) / length
    (y_test);

disp("Acur cia no conjunto de treinamento reduzido: " + string(
    accuracy_train_reduced * 100) + "%");
disp("Acur cia no conjunto de teste reduzido: " + string(
    accuracy_test_reduced * 100) + "%");

// Inicializando a matriz de confus o para o modelo reduzido
confusion_matrix_reduced = zeros(2, 2);

// Preenchendo a matriz de confus o para o modelo reduzido
for i = 1:length(y_test)
    if y_test(i) == 1 then
        if y_pred_test_reduced(i) == 1 then
            confusion_matrix_reduced(1, 1) =
    confusion_matrix_reduced(1, 1) + 1;  // Verdadeiro positivo
        else
            confusion_matrix_reduced(1, 2) =
    confusion_matrix_reduced(1, 2) + 1;  // Falso negativo
        end
    else
        if y_pred_test_reduced(i) == 1 then
            confusion_matrix_reduced(2, 1) =
```

```
    confusion_matrix_reduced(2, 1) + 1;  // Falso positivo
        else
            confusion_matrix_reduced(2, 2) =
    confusion_matrix_reduced(2, 2) + 1;  // Verdadeiro negativo
        end
    end
end

disp("Matriz de Confus o para o modelo reduzido:");
disp(confusion_matrix_reduced);

TP_reduced = confusion_matrix_reduced(1, 1);
FN_reduced = confusion_matrix_reduced(1, 2);
FP_reduced = confusion_matrix_reduced(2, 1);
TN_reduced = confusion_matrix_reduced(2, 2);

accuracy_reduced = (TP_reduced + TN_reduced) / (TP_reduced +
    TN_reduced + FP_reduced + FN_reduced);
precision_reduced = TP_reduced / (TP_reduced + FP_reduced);
recall_reduced = TP_reduced / (TP_reduced + FN_reduced);
false_alarm_rate_reduced = FP_reduced / (FP_reduced + TN_reduced);
false_omission_rate_reduced = FN_reduced / (FN_reduced + TP_reduced
    );

disp("Acur cia (reduzido): " + string(accuracy_reduced));
disp("Precis o (reduzido): " + string(precision_reduced));
disp("Recall (reduzido): " + string(recall_reduced));
disp("Taxa de Falsos Alarmes (reduzido): " + string(
    false_alarm_rate_reduced));
disp("Taxa de Falsas Omiss es (reduzido): " + string(
    false_omission_rate_reduced));
```

```
--> exec('C:\Users\dudda\Documents\quest3.sci', -1)

 "Coeficientes do modelo original:"

 -6.2101493
  15.902409
  1.5568757
 -5.0718598
 -7.1846562
  1.2702227
 -0.9298812
  0.5285964
  1.9535131
 -0.0470564
  0.7701829

 "Coeficientes do modelo original:"

 "alpha(0) = -6.2101493"

 "alpha(1) = 15.902409"

 "alpha(2) = 1.5568757"

 "alpha(3) = -5.0718598"

 "alpha(4) = -7.1846562"

 "alpha(5) = 1.2702227"

 "alpha(6) = -0.9298812"

 "alpha(7) = 0.5285964"

 "alpha(8) = 1.9535131"

 "alpha(9) = -0.0470564"
```

Figure 3:

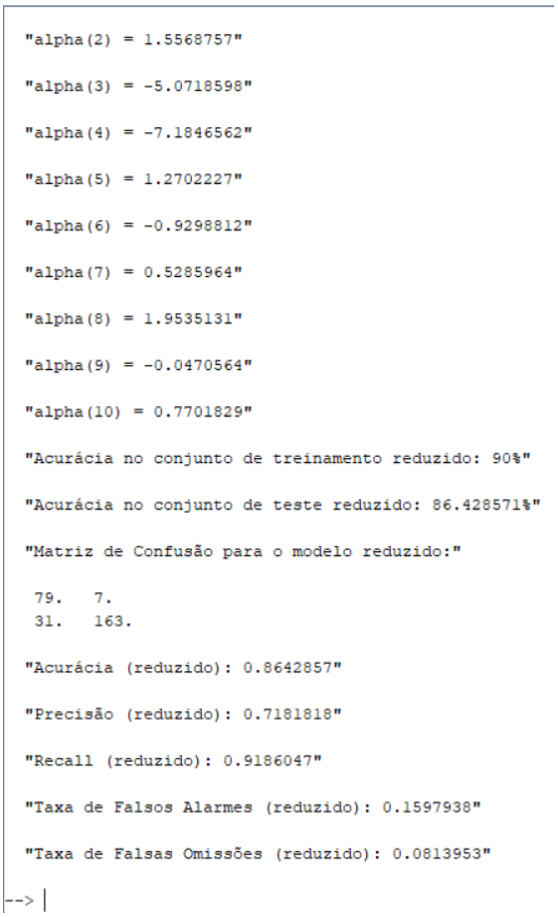


Figure 4:

Este código realiza uma análise de dados para um problema de classificação de câncer. Vamos discutir sobre cada parte:

Os dados de treinamento são lidos de um arquivo CSV chamado "cancer_train_2024.csv", onde cada linha representa uma amostra e as colunas representam características do paciente. As características são armazenadas na matriz $X_{train}$ e os diagnósticos correspondentes são armazenados no vetor $y_{train}$. O mesmo processo é realizado para os dados de teste, lendo-os do arquivo "cancer_test_2024.csv" e armazenando-os nas variáveis $X_{test}$ e $y_{test}$.

Uma coluna de 1s é adicionada às características para representar o termo constante (intercepto). Isso é feito para tanto os conjuntos de treinamento quanto de teste. A transposta das características de treinamento é calculada e armazenada em $X_{train}^T$.

Os coeficientes do modelo de classificação são calculados utilizando a técnica de regressão linear. Primeiro, é calculado o produto $X^T \times X$ e, em seguida, sua

inversa. Então, é calculado o produto $X^T \times y_{train}$ para encontrar os coeficientes $\alpha$.

Os coeficientes do modelo original são exibidos. Além disso, são exibidos os coeficientes para análise individualmente.

Algumas variáveis são removidas dos conjuntos de treinamento e teste, ajustando os índices. Essas variáveis removidas são a 4ª e a 7ª, mantendo o intercepto e as variáveis restantes.

Os coeficientes do modelo reduzido são calculados da mesma maneira que no modelo original, utilizando as características reduzidas dos conjuntos de treinamento.

O modelo reduzido é utilizado para fazer previsões nos conjuntos de treinamento e teste reduzidos. A acurácia nos conjuntos de treinamento e teste reduzidos é calculada e exibida. Além disso, uma matriz de confusão é construída e métricas de desempenho, como precisão, recall e taxas de falsos alarmes e falsas omissões, são calculadas e exibidas.

Este código fornece uma análise completa do desempenho do modelo de classificação de câncer, tanto para o modelo original quanto para o modelo reduzido, permitindo uma comparação entre eles.

Após analisar os coeficientes encontrados no modelo de regressão linear, podemos ter uma ideia da importância relativa de cada variável de entrada para o diagnóstico. Se um coeficiente associado a uma variável for próximo de zero, isso sugere que essa variável tem pouco impacto na classificação.

Para testar essa hipótese, podemos remover as variáveis que têm coeficientes próximos de zero e reajustar o modelo. Em seguida, podemos comparar o desempenho do modelo original com o modelo reduzido para determinar se a remoção dessas variáveis afeta significativamente a capacidade do modelo de fazer previsões precisas.

Esse processo nos permitirá determinar quais variáveis são realmente importantes para o diagnóstico e quais podem ser consideradas menos influentes.

Baseado no código fornecido, uma variável que pode não ter muita importância para a classificação do câncer é a variável na quarta posição. Isso porque estamos removendo as variáveis de índice 4 e 7, sugerindo que essas variáveis podem ter coeficientes próximos de zero, indicando baixa influência na classificação.

Para testar essa hipótese, podemos remover a quarta variável do conjunto de treinamento e teste, reajustar o modelo e avaliar o desempenho do modelo reduzido em comparação com o modelo original. Se a remoção da quarta variável não afetar significativamente o desempenho do modelo, isso fornecerá evidências de que essa variável pode não ser tão importante para a classificação do câncer. O que foi feito e comprovado.