Questão 1 -

Inicialmente devemos modelar o problema. Temos uma função de base, da forma:

$$P = b \cdot L^{\alpha} \cdot K^{1-\alpha}$$

Então, tomamos o **logaritmo natural** em ambos os lados, ficando com:

$$ln(P) = ln(b) + \alpha ln(L) + (1 - \alpha) ln(K)$$

Agora, retomaremos nossas variáveis, apenas para elucidar a explicação:

$$Y = ln(P), X_1 = ln(L), X_2 = ln(K)$$

Resultando na seguinte equação:

$$Y = ln(b) + \alpha X_1 + (1 - \alpha) X_2$$

Portanto, fica claro que chegamos a uma equação linear, cujo termo independente é $ln(b)$ e possui as variáveis X_1 e X_2 com coeficientes α e $\beta = (1 - \alpha)$. Agora, podemos usar os mínimos quadrados para achar as melhores soluções para $ln(b)$, α *e* β. Então, fui ao scilab criar uma função para aproximar a solução usando o método dos mínimos quadrados, que se dá pela resolução seguinte equação:

$$A^T A\bar{x} = A^T b$$

Onde A = $\begin{bmatrix} 1 & X_1 & X_2 \end{bmatrix}$, para cada linha de dados e contém o resultado real observado.

A função que resolve esse sistema em scilab ficou da seguinte forma:

```
////////////////////////////////////////////////////////////////////////
//Variáveis de saída:
//alpha: solução do sistema A_tAx=A_tb (assumimos que tal solução existe).
////////////////////////////////////////////////////////////////////////
function alpha = minimos_quadrados(X, b)

    // Calcular (X^T * X)
    X_transp_X = X' * X;

    // Calcular (X^T * y)
    X_transp_b = X' * b;

    alpha = gaussian_elimination(X_transp_X, X_transp_b);

    endfunction
```

Agora, basta usarmos os dados fornecidos no enunciado e realizar nossa modelagem no scilab, então, poderemos usar nossa função para achar $ln(b)$, α *e* β.

```
--> P = [100, 101, 112, 122, 124, 122, 143, 152, 151, 126, 155, 159,

--> L = [100, 105, 110, 117, 122, 121, 125, 134, 140, 123, 143, 147,

--> K = [100, 107, 114, 122, 131, 138, 149, 163, 176, 185, 198, 208,

--> lnP = log(P); //Y = log(P)

--> lnL = log(L); //X_1 = log(L)

--> lnK = log(K); //X_2 = log(K)

--> A = [ones(lnP') lnL' lnK'];

--> y = lnP';

--> alphas = minimos_quadrados(A, y)
 alphas  =

  -0.0692144
   0.7688664
   0.2471110
```

Note que P, L e K continuam para além do print. O que importa aqui é que chegamos aos valores requeridos. Pela nossa modelagem, $\bar{x} = [ln(b)\ \alpha\ \beta]^T$, então sabemos que:

$$ln(b) = -0,0692; \alpha = 0,7688; \beta = 0,2472 \approx 1 - \alpha$$

Aqui, nos deparamos com um problema. Como modelamos β independente de α, chegamos a um resultado diferente do esperado, pois vemos que β não é exatamente 1 - α, como proposto no problema inicial. Isso ocorreu exatamente pela forma que utilizamos para resolver o problema, usando uma variável β que não existe. Então, pare adequar a equação base, usaremos apenas o α, que é uma variável existente na equação de base. Com isso, ficamos com:

```
"Os parametros aprendidos foram: b = 0.9331266 e alpha = 0.7688664"
"A função de produção estimada é P = 0.9331266 * L ^ 0.7688664 * K ^ 0.2311336"
```

Aqui, você deve ter notado uma mudança no valor de b. Isso ocorreu já que calculamos $ln(b)$, contudo, precisamos de b. Portanto, fazemos $e^{ln(b)} = b$, por isso a diferença.

Agora, podemos achar os valores propostos na questão b, sendo eles referentes aos anos de 1910 e 1920. Para isso, basta substituir os valores correspondentes na equação encontrada:

```
--> L_1910 = 147;
--> K_1910 = 208;
--> L_1920 = 194;
--> K_1920 = 407;
--> P_1910_est = b * L_1910 ^ alpha * K_1910 ^ (1 - alpha);
--> P_1920_est = b * L_1920 ^ alpha * K_1920 ^ (1 - alpha);
--> disp("Produção estimada para 1910: " + string(P_1910_est) + " (Produção real: 159)");
  "Produção estimada para 1910: 148.62791 (Produção real: 159)"
--> disp("Produção estimada para 1920: " + string(P_1920_est) + " (Produção real: 231)");
  "Produção estimada para 1920: 214.8421 (Produção real: 231)"
```

Para melhor visualização, focaremos no resultado dessas linhas de código:

```
"Produção estimada para 1910: 148.62791 (Produção real: 159)"
"Produção estimada para 1920: 214.8421 (Produção real: 231)"
```

Aqui, podemos comparar o valor real, com o valor obtido. Vemos que para o ano de 1910, nosso modelo previu 148,62, enquanto o valor real seria 159, um erro absoluto de aproximadamente 10. Enquanto para 1920, nosso modelo previu 214.84, enquanto o valor real é 231, com erro absoluto de aproximadamente 16.

Por mais que seja um erro grande, esse erro já era esperado, pois como dito no próprio enunciado “[...] existirem muitos outros fatores afetando o desempenho da economia”, ou seja, o dado contém muitos ruídos em relação a equação base. Além disso, pensando em um modelo generalista, o fit perfeito aos dados seria ruim, já que previriamos muito bem os dados já usados, mas muito mal novos dados, que é a ideia da equação proposta.

Analisando o uso do β, descartado pela natureza do problema, vemos que conseguimos chegar mais próximo dos dados reais:

```
"A função de produção estimada é P = 0.9331266 * L ^ 0.7688664 * K ^ 0.247111"
"Produção estimada para 1910: 161.85909 (Produção real: 159)"
"Produção estimada para 1920: 236.49068 (Produção real: 231)"
```

Agora, observamos erros de 3 e 5, que são bem menores que 10 e 16 encontrados anteriormente. Contudo, como já comentado durante o relatório, não usaremos esses resultados, já que não seguem a modelagem proposta e não há garantias que um menor erro nesses resultados levará a um melhor modelo para previsões futuras.

Questão 2 -

Para realização dessa tarefa, será usada nossa função de mínimos quadrados criada na questão anterior e precisaremos construir uma função para realizar a predição de acordo com o plano encontrado, sendo ela construída da seguinte forma:

```
// Função de classificação
function y_pred = classificar(X, weights)
    y_pred = X * weights; // Determina o hiperplano
    y_pred = sign(y_pred); // Classifica baseado no sinal de h(x)
endfunction
```

Sendo os “weights” apreendidos usando a função de mínimos quadrados. Além disso, precisamos de funções para calcular a matriz de confusão e as métricas pedidas, sendo elas construídas da seguinte forma:

```
////////////////////////////////////////////////////////////////////////////
//Variáveis de saída:
//matriz_confusao: [tp, fp;
//                  fn, tn]
//onde:
//tp = Verdadeiros Positivos (True Positives)
//fp = Falsos Positivos (False Positives)
//fn = Falsos Negativos (False Negatives)
//tn = Verdadeiros Negativos (True Negatives)
////////////////////////////////////////////////////////////////////////////
function matriz_confusao = calcular_matriz_confusao(y_true, y_pred)
    tp = sum((y_true == 1) & (y_pred == 1));
    tn = sum((y_true == -1) & (y_pred == -1));
    fp = sum((y_true == -1) & (y_pred == 1));
    fn = sum((y_true == 1) & (y_pred == -1));

    matriz_confusao = [tp, fp; fn, tn];
endfunction
```

Aqui, calculamos as taxas necessárias para criar a matriz de confusão e organizamos esses números para que sejam mostrados na organização correta. Tudo isso, baseando-se nas previsões do modelo e os dados reais correspondentes. Também temos a função para as métricas, sendo que essa função recebe a matriz de confusão resultante da função anterior:

```
////////////////////////////////////////////////////////////////////////
//Variáveis de saída:
//precisao: Precision (precisao)
//recall: Recall (sensibilidade)
//fpr: False positive rate (probabilidade de falso alarme)
//fnr: False negative rate (probabilidade de falsa omissão de alarme)
////////////////////////////////////////////////////////////////////////
function [precisao, recall, fpr, fnr] = calcular_metricas(matriz_confusao)
    tp = matriz_confusao(1, 1);
    fp = matriz_confusao(1, 2);
    fn = matriz_confusao(2, 1);
    tn = matriz_confusao(2, 2);

    precisao = tp / (tp + fp);
    recall = tp / (tp + fn);
    fpr = fp / (fp + tn);
    fnr = fn / (fn + tp);
endfunction
```

Aqui, basicamente usamos as taxas já calculadas para criar métricas conhecidas no machine learning, como precision, recall, false negative rate e false positive rate.

Agora, com essas funções em mãos, podemos criar um código para carregar os dados, aprender os pesos, realizar as previsões e calcular as métricas que queremos. Não irei explicar parte a parte desse código, mas segue um print de como ele ficou:

```
// Carregar os dados de treinamento
dados_treinamento = csvRead('data\cancer_train_2024.csv', ';');

// Separação das colunas de features e o alvo
X_train = dados_treinamento(:, 1:10);
y_train = dados_treinamento(:, 11);

// Adicionar a coluna de 1s para o bias
X_train = [ones(size(X_train, 1), 1), X_train];

// Calcular os coeficientes usando minimos quadrados
weights = minimos_quadrados(X_train, y_train);
```

```
// Carregar os dados de teste
dados_teste = csvRead('data/cancer_test_2024.csv', ';');
X_test = dados_teste(:, 1:10);
y_test = dados_teste(:, 11);

// Adicionar a coluna de 1s para o bias
X_test = [ones(size(X_test, 1), 1), X_test];

// Classificar os dados de treinamento e teste
y_train_pred = classificar(X_train, weights);
y_test_pred = classificar(X_test, weights);

// Acurácia
acuracia_treinamento = sum(y_train == y_train_pred) / length(y_train);
acuracia_teste = sum(y_test == y_test_pred) / length(y_test);

// Matriz de confusao
matriz_confusao = calcular_matriz_confusao(y_test, y_test_pred);

[precisao, recall, fpr, fnr] = calcular_metricas(matriz_confusao);

// Exibir os resultados
disp("Matriz de Confusão:");
disp(matriz_confusao);
disp("--------------------------------");
disp("Acurácia no conjunto de treinamento: " + string(acuracia_treinamer
disp("Acurácia no conjunto de teste: " + string(acuracia_teste));
disp("--------------------------------");
disp("Precisão: " + string(precisao));
disp("Recall: " + string(recall));
disp("Probabilidade de Falso Alarme (FPR): " + string(fpr));
disp("Probabilidade de Falsa Omissão (FNR): " + string(fnr));
```

Executando ele, temos os seguintes resultados:

```
"Matriz de Confusão:"

 80.   25.
 6.    169.

"------------------------------"

"Acurácia no conjunto de treinamento: 0.9214286"

"Acurácia no conjunto de teste: 0.8892857"

"------------------------------"

"Precisão: 0.7619048"

"Recall: 0.9302326"

"Probabilidade de Falso Alarme (FPR): 0.128866"

"Probabilidade de Falsa Omissão (FNR): 0.0697674"
```

Vamos agora analisar parte a parte dessas medidas. Inicialmente, vamos focar na acurácia, comparando o desempenho no conjunto de treino e de teste. Vemos que houve uma maior acurácia no conjunto de treino, o que já era esperado, mas o interessante aqui é que o desempenho no teste não foi muito abaixo do treino, dando indícios de uma boa generalização do modelo. Mas essa métrica sozinha não é suficiente para uma completa análise do modelo que criamos. A partir de agora, as métricas são referentes ao conjunto de teste.

Analisando a matriz de confusão, ela nos mostrou que dos 280 registros, 80 foram corretamente classificados como positivo e 169 foram corretamente classificados como negativos, isso significa que o modelo acertou em 249 das 280 previsões (88,9%). Também vemos que 6 foram erroneamente classificados como negativo e 25 erroneamente classificados como positivos. Contudo, devemos ter em mente que as classes não são balanceadas. Então, para entendermos melhor esses casos, vamos agora analisar nossas outras métricas.

A precisão (76,19%) indica a porcentagem dos classificados como positivos que foram corretamente classificados, o recall (93,02%) indica a porcentagem dos positivos reais que foram classificados como positivos. Isso pode ser confuso em um primeiro momento, mas, de maneira geral, a precisão nos indica qual a probabilidade de você estar com câncer se você for classificado como positivo, enquanto o recall indica a porcentagem de pessoas com câncer que vão ser

identificadas. Com isso, vemos que um bom recall é interessante, já que queremos prever o câncer na maior parte de pessoas possíveis. Uma precisão mais baixa não é tão preocupante, já que prever erroneamente que alguém tem câncer, não tem um impacto tão significativo quanto não identificar a doença, já que exames médicos posteriores corrigiriam esse erro. Já a probabilidade de falso alarme (12,89%) ou seja, dizer que alguém que não tem câncer o tem, obteve um resultado que, dado a característica do problema, não é tão preocupante. Já a probabilidade de false omissão é algo mais seŕio no nosso contexto, e o resultado de 6,98% foi um ótimo resultado levando em conta a simplicidade do modelo empregado.

Essas métricas nos mostram que, apesar de muito simples, o modelo construído consegue performar bem para a tarefa proposta, levando em conta o seu contexto. As diferenças entre a acurácia no treino e no teste, bem como as outras métricas no conjunto de teste, mostraram que o modelo obteve uma boa capacidade de generalização.

Questão 3 -

Para identificar a relevância das features e entender quais delas são mais importantes para o diagnóstico precisamos (1) analisar a escala dos dados, (2) observar quais coeficientes com os maiores valores em módulo. Ao observar os dados, vemos que eles se encontram normalizados, ou seja, todos estão entre 0 e 1. Dessa forma, podemos olhar diretamente para os coeficientes do plano,levando em consideração que o primeiro é o bias:

```
"Coeficientes:"

"-6.2101493: bias"

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

Aqui, quanto maior o coeficiente em módulo, mais a feature será relevante para resolução do problema. Então, vemos que a 1º (15,9), 4º (-7,18) e a 3º (-5,07) são as 3 features mais importantes. Podemos dizer ainda que a 9º feature é irrelevante, já que seu coeficiente é -0,04. Poderíamos tirar apenas essa feature e analisar o resultado, mas não teria graça, então vamos reduzir o número de features para 6 e ver o que acontece. Para isso, devemos tirar as 4 features menos relevantes, sendo elas a 6º, 7º, 9º e 10º. Fazendo isso e treinando novamente o modelo obtemos:

```
"Matriz de Confusão:"

 81.   36.
 5.    158.

"------------------------------"

"Acurácia no conjunto de treinamento: 0.9178571"

"Acurácia no conjunto de teste: 0.8535714"

"------------------------------"

"Precisão: 0.6923077"

"Recall: 0.9418605"

"Probabilidade de Falso Alarme (FPR): 0.185567"

"Probabilidade de Falsa Omissão (FNR): 0.0581395"
```

Comparando com o resultado anterior, ao tirar essas 4 features, os resultados se alteram levemente. Tivemos uma piora na acurácia, nos falsos positivos, na precisão e na probabilidade de falso alarme. Contudo, uma leve melhora no recall e na probabilidade de falsa omissão. Apesar disso, essa diferença é pouco significativa, apesar de retirarmos quase metade dos dados presentes. Como efeito, chegamos a um modelo bem mais simples, mas que (praticamente) manteve a sua performance.

Visto isso, concluímos que nem todas as variáveis de entrada são igualmente relevantes. Também, vimos que nesse problema, temos uma variável que é irrelevante (9º) e outras que são pouco relevantes, a ponto de que a sua exclusão não afeta significativamente o desempenho do modelo.

Gustavo Tironi