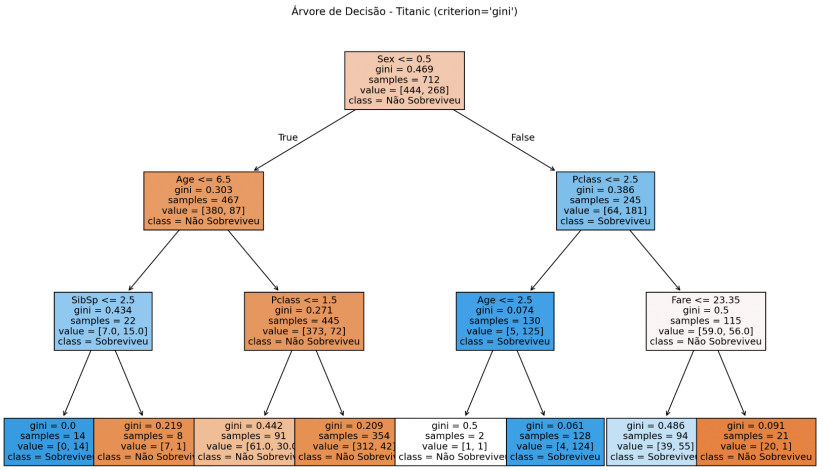
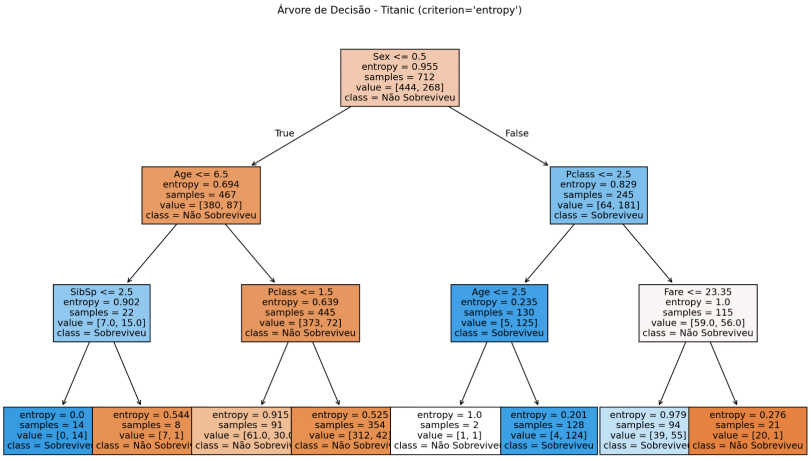
1. https://github.com/acmachado48/IA/blob/main/Lista%203/Q1.ipynb






```
🔷 Treinando Árvore de Decisão com criterion='entropy' 🔷
Acurácia na validação: 0.7989
Matriz de Confusão:
 [[92 13]
 [23 51]]
2025-03-08 19:52:00.235 Python[21513:1465696] +[IMKClient subclass]: chose IMKClient_Modern
2025-03-08 19:52:00.235 Python[21513:1465696] +[IMKInputSession subclass]: chose IMKInputSession_Modern

🔷 Treinando Árvore de Decisão com criterion='gini' 🔷
Acurácia na validação: 0.7989
Matriz de Confusão:
 [[92 13]
 [23 51]]

✅ Arquivo de submissão gerado: submission_entropy.csv
meu_ambienteanacarolinamachado@Anas-MacBook-Pro IA %
```

Ambas as árvores têm estrutura similar, isso indica que os critérios chegaram a divisões parecidas. As primeiras divisões são idênticas: a variável "Sex" é usada como raiz.

As duas árvores resultaram em uma acurácia de 79.89%, mostrando que, nesse caso, o critério de divisão não impactou significativamente a precisão do modelo. A matriz de confusão indica que os erros de classificação foram praticamente os mesmos.

2. max_depth

Se muito baixo, a árvore pode ficar muito simples e sofrer de *underfitting* (baixo desempenho). Se muito alto, a árvore pode memorizar os dados e sofrer de *overfitting*.

max_features

Define quantas colunas serão consideradas ao procurar a melhor divisão.

Se muito baixo, pode reduzir o poder preditivo da árvore. Se muito alto, pode levar ao *overfitting* por capturar padrões específicos demais.

min_samples_leaf

Se baixo, pode gerar folhas muito pequenas e *overfitting*. Se alto, força nós a terem mais exemplos, reduzindo a complexidade da árvore.

min_samples_split

Se baixo, a árvore cresce muito (muito específica). Se alto, a árvore fica menor e pode perder padrões importantes.

criterion

"`gini`": mede a impureza de Gini (quantos erros de classificação a divisão criaria).

"`entropy`": mede a entropia (grau de desordem dos dados antes e depois da divisão).

Gini costuma ser mais rápido e é preferido para grandes conjuntos de dados. Entropy pode ser mais interpretável, pois está relacionado à teoria da informação.

3. https://github.com/acmachado48/IA/blob/main/Lista%203/Q3.ipynb

O resultado mostra que GridSearchCV e RandomizedSearchCV encontraram soluções muito próximas, com a mesma acurácia. BayesSearchCV, apesar de ser mais eficiente computacionalmente, encontrou uma solução um pouco pior.

```
  train_df["Embarked"].fillna(train_df["Embarked"].mode()[0], inplace=True)
🔹 Melhor combinação GridSearchCV: {'criterion': 'gini', 'max_depth': 3, 'min_samples_leaf': 5, 'min_samples_split': 2}
Acurácia na validação: 0.7988826815642458
🔹 Melhor combinação RandomizedSearchCV: {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 7, 'min_samples_split': 12}
Acurácia na validação: 0.7988826815642458
🔹 Melhor combinação BayesSearchCV: OrderedDict({'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 17, 'min_samples_split': 11})
Acurácia na validação: 0.7932960893854749
meu_ambienteanacarolinamachado@Anas-MacBook-Pro_iA %
```

4. A) Apenas I e II

5. A) Apenas I e II

6. O algoritmo C4.5 é uma evolução do ID3, as principais diferenças são o suporte a atributos contínuos, ID3 só lida com atributos discretos, já o C4.5 pode lidar com atributos contínuos, criando pontos de corte para dividir os dados.

O ID3 não trata valores ausentes. Se houver dados faltando, precisam ser preenchidos ou descartados. C4.5 consegue trabalhar com valores ausentes, estimando as probabilidades das classes com base nos dados disponíveis.

ID3 não implementa poda, o que pode resultar em overfitting. C4.5 realiza poda reduzindo ramos irrelevantes e tornando a árvore mais generalizável. Além disso, o C4.5 pode transformar a árvore em uma estrutura binária, facilitando sua interpretação e otimização.

7. O Ganho de Informação, utilizado pelo ID3, mede a redução na entropia ao dividir os dados com um atributo, mas tende a favorecer atributos com muitos valores únicos, como ID ou CPF, o que pode resultar em divisões ruins.

Razão de Ganho, utilizado pelo C4.5, ajusta o ganho de informação levando em conta a quantidade de valores distintos do atributo. Isso penaliza atributos com muitos valores diferentes, tornando a seleção mais equilibrada.