```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from skopt import BayesSearchCV
from skopt.space import Integer, Categorical

# Carregar os dados
train_data = pd.read_csv(r'Lista 3\train.csv')
test_data = pd.read_csv(r'Lista 3\test.csv')
gender_submission = pd.read_csv(r'Lista 3\gender_submission.csv')

# Pré-processamento básico
def preprocess_data(df):
    df = df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)  #
Remover colunas irrelevantes
    df['Age'].fillna(df['Age'].median(), inplace=True)  # Preencher
valores faltantes em 'Age'
    df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)  #
Preencher valores faltantes em 'Embarked'
    df['Fare'].fillna(df['Fare'].median(), inplace=True)  # Preencher
valores faltantes em 'Fare'
    df = pd.get_dummies(df, columns=['Sex', 'Embarked'],
drop_first=True)  # Codificar variáveis categóricas
    return df

train_data = preprocess_data(train_data)
test_data = preprocess_data(test_data)

# Definir features e target
X = train_data.drop('Survived', axis=1)
y = train_data['Survived']

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Questão 01: Árvore de decisão com critério Entropy e Gini
def questao_01():
    # Árvore com critério Entropy
    tree_entropy = DecisionTreeClassifier(criterion='entropy',
random_state=42)
```

```
    tree_entropy.fit(X_train, y_train)
    y_pred_entropy = tree_entropy.predict(X_test)
    print("Acurácia (Entropy):", accuracy_score(y_test,
y_pred_entropy))

    # Árvore com critério Gini
    tree_gini = DecisionTreeClassifier(criterion='gini',
random_state=42)
    tree_gini.fit(X_train, y_train)
    y_pred_gini = tree_gini.predict(X_test)
    print("Acurácia (Gini):", accuracy_score(y_test, y_pred_gini))


# Questão 02: Impacto de outros hiperparâmetros
def questao_02():
    # Testar diferentes valores para max_depth, max_features e
min_samples_leaf
    params = {
        'max_depth': [3, 5, 7, None],
        'max_features': ['sqrt', 'log2', None],
        'min_samples_leaf': [1, 2, 4]
    }

    tree = DecisionTreeClassifier(random_state=42)
    grid_search = GridSearchCV(tree, params, cv=5, scoring='accuracy')
    grid_search.fit(X_train, y_train)

    print("\nMelhores hiperparâmetros encontrados:")
    print(grid_search.best_params_)
    print("Acurácia com melhores hiperparâmetros:",
grid_search.best_score_)


# Questão 03: Otimizadores de hiperparâmetros
def questao_03():
    # GridSearchCV
    param_grid = {
        'max_depth': [3, 5, 7, None],
        'min_samples_leaf': [1, 2, 4],
        'criterion': ['gini', 'entropy']
    }
    grid_search = GridSearchCV(DecisionTreeClassifier(), param_grid,
cv=5, scoring='accuracy')
```

```
    grid_search.fit(X_train, y_train)
    print("\nMelhores parâmetros (GridSearchCV):",
grid_search.best_params_)
    print("Acurácia (GridSearchCV):", grid_search.best_score_)

    # RandomizedSearchCV
    randomized_search = RandomizedSearchCV(DecisionTreeClassifier(),
param_grid, cv=5, n_iter=10, scoring='accuracy', random_state=42)
    randomized_search.fit(X_train, y_train)
    print("\nMelhores parâmetros (RandomizedSearchCV):",
randomized_search.best_params_)
    print("Acurácia (RandomizedSearchCV):",
randomized_search.best_score_)

    # BayesSearchCV
    bayes_search = BayesSearchCV(
        DecisionTreeClassifier(),
        {
            'max_depth': Integer(3, 10),
            'min_samples_leaf': Integer(1, 5),
            'criterion': Categorical(['gini', 'entropy'])
        },
        cv=5,
        n_iter=10,
        scoring='accuracy',
        random_state=42
    )
    bayes_search.fit(X_train, y_train)
    print("\nMelhores parâmetros (BayesSearchCV):",
bayes_search.best_params_)
    print("Acurácia (BayesSearchCV):", bayes_search.best_score_)

# Executar as questões
print("=== Questão 01 ===")
questao_01()

print("\n=== Questão 02 ===")
questao_02()

print("\n=== Questão 03 ===")
questao_03()
```

Questão 01
Acurácia (Entropy): 0.7574626865671642
Acurácia (Gini): 0.746268656716418

Explicação do critério Gini:
O critério Gini mede a impureza de um nó. Ele calcula a probabilidade de um elemento ser classificado incorretamente se for escolhido aleatoriamente.
Fórmula do Gini: Gini = 1 - Σ(p_i)^2, onde p_i é a proporção de elementos da classe i no nó.

Questão 02
Melhores hiperparâmetros encontrados:
{'max_depth': 3, 'max_features': None, 'min_samples_leaf': 4}
Acurácia com melhores hiperparâmetros: 0.8105290322580647
Discussão dos hiperparâmetros:
1. max_depth: Controla a profundidade máxima da árvore. Valores menores evitam overfitting.
2. max_features: Limita o número de features consideradas para dividir um nó. Pode melhorar a generalização.
3. min_samples_leaf: Define o número mínimo de amostras em uma folha. Aumentar esse valor pode reduzir overfitting.

Questão 03
Melhores parâmetros (GridSearchCV): {'criterion': 'entropy', 'max_depth': 3, 'min_samples_leaf': 2}
Acurácia (GridSearchCV): 0.8122193548387099
Melhores parâmetros (RandomizedSearchCV): {'min_samples_leaf': 4, 'max_depth': 5, 'criterion': 'gini'}
Acurácia (RandomizedSearchCV): 0.8122322580645163
Melhores parâmetros (BayesSearchCV): OrderedDict({'criterion': 'entropy', 'max_depth': 8, 'min_samples_leaf': 2})
Acurácia (BayesSearchCV): 0.8154322580645161
PS C:\Users\Felipe\Desktop\Faculdade\Inteligencia artifical>

Questão 04
A alternativa correta é A)

Questão 05
A alternativa correta é A)

Questão 06
As diferenças entre C45 e ID3 são:
Tratamento de atributos contínuos: O C45 pode lidar com atributos contínuos, enquanto o ID3 só lida com atributos categóricos.
Podagem (pruning): O C45 realiza podagem para evitar overfitting, enquanto o ID3 não.
Uso de Razão de Ganho (Gain Ratio): O C45 usa Gain Ratio para evitar viés em atributos com muitos valores.

Questão 07

Ganho de Informação: Mede a redução na entropia após a divisão de um nó. Pode ser tendencioso para atributos com muitos valores.

Razão de Ganho (Gain Ratio): Corrige o viés do Ganho de Informação ao normalizá-lo pela entropia intrínseca do atributo.