Questão 01

Considerando-se os códigos
(Lendo_e_tratando_arquivo_v2.ipynb e DecisionTree.ipynb)
disponibilizados no CANVAS, pede-se:

Encontrar o padrão de pessoas que sobreviveram ao desastre do TITANIC, que matou mais de 1.500 pessoas em 1912.

1. Pré-processamento e Codificação dos Dados

Tratamento de Valores Ausentes

- **Idade (`Age`)**: Preenchida com a mediana da coluna.
- **Porto de Embarque (`Embarked`)**: Preenchido com a moda (valor mais frequente).
- **Tarifa (`Fare`)**: Preenchida com a mediana (caso existam valores ausentes na base de teste).

Codificação One-Hot

- **Sexo (`Sex`)**: Transformado em `Sex_male` (1 para masculino, 0 para feminino).
- **Porto de Embarque (`Embarked`)**: Transformado em `Embarked_Q` e `Embarked_S` (referência: `Embarked_C`).
- **Classe (`Pclass`)**: Transformado em `Pclass_2` e `Pclass_3` (referência: `Pclass_1`).

Colunas Removidas

- `PassengerId`, `Name`, `Ticket`, `Cabin` (não relevantes para o modelo).

O padrão foi, mulheres, pessoas que estavam na classe alta, pagaram a maior tarifa e com idade menor ou igual a 6.5 anos.

```
Importância dos Atributos:
    Feature  Importance
   Sex_male    0.631540
   Pclass_3    0.158172
       Fare    0.107148
        Age    0.058023
      SibSp    0.045117
      Parch    0.000000
 Embarked_Q    0.000000
 Embarked_S    0.000000
   Pclass_2    0.000000
```

```
                precision    recall  f1-score

Não Sobreviveu       0.82      0.89      0.86
    Sobreviveu       0.80      0.69      0.74

      accuracy                           0.82
```

```python
def preprocess_data(df, is_train=True):
    Clique para recolher o intervalo.

    # Preenchimento de valores ausentes
    df['Age'].fillna(df['Age'].median(), inplace=True)
    df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
    if 'Fare' in df.columns:
        df['Fare'].fillna(df['Fare'].median(), inplace=True)

    # Removendo colunas desnecessárias
    drop_cols = ['PassengerId', 'Name', 'Ticket', 'Cabin']
    df.drop(columns=[col for col in drop_cols if col in df.columns], inplace=True)

    # Aplicando one-hot encoding para variáveis categóricas
    df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
    df = pd.get_dummies(df, columns=['Pclass'], drop_first=True)

    # Se for conjunto de treino, separa a variável target
    if is_train and 'Survived' in df.columns:
        X = df.drop('Survived', axis=1)
        y = df['Survived']
        return X, y
    else:
        return df
```

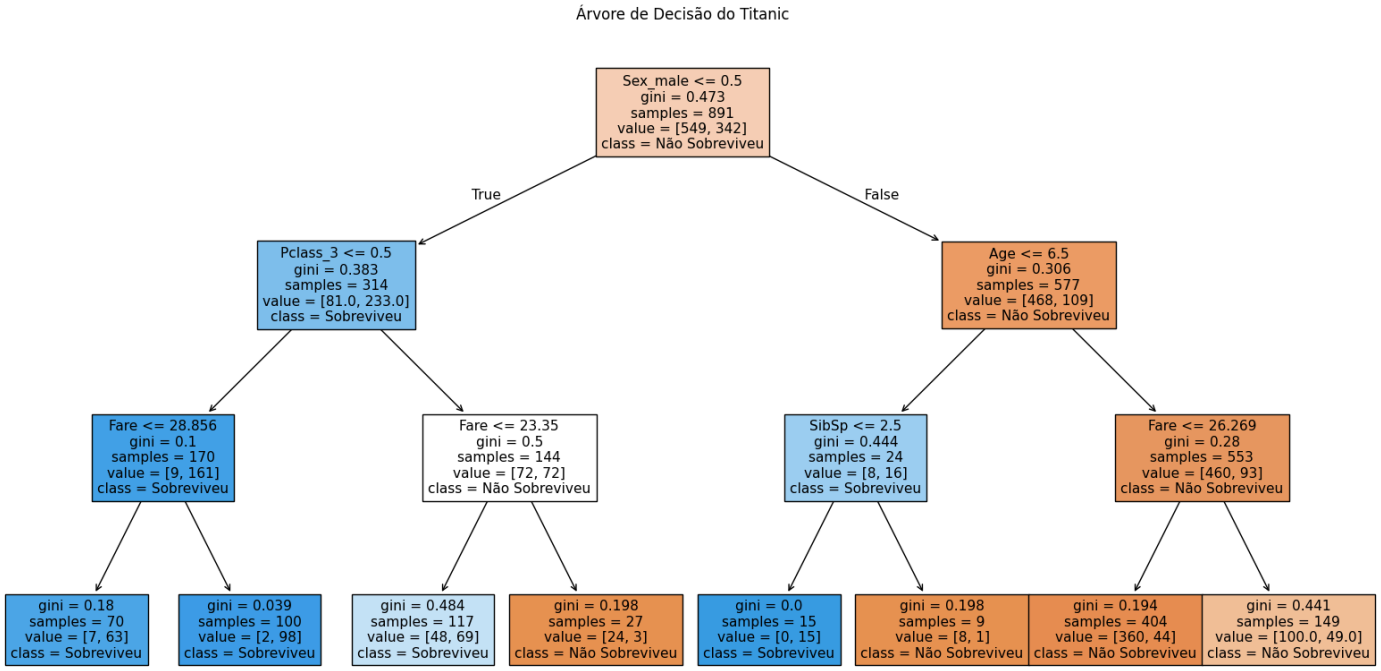
```python
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

```python
if 'Survived' in test_df.columns:
    y_test = test_df['Survived']
    if 'Survived' in X_test.columns:
        X_test = X_test.drop('Survived', axis=1)
else:
    y_test = None

# Garantindo que X_test tenha as mesmas features que X_train
X_test = X_test.reindex(columns = X_train.columns, fill_value=0)

# Treinando o classificador de árvore de decisão com restrições para limitar a altura
clf = DecisionTreeClassifier(
    random_state=42,
    max_depth=3,  # Define uma altura máxima da árvore
    min_samples_split=15,  # Pelo menos 10 amostras para dividir um nó
    min_samples_leaf=5  # Pelo menos 5 amostras em cada folha
)
clf.fit(X_train, y_train)

# Realizando predições na base de teste
y_pred = clf.predict(X_test)
```




```
print("\n\n=== Relatório de Classificação na Base de Treino ===")
y_pred_train = clf.predict(X_train)
print(classification_report(y_train, y_pred_train, target_names=['Não Sobreviveu', 'Sobreviveu']))

# Se a base de teste tiver rótulos
if y_test is not None:
    print("\n\n=== Relatório de Classificação na Base de Teste ===")
    print(classification_report(y_test, y_pred, target_names=['Não Sobreviveu', 'Sobreviveu']))
else:
    # Caso não tenha rótulos no teste, use validação cruzada
    from sklearn.model_selection import cross_val_predict
    print("\n\n=== Relatório com Validação Cruzada (5 folds) ===")
    y_pred_cv = cross_val_predict(clf, X_train, y_train, cv=5)
    print(classification_report(y_train, y_pred_cv, target_names=['Não Sobreviveu', 'Sobreviveu']))
```

```
# Plotando a árvore de decisão
plt.figure(figsize=(20,10))
plot_tree(clf, feature_names=X_train.columns, class_names=['Não Sobreviveu', 'Sobreviveu'], filled=True)
plt.title("Árvore de Decisão do Titanic")
plt.show()
```

```
print("Importância dos Atributos:")
print(importance_df.to_string(index=False))
```

Questão 02

Leia o artigo “A_comparative_study_of_decision_tree_ID3_and_C4.5.pdf” que está no CANVAS e responda:

1) Quais as diferenças entre os algoritmos de árvore ID3 e C4.5?

Os algoritmos ID3 e C4.5 são métodos para a construção de árvores de decisão, mas o C4.5 trouxe várias melhorias em relação ao ID3. O ID3, por exemplo, funciona bem para dados categóricos, mas tem algumas limitações. Ele não lida

bem com valores ausentes, não trabalha diretamente com atributos numéricos e pode ser enviesado ao escolher atributos com muitos valores distintos.

Já o C4.5 veio para resolver esses problemas. Ele consegue trabalhar com dados numéricos transformando-os em intervalos, ou seja, ao invés de lidar com valores contínuos como 10, 20 ou 30, ele divide os dados em faixas, como "menor ou igual a 15" e "maior que 15". Assim, o algoritmo consegue tomar decisões mais eficazes. Além disso, o C4.5 tem um mecanismo para lidar com dados faltantes, distribuindo os exemplos proporcionalmente entre os valores conhecidos. Outra vantagem é que ele
usa uma métrica chamada **gain ratio**, que ajuda a evitar o viés para atributos com muitos valores distintos.

2) Como o algoritmo C4.5 lida com os atributos de entrada que são numéricos? Primeiro, ele ordena todos os valores numéricos daquele atributo. Depois, identifica possíveis "pontos de corte" para dividir os dados em dois grupos – por exemplo, "menor ou igual a 50" e "maior que 50". Para cada ponto de corte, ele calcula qual deles separa melhor as classes e escolhe o que traz o maior ganho de informação.

Esse processo permite que o C4.5 lide com números de forma eficiente, sem precisar transformá-los manualmente em categorias antes do treinamento. Isso faz com que a árvore de decisão fique mais flexível e consiga se adaptar melhor a diferentes tipos de dados.

Questão 03

Com base nestas informações, qual as saídas da árvore para os seguintes registros de teste, respectivamente?

Afirmativa (C)

Questão 04

Considerando a árvore da questão anterior, e as seguintes

afirmações: I. Esta árvore possui 5 regras de classificação

II. Das regras geradas, há apenas uma com cobertura por classe de 100%

III. A menor cobertura por classe é de 6.8% e corresponde à classe

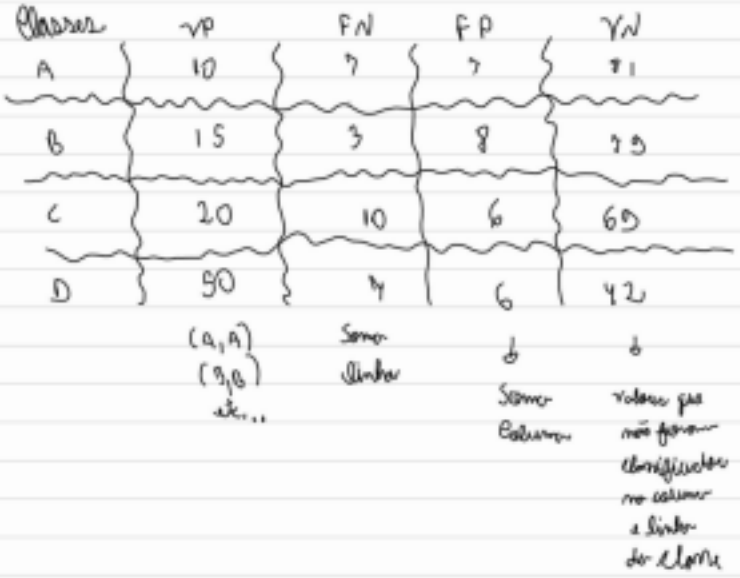
Iris_Virgínica

É correto o que se afirma em:
c) I e II, apenas.

1- Possui 5 folhas, 2 - Há apenas uma de 100%, 3 - Existe uma menor com 2,7%

Questão 05

Imagem na prox pag



	Precisão	Recall	F1Score	TVP	TFN	TFP	TVN
A	0.58	0.58	0.58	0.58	0.41	0.06	0.93
B	0.65	0.83	0.73	0.83	0.16	0.07	0.92
C	0.76	0.66	0.71	0.66	0.33	0.06	0.9398
D	0.89	0.87	0.88	0.87	0.122	0.09	0.9077