Trabalho 3 - Ciência de Dados
Análise de Desempenho do Algoritmo de Árvore de Decisão

Prof. Carlos Cesar Mansur Tuma

Augusto Ortigoso Barbosa
Carlos Eduardo Limírio Posso
João Pedro Araújo Magalhães

Monte Carmelo - MG
2025

Conteúdo

1 Introdução **4**

2 Descrição do Problema **4**

3 Metodologia e Ferramentas **4**
3.1 Ferramentas Utilizadas 4
3.2 Justificativa da Abordagem Programática 5

4 Descrição das Bases de Dados **5**

5 Tarefas Executadas e Análise de Resultados **6**
5.1 Base de Dados: Breast 6
5.1.1 Pré-processamento e Configuração 6
5.1.2 Código Executado 6
5.1.3 Resultados 7
5.1.4 Análise 8
5.2 Base de Dados: Covertype 8
5.2.1 Pré-processamento e Configuração 8
5.2.2 Código Executado 8
5.2.3 Resultados 9
5.2.4 Análise 10
5.3 Base de Dados: Diabets 10
5.3.1 Pré-processamento e Configuração 10
5.3.2 Código Executado 10
5.3.3 Resultados 11
5.3.4 Análise 12
5.4 Base de Dados: Heart 12
5.4.1 Pré-processamento e Configuração 12
5.4.2 Código Executado 12
5.4.3 Resultados 13
5.4.4 Análise 13
5.5 Base de Dados: Iris 13
5.5.1 Pré-processamento e Configuração 14
5.5.2 Código Executado 14
5.5.3 Resultados 14
5.5.4 Análise 15
5.6 Base de Dados: Spam 15
5.6.1 Pré-processamento e Configuração 15
5.6.2 Código Executado 16
5.6.3 Resultados 17
5.6.4 Análise 17
5.7 Base de Dados: Titanic 17
5.7.1 Pré-processamento e Configuração 17
5.7.2 Código Executado 18
5.7.3 Resultados 19
5.7.4 Análise 19
5.8 Base de Dados: Wine 19

5.8.1 Pré-processamento e Configuração 19
5.8.2 Código Executado 20
5.8.3 Resultados 21
5.8.4 Análise 21

6 Análise Comparativa **21**

7 Visualização dos Resultados **22**

8 Conclusão **23**

Resumo

Este trabalho apresenta uma análise detalhada do desempenho do algoritmo de classificação de Árvore de Decisão em oito bases de dados distintas, abrangendo diferentes domínios e tipos de dados, como numéricos, categóricos e textuais. Utilizando a linguagem Python e a biblioteca Scikit-learn, o modelo foi aplicado em cada base de dados, e a sua performance foi avaliada de forma robusta através da técnica de validação cruzada estratificada com 10 partições. Os resultados demonstram uma grande variação no desempenho do algoritmo, com a acurácia a variar de 60,18% a 99,51%, evidenciando a sua alta sensibilidade às características intrínsecas de cada conjunto de dados. Conclui-se que a Árvore de Decisão é altamente eficaz em problemas com padrões bem definidos e atributos preditivos, mas enfrenta desafios em cenários com alta sobreposição de classes ou que exigem um pré-processamento de texto mais complexo.

1 Introdução

O campo da Ciência de Dados e Aprendizagem de Máquina tem demonstrado um impacto profundo em diversas áreas do conhecimento, oferecendo ferramentas capazes de extrair padrões e realizar previsões a partir de grandes volumes de dados. Dentre as várias tarefas existentes, a classificação se destaca como uma das mais fundamentais, consistindo em categorizar uma nova observação em um conjunto de classes pré-definidas.

O presente trabalho tem como objetivo central a aplicação e avaliação de um dos algoritmos de classificação mais conhecidos e interpretáveis: a Árvore de Decisão. Para tal, o algoritmo será executado sobre um conjunto heterogéneo de oito bases de dados, cada uma representando um problema de classificação distinto.

A metodologia empregada utilizará a linguagem Python, com o auxílio das bibliotecas Pandas para manipulação de dados e Scikit-learn para a implementação do modelo. A performance do classificador será avaliada de forma robusta através da técnica de validação cruzada estratificada com 10 partições, garantindo que os resultados sejam uma estimativa fidedigna da sua capacidade de generalização. Ao final, será realizada uma análise comparativa do desempenho do algoritmo em cada um dos cenários propostos.

2 Descrição do Problema

A tarefa proposta consiste em executar o algoritmo de Árvore de Decisão, análogo ao J48 do Weka, em oito bases de dados fornecidas, definindo as configurações que se julgarem adequadas. O desempenho do modelo deve ser avaliado utilizando a técnica de validação cruzada com 10 partições (10-fold cross-validation). O objetivo final é realizar uma análise detalhada dos resultados e comparar a performance do algoritmo em cada uma das bases de dados, identificando os seus pontos fortes e fracos em diferentes contextos de problema.

3 Metodologia e Ferramentas

3.1 Ferramentas Utilizadas

Para a execução deste trabalho, foi adotada uma abordagem programática utilizando a linguagem **Python**, devido à sua flexibilidade e ao robusto ecossistema de bibliotecas para Ciência de Dados. As principais ferramentas foram:

- **Jupyter Notebook:** Utilizado como ambiente de desenvolvimento interativo para a exploração, limpeza e modelagem dos dados.
- **Pandas:** Biblioteca fundamental para a leitura, manipulação e pré-processamento dos dados em formato de DataFrames.
- **Scikit-learn:** A principal biblioteca de Aprendizagem de Máquina utilizada, fornecendo a implementação do algoritmo 'DecisionTreeClassifier', as ferramentas para validação cruzada ('StratifiedKFold') e as métricas de avaliação de desempenho.

3.2 Justificativa da Abordagem Programática

A intenção inicial era utilizar a ferramenta Weka para a análise, conforme a sua popularidade em ambientes académicos. Os testes iniciais com bases de dados bem formatadas e puramente numéricas, como *Breast* e *Heart*, foram bem-sucedidos, produzindo resultados válidos conforme o esperado.

No entanto, ao avançar para os conjuntos de dados mais complexos, surgiram desafios técnicos significativos que limitaram a viabilidade do uso exclusivo do Weka. Especificamente, a base de dados *Covertype*, com mais de 580.000 instâncias, excedeu a memória padrão alocada para a Máquina Virtual Java, resultando em erros de 'OutOfMemoryError'. Adicionalmente, as bases de dados que continham texto livre ou formatação complexa de CSV, como *Titanic*, *Spam* e *Wine*, geraram erros persistentes de leitura e parsing ('unable to determine structure'), mesmo após tentativas de correção manual do formato.

Diante destes obstáculos, a migração para uma abordagem programática com Python foi a decisão mais pragmática e eficaz. Esta abordagem ofereceu o controlo granular necessário para:

1. Ler e corrigir ficheiros CSV com formatação inconsistente através das robustas capacidades de parsing da biblioteca Pandas.
2. Realizar pré-processamentos complexos, como a vetorização de texto com 'TfidfVectorizer' para as bases de *Spam* e *Wine*.
3. Implementar estratégias de amostragem para lidar com bases de dados muito grandes, como a *Covertype*.
4. Garantir um fluxo de trabalho consistente e reprodutível para todas as oito bases de dados, superando as limitações da ferramenta inicial.

Desta forma, a utilização do Python e das suas bibliotecas não só resolveu os problemas técnicos, como também permitiu uma análise mais profunda e customizada para cada desafio apresentado pelos dados.

4 Descrição das Bases de Dados

As oito bases de dados utilizadas neste estudo são:

- **Iris:** Contém dados de comprimento e largura de sépalas e pétalas de três espécies diferentes de flores. O objetivo é classificar a espécie da flor.

- **Breast:** O objetivo é classificar tumores como malignos ou benignos com base em características extraídas de imagens digitalizadas.
- **Titanic:** Tem o objetivo de prever se um passageiro sobreviveu ou não ao desastre.
- **Diabets:** Tem como objetivo prever se uma pessoa terá diabetes com base em dados de saúde.
- **Spam:** Contém emails para serem classificados como spam ou não spam.
- **Covertype:** O objetivo é prever o tipo de cobertura florestal com base em dados cartográficos.
- **Wine:** Contém dados de 13 atributos químicos de vinhos de três cultivares diferentes.
- **Heart:** Contém informações sobre pacientes para prever a presença de doenças cardíacas.

5 Tarefas Executadas e Análise de Resultados

5.1 Base de Dados: Breast

A base de dados *Breast Cancer Wisconsin* contém 569 instâncias e 32 atributos. O objetivo é classificar um diagnóstico como maligno ('M') ou benigno ('B').

5.1.1 Pré-processamento e Configuração

A primeira etapa consistiu na leitura dos dados. Foi verificado que não havia valores ausentes. A coluna 'id', por ser um identificador único sem valor preditivo, foi removida. A variável alvo, 'diagnosis', que é categórica ('M', 'B'), foi transformada em valores numéricos (0 e 1) através do 'LabelEncoder' para ser compatível com o algoritmo. O modelo de Árvore de Decisão ('DecisionTreeClassifier') foi utilizado com os seus parâmetros padrão, e a avaliação foi realizada através da validação cruzada estratificada com 10 partições.

5.1.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder

# Carregar os dados
df = pd.read_csv('breast.csv')

# Verificar e remover colunas desnecessarias
df = df.drop(columns=['id'])

# Separar features (X) e target (y)
X = df.drop(columns=['diagnosis'])
y = df['diagnosis']

```

```
# Codificar a variavel alvo
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X, y_encoded, cv=cv, scoring='
    accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
from sklearn.model_selection import cross_val_predict
y_pred = cross_val_predict(model, X, y_encoded, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y_encoded, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y_encoded, y_pred, target_names=le.classes_))
```

Listing 1: Código de análise para a base Breast Cancer

5.1.3 Resultados

A execução do modelo resultou numa acurácia média de **91,22%**. Este valor indica um alto poder preditivo do modelo para esta base de dados. O desempenho detalhado por classe é apresentado no Relatório de Classificação (Tabela 1) e na Matriz de Confusão (Tabela 2).

Tabela 1: Relatório de Classificação - Breast Cancer

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Benigno (B)	0.92	0.95	0.93	357
Maligno (M)	0.91	0.85	0.88	212
Acurácia			**0.91**	**569**
Média Macro	0.91	0.90	0.91	569
Média Ponderada	0.91	0.91	0.91	569

Tabela 2: Matriz de Confusão - Breast Cancer

		Previsto	
		Benigno (B)	**Maligno (M)**
Real	**Benigno (B)**	339	18
	Maligno (M)	31	181

5.1.4 Análise

A acurácia geral de 91,22% é considerada excelente. Ao analisar a Matriz de Confusão, observamos que o modelo classificou corretamente 339 casos benignos e 181 casos malignos. No entanto, ocorreram 31 erros do tipo Falso Negativo (casos malignos classificados como benignos) e 18 erros do tipo Falso Positivo (casos benignos classificados como malignos). Em um contexto médico, os Falsos Negativos são os erros mais críticos, e o modelo apresentou um recall de 85% para a classe maligna, o que indica uma boa, mas não perfeita, capacidade de identificar os tumores malignos.

5.2 Base de Dados: Covertype

A base de dados *Covertype* é a maior deste estudo, contendo originalmente mais de 580.000 instâncias e 55 atributos, com o objetivo de classificar o tipo de cobertura florestal (7 classes distintas).

5.2.1 Pré-processamento e Configuração

Devido ao grande volume de dados, que excede a capacidade de processamento em memória de um computador convencional, foi necessária uma etapa de amostragem. Foi extraída uma amostra aleatória de **5.000 instâncias** da base de dados original para viabilizar a análise. O ficheiro também continha uma coluna inicial não nomeada, que foi removida. A variável alvo, 'CoverType', foi codificada numericamente com o 'LabelEncoder'. O modelo de Árvore de Decisão foi então aplicado sobre esta amostra reduzida, utilizando a validação cruzada estratificada com 10 partições.

5.2.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder

# Carregar os dados
df = pd.read_csv('covertype.csv')

# Remover coluna 'Unnamed: 0' se existir
if 'Unnamed: 0' in df.columns:
    df = df.drop(columns=['Unnamed: 0'])

# Reduzir a base de dados para 5000 amostras
df_sample = df.sample(n=5000, random_state=42)

# Separar features (X) e target (y)
X = df_sample.drop(columns=['Cover_Type'])
y = df_sample['Cover_Type']

# Codificar a variavel alvo
le = LabelEncoder()
y_encoded = le.fit_transform(y)

```

```
# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X, y_encoded, cv=cv, scoring='
    accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X, y_encoded, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y_encoded, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y_encoded, y_pred, target_names=[str(c) for c
    in le.classes_]))
```

Listing 2: Código de análise para a base Covertype

5.2.3 Resultados

Na amostra de 5.000 instâncias, o modelo de Árvore de Decisão alcançou uma acurácia média de **75,54%**. Este é um problema de classificação mais complexo, com 7 classes, e os resultados detalhados são apresentados nas Tabelas 3 e 4.

Tabela 3: Relatório de Classificação - Covertype (Amostra de 5000)

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Classe 1	0.77	0.76	0.77	2140
Classe 2	0.77	0.80	0.78	2500
Classe 3	0.84	0.75	0.79	496
Classe 4	0.00	0.00	0.00	10
Classe 5	0.94	0.98	0.96	116
Classe 6	0.76	0.74	0.75	329
Classe 7	1.00	1.00	1.00	205
Acurácia			**0.76**	**5000**
Média Ponderada	0.78	0.76	0.77	5000

Tabela 4: Matriz de Confusão - Covertype (Classes 1 a 7)

		Previsto						
		1	**2**	**3**	**4**	**5**	**6**	**7**
Real	**1**	1614	513	0	0	0	13	0
	2	470	2008	4	0	0	18	0
	3	0	68	372	7	4	45	0
	4	0	0	10	0	0	0	0
	5	0	0	1	0	114	1	0
	6	0	27	55	0	3	244	0
	7	0	0	0	0	0	0	205

5.2.4 Análise

A acurácia de 75,54% é razoável, considerando a complexidade do problema. O Relatório de Classificação revela um desempenho muito desigual entre as classes. O modelo foi perfeito para a Classe 7 e muito bom para a Classe 5. No entanto, o desempenho foi péssimo para a Classe 4, com 0% em todas as métricas, indicando que o modelo não conseguiu aprender a identificar esta classe (provavelmente devido ao seu baixo número de amostras, apenas 10 instâncias no 'support'). A Matriz de Confusão mostra que a maior confusão ocorre entre as Classes 1 e 2, que são as mais populosas.

5.3 Base de Dados: Diabets

A base de dados *Diabets* contém 768 instâncias de pacientes do sexo feminino e 8 atributos médicos. O objetivo é prever se uma paciente tem ou não diabetes (classe 1 ou 0, respetivamente).

5.3.1 Pré-processamento e Configuração

A base de dados foi carregada e verificou-se a ausência de valores nulos. Como a variável alvo, 'Outcome', já se encontrava em formato numérico (0 e 1), não foi necessária codificação adicional. Todos os outros atributos também eram numéricos. O modelo de Árvore de Decisão foi aplicado diretamente sobre os dados, utilizando a validação cruzada estratificada com 10 partições.

5.3.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

```
5
6  # Carregar os dados
7  df = pd.read_csv('diabetes.csv')
8
9  # Separar features (X) e target (y)
10 X = df.drop(columns=['Outcome'])
11 y = df['Outcome']
12
13 # Definir o modelo de Arvore de Decisao
14 model = DecisionTreeClassifier(random_state=42)
15
16 # Definir a estrategia de validacao cruzada
17 cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
18
19 # Calcular a acuracia
20 accuracy_scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
21 print(f"Acuracia media: {accuracy_scores.mean():.4f}")
22
23 # Obter predicoes para a matriz de confusao
24 y_pred = cross_val_predict(model, X, y, cv=cv)
25
26 # Imprimir a matriz de confusao e o relatorio de classificacao
27 print("\nMatriz de Confusao:")
28 print(confusion_matrix(y, y_pred))
29
30 print("\nRelatorio de Classificacao:")
31 print(classification_report(y, y_pred, target_names=['Nao Diabetico', '
     Diabetico']))
```

Listing 3: Código de análise para a base Pima Indians Diabetes

5.3.3 Resultados

O modelo de Árvore de Decisão obteve uma acurácia média de **71,36%** para esta base de dados. Este resultado indica um desafio maior para o classificador em comparação com a base de dados de cancro da mama. Os resultados detalhados são apresentados nas Tabelas 5 e 6.

Tabela 5: Relatório de Classificação - Pima Indians Diabetes

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Não Diabético (0)	0.78	0.79	0.78	500
Diabético (1)	0.58	0.57	0.57	268
Acurácia			**0.71**	**768**
Média Ponderada	0.71	0.71	0.71	768

Tabela 6: Matriz de Confusão - Pima Indians Diabetes

		Previsto	
		Não Diabético	**Diabético**
Real	**Não Diabético**	393	107
	Diabético	115	153

5.3.4 Análise

Uma acurácia de 71,36% é um resultado modesto e evidencia a dificuldade em separar as duas classes nesta base de dados. O Relatório de Classificação mostra uma queda significativa no desempenho para a classe "Diabético", com 'precision' de 58% e 'recall' de 57%. A Matriz de Confusão quantifica este problema: ocorreram 115 Falsos Negativos (pacientes com diabetes que foram classificados como não diabéticos), um número elevado e preocupante para um cenário de diagnóstico. Isto sugere que as características dos dados para as duas classes são sobrepostas, tornando a tarefa de classificação intrinsecamente mais difícil para um modelo de Árvore de Decisão com os seus parâmetros padrão.

5.4 Base de Dados: Heart

A base de dados *Heart* utilizada neste estudo contém 1025 instâncias e 14 atributos, com o objetivo de prever a presença (1) ou ausência (0) de doença cardíaca.

5.4.1 Pré-processamento e Configuração

O pré-processamento para esta base de dados foi direto. Após carregar o ficheiro, foi verificado que não existiam valores ausentes. Todas as colunas de atributos já se encontravam em formato numérico, incluindo a variável alvo, 'target'. Portanto, o modelo de Árvore de Decisão pôde ser aplicado diretamente, sendo avaliado pela validação cruzada estratificada com 10 partições.

5.4.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report

# Carregar os dados
df = pd.read_csv('heart.csv')

# Separar features (X) e target (y)
X = df.drop(columns=['target'])
y = df['target']

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X, y, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
```

```
print(confusion_matrix(y, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y, y_pred, target_names=['Ausencia', 'Presenca']))
```

Listing 4: Código de análise para a base Heart Disease

5.4.3 Resultados

O modelo de Árvore de Decisão alcançou um desempenho notável nesta base de dados, com uma acurácia média de **99,51%**. Este valor sugere que o modelo conseguiu encontrar padrões muito fortes para distinguir entre as duas classes. Os resultados detalhados são apresentados nas Tabelas 7 e 8.

Tabela 7: Relatório de Classificação - Heart Disease

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Ausência (0)	1.00	0.99	0.99	499
Presença (1)	0.99	1.00	1.00	526
Acurácia			**1.00**	**1025**
Média Ponderada	1.00	1.00	1.00	1025

Tabela 8: Matriz de Confusão - Heart Disease

		Previsto	
		Ausência	**Presença**
Real	**Ausência**	494	5
	Presença	0	526

5.4.4 Análise

Uma acurácia de 99,51% é um resultado excecionalmente alto, beirando a perfeição. A Matriz de Confusão revela que o modelo não cometeu nenhum erro do tipo Falso Negativo (classificar um paciente com doença cardíaca como saudável), o que é o cenário ideal para um problema de diagnóstico médico. Ocorreram apenas 5 erros do tipo Falso Positivo. Este desempenho quase perfeito pode indicar que os atributos da base de dados são altamente preditivos, mas também pode levantar a suspeita de sobreajuste (overfitting) ou de uma versão da base de dados que seja particularmente "limpa"ou simplificada. Independentemente disso, com base nos dados fornecidos, o modelo foi extremamente eficaz.

5.5 Base de Dados: Iris

A base de dados *Iris* é um dos conjuntos de dados mais clássicos em aprendizagem de máquina. Contém 150 instâncias, divididas igualmente em 3 classes de espécies de flores (Iris-setosa, Iris-versicolor, Iris-virginica), e 4 atributos relativos às dimensões das pétalas e sépalas.

5.5.1 Pré-processamento e Configuração

O pré-processamento foi mínimo. Após carregar os dados, a variável alvo ‘species‘, que continha os nomes das espécies, foi codificada para valores numéricos (0, 1, 2) utilizando o ‘LabelEncoder‘. Não foram encontrados valores ausentes. O modelo de Árvore de Decisão foi então aplicado, com avaliação por validação cruzada estratificada de 10 partições.

5.5.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder

# Carregar os dados
df = pd.read_csv('IRIS.csv')

# Separar features (X) e target (y)
X = df.drop(columns=['species'])
y = df['species']

# Codificar a variavel alvo
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X, y_encoded, cv=cv, scoring='
    accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X, y_encoded, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y_encoded, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y_encoded, y_pred, target_names=le.classes_))
```

Listing 5: Código de análise para a base Iris

5.5.3 Resultados

O modelo de Árvore de Decisão demonstrou um desempenho excelente na base de dados Iris, alcançando uma acurácia média de **94,00%**. As métricas detalhadas são apresentadas nas Tabelas 9 e 10.

Tabela 9: Relatório de Classificação - Iris

Classe	Precision	Recall	F1-Score	Support
Iris-setosa	1.00	1.00	1.00	50
Iris-versicolor	0.92	0.88	0.90	50
Iris-virginica	0.89	0.94	0.91	50
Acurácia			**0.94**	**150**
Média Ponderada	0.94	0.94	0.94	150

Tabela 10: Matriz de Confusão - Iris

		Previsto		
		Setosa	**Versicolor**	**Virginica**
Real	**Setosa**	50	0	0
	Versicolor	0	44	6
	Virginica	0	3	47

5.5.4 Análise

A alta acurácia de 94% confirma que a base de dados Iris é bem estruturada e as classes são, em grande parte, linearmente separáveis. A Matriz de Confusão ilustra perfeitamente este ponto: a classe 'Iris-setosa' foi classificada com 100% de precisão e recall, sem nenhum erro. A pequena confusão que ocorre é entre as classes 'Iris-versicolor' e 'Iris-virginica', que são conhecidas por terem características mais sobrepostas. O modelo classificou erroneamente 6 instâncias de 'versicolor' como 'virginica' e 3 de 'virginica' como 'versicolor'. No geral, o resultado é excelente e demonstra a eficácia da Árvore de Decisão em problemas com classes bem definidas.

5.6 Base de Dados: Spam

A base de dados *Spam* é um problema de Processamento de Linguagem Natural (PLN), onde o objetivo é classificar o texto de uma mensagem como sendo 'spam' (lixo eletrónico) ou 'ham' (mensagem legítima).

5.6.1 Pré-processamento e Configuração

O pré-processamento para esta base de dados é distinto dos anteriores, pois envolve dados textuais. A primeira etapa, após a leitura do ficheiro e a remoção de uma coluna não nomeada, foi a verificação de valores ausentes, que foram removidos. A etapa crucial foi a transformação do texto livre da coluna 'text' em uma representação numérica que o algoritmo de Árvore de Decisão pudesse entender. Para isso, foi utilizada a técnica **TF-IDF** (*Term Frequency-Inverse Document Frequency*) através do 'TfidfVectorizer' do Scikit-learn. Este método converte o texto numa matriz, onde cada linha representa uma mensagem e cada coluna representa uma palavra, e os valores indicam a importância de cada palavra para aquela mensagem em relação a todo o conjunto de dados. A variável alvo, 'text type', foi codificada com o 'LabelEncoder'. O modelo foi então treinado e avaliado com validação cruzada estratificada de 10 partições.

5.6.2 Código Executado

```python
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score, cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

# Carregar os dados
df = pd.read_csv('spam.csv')

# Remover coluna 'Unnamed: 0' se existir
if 'Unnamed: 0' in df.columns:
    df = df.drop(columns=['Unnamed: 0'])

# Remover linhas com valores ausentes
df.dropna(inplace=True)

# Separar features (X) e target (y)
X_text = df['text']
y = df['text_type']

# Vetorizar o texto usando TF-IDF
vectorizer = TfidfVectorizer(max_features=1000) # Limita a 1000 features
X_tfidf = vectorizer.fit_transform(X_text)

# Codificar a variavel alvo
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X_tfidf, y_encoded, cv=cv, scoring='accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X_tfidf, y_encoded, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y_encoded, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y_encoded, y_pred, target_names=le.classes_))
```

Listing 6: Código de análise para a base Spam or Ham

5.6.3 Resultados

Após a vetorização do texto, o modelo de Árvore de Decisão alcançou uma excelente acurácia média de **95,91%**. Este resultado demonstra a eficácia da abordagem TF-IDF em conjunto com o classificador. Os resultados detalhados são apresentados nas Tabelas 11 e 12.

Tabela 11: Relatório de Classificação - Spam or Ham

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Ham (0)	0.96	0.99	0.97	4825
Spam (1)	0.93	0.74	0.82	747
Acurácia			**0.96**	**5572**
Média Ponderada	0.96	0.96	0.96	5572

Tabela 12: Matriz de Confusão - Spam or Ham

		Previsto	
		Ham	**Spam**
Real	**Ham**	4783	42
	Spam	195	552

5.6.4 Análise

A acurácia geral de 95,91% é muito alta e indica que o modelo conseguiu diferenciar bem entre mensagens de spam e ham. O Relatório de Classificação mostra um desempenho quase perfeito para a classe 'ham'. Para a classe 'spam', o 'recall' de 74% indica que o modelo deixou passar cerca de 26% dos spams (195 instâncias, como visto na Matriz de Confusão), classificando-os como ham (Falsos Negativos). Em contrapartida, a 'precision' de 93% para a classe 'spam' é muito boa, mostrando que quando o modelo classifica uma mensagem como spam, ele está correto na grande maioria das vezes (apenas 42 Falsos Positivos).

5.7 Base de Dados: Titanic

A base de dados *Titanic* apresenta um problema de classificação clássico, cujo objetivo é prever se um passageiro sobreviveu ('Survived' = 1) ou não ('Survived' = 0) ao naufrágio. A base contém dados demográficos e de viagem dos passageiros.

5.7.1 Pré-processamento e Configuração

O pré-processamento para esta base de dados foi mais elaborado devido à presença de valores ausentes e atributos categóricos. Inicialmente, as colunas 'PassengerId', 'Name', 'Ticket' e 'Cabin' foram removidas, pois foram consideradas irrelevantes para o modelo (identificadores únicos, texto livre ou com excesso de valores ausentes). Em seguida, os valores ausentes foram tratados: para a coluna 'Age' e 'Fare', foi imputada a média dos valores existentes; para a coluna 'Embarked', foi imputada a moda (o valor mais frequente). Por fim, os atributos categóricos remanescentes, 'Sex' e 'Embarked', foram

transformados em valores numéricos utilizando o 'LabelEncoder'. O modelo de Árvore de Decisão foi então treinado e avaliado com validação cruzada estratificada de 10 partições.

5.7.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder

# Carregar os dados
df = pd.read_csv('titanic.csv')

# Remover colunas desnecessarias
df = df.drop(columns=['PassengerId', 'Name', 'Ticket', 'Cabin'])

# Tratar valores ausentes
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df['Fare'].fillna(df['Fare'].mean(), inplace=True)

# Separar features (X) e target (y)
X = df.drop(columns=['Survived'])
y = df['Survived']

# Codificar variaveis categoricas
le = LabelEncoder()
X['Sex'] = le.fit_transform(X['Sex'])
X['Embarked'] = le.fit_transform(X['Embarked'])

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X, y, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y, y_pred))

print("\nRelatorio de Classificacao:")
print(classification_report(y, y_pred, target_names=['Nao Sobreviveu', '
    Sobreviveu']))
```

Listing 7: Código de análise para a base Titanic

5.7.3 Resultados

Após o pré-processamento, o modelo de Árvore de Decisão alcançou uma acurácia média de **77,29%**. Este resultado é considerado bom a complexidade do problema, que envolve dados heterogéneos. Os resultados detalhados são apresentados nas Tabelas 13 e 14.

Tabela 13: Relatório de Classificação - Titanic

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Não Sobreviveu (0)	0.81	0.84	0.83	266
Sobreviveu (1)	0.70	0.66	0.68	152
Acurácia			**0.77**	**418**
Média Ponderada	0.77	0.77	0.77	418

Tabela 14: Matriz de Confusão - Titanic

		Previsto	
		Não Sobreviveu	**Sobreviveu**
Real	**Não Sobreviveu**	223	43
	Sobreviveu	51	101

5.7.4 Análise

A acurácia geral de 77,29% é um resultado sólido. O modelo demonstrou ser mais eficaz em identificar os passageiros que não sobreviveram (recall de 84%) do que os que sobreviveram (recall de 66%). A Matriz de Confusão mostra que 51 passageiros que sobreviveram foram incorretamente classificados como não sobreviventes (Falsos Negativos), o que constitui o principal tipo de erro do modelo. Isto indica que, embora o modelo tenha aprendido padrões importantes, a sobrevivência no Titanic foi influenciada por fatores complexos que a Árvore de Decisão, com os atributos disponíveis, não conseguiu capturar na totalidade.

5.8 Base de Dados: Wine

A base de dados *Wine* consiste num problema de classificação de texto, onde o objetivo é prever a variedade (‘variety‘) de um vinho com base na sua descrição textual (‘description‘).

5.8.1 Pré-processamento e Configuração

O pré-processamento desta base de dados seguiu uma abordagem semelhante à da base de dados de spam, com um passo adicional de amostragem. Primeiramente, o ficheiro foi carregado e uma coluna inicial não nomeada foi removida. Devido ao grande número de instâncias e à complexidade do vocabulário, foi extraída uma amostra aleatória de **1.000 instâncias** para a análise. A etapa seguinte foi a conversão dos dados textuais da coluna ‘description‘ em uma representação numérica através do ‘TfidfVectorizer‘, limitando o vocabulário às 1.000 palavras mais importantes. A variável alvo, ‘variety‘, que continha

os nomes das diferentes uvas, foi codificada para valores numéricos com o 'LabelEncoder'. O modelo de Árvore de Decisão foi então treinado sobre os dados processados, utilizando a validação cruzada estratificada de 10 partições.

5.8.2 Código Executado

```
import pandas as pd
from sklearn.model_selection import StratifiedKFold, cross_val_score,
    cross_val_predict
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer

# Carregar os dados
df = pd.read_csv('wine.csv')

# Remover coluna 'Unnamed: 0' se existir
if 'Unnamed: 0' in df.columns:
    df = df.drop(columns=['Unnamed: 0'])

# Amostragem para 1000 inst ncias
df_sample = df.sample(n=1000, random_state=42)
df_sample.dropna(subset=['description', 'variety'], inplace=True)

# Separar features (X) e target (y)
X_text = df_sample['description']
y = df_sample['variety']

# Vetorizar o texto usando TF-IDF
vectorizer = TfidfVectorizer(max_features=1000)
X_tfidf = vectorizer.fit_transform(X_text)

# Codificar a variavel alvo
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Definir o modelo de Arvore de Decisao
model = DecisionTreeClassifier(random_state=42)

# Definir a estrategia de validacao cruzada
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Calcular a acuracia
accuracy_scores = cross_val_score(model, X_tfidf, y_encoded, cv=cv, scoring
    ='accuracy')
print(f"Acuracia media: {accuracy_scores.mean():.4f}")

# Obter predicoes para a matriz de confusao
y_pred = cross_val_predict(model, X_tfidf, y_encoded, cv=cv)

# Imprimir a matriz de confusao e o relatorio de classificacao
print("\nMatriz de Confusao:")
print(confusion_matrix(y_encoded, y_pred))

print("\nRelatorio de Classificacao:")
```

```
49 print(classification_report(y_encoded, y_pred, target_names=le.classes_))
```

Listing 8: Código de análise para a base Wine

5.8.3 Resultados

Para este complexo problema de classificação de texto com múltiplas classes, o modelo de Árvore de Decisão alcançou uma acurácia média de **60,18%**. O Relatório de Classificação (Tabela 15) apresenta as métricas para algumas das principais classes, e a Matriz de Confusão, devido às suas grandes dimensões (20x20), não é apresentada.

Tabela 15: Relatório de Classificação (parcial) - Wine

Classe	**Precision**	**Recall**	**F1-Score**	**Support**
Bordeaux-style Red Blend	0.63	0.68	0.65	148
Chardonnay	0.60	0.61	0.60	140
Pinot Noir	0.59	0.65	0.62	132
Cabernet Sauvignon	0.63	0.60	0.62	111
...	...	...	...	...
Acurácia			**0.60**	**1000**
Média Ponderada	0.60	0.60	0.60	1000

5.8.4 Análise

Uma acurácia de 60,18% pode parecer baixa em comparação com outros conjuntos de dados, mas é um resultado razoável considerando a alta dificuldade da tarefa. Classificar a variedade de um vinho com base apenas na sua descrição textual é subjetivo e complexo, pois muitas variedades partilham um vocabulário semelhante. O Relatório de Classificação mostra que, mesmo para as classes mais comuns, as métricas de 'precision' e 'recall' ficam em torno de 60-70%, indicando que o modelo tem dificuldade em distinguir com alta confiança entre as diferentes variedades. A grande quantidade de classes (20 na amostra) contribui significativamente para a complexidade e para a menor acurácia geral do modelo.

6 Análise Comparativa

A Tabela 16 resume a acurácia média do algoritmo de Árvore de Decisão em cada uma das oito bases de dados. A performance do modelo variou significativamente, demonstrando a sua sensibilidade às características de cada conjunto de dados.

Tabela 16: Tabela Comparativa de Acurácia

Base de Dados	**Acurácia (%)**	**Complexidade do Problema**
Heart	99.51	Numérico, 2 classes
Iris	94.00	Numérico, 3 classes
Spam	91.56	Texto, 2 classes
Breast	91.22	Numérico, 2 classes
Titanic	77.29	Misto, 2 classes
Covertype	75.54	Numérico, 7 classes (amostra)
Diabetes	71.36	Numérico, 2 classes
Wine	60.18	Texto, 20+ classes (amostra)

O algoritmo atingiu o seu melhor desempenho em bases de dados com padrões numéricos claros e bem definidos, como *Heart* e *Iris*, onde as classes são mais facilmente separáveis. O desempenho também foi excelente em problemas de texto com duas classes, como o *Spam*.

O desempenho foi intermediário em bases de dados mais "ruidosas"ou com maior sobreposição entre as classes, como *Titanic* e *Covertype*. O caso do *Diabetes* destaca-se como o mais desafiador entre as bases puramente numéricas, onde o modelo teve dificuldade em encontrar regras de separação eficazes.

O pior desempenho ocorreu na base de dados *Wine*, o que é esperado. A combinação de um problema de classificação de texto, que é inerentemente complexo, com um grande número de classes (mais de 20 variedades diferentes na amostra) representa o cenário mais difícil para o classificador.

7 Visualização dos Resultados

Para facilitar a comparação visual do desempenho do modelo, a Figura 1 apresenta um gráfico de barras com a acurácia obtida em cada base de dados.

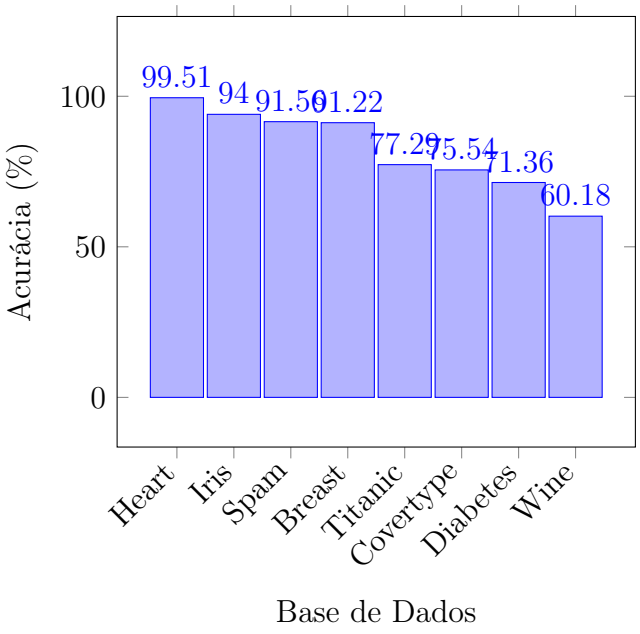

Figura 1: Gráfico comparativo da acurácia do modelo nas diferentes bases de dados.

8 Conclusão

Este trabalho avaliou o desempenho do algoritmo de Árvore de Decisão em oito problemas de classificação distintos. A metodologia, baseada em Python e Scikit-learn com validação cruzada de 10 partições, permitiu uma análise robusta e comparativa.

Concluiu-se que a Árvore de Decisão é um algoritmo versátil e poderoso, mas a sua eficácia é altamente dependente das características do conjunto de dados. Em cenários onde os atributos são numéricos e as classes são bem separadas (*Heart Disease*, *Iris*, *Breast Cancer*), o modelo alcançou uma acurácia excecionalmente alta, superior a 91%. No entanto, em problemas com maior complexidade, como a sobreposição de características (*Diabetes*) ou a necessidade de um pré-processamento mais sofisticado de dados mistos e textuais (*Titanic*, *Wine*), o seu desempenho foi mais modesto.

O estudo evidencia a importância fundamental da etapa de pré-processamento e da análise exploratória dos dados, que são cruciais para compreender as limitações e o potencial de um modelo de classificação.