UNIVERSIDADE FEDERAL FLUMINENSE
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
TCC00297 - INTELIGÊNCIA ARTIFICIAL

Trabalho de Classificação

BEATRIZ DE OLIVEIRA PIEDADE

NITERÓI
2024

Contents

1 Introdução

O objetivo deste trabalho é realizar a classificação do conjunto de dados "Secondary Mushroom", buscando prever se um cogumelo é comestível ou venenoso. O dataset foi analisado com foco na preparação, construção e avaliação de modelos de Machine Learning que maximizem a performance em métricas relevantes.

O conjunto de dados contém 61.068 registros e 20 atributos. A tabela abaixo apresenta uma visão geral dos atributos disponíveis no conjunto:

Nome	**Papel**	**Tipo**	**Valores ausentes**
class	Target	Categorical	no
cap-diameter	Feature	Continuous	no
cap-shape	Feature	Categorical	no
cap-surface	Feature	Categorical	yes
cap-color	Feature	Categorical	no
does-bruise-or-bleed	Feature	Categorical	no
gill-attachment	Feature	Categorical	yes
gill-spacing	Feature	Categorical	yes
gill-color	Feature	Categorical	no
stem-height	Feature	Continuous	no
stem-width	Feature	Continuous	no
stem-root	Feature	Categorical	yes
stem-surface	Feature	Categorical	yes
stem-color	Feature	Categorical	no
veil-type	Feature	Categorical	yes
veil-color	Feature	Categorical	yes
has-ring	Feature	Categorical	no
ring-type	Feature	Categorical	yes
spore-print-color	Feature	Categorical	yes
habitat	Feature	Categorical	no
season	Feature	Categorical	no

1.1 Coleta de dados

O conjunto de dados foi obtido do Repositório de Machine Learning da UCI.

```
# CARREGANDO DADOS
from ucimlrepo import fetch_ucirepo

# importando dataset
dataset = fetch_ucirepo(id=763)

# coletando as informações
data_frame = dataset.data.original
```

O conjunto de dados, assim como todas as tabelas derivadas, está estruturado no formato *DataFrame*, uma poderosa estrutura de dados fornecida pela biblioteca *pandas*. Essa estrutura simplifica a manipulação, análise e pré-processamento dos dados, permitindo operações eficientes.

1.2 Pré-processamento de dados

O conjunto de dados foi tratado para minimizar o impacto de dados nulos, duplicados ou mal formatados na performance dos modelos. O pré-processamento envolveu as seguintes etapas:

1. A remoção de colunas com muitos valores nulos, utilizando o parâmetro *thresh* para evitar a perda excessiva de informações. O valor do *thresh* é dado pela variável *tolerancia*, que garante que colunas com 70% de dados não nulos sejam mantidas.

2. A remoção de dados duplicados para evitar redundância e influências desproporcionais na modelagem.

3. A transformação de variáveis categóricas em valores numéricos, garantindo que o modelo possa interpretar essas variáveis.

```
# TRATANDO DADOS
import pandas
from sklearn.preprocessing import LabelEncoder

# removendo colunas com muitos nulos
tolerancia = len(data_frame) * 0.7
data_frame = data_frame.dropna(axis=1, thresh=tolerancia)

# removendo dados duplicados
```

```
data_frame = data_frame.drop_duplicates()

# convertendo colunas categóricas em valores inteiros
for coluna in data_frame.columns:
if (data_frame[coluna].dtype == type(object)):
 conversor = LabelEncoder()
 data_frame[coluna] = conversor.fit_transform(
                                        data_frame[coluna])
```

Ao término do processo, o conjunto de dados foi reduzido a 60.922 registros tratados.

1.3 Divisão de dados

Dado o tamanho do conjunto de dados e o número de atributos, ele será dividido em 80% para treinamento e 20% para teste, utilizando a função *train_test_split()* do pacote *sklearn.model_selection*.

```
# DIVIDINDO O DATASET TRATADO
import numpy
from sklearn.model_selection import train_test_split

atributos = data_frame.drop(["class"], axis=1)
respostas = data_frame[["class"]]

a_treino, a_teste, r_treino, r_teste = train_test_split(
                                        atributos,
                                        respostas,
                                        test_size=0.2,
                                        random_state=42)

# convertendo de (N, 1) para (N,)
r_treino = numpy.ravel(r_treino)
r_teste = numpy.ravel(r_teste)
```

Além disso, as tabelas resultantes das respostas de treino e teste serão convertidas para o formato *(n_sample,)*, que é o formato esperado pelos classificadores para que possam processar os dados corretamente.

1.4 Treinamento e avaliação do modelo

Para este trabalho, serão utilizados os modelos de aprendizado de máquina: Árvore de Decisão (DT), Random Forest (RF) e Rede Neural Multilayer Perceptron (MLP). Os modelos passarão por ajustes nos parâmetros do classificador e serão avaliados com base na performance dos algoritmos e na sua explicabilidade.

Na avaliação da performance, serão utilizadas as métricas *Acurácia* e *F1-Score*, onde a *Acurácia* é dada por

$$Acuracia = \frac{PrevisoesCorretas}{PrevisoesTotais}$$

e o *F1-Score* é calculado como

$$F1 = 2 \cdot \frac{Precisao \cdot Recall}{Precisao + Recall}$$

$$Precisao = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosPostivos}$$

$$Recall = \frac{VerdadeirosPositivos}{VerdadeirosPositivos + FalsosNegativos}$$

2 Árvore de decisão

2.1 Algoritmo utilizado

2.2 Variações de parâmetros

2.3 Performance

3 Random Forest

3.1 Algoritmo utilizado

3.2 Variações de parâmetros

3.3 Performance

4 Rede Neural Multilayer perceptron

4.1 Algoritmo utilizado

4.2 Variações de parâmetros

4.3 Performance

5 Conclusão

- Uma análise crítica de escolher e colocar um modelo em produção para o domínio do dataset selecionado.