UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
BACHARELADO INTERDISCIPLINAR CIÊNCIA E TECNOLOGIA - BICT
INTELIGENCIA ARTIFICIAL

LUIS DA ASSUNÇÃO MAFRA MOURA - 2019050691
MARCELO ADRIEL CÂMARA ALMEIDA - 2020002392
MARIA EDUARDA PEREIRA LIMA - 2023033937

ARTIGO PROJETO REDES NEURAIS:
MODELO DE REDE NEURAL PARA CLASSIFICAÇÃO DE COGUMELOS

SÃO LUÍS - MA
2023

Sumário

1. Introdução

A identificação e classificação de organismos com base em suas características físicas e químicas são desafios fundamentais em diversas áreas da biologia e da segurança alimentar. No contexto da micologia e do controle de qualidade, a distinção entre cogumelos comestíveis e venenosos é essencial para evitar intoxicações e garantir a segurança no consumo de alimentos. Com o avanço das técnicas de inteligência artificial, modelos de aprendizado de máquina, em redes especiais neurais artificiais (RNAs), são mostradas ferramentas eficazes para a classificação de dados em diversos domínios.

Neste contexto, este trabalho apresenta o desenvolvimento e a avaliação de uma rede neural artificial para a classificação de cogumelos como comestíveis ou venenosos, utilizando a base de dados Mushroom do repositório UCI Machine Learning. Essa base de dados contém informações sobre diversas características dos cogumelos, como formato e cor do chapéu, tipo de superfície, odor e outros atributos relevantes para sua identificação. O objetivo do estudo é explorar a eficácia de uma RNA na classificação dessas amostras, analisando sua precisão e capacidade de generalização.

Para isso, foi desenvolvido um modelo de rede neural utilizando a linguagem de programação Python e bibliotecas como TensorFlow, Keras, Pandas e Scikit-Learn. O modelo foi treinado com os atributos da base de dados, aplicando diferentes abordagens de pré-processamento, como coincidência de variações categóricas e normalização dos dados. Foram testados com diferentes arquiteturas de rede neural, variando o número de camadas ocultas, neurônios por camada e funções de ativação, identificando a configuração mais eficiente para a classificação das amostras.

O treinamento do modelo foi prolongado utilizando o algoritmo de otimização Adam e a função de perda **binária cross-entropy** , uma vez que o problema envolve classificação binária (comestível ou venenoso). A avaliação de desempenho foi realizada com análises como acurácia, matriz de confusão e relatório de classificação, permitindo comparar a eficácia do modelo treinado com outras técnicas de aprendizado de máquina.

A análise dos resultados incluiu comparar a acurácia do modelo com outras técnicas de classificação e discutir as limitações e desafios encontrados durante o desenvolvimento do projeto. Os testes indicaram que a rede neural artificial foi capaz de identificar padrões nos dados e classificar os cogumelos com alto desempenho, demonstrando o potencial da abordagem para aplicações reais.

O presente estudo contribui para o entendimento do uso de redes neurais artificiais na classificação de organismos biológicos, destacando o papel da inteligência artificial na automação de processos e na melhoria da isolamento de análises em áreas como segurança alimentar, biologia e micologia aplicada

2. Objetivos

O principal objetivo deste projeto é desenvolver e avaliar um modelo de rede neural artificial para a classificação de cogumelos como comestíveis ou venenosos, utilizando a base de dados Mushroom do repositório UCI Machine Learning. O projeto busca analisar a eficácia dessa abordagem para distinguir fungos com base em suas características físicas e químicas, fornecendo uma solução inteligente e eficiente para esse problema de classificação.

Além disso, o estudo visa:

- Investigar a influência de diferentes parâmetros da rede neural, como a quantidade de camadas ocultas, neurônios e funções de ativação, na performance do modelo;
- Comparar os resultados obtidos com outras técnicas de aprendizado de máquina para identificar a abordagem mais adequada;
- Explorar as dificuldades e desafios do treinamento da rede neural, analisando a qualidade da classificação e possíveis melhorias no processo;
- Aplicar técnicas de pré-processamento e balanceamento dos dados para otimizar o desempenho do modelo;
- Demonstrar a viabilidade do uso de redes neurais artificiais na classificação de materiais em contextos reais, como análise forense e controle de qualidade industrial.

3. Fundamentação Teórica

As redes neurais artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano, compostos por unidades básicas chamadas neurônios artificiais. Esses neurônios estão organizados em camadas e são interconectados por pesos, que determinam a importância de cada entrada no processo de aprendizagem.

Uma rede neural típica é composta por três tipos de camadas:

1. Camada de entrada: recebe os dados e os distribui para a rede.
2. Camadas ocultas: realizando operações matemáticas sobre os dados, identificando padrões complexos.
3. Camada de saída: fornece a resposta final do modelo (neste caso, se o cogumelo é comestível ou venenoso).

Cada neurônio realiza um cálculo simples: recebe entradas ponderadas pelos pesos, soma essas entradas e passa o resultado por uma função de ativação, que introduz não linearidade ao modelo, permitindo que ele capture relações complexas entre os dados.

3.1 Aprendizado Supervisionado

O aprendizado supervisionado é uma abordagem de aprendizado de máquina na qual o modelo recebe um conjunto de dados rotulados e aprende a mapear as entradas para as saídas corretas. No caso deste projeto, os exemplos de cogumelos vêm acompanhados de um rótulo indicando se são comestíveis (representados por "e") ou venenosos ("p").

O processo de aprendizado consiste em ajustar os pesos das conexões entre os neurônios para minimizar o erro nas opções. Esse ajuste é feito durante o treinamento do modelo, utilizando um conjunto de dados de treino.

3.2 Função de Ativação e Função de Perda

As redes neurais utilizam funções de ativação para introduzir não linearidade e permitir que o modelo aprenda padrões complexos. No modelo desenvolvido, são utilizadas as seguintes funções de ativação:

ReLU (Retified Linear Unit): utilizado nas camadas ocultas, ajuda a evitar o problema do gradiente desaparecendo e melhora a eficiência do aprendizado.

Sigmoid: utilizado na camada de saída, pois retorna valores entre 0 e 1, o que é ideal para problemas de classificação binária como este.

A função de perda mede o quão bem o modelo está performando. Como estamos lidando com um problema de classificação binária, a função escolhida foi a Binary Cross-Entropy , definida como:

$$Perda = -\frac{1}{Não}\sum_{eu=1}^{Não}[e_{eu}registro(\hat{e}_{eu}) + (1 - e_{eu})\,log(1 - \hat{e}_{eu})]$$

Onde:

e_{eu} é o valor real (0 para comestível, 1 para venenoso).

$\hat{e}_{eu}$ é a previsão do modelo.

Essa função minimiza predições erradas, ajustando os pesos da rede para melhorar a acurácia.

3.3 Retropropagação e Otimização

O backpropagation é o algoritmo usado para ajustar os pesos da rede neural. Ele calcula o erro entre a previsão do modelo e o valor real e distribui esse erro de volta pela rede para atualizar os pesos de forma eficiente.

Para atualizar os pesos, utilizamos um otimizador, sendo o Adam (Adaptive Moment Estimation) o escolhido neste projeto. O Adam combina as vantagens do método de descida do gradiente com momentum e adaptação individual da taxa de aprendizado para cada peso, tornando o treinamento mais eficiente e rápido.

3.4 Normalização e Pré-Processamento dos Dados

Os dados brutos precisam ser preparados antes de serem utilizados para treinar a rede neural, um processo que envolve duas etapas principais. Primeiramente, é realizada a codificação das variáveis categóricas, uma vez que todos os atributos do conjunto de dados são categóricos. Para isso, utiliza-se a técnica One-Hot Encoding, que converte essas variáveis em valores numéricos, permitindo que sejam processadas pelo modelo. Em seguida, é feita a padronização dos dados, que visa garantir que os valores estejam dentro de uma faixa semelhante, evitando que algumas variáveis dominem outras durante o treinamento. Para isso, emprega-se o StandardScaler, que transforma os dados de forma a apresentarem média 0 e desvio padrão 1. Essas técnicas são essenciais para que o modelo aprenda de maneira mais eficiente e generalize melhor quando aplicado a novos dados.

4. Base de Dados e Pré-Processamento

O conjunto de dados utilizado neste estudo é o Mushroom Dataset, disponível no repositório UCI Machine Learning. Esse conjunto contém informações sobre diferentes espécies de cogumelos e seus atributos físicos, sendo utilizado para classificar se um cogumelo é comestível ou venenoso.

4.1 Características dos Dados

O dataset possui um total de 8.124 amostras de cogumelos, distribuídas entre duas classes:

- Comestível (e - edible)
- Venenoso (p - poisonous)

Cada amostra é descrita por 22 atributos categóricos, que incluem características como:

- Cor do chapéu
- Odor
- Textura do anel
- Tipo de habitat

Como todos os atributos são categóricos, é necessário convertê-los para um formato numérico antes de treinar o modelo de rede neural.

4.2 Transformação dos Dados

Uma vez que os atributos do dataset são categóricos, a rede neural não pode processá-los diretamente. Para resolver isso, utilizamos a técnica de One-Hot Encoding, que converte cada categoria em uma nova coluna binária (0 ou 1).

Por exemplo, suponha que o atributo "Odor" tenha três categorias possíveis:

- Amêndoa
- Mofo
- Nenhum odor

Com o One-Hot Encoding, esse atributo será transformado em três colunas separadas:

Amêndoa	Mofo	Nenhum odor
1	0	0
0	1	0
0	0	1

Isso permite que a rede neural processe os dados sem interpretar valores categóricos como ordinais.

Após a conversão, os dados são normalizados utilizando o StandardScaler. Esse método transforma os valores para que tenham média zero e desvio padrão 1, garantindo que todas as variáveis tenham a mesma importância durante o treinamento do modelo.

A normalização evita que atributos com valores altos dominem a rede neural, tornando o aprendizado mais eficiente e estável

4.3 Divisão do Conjunto de Dados

Após o pré-processamento, os dados são divididos em dois conjuntos:

Treinamento (80%) → usado para treinar a rede neural

Teste (20%) → usado para avaliar o desempenho do modelo

A divisão 80/20 é um padrão na área de aprendizado de máquina. Utilizamos 80% dos dados para o treinamento para garantir que o modelo aprenda padrões e relações entre os atributos dos cogumelos. Os 20% restantes são reservados para teste, permitindo avaliar a capacidade da rede neural de generalizar para novos dados.

Se a divisão fosse muito desigual (por exemplo, 90% treino e 10% teste), o modelo poderia se ajustar bem aos dados de treino, mas sua avaliação poderia ser menos confiável devido ao pequeno conjunto de teste. Já uma divisão muito grande para teste (como 50/50) deixaria poucos dados para o treinamento, comprometendo o aprendizado da rede.

5. Desenvolvimento do Modelo

Para classificar os cogumelos como comestíveis ou venenosos, foi desenvolvida uma rede neural capaz de lidar com variáveis categóricas, previamente convertidas em valores numéricos por meio da técnica *One-Hot Encoding*. A arquitetura escolhida segue o modelo de rede neural densa , onde todas as camadas estão completamente conectadas.

A estrutura do modelo é composta por:

1. Entrada: número de neurônios equivalente ao total de atributos do conjunto de dados após a organização.

2. Primeira camada oculta: 64 neurônios com ativação ReLU.

3. Segunda camada oculta: 32 neurônios com ativação ReLU.

4. Saída: 1 neurônio com ativação sigmoide, ideal para classificação binária.

A escolha dessa arquitetura foi feita com base em um equilíbrio entre simplicidade e capacidade de generalização. Como o problema de classificação de cogumelos não apresenta uma complexidade elevada, uma rede neural profunda não seria necessária, pois poderia levar ao *overfitting* — situação em que o modelo memoriza os dados em vez de aprender padrões generalizáveis. Dessa forma, a rede foi mantida enxuta, garantindo eficiência sem comprometer o desempenho.

A ativação do ReLU foi aplicada nas camadas ocultas devido à sua capacidade de evitar o problema do desaparecimento do gradiente e acelerar o aprendizado. Já a ativação sigmoide na saída é fundamental para transformar os valores numéricos em probabilidades entre 0 e 1, permitindo interpretar os resultados de maneira clara: valores próximos de 1 indicam que um cogumelo é comestível, enquanto valores próximos de 0 indicam que é venenoso.

Essa configuração foi desenvolvida para garantir um equilíbrio entre desempenho e generalização, resultando em um modelo eficiente e confiável para a classificação de cogumelos.

6. Implementação do modelo

A seguir, apresenta-se o código completo desenvolvido para a construção, treinamento e avaliação da rede neural artificial.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/mushroo
m/agaricus-lepiota.data"
columns = ["class", "cap-shape", "cap-surface", "cap-color",
"bruises", "odor", "gill-attachment",
           "gill-spacing", "gill-size", "gill-color",
"stalk-shape", "stalk-root", "stalk-surface-above-ring",
           "stalk-surface-below-ring", "stalk-color-above-ring",
"stalk-color-below-ring", "veil-type",
           "veil-color", "ring-number", "ring-type",
"spore-print-color", "population", "habitat"]

data = pd.read_csv(url, names=columns)

X = data.drop("class", axis=1)
y = data["class"]

label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

X = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

model = keras.Sequential([
    layers.Dense(32, activation='relu',
input_shape=(X_train.shape[1],)),
    layers.Dense(16, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

epochs = 20
history = model.fit(X_train, y_train, epochs=epochs,
batch_size=16, validation_data=(X_test, y_test))

loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

6.1 Explicação do código

A seguir, será apresentada uma explicação detalhada de cada parte do código, abordando a lógica por trás de cada etapa e as decisões tomadas para a construção da rede neural.

1. Importação das Bibliotecas

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

- pandas e numpy: usados para manipulação e processamento de dados.
- sklearn.model_selection: permite dividir o conjunto de dados em treino e teste.
- sklearn.preprocessing: contém ferramentas para transformar variáveis categóricas e padronizar os dados.
- tensorflow e keras: utilizados para a construção e treinamento da rede neural.

2. Carregamento do Conjunto de Dados

```
url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/mushroo
m/agaricus-lepiota.data"
columns = ["class", "cap-shape", "cap-surface", "cap-color",
"bruises", "odor", "gill-attachment",
           "gill-spacing", "gill-size", "gill-color",
"stalk-shape", "stalk-root", "stalk-surface-above-ring",
           "stalk-surface-below-ring", "stalk-color-above-ring",
"stalk-color-below-ring", "veil-type",
           "veil-color", "ring-number", "ring-type",
"spore-print-color", "population", "habitat"]

data = pd.read_csv(url, names=columns)
```

- O conjunto de dados é carregado diretamente do UCI Machine Learning Repository e armazenado em um DataFrame .
- As colunas são nomeadas conforme a descrição da base de dados

3. Separação entre rótulos e atributos

```
X = data.drop("class", axis=1)  # Remove a coluna "class" (rótulo)
y = data["class"]  # Define os rótulos ("e" para comestível, "p"
para venenoso)
```

- X: contém os atributos dos cogumelos.
- y: contém as classes ("e" = comestível, "p" = venenoso).

4. Codificação dos Rótulos

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

Transformação y em valores numéricos:

- 0 → comestível ("e")
- 1 → venenoso ("p")

5. Conversão de Variáveis Categóricas

```
X = pd.get_dummies(X)
```

Como todos os atributos são categóricos, o One-Hot Encoding é aplicado, convertendo cada categoria em uma variável binária.

6. Divisão do Conjunto de Dados

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

- 80% dos dados são usados para treino e 20% para teste .
- A divisão garante que o modelo aprenda com a maioria dos dados, mas seja testado em exemplos nunca vistos
- “random_state=42” assegura que a divisão seja sempre a mesma em execuções diferentes.

7. Padronização dos Dados

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

- O StandardScalernormaliza os dados, garantindo que cada variável tenha média 0 e desvio padrão 1 .

- Isso melhorou a convergência da rede neural, especialmente ao usar a ativação ReLU.

8. Definição do Modelo e Hiperparâmetros

```
model = keras.Sequential([
    layers.Dense(64, activation='relu',
input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

- Camada de entrada: recebe os atributos processados.
- Primeira camada oculta: 64 neurônios, ativação ReLU .
- Segunda camada oculta: 32 neurônios, ativação ReLU .
- Camada de saída: 1 neurônio com ativação sigmoide , produzindo uma probabilidade entre 0 e 1.

9. Compilação do Modelo

```
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

- Otimizador Adam : ajusta os pesos da rede para minimizar o erro.
- Função de perda : binary_crossentropy, usada para classificação binária.
- Métrica de avaliação : accuracy, que mede a taxa de acertos.

10. Treinamento e Validação do Modelo

```
epochs = 20
history = model.fit(X_train, y_train, epochs=epochs,
batch_size=16, validation_data=(X_test, y_test))
```

- epochs=20: o modelo passa pelos dados 20 vezes.
- batch_size=16: os dados são processados em lotes de 16 exemplos por vez.
- validation_data=(X_test, y_test): permite monitorar o desempenho com dados nunca vistos.

11. Avaliação do Modelo

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

- Avalia o modelo no conjunto de teste.
- Retorna a precisão final.

Após o treinamento, a rede neural é testada com novos dados. O resultado esperado inclui uma acurácia elevada (próxima de 100%), pois o conjunto de dados é bem estruturado e não apresenta grande complexidade.

A partir da análise da matriz de confusão e do relatório de classificação, podemos confirmar que a rede neural é capaz de identificar corretamente cogumelos comestíveis e venenosos, demonstrando a eficácia do modelo.

7. Resultados e análise

Após o treinamento, avaliamos o modelo no conjunto de teste para verificar sua capacidade de generalização. A métrica principal utilizada foi a acurácia , que mede a proporção de predições corretas em relação ao total de exemplos testados.

A avaliação do modelo resultou nos seguintes resultados:

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

O modelo alcançou uma acurácia superior a 95% , o que indica um bom desempenho na tarefa de classificação. Essa alta precisão sugere que a rede neural conseguiu capturar padrões relevantes nos dados e distinguir corretamente as classes de cogumelos.

Se compararmos a rede neural com modelos clássicos de aprendizado de máquina, como Árvores de Decisão , Random Forest e Support Vector Machines (SVM) , podemos observar que:

- Árvores de decisão tendem a ter um desempenho sólido e interpretável, mas podem ser sensíveis a dados ruidosos.
- Random Forest oferece robustez e estabilidade, com overfitting mais rápido.
- SVM pode ser eficiente, especialmente com ajustes adequados de hiperparâmetros.

Uma comparação experimental poderia ser realizada para verificar qual abordagem se sai melhor na classificação dos cogumelos.

7.1 Matriz de Confusão e Análise dos Erros

Para analisar o desempenho do modelo de rede neural, utilizou-se a matriz de confusão, que é uma ferramenta visual e intuitiva para entender como o modelo está classificando os dados. Ela organiza os resultados em quatro categorias principais: Verdadeiros Positivos (VP), onde o modelo acertou a classe positiva; Falsos Positivos (FP), casos em que o modelo previu positivo, mas a resposta correta era negativa; Verdadeiros Negativos (VN), quando o modelo acertou a classe negativa; e Falsos Negativos (FN), onde o modelo errou ao prever a classe negativa. Com essa estrutura, é possível identificar padrões de acertos e erros, calcular métricas como precisão, recall e acurácia, e entender melhor onde o modelo pode estar falhando ou se saindo bem. Em resumo, a matriz de confusão é uma maneira clara e eficiente de avaliar a qualidade das previsões do modelo. O código abaixo gera a matriz de confusão a partir das variações do modelo:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype("int32")
```

```
cm = confusion_matrix(y_test, y_pred_classes)

plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=['Comestível', 'Venenoso'], yticklabels=['Comestível',
'Venenoso'])
plt.xlabel('Predito')
plt.ylabel('Real')
plt.title('Matriz de Confusão')
plt.show()
```

A matriz de confusão permite analisar onde o modelo teve mais dificuldades.

- Verdadeiros Positivos (TP): Número de cogumelos venenosos corretamente classificados como venenosos.
- Verdadeiros Negativos (TN): Número de cogumelos comestíveis corretamente classificados como comestíveis.
- Falsos Positivos (FP): Cogumelos comestíveis erroneamente classificados como venenosos.
- Falsos Negativos (FN): Cogumelos venenosos erroneamente classificados como comestíveis (o erro mais crítico).

Um pequeno número de falsificações negativas pode representar um problema grave, pois cogumelos venenosos encontrados incorretamente como comestíveis podem causar intoxicações. Esse fator deve ser minimizado para garantir a segurança da classificação.

7.2 Limitações do modelo

O modelo desenvolvido, apesar de apresentar um bom desempenho na classificação de cogumelos como comestíveis ou venenosos, possui algumas limitações que devem ser consideradas. Uma das questões principais é o possível desequilíbrio nos dados, onde a distribuição entre as classes pode não ser uniforme. Se houver muito mais exemplos de uma classe do que da outra, o modelo pode se tornar tendencioso, favorecendo a classe majoritária e resultando em uma menor capacidade de generalização. Para mitigar esse problema, podem ser utilizadas técnicas como oversampling , que aumenta a quantidade de exemplos da classe minoritária, e undersampling , que reduz a quantidade de exemplos da classe majoritária, tornando o conjunto de dados mais balanceado.

Outra limitação relevante é o overfitting , que ocorre quando o modelo se ajusta aos dados de treinamento, apresentando um desempenho muito alto nesta fase, mas inferior na validação e em novos dados. Isso pode ser combinado com a aplicação de técnicas como Dropout , que desativa aleatoriamente neurônios durante o treinamento, evitando a dependência excessiva de padrões específicos, e regularização L2 , que penaliza pesos elevados na rede, favorecendo uma melhor generalização.

Além disso, redes neurais artificiais são modelos de difícil interpretabilidade, muitas vezes chamadas de caixa-preta . Isso significa que, embora o modelo consiga fazer variações precisas, entender exatamente quais fatores levaram a determinada classificação pode ser solicitado. Para contornar essa limitação, podem ser utilizadas técnicas como SHAP

(SHapley Additive exPlanations) , que ajudam a interpretar a importância de cada atributo na decisão final do modelo, tornando suas limitações mais compreensíveis.

Para melhorar o desempenho do modelo, algumas estratégias podem ser adotadas:

1. Ajuste de Hiperparâmetros
    - Testar diferentes detalhes de neurônios e camadas ocultas.
    - Experimentar funções de ativação alternativas, como ReLU, Sigmoid e Tanh.
    - Ajustar a taxa de aprendizagem do otimizador Adam.
2. Uso de Técnicas de Regularização
    - Adicione camadas Dropout para evitar overfitting.
    - Aplicar regularização L1/L2 para penalizar pesos elevados na rede.
3. Aumento do Conjunto de Dados
    - Explore outras bases de dados sobre cogumelos.
    - Gerar novos exemplos sintéticos a partir dos dados existentes.
4. Exploração de Outras Abordagens
    - Comparar a RNA com outros algoritmos de aprendizado de máquina.
    - Aplicar técnicas de IA explicável (XAI) para tornar o modelo mais interpretável.

Os resultados demonstram que a rede neural artificial foi eficaz na classificação de cogumelos comestíveis e venenosos. No entanto, para garantir uma segurança, é fundamental minimizar as falsas negativas e explorar técnicas que aumentem a confiabilidade do modelo.

8. Conclusão

Neste trabalho, desenvolvemos um modelo de rede neural artificial para a classificação de cogumelos como comestíveis ou venenosos, utilizando a base de dados *Mushroom Dataset* do U*Codificação One-Hot* e padro*StandardScaler* . Além disso

Os resultados obtidos demonstraram que a rede neural foi altamente eficaz na classificação dos cogumelos, atingindo uma acurácia de 100% no conjunto de testes. A matriz de confusão indicou que o modelo conseguiu separar corretamente todas as amostras, sem erros de classificação. Isso mostra que a rede neural conseguiu identificar corretamente os padrões presentes nos atributos dos cogumelos, diferenciando de maneira precisa os venenosos dos comestíveis.

A alta precisão sugere que o modelo pode ser uma ferramenta útil para auxiliar na identificação de cogumelos, porém, para aplicações reais, seria necessário realizar mais testes com dados coletados em ambientes naturais, fora do conjunto de dados utilizados. Além disso, trabalhos futuros podem explorar técnicas como *aumento de dados* , regularização e experimentação com diferentes arquiteturas para validar a robustez do modelo em diferentes cenários.

Com isso, o estudo contribui para a compreensão do uso de redes neurais na classificação de dados categóricos e reforçar a importância da inteligência artificial no reconhecimento de padrões e na tomada de decisões automatizadas.

9. Referências Bibliográficas

UCI MACHINE LEARNING REPOSITORY. *Mushroom Database*. University of California, Irvine, 1987. Disponível em: https://archive.ics.uci.edu/dataset/73/mushroom. Acesso em: 21 fev. 2025.

BISHOP, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. Cambridge: MIT Press, 2016.

HAYKIN, Simon. *Neural Networks and Learning Machines*. 3. ed. New York: Pearson, 2009.