Questao 5

Codigo Fonte:

```
# --- Bibliotecas Necessárias ---
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
import io
from contextlib import redirect_stdout
from fpdf import FPDF
from fpdf.enums import XPos, YPos
import warnings
warnings.filterwarnings('ignore')


def solve_and_capture_output():
    """
    Executa a classificação dos Ipês com Keras e captura o output.
    """
    print("Questao 5: Classificacao de Ipes com Redes Neurais (TensorFlow/Keras)")
    print("=" * 75)

    # --- 1. Declaração dos Dados ---
    # Criando os DataFrames diretamente no código.

    # Dados de Treinamento
    dados_treino_dict = {
        'Comprimento': [26, 19, 18, 17, 24, 17, 15, 16, 14, 23, 17, 18, 17, 25, 12, 22],
         'Largura': [1.50, 1.70, 1.80, 1.30, 1.30, 1.20, 1.60, 1.40, 1.45, 1.70, 1.35, 1.90, 1.75, 1.80, 1.25,
1.50],
           'Tipo': ['Branco', 'Rosa', 'Rosa', 'Amarelo', 'Branco', 'Amarelo', 'Rosa', 'Amarelo', 'Amarelo',
'Branco', 'Amarelo', 'Rosa', 'Rosa', 'Branco', 'Amarelo', 'Branco']
    }
    dados_treino = pd.DataFrame(dados_treino_dict)

    # Dados de Teste
    dados_teste_dict = {
        'Comprimento': [15, 16, 17],
        'Largura': [1.77, 1.87, 1.98]
    }
    dados_teste = pd.DataFrame(dados_teste_dict)

    print("1. Preparacao dos Dados")
    print("\nDados de Treinamento:")
    print(dados_treino)
```

```
# --- 2. Pré-processamento dos Dados ---
# Separar características (X) e rótulos (y)
X_train = dados_treino[['Comprimento', 'Largura']].values
y_train_labels = dados_treino['Tipo'].values

# Codificar rótulos de texto para números (Branco->0, Rosa->1, Amarelo->2)
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train_labels)

# Converter rótulos numéricos para o formato one-hot encoding
# Ex: 0 -> [1,0,0], 1 -> [0,1,0], 2 -> [0,0,1]
y_train_one_hot = to_categorical(y_train_encoded)

# Normalizar as características de entrada (X)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

print("\nLabels transformados para One-Hot Encoding (Branco, Rosa, Amarelo):")
print(y_train_one_hot)
print("\nCaracteristicas de Treino Normalizadas:")
print(X_train_scaled)

# --- 3. Construção do Modelo da Rede Neural ---
print("\n" + "=" * 75)
print("2. Topologia e Treinamento da Rede")

# Definindo o modelo sequencial
modelo = Sequential()
# Camada oculta com 5 neurônios, função de ativação 'relu' e 2 entradas
modelo.add(Dense(5, input_dim=2, activation='relu'))
# Camada de saída com 3 neurônios (um para cada tipo de Ipê) e ativação 'sigmoid'
modelo.add(Dense(3, activation='sigmoid'))

# Compilando o modelo
# Loss 'binary_crossentropy' é usado com sigmoid na saída, como no exemplo.
modelo.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Treinando o modelo
print("\nIniciando o treinamento...")
# verbose=0 para não poluir o output com o log de cada época
modelo.fit(X_train_scaled, y_train_one_hot, epochs=2000, batch_size=5, verbose=0)
print("Treinamento concluido.")

# --- 4. Análise do Modelo Treinado ---
print("\n" + "=" * 75)
print("3. Analise do Modelo Treinado")

print("\nTopologia da Rede (Resumo do Modelo):")
# Capturar o resumo do modelo para exibição
string_stream = io.StringIO()
modelo.summary(print_fn=lambda x: string_stream.write(x + '\n'))
model_summary = string_stream.getvalue()
print(model_summary)
```

```
    print("Tipo de Treinamento: Supervisionado, com algoritmo de backpropagation e otimizador 'adam'.")

    print("\nPesos (Wij) e Bias (bi) Finais:")
    for i, layer in enumerate(modelo.layers):
        weights, biases = layer.get_weights()
        print(f"\n--- Camada {i+1} ---")
        print("Bias (bi):")
        for j, b in enumerate(biases):
            print(f"  Neuronio {j+1}: {b:.4f}")
        print("Pesos (Wij):")
        for j, w_neuron in enumerate(weights.T): # Transpor para formato Neuronio -> Conexão
            for k, w in enumerate(w_neuron):
                print(f"  Entrada {k+1} -> Neuronio {j+1}: {w:.4f}")

    # --- 5. Classificação dos Dados de Teste ---
    print("\n" + "=" * 75)
    print("4. Classificacao dos Ipes de Teste")

    print("\nDados de Teste:")
    print(dados_teste)

    X_test = dados_teste[['Comprimento', 'Largura']].values

    # Normalizar os dados de teste COM O MESMO SCALER usado no treino
    X_test_scaled = scaler.transform(X_test)

    print("\nDados de Teste Normalizados:")
    print(X_test_scaled)

    # Fazer a predição
    predicoes = modelo.predict(X_test_scaled)

    # Encontrar a classe com maior probabilidade para cada predição
    predicoes_classes = np.argmax(predicoes, axis=1)

    # Decodificar os resultados de volta para os nomes originais
    resultado_final = label_encoder.inverse_transform(predicoes_classes)

    print("\nResultado da Classificacao:")
    dados_teste['Tipo de Ipe Predito'] = resultado_final
    print(dados_teste)
    print("=" * 75)

def generate_pdf_report(code_content, output_content):
    """Gera um PDF com o código e o output."""
    pdf = FPDF()
    pdf.add_page()

    pdf.set_font("Courier", 'B', 16)
    pdf.cell(0, 10, 'Questao 5', new_x=XPos.LMARGIN, new_y=YPos.NEXT, align='C')
    pdf.ln(10)
```

```
    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Codigo Fonte:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 8)
    pdf.multi_cell(0, 5, code_content)

    pdf.add_page()

    pdf.set_font("Courier", 'B', 12)
    pdf.cell(0, 10, 'Output da Execucao:', new_x=XPos.LMARGIN, new_y=YPos.NEXT)
    pdf.set_font("Courier", '', 9)
    output_content_safe = output_content.encode('latin-1', 'replace').decode('latin-1')
    pdf.multi_cell(0, 5, output_content_safe)

    pdf_file_name = "resultado_questao_5.pdf"
    pdf.output(pdf_file_name)
    return pdf_file_name

# --- Bloco Principal de Execução ---
if __name__ == "__main__":
    output_buffer = io.StringIO()
    with redirect_stdout(output_buffer):
        solve_and_capture_output()
    output_content = output_buffer.getvalue()

    print("--- [INICIO] Resultado da Execucao no Terminal ---")
    print(output_content)
    print("--- [FIM] Resultado da Execucao no Terminal ---")

    try:
        with open(__file__, 'r', encoding='utf-8') as f:
            code_content = f.read()
    except Exception as e:
        code_content = f"Nao foi possivel ler o arquivo do codigo: {e}"

    try:
        pdf_file = generate_pdf_report(code_content, output_content)
        print(f"\nPDF '{pdf_file}' gerado com sucesso no diretorio atual!")
    except Exception as e:
        print(f"\nOcorreu um erro ao gerar o PDF: {e}")
```

Output da Execucao:

```
Questao 5: Classificacao de Ipes com Redes Neurais (TensorFlow/Keras)
================================================================================
1. Preparacao dos Dados

Dados de Treinamento:
    Comprimento  Largura     Tipo
0            26     1.50   Branco
1            19     1.70     Rosa
2            18     1.80     Rosa
3            17     1.30  Amarelo
4            24     1.30   Branco
5            17     1.20  Amarelo
6            15     1.60     Rosa
7            16     1.40  Amarelo
8            14     1.45  Amarelo
9            23     1.70   Branco
10           17     1.35  Amarelo
11           18     1.90     Rosa
12           17     1.75     Rosa
13           25     1.80   Branco
14           12     1.25  Amarelo
15           22     1.50   Branco

Labels transformados para One-Hot Encoding (Branco, Rosa, Amarelo):
[[0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]]

Caracteristicas de Treino Normalizadas:
[[ 1.83046377 -0.1448819 ]
 [ 0.06311944  0.78236224]
 [-0.18935832  1.24598431]
 [-0.44183608 -1.07212604]
 [ 1.32550825 -1.07212604]
 [-0.44183608 -1.53574811]
 [-0.9467916   0.31874017]
 [-0.69431384 -0.60850397]
 [-1.19926937 -0.37669293]
```

```
 [ 1.07303049  0.78236224]
 [-0.44183608 -0.840315  ]
 [-0.18935832  1.70960638]
 [-0.44183608  1.01417328]
 [ 1.57798601  1.24598431]
 [-1.70422489 -1.30393707]
 [ 0.82055272 -0.1448819 ]]

================================================================================
2. Topologia e Treinamento da Rede

Iniciando o treinamento...
Treinamento concluido.

================================================================================
3. Analise do Modelo Treinado

Topologia da Rede (Resumo do Modelo):
Model: "sequential"
????????????????????????????????????????????????????????????????????????????????
? Layer (type)                         ? Output Shape                ?         Param # ?
????????????????????????????????????????????????????????????????????????????????
? dense (Dense)                        ? (None, 5)                   ?              15 ?
????????????????????????????????????????????????????????????????????????????????
? dense_1 (Dense)                      ? (None, 3)                   ?              18 ?
????????????????????????????????????????????????????????????????????????????????
 Total params: 101 (408.00 B)
 Trainable params: 33 (132.00 B)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 68 (276.00 B)


Tipo de Treinamento: Supervisionado, com algoritmo de backpropagation e otimizador 'adam'.

Pesos (Wij) e Bias (bi) Finais:

--- Camada 1 ---
Bias (bi):
  Neuronio 1: 0.5171
  Neuronio 2: 0.7782
  Neuronio 3: 0.9143
  Neuronio 4: 0.7525
  Neuronio 5: 0.6157
Pesos (Wij):
  Entrada 1 -> Neuronio 1: -0.4101
  Entrada 2 -> Neuronio 1: -2.6916
  Entrada 1 -> Neuronio 2: -0.4425
  Entrada 2 -> Neuronio 2: 2.4612
  Entrada 1 -> Neuronio 3: 3.5961
  Entrada 2 -> Neuronio 3: -0.4170
  Entrada 1 -> Neuronio 4: -1.7158
  Entrada 2 -> Neuronio 4: -1.9677
  Entrada 1 -> Neuronio 5: -1.7640
```

```
  Entrada 2 -> Neuronio 5: 2.1172

--- Camada 2 ---
Bias (bi):
  Neuronio 1: 0.2454
  Neuronio 2: -0.1435
  Neuronio 3: 0.2779
Pesos (Wij):
  Entrada 1 -> Neuronio 1: 2.1435
  Entrada 2 -> Neuronio 1: -2.0336
  Entrada 3 -> Neuronio 1: -2.4506
  Entrada 4 -> Neuronio 1: 1.2856
  Entrada 5 -> Neuronio 1: -1.4851
  Entrada 1 -> Neuronio 2: -0.2708
  Entrada 2 -> Neuronio 2: -0.4587
  Entrada 3 -> Neuronio 2: 1.8100
  Entrada 4 -> Neuronio 2: -2.1930
  Entrada 5 -> Neuronio 2: -2.5057
  Entrada 1 -> Neuronio 3: -2.7496
  Entrada 2 -> Neuronio 3: 1.0212
  Entrada 3 -> Neuronio 3: -1.9034
  Entrada 4 -> Neuronio 3: -0.9530
  Entrada 5 -> Neuronio 3: 1.6078

================================================================================
4. Classificacao dos Ipes de Teste

Dados de Teste:
   Comprimento  Largura
0           15     1.77
1           16     1.87
2           17     1.98

Dados de Teste Normalizados:
[[-0.9467916   1.10689769]
 [-0.69431384  1.57051976]
 [-0.44183608  2.08050404]]
 [1m1/1 [0m                    [32m??????????????????? [0m [37m [0m                     [1m0s [0m
114ms/step                                    [1m1/1 [0m   [32m??????????????????? [0m [37m [0m
 [1m0s [0m 126ms/step

Resultado da Classificacao:
   Comprimento  Largura Tipo de Ipe Predito
0           15     1.77                Rosa
1           16     1.87                Rosa
2           17     1.98                Rosa
================================================================================
```