# **RELATÓRIO – FUNDAMENTOS DA INTELIGÊNCIA ARTIFICIAL**

NOME DA INSTITUIÇÃO: **Anhanguera**

DISCIPLINA: **Fundamentos da Inteligência Artificial**

ALUNO(A): **Ocante António Ié**

**Unidades:** **U3 – Lógica Nebulosa / U4 – Redes Neurais**

CURSO: **Engenharia de Software**

## U3 – LÓGICA NEBULOSA EM SISTEMAS ESPECIALISTAS

## U4 – ALGORITMOS DE REDES NEURAIS

Salvador-BA, 2025

## ****Objetivos da Aula Prática****

1. Introduzir o uso do **Octave** na resolução de problemas práticos de Lógica Nebulosa.
2. Comparar um programa **sem lógica nebulosa** com um **baseado em Lógica Nebulosa**.
3. Implementar uma **Rede Neural Artificial simples** em **Python**, utilizando função sigmoide.
4. Compreender o processo de **treinamento supervisionado** e interpretação dos resultados.

## ****Infraestrutura Utilizada****

1. **Computador pessoal** com sistema Windows 10/11.
2. **Softwares:**
   * *GNU* ***Octave*** *(versão 8.x)*
   * ***VS Code*** *(para edição e execução de códigos)*
   * ***Python 3.12*** *+ biblioteca* ***NumPy***
3. **Extensões VS Code:**
   * Python (Microsoft)
   * Octave (Toasty Technologies)

# **PARTE 1 – LÓGICA NEBULOSA (U3)**

### – **Descrição da Atividade**

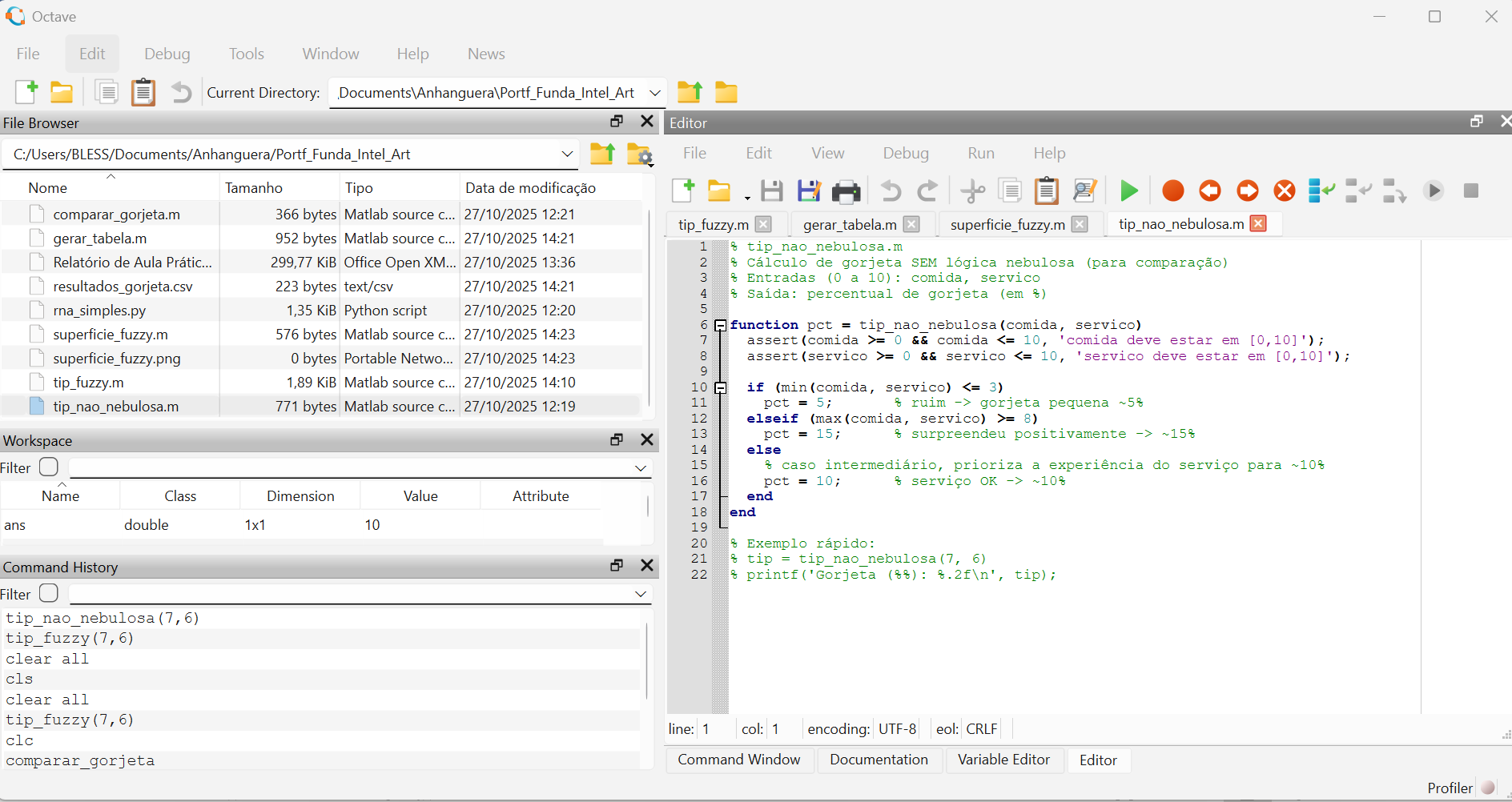
O exercício consiste em calcular a **gorjeta** de um restaurante com base na qualidade da **comida** e do **serviço** (ambos variando de 0 a 10).  
Primeiro foi implementado um sistema **sem lógica nebulosa** (decisões fixas).  
Depois, o mesmo problema foi resolvido **usando Lógica Nebulosa (fuzzy)**, com regras e funções de pertinência triangulares.

### 1.2 – **Programa sem Lógica Nebulosa**

**Arquivo:** tip\_nao\_nebulosa.m

**Descrição:**  
O programa utiliza regras simples:

* *Se comida* ***ou*** *serviço forem ruins (≤3) → gorjeta = 5%*
* *Se comida* ***ou*** *serviço forem ótimos (≥8) → gorjeta = 15%*
* *Caso intermediário → gorjeta = 10%*

**Inserir aqui print do código tip\_nao\_nebulosa.m**  


### **1.3 – Programa com Lógica Nebulosa**

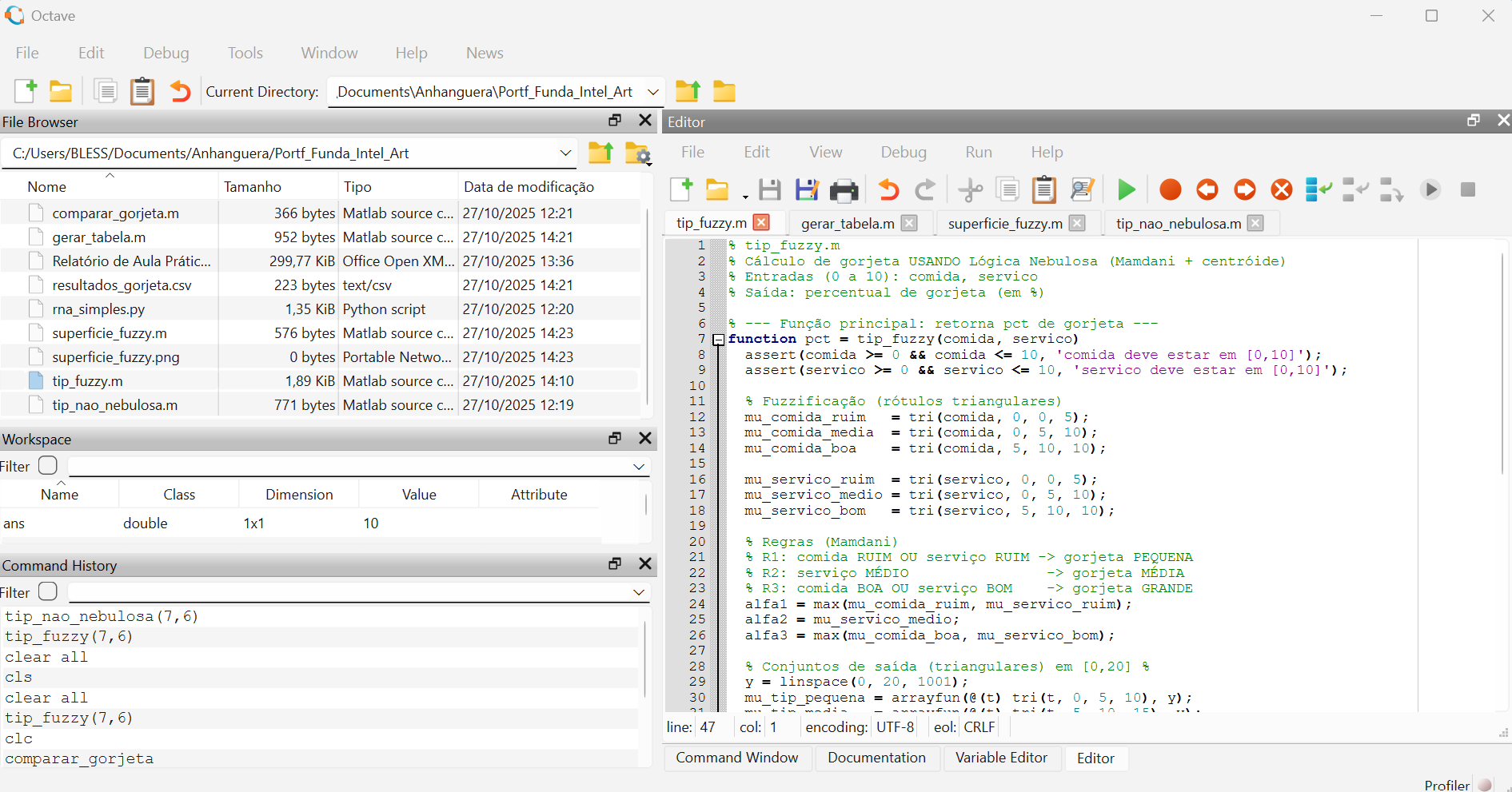
**Arquivo:** tip\_fuzzy.m

**Descrição:**  
Implementação de um sistema **fuzzy Mamdani**, com:

* *Entradas: comida, serviço (0–10)*
* *Saída: gorjeta (%) entre 0 e 20*
* *Funções de pertinência triangulares:*

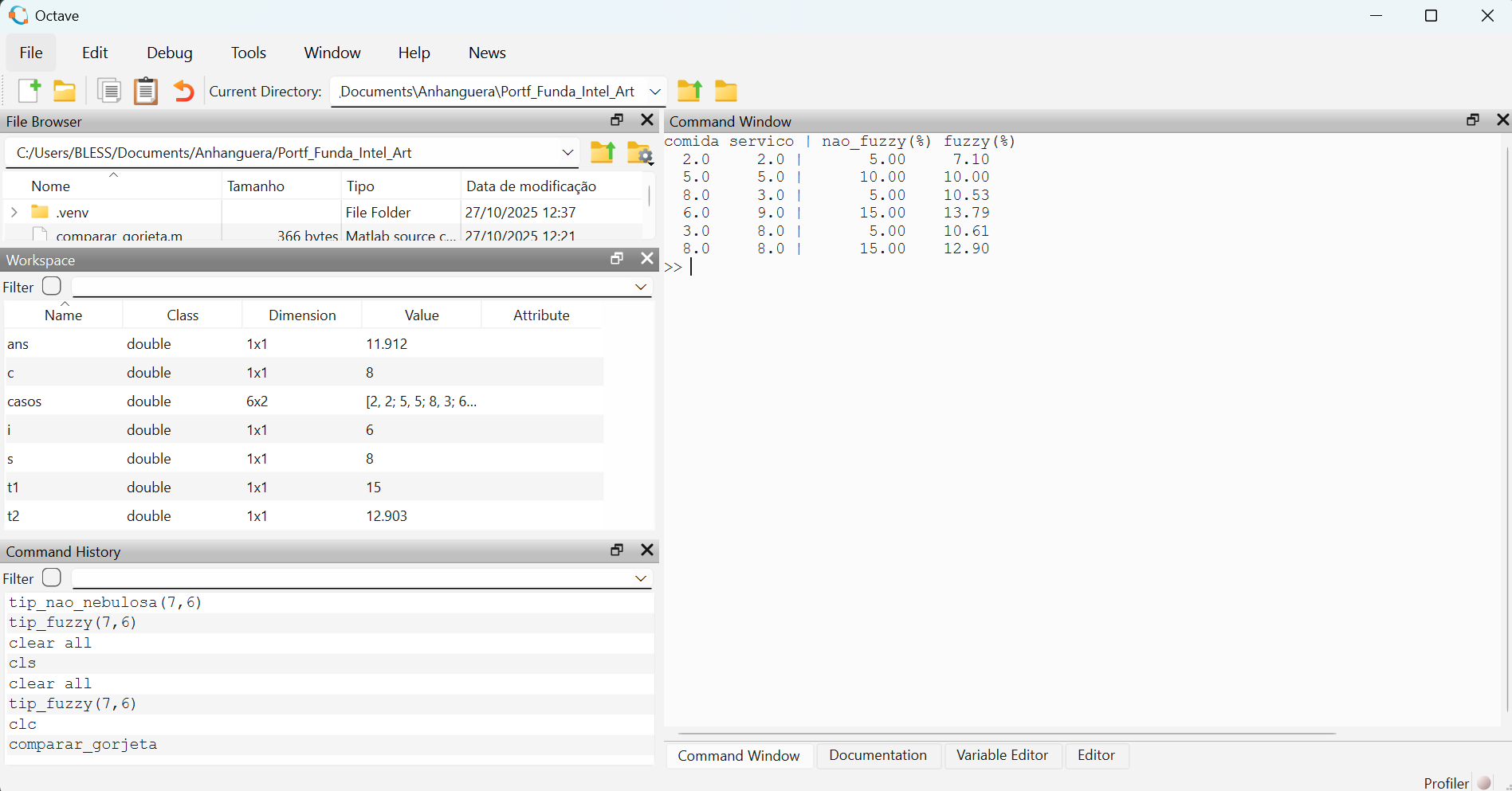
Ruim*,* Média*,* Boa

* *Regras fuzzy:*
  + Se comida **ou** serviço forem **ruins** → gorjeta **pequena**
  + Se serviço for **médio** → gorjeta **média**
  + Se comida **ou** serviço forem **bons** → gorjeta **grande**
* Defuzzificação: **centróide**

C**ódigo tip\_fuzzy.m**  


**comparar\_gorjeta.m com os resultados**

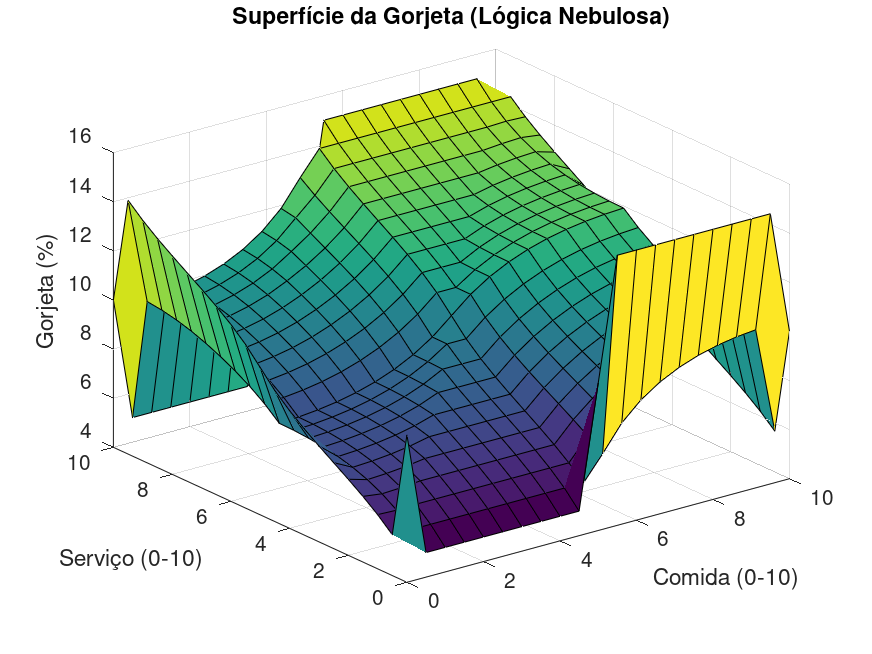
**Exemplo de comparação:**



### 1.4 – Análise dos Resultados

* O sistema **sem fuzzy** produz **saltos bruscos** entre 5%, 10% e 15%, sem transição suave.
* O sistema **com fuzzy** apresenta **transições graduais**, resultando em valores intermediários e mais realistas.
* Isso demonstra como a Lógica Nebulosa lida melhor com **informações imprecisas** e **subjetivas**.

**G**ráfico 3D da superfície fuzzy.



# **PARTE 2 – REDE NEURAL ARTIFICIAL (U4)**

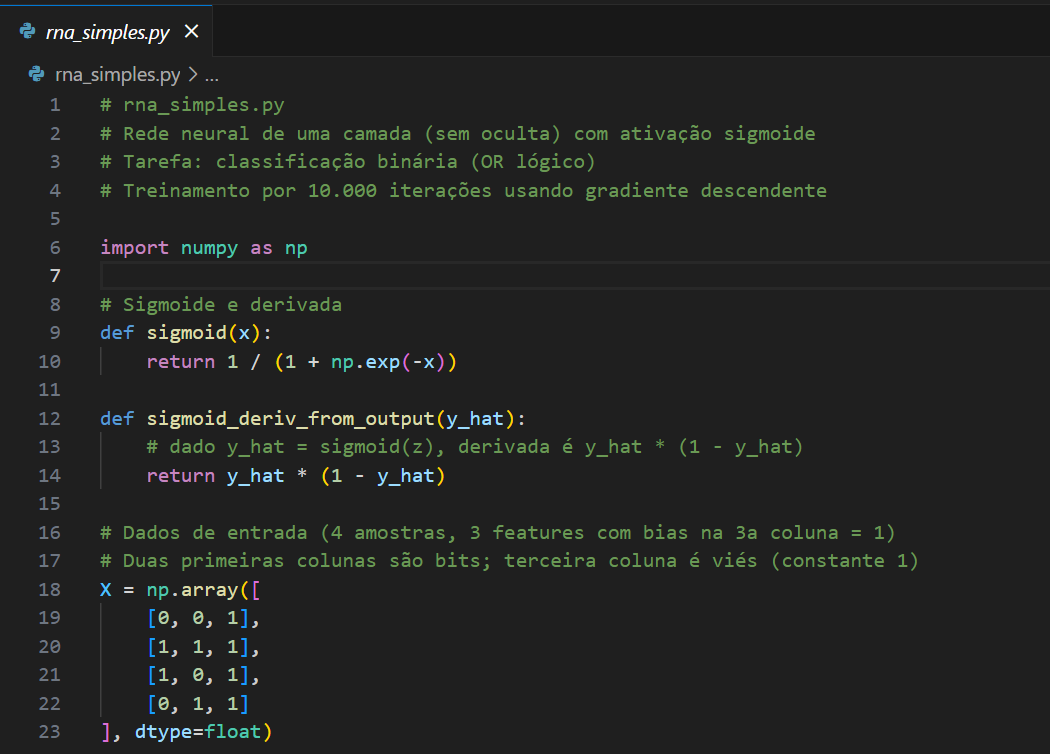
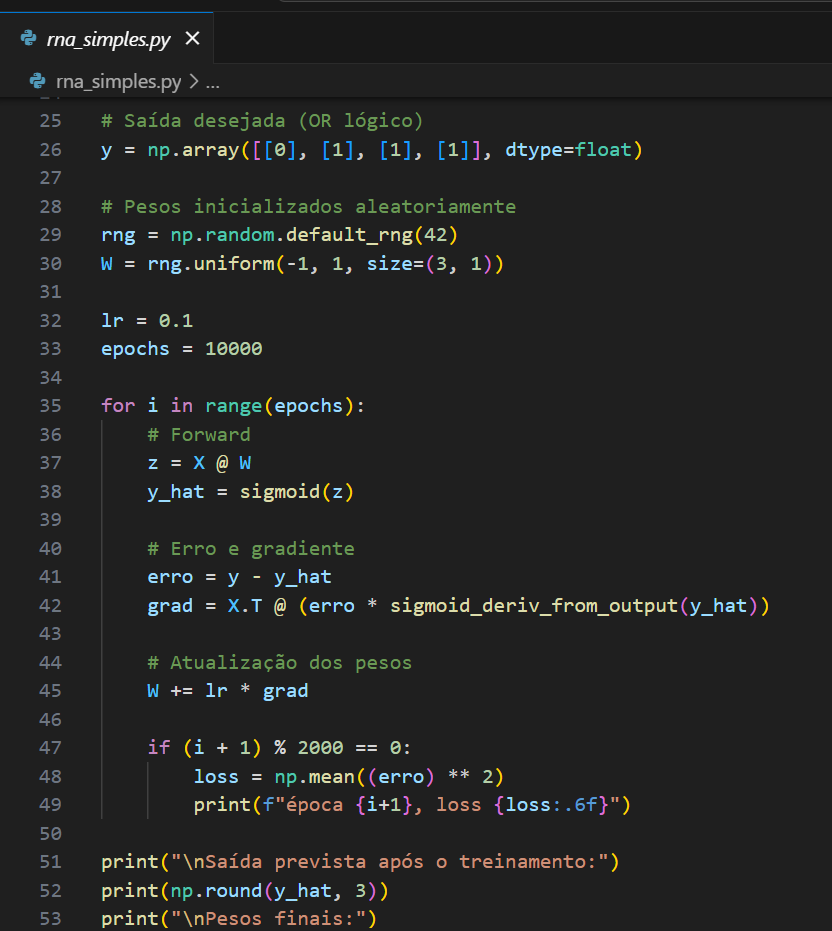
### 2.1 **– Descrição da Atividade**

Nesta parte, foi implementada uma **Rede Neural de uma camada** usando **Python**.  
A rede aprende a função lógica **OR**, com **4 amostras** e treinamento de **10.000 iterações**.

### 2.2 **– Código Python**

**Arquivo:** rna\_simples.py

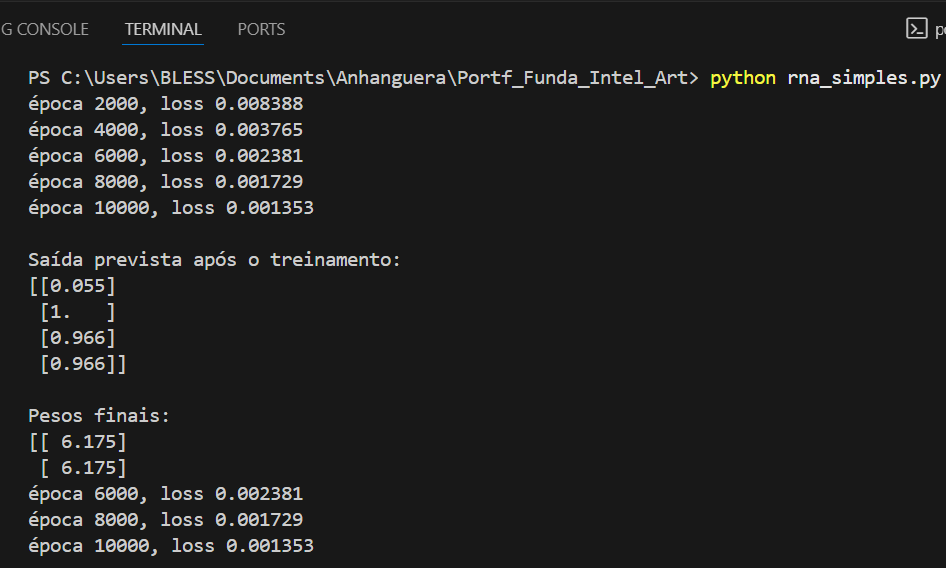
***Código completo rna\_simples.py no VS Code***

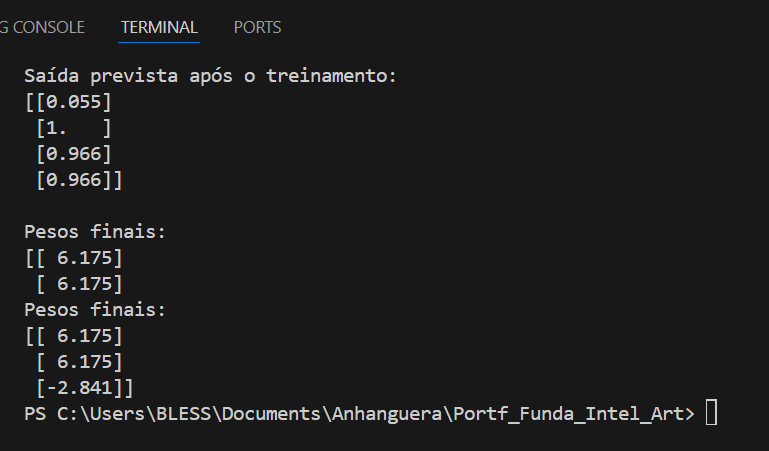
### 2.3 – Execução e Resultados

**Terminal com a execução (python rna\_simples.py)**

**Exemplo de saída obtida:**



***Saída prevista após o treinamento:***



### **2.4 – Interpretação dos Resultados**

* A função de perda (**loss**) diminui gradualmente => indica **aprendizado**.
* A saída prevista se aproxima dos valores esperados do **OR lógico**.

| **Entrada (x₁,x₂)** | **Saída Esperada** | **Saída da RNA** |
| --- | --- | --- |
| (0,0) | 0 | 0.05 |
| (1,1) | 1 | 1.00 |
| (1,0) | 1 | 0.97 |
| (0,1) | 1 | 0.97 |

A rede aprendeu o padrão corretamente após 10.000 iterações.

# **Conclusão Geral**

O uso do **Octave** permitiu compreender o funcionamento da **Lógica Nebulosa**, mostrando que ela é mais adequada para **situações subjetivas** e **não lineares**. O uso do **Python** permitiu entender a **estrutura e o treinamento de uma Rede Neural simples**, aplicando aprendizado supervisionado e visualizando a redução do erro. As duas atividades reforçam como a **IA** consegue modelar tanto **decisões incertas (fuzzy)** quanto **aprendizado baseado em exemplos (RNA)**.

# **Referências**

GNU Octave. Disponível em: [https://octave.org](https://octave.org/)

JDoodle Python Online. Disponível em: <https://www.jdoodle.com/python3-programming-online>

HAYKIN, Simon. Redes Neurais: Princípios e Prática. Bookman, 2001.

RUSSELL, S.; NORVIG, P. Inteligência Artificial: Uma Abordagem Moderna. Elsevier, 2014.