

# DESAFIO DE CÓDIGO – SIMULANDO DESAFIO COM IA GENERATIVAS

## USO DA FERRAMENTA DO CHATGPT

### DESAFIO 1

#### Descrição

Imagine que você foi designado a criar um algoritmo para analisar o sentimento de um comentário fornecido pelo usuário, simulando análises de sentimentos, um assunto muito comentado dentro do Machine Learning. O programa solicitará ao usuário que insira um comentário, e em seguida, dividirá esse comentário em palavras individuais.

Após isso, ele contará o número de palavras positivas, negativas e neutras dentro do comentário, baseando-se em uma lista pré-definida de palavras-chave. As palavras consideradas positivas incluem "bom", "ótimo", "excelente", "maravilhoso", "gostei" e "incrível" enquanto as palavras negativas incluem "ruim", "péssimo", "horrível", "terrível" e "odeio". Já as palavras neutras incluem "mas", "deixou", "apesar" e "embora".

Depois de calcular as contagens de palavras positivas e negativas, o programa determinará o sentimento predominante do comentário. Se houver mais palavras positivas do que negativas, o sentimento será considerado positivo. Se houver mais palavras negativas do que positivas, o sentimento será considerado negativo. Caso contrário, se houver um número igual de palavras positivas e negativas, o sentimento será neutro.

#### Entrada

O usuário será solicitado a fornecer um comentário como entrada para o programa.

#### Saída

O programa exibirá o sentimento do comentário inserido pelo usuário, que pode ser "Positivo", "Negativo" ou "Neutro", dependendo da análise das palavras-chave no comentário.

#### Exemplos

A tabela abaixo apresenta exemplos com alguns dados de entrada e suas respectivas saídas esperadas. Certifique-se de testar seu programa com esses exemplos e com outros casos possíveis.

Entrada	Saída
A mentoria foi incrível, aprendi muito!	Sentimento: Positivo
O clima hoje está terrível, odeio dias quentes.	Sentimento: Negativo
A comida estava boa, mas o serviço deixou a desejar.	Sentimento: Neutro

## CÓDIGO IMCOMPLETO - PROMPT DESAFIO 1

Preciso que analise o código abaixo em python e faça correção no que for preciso para ter as devidas entradas e saídas solicitadas.

```
# Importa o módulo re, que é a biblioteca de expressões regulares do Python.  
import re
```

```
def analyze_sentiment():
```

```
    # Entrada do usuário  
    comentario = input()
```

```
    # Divisão do comentário em palavras  
    palavras = re.findall(r'\b\w+\b', comentario.lower())
```

```
    # Lista de palavras positivas, negativas e neutras
```

```
    positivas = ["bom", "boa", "ótimo", "excelente", "maravilhoso", "gostei", "incrível"]
```

```
    negativas = ["ruim", "péssimo", "horrível", "terrível", "odeio"]
```

```
    neutras = ["mas", "deixou", "apesar", "embora"]
```

```
    # Contagem de palavras positivas, negativas e neutras
```

```
    count_positivo = sum(palavra in positivas for palavra in palavras)
```

```
    count_negativo = sum(palavra in negativas for palavra in palavras)
```

```
    # TODO: Conte quantas palavras neutras estão presentes no comentário.
```

```
    # Verifica se há mais palavras positivas do que negativas no comentário e se não há palavras  
    neutras. Se essa condição for verdadeira, o comentário é considerado positivo.
```

```
    if count_positivo > count_negativo and count_neutro == 0:  
        return "Positivo"
```

```
    # TODO: Complete a codição para determinar o sentimento com base na contagem de  
    palavras
```

```
# Saída esperada
```

```
sentimento = analyze_sentiment()  
print("Sentimento:", sentimento)
```

```
{REGRAS}
```

```
> Entrada, O usuário será solicitado a fornecer um comentário como entrada para o programa.
```

```
> Saída, O programa exibirá o sentimento do comentário inserido pelo usuário, que pode ser  
"Positivo", "Negativo" ou "Neutro", dependendo da análise das palavras-chave no comentário.
```

## RETORNO DA CORREÇÃO DA IA PELO CHATGPT

# Importa o módulo re, que é a biblioteca de expressões regulares do Python.

```
import re
```

```
import re
```

```
def analyze_sentiment():
```

```
    # Entrada do usuário
```

```
    comentario = input(" ")
```

```
    # Divisão do comentário em palavras
```

```
    palavras = re.findall(r'\b\w+\b', comentario.lower( ))
```

```
    # Lista de palavras positivas, negativas e neutras
```

```
    positivas = ["bom", "boa", "ótimo", "excelente", "maravilhoso", "gostei", "incrível"]
```

```
    negativas = ["ruim", "péssimo", "horrível", "terrível", "odeio"]
```

```
    neutras = ["mas", "deixou", "apesar", "embora"]
```

```
    # Contagem de palavras positivas, negativas e neutras
```

```
    count_positivo = sum(palavra in positivas for palavra in palavras)
```

```
    count_negativo = sum(palavra in negativas for palavra in palavras)
```

```
    count_neutro = sum(palavra in neutras for palavra in palavras)
```

```
    # Verifica o sentimento do comentário
```

```
    if count_positivo > count_negativo and count_neutro == 0:
```

```
        return "Positivo"
```

```
    elif count_negativo > count_positivo and count_neutro == 0:
```

```
        return "Negativo"
```

```
    else:
```

```
        return "Neutro"
```

```
# Saída esperada
```

```
sentimento = analyze_sentiment( )
```

```
print("Sentimento:", sentimento)
```

## DESAFIO 2

### Descrição

Neste desafio, você será solicitado a criar um algoritmo que avalia se um prompt fornecido pelo usuário está adequado. O programa solicitará ao usuário que insira um prompt e, em seguida, verificará se o prompt contém palavras-chave relevantes. As palavras-chave consideradas relevantes serão "inteligência artificial", "sistemas de recomendação online", "exemplo de conversação", "explique conceitos" e "dicas de tecnologia". Se o prompt incluir pelo menos uma dessas palavras-chave, o programa informará que o prompt está adequado; caso contrário, ele indicará que o prompt não está adequado e sugerirá ao usuário que inclua palavras-chave relevantes.

### Entrada

O usuário será solicitado a inserir um prompt como entrada para o programa.

### Saída

O programa exibirá feedback para o usuário com base na avaliação do prompt inserido. Se o prompt contiver palavras-chave relevantes, o programa informará que o prompt está adequado. Caso contrário, ele indicará que o prompt não está adequado e sugerirá ao usuário que inclua palavras-chave relevantes.

### Exemplos

A tabela abaixo apresenta exemplos com alguns dados de entrada e suas respectivas saídas esperadas. Certifique-se de testar seu programa com esses exemplos e com outros casos possíveis.

Entrada	Saída
Por favor, explique conceitos de inteligência artificial.	O prompt está adequado.
Crie exemplos de conversação.	O prompt está adequado.
Qual é a coisa mais bonita do mundo?	O prompt não está adequado. Inclua palavras-chave relevantes.

## CÓDIGO IMCOMPLETO - PROMPT DESAFIO 2

Preciso que analise o código abaixo em python e faça correção no que for preciso para ter as devidas entradas e saídas solicitadas.

Entrada do usuário

```
prompt_usuario = input()
```

# Função para avaliar se o prompt está adequado

```
def avaliar_prompt(prompt):
```

```
    # Verifica se o prompt contém palavras-chave relevantes
```

```
    palavras_chave = ["inteligência artificial", "sistemas de recomendação online", "exemplos de conversação", "explique conceitos", "dicas de tecnologia" ]
```

```
    # TODO: Aplique a condição necessária para verificar se o prompt está ou não adequado de acordo com o enunciado
```

```
# Avaliar o prompt do usuário
```

```
feedback_usuario = avaliar_prompt(prompt_usuario)
```

```
# Exibir feedback
```

```
print(feedback_usuario)
```

{REGRAS}

> Entrada, O usuário será solicitado a inserir um prompt como entrada para o programa.

> Saída, O programa exibirá feedback para o usuário com base na avaliação do prompt inserido. Se o prompt contiver palavras-chave relevantes, o programa informará que o prompt está adequado. Caso contrário, ele indicará que o prompt não está adequado e sugerirá ao usuário que inclua palavras-chave relevantes.

> Exemplos de entrada e saída 01: Entrada, Por favor, explique conceitos de inteligência artificial. Saída, O prompt está adequado.

> Exemplos de entrada e saída 02: Entrada, Crie exemplos de conversação. Saída, O prompt está adequado.

> Exemplos de entrada e saída 03: Entrada, Qual é a coisa mais bonita do mundo? Saída, O prompt não está adequado. Inclua palavras-chave relevantes.

## RETORNO DA CORREÇÃO DA IA PELO CHATGPT

```
# Entrada do usuário
prompt_usuario = input(" ")

# Função para avaliar se o prompt está adequado
def avaliar_prompt(prompt):
    # Verifica se o prompt contém palavras-chave relevantes
    palavras_chave = ["inteligência artificial", "sistemas de recomendação online", "exemplos de conversação", "explique conceitos", "dicas de tecnologia"]

    # Verifica se qualquer palavra-chave está no prompt
    for palavra in palavras_chave:
        if palavra.lower() in prompt.lower():
            return "O prompt está adequado."

    return "O prompt não está adequado. Inclua palavras-chave relevantes."

# Avaliar o prompt do usuário
feedback_usuario = avaliar_prompt(prompt_usuario)

# Exibir feedback
print(feedback_usuario)
```

## DESAFIO 3

### Descrição

Neste desafio, você será solicitado a fornecer uma breve descrição de boas práticas de refatoração de código em três áreas específicas: identificação clara de funções, separação da entrada de dados e uso de nomes descritivos para variáveis e funções. Cada descrição deve indicar como o código pode ser melhorado nesses aspectos. O objetivo é simularmos a ação de pesquisas em IAs Generativas de busca e pesquisa, dessa forma, cada entrada simula um bom prompt de pesquisa.

### Entrada:

A entrada será uma string de texto que representa um bom prompt de pesquisa para IAs Generativas. Após inserir sua descrição, o programa irá processá-la e fornecer um retorno sobre a prática de refatoração sugerida. Se a entrada corresponder a uma das três áreas especificadas, o programa irá sugerir a respectiva melhoria. Caso contrário, informará que a opção é inválida.

### Saída:

Após inserir sua descrição, o programa irá processá-la e fornecer um retorno sobre a prática de refatoração sugerida. Se a entrada corresponder a uma das três áreas especificadas, o programa irá sugerir a respectiva melhoria. Caso contrário, informe: **Opção inválida.**

### Exemplos

A tabela abaixo apresenta exemplos com alguns dados de entrada e suas respectivas saídas esperadas. Certifique-se de testar seu programa com esses exemplos e com outros casos possíveis.

Entrada	Saída
Dica de boas práticas de refatoração de código, nas funções.	Separe funções em unidades coesas e com responsabilidades únicas.
Dica de boas práticas de refatoração de código, nas entrada de dados.	Valide e normalize as entradas para evitar inconsistências.
Dica de boas práticas de refatoração de código, nomenclaturas.	Use nomes descritivos para variáveis e funções.

## CÓDIGO IMCOMPLETO - PROMPT DESAFIO 3

Preciso que analise o código abaixo em python e faça correção no que for preciso para ter as devidas entradas e saídas solicitadas. O objetivo é simular a ação de pesquisas em IA Generativas de busca e pesquisa, dessa forma, cada entrada simula um bom prompt de pesquisa.

```
def identificar_funcoes(texto):
    # Retorna uma mensagem sobre separar funções em unidades coesas e com responsabilidades únicas.
    return "Separe funções em unidades coesas e com responsabilidades únicas."

def entrada_de_dados(texto):
    # TODO: Retorne uma mensagem sobre validar e normalizar as entradas para evitar inconsistências.
    return "TODO"

def nomenclatura_significativa(texto):
    # TODO: Retorne uma mensagem sobre usar nomes descritivos para variáveis e funções.
    return "TODO"

def processar_entrada(texto):
    # Dicionário mapeando textos para funções
    opcoes = {
        "Dica de boas práticas de refatoração de código, nas funções.": identificar_funcoes,
        "Dica de boas práticas de refatoração de código, nas entrada de dados.": entrada_de_dados,
        "Dica de boas práticas de refatoração de código, nomenclaturas.": nomenclatura_significativa
    }

    # Verifica se o texto está presente nas opções
    if texto in opcoes:
        # Chama a função correspondente ao texto e retorna o resultado
        return opcoes[texto](texto)
    else:
        # TODO: Retorne uma mensagem de opção inválida se o texto não estiver nas opções
        return ""

def desafio():
    # Solicita uma entrada do usuário
    entrada = input()
    # Processa a entrada e obtém a saída
    saida = processar_entrada(entrada)
    # Exibe a saída
    print(saida)

desafio()
```



## {REGRAS}

> Entrada, A entrada será uma string de texto que representa um bom prompt de pesquisa para IAs Generativas. Após inserir sua descrição, o programa irá processá-la e fornecer um retorno sobre a prática de refatoração sugerida. Se a entrada corresponder a uma das três áreas especificadas, o programa irá sugerir a respectiva melhoria. Caso contrário, informará que a opção é inválida.

> Saída, após inserir sua descrição, o programa irá processá-la e fornecer um retorno sobre a prática de refatoração sugerida. Se a entrada corresponder a uma das três áreas especificadas, o programa irá sugerir a respectiva melhoria. Caso contrário, informe: Opção inválida.

> Exemplos de entrada e saída 01: Entrada, PDica de boas práticas de refatoração de código, nas funções. Saída, separe funções em unidades coesas e com responsabilidades únicas.

> Exemplos de entrada e saída 02: Entrada, Dica de boas práticas de refatoração de código, na entrada de dados. Saída, Valide e normalize as entradas para evitar inconsistências.

> Exemplos de entrada e saída 03: Dica de boas práticas de refatoração de código, nomenclaturas. Saída, use nomes descritivos para variáveis e funções.

## RETORNO DA CORREÇÃO DA IA PELO CHATGPT

```
def identificar_funcoes(texto):
```

```
    # Retorna uma mensagem sobre separar funções em unidades coesas e com responsabilidades únicas.
```

```
    return "Separe funções em unidades coesas e com responsabilidades únicas."
```

```
def entrada_de_dados(texto):
```

```
    # Retorna uma mensagem sobre validar e normalizar as entradas para evitar inconsistências.
```

```
    return "Valide e normalize as entradas para evitar inconsistências."
```

```
def nomenclatura_significativa(texto):
```

```
    # Retorna uma mensagem sobre usar nomes descritivos para variáveis e funções.
```

```
    return "Use nomes descritivos para variáveis e funções."
```

```
def processar_entrada(texto):
```

```
    # Dicionário mapeando textos para funções
```

```
    opcoes = {
```

```
        "Dica de boas práticas de refatoração de código, nas funções.": identificar_funcoes,
```

```
        "Dica de boas práticas de refatoração de código, na entrada de dados.": entrada_de_dados,
```

```
        "Dica de boas práticas de refatoração de código, nomenclaturas.":
```

```
        nomenclatura_significativa
```

```
    }
```

```
    # Verifica se o texto está presente nas opções
```

```
    if texto in opcoes:
```

```
        # Chama a função correspondente ao texto e retorna o resultado
```

```
        return opcoes[texto](texto)
```

```
    else:
```

```
        # Retorna uma mensagem de opção inválida se o texto não estiver nas opções
```

```
        return "Opção inválida."
```

```
def desafio():
```

```
    # Solicita uma entrada do usuário
```

```
    entrada = input()
```

```
    # Processa a entrada e obtém a saída
```

```
    saida = processar_entrada(entrada)
```

```
    # Exibe a saída
```

```
    print(saida)
```

```
desafio()
```

## DESAFIO 4

### Descrição

Neste desafio, você deve criar um programa que simule o auxílio de vendas de um site de catálogos de cogumelos utilizando inteligência artificial. O intuito é oferecer aos clientes sugestões de cogumelos que estão em promoção. Dessa forma, **o programa deve permitir que o usuário informe o nome de um cogumelo desejado e, com base nessa informação, deve sugerir até dois cogumelos adicionais da lista, cujos valores sejam iguais ou menores que o do cogumelo selecionado pelo cliente.** No caso de não houver sugestões disponíveis, ou seja, se o cogumelo escolhido for o mais caro, o programa deve exibir uma mensagem indicando que não há sugestões.

A baixo apresentamos a lista de cogumelos oferecidos pela loja com todos os seus valores. **Considere que essa lista já está ordenada por prioridade, ou seja, você deve oferecer como alternativas nessa ordem:**

Cogumelo	Valor
Shitake	10
Portobello	8
Shimeji	6
Champignon	12
Funghi	16
Porcini	16

## CÓDIGO IMCOMPLETO - PROMPT DESAFIO 4

Preciso que analise o código abaixo em python e faça correção no que for preciso para ter as devidas entradas e saídas solicitadas. O programa deve permitir que o usuário informe o nome de um cogumelo desejado e, com base nessa informação, deve sugerir até dois cogumelos adicionais da lista, cujos valores sejam iguais ou menores que o do cogumelo selecionado pelo cliente. No caso de não houver sugestões disponíveis, ou seja, se o cogumelo escolhido for o mais caro, o programa deve exibir uma mensagem indicando que não há sugestões. O código deve usar as regras impostas com os exemplos de entrada e saída.

```
# Entrada do usuário
```

```
cogumelo_desejado = input()
```

```
# Função para sugerir cogumelos com preços mais baixos com base em um cogumelo desejado.
```

```
def sugerir_cogumelos(cogumelo_desejado):
```

```
    #TODO: Defina um dicionário onde as chaves são os tipos de cogumelos e os valores são os  
    preços correspondentes
```

```
    catalogo = {  
        "Shitake": 10,  
    }
```

```
    # Verifica se o cogumelo desejado estão no catálogo
```

```
    if cogumelo_desejado in catalogo:
```

```
        # TODO: Se estiver no catálogo, armazene o preço do cogumelo desejado e crie uma lista  
        vazia para as sugestões
```

```
        valor_desejado =
```

```
        sugestoos =
```

```
    # Procura por cogumelos mais baratos no catálogo
```

```
    for cogumelo, valor in catalogo.items():
```

```
        if valor <= valor_desejado and cogumelo != cogumelo_desejado:
```

```
            sugestoos.append((cogumelo, valor)) # Adiciona uma tupla (cogumelo, valor)
```

```
            if len(sugestoos) == 2:
```

```
                break
```

```
    if not sugestoos:
```

```
        # TODO: Se não houver sugestões, exiba a mensagem indicada no enunciado
```

```
        print("")
```

```
    else:
```

```
        for sugestao, valor_sugestao in sugestoos:
```

```
            print(f"{sugestao} - Valor: {valor_sugestao}")
```

```
    else:
```

```
        # TODO: Se o cogumelo desejado não estiver no catálogo, exiba uma mensagem de erro  
        indicada no enunciado
```

```
        print("")
```

```
# Chamada da função para sugerir cogumelos
```

```
sugerir_cogumelos(cogumelo_desejado)
```

#### {REGRAS}

- > Considere que essa lista já está ordenada por prioridade, ou seja, você deve oferecer como alternativas nessa ordem: Shitake - Valor: 10, Portobello - Valor: 8, Shimeji - Valor: 6, Champignon - Valor: 12, Funghi - Valor: 16 e Porcini - Valor: 16
- > Entrada, A entrada será uma string representando o nome do cogumelo desejado pelo usuário.
- > Saída, Uma lista com no máximo 2 sugestões de cogumelos mais baratos do que o enviado como entrada. Lembrando que a sugestão das alternativas deve considerar a lista de cogumelos na ordem descrita na tabela supracitada neste desafio.
- > Cada saída deve estar em linhas diferentes
- > Exemplos de entrada e saída 01: Entrada, Shitake. Saída, Portobello - Valor: 8  
Shimeji - Valor: 6
- > Exemplos de entrada e saída 02: Entrada, Champignon. Saída, Shitake - Valor: 10  
Portobello - Valor: 8.
- > Exemplos de entrada e saída 03: Entrada, Portobello. Saída, Shimeji - Valor: 6.
- > Exemplos de entrada e saída 04: Entrada, Shimeji. Saída, Desculpe, não há sugestões disponíveis.

## RETORNO DA CORREÇÃO DA IA PELO CHATGPT

```
# Entrada do usuário
cogumelo_desejado = input()

# Função para sugerir cogumelos com preços mais baixos com base em um cogumelo desejado.
def sugerir_cogumelos(cogumelo_desejado):
    # Definindo um dicionário onde as chaves são os tipos de cogumelos e os valores são os
    # preços correspondentes
    catalogo = {
        "Shitake": 10,
        "Portobello": 8,
        "Shimeji": 6,
        "Champignon": 12,
        "Funghi": 16,
        "Porcini": 16
    }

    # Verifica se o cogumelo desejado está no catálogo
    if cogumelo_desejado in catalogo:
        # Armazena o preço do cogumelo desejado e cria uma lista vazia para as sugestões
        valor_desejado = catalogo[cogumelo_desejado]
        sugestoes = []

        # Lista de cogumelos em ordem de prioridade
        lista_prioridade = ["Shitake", "Portobello", "Shimeji", "Champignon", "Funghi", "Porcini"]

        # Procura por cogumelos mais baratos no catálogo, seguindo a ordem de prioridade
        for cogumelo in lista_prioridade:
            if catalogo[cogumelo] < valor_desejado:
                sugestoes.append((cogumelo, catalogo[cogumelo]))
                if len(sugestoes) == 2:
                    break

        if not sugestoes:
            # Se não houver sugestões, exibe a mensagem indicada no enunciado
            print("Desculpe, não há sugestões disponíveis.")
        else:
            for sugestao, valor_sugestao in sugestoes:
                print(f"{sugestao} - Valor: {valor_sugestao}")
    else:
        # Se o cogumelo desejado não estiver no catálogo, exibe uma mensagem de erro indicada
        # no enunciado
        print("Cogumelo não encontrado no catálogo.")

# Chamada da função para sugerir cogumelos
sugerir_cogumelos(cogumelo_desejado)
```

## DESAFIO 5

### Descrição

Você faz parte de uma equipe que está desenvolvendo modelos de Machine Learning para identificar a probabilidade de inadimplência em empréstimos concedidos por uma instituição financeira. Após treinar os modelos, sua tarefa é avaliar seu desempenho usando algumas métricas de avaliação. Nesse contexto, o desafio é criar um algoritmo que receba  $n$  matrizes de confusão e retorne o índice, precisão e acurácia da matriz que apresenta o melhor desempenho com base no cálculo dessas métricas. Lembrando que:

- **Acurácia é calculada pela fórmula:**  $(VP + VN) / (VP + FP + FN + VN)$
- **Precisão é calculada pela fórmula:**  $VP / (VP + FP)$

Onde:

- **VP (Verdadeiro Positivo):** Casos em que o modelo previu corretamente a classe positiva.
- **FP (Falso Positivo ou Erro Tipo I):** Casos em que o modelo previu incorretamente a classe positiva.
- **FN (Falso Negativo ou Erro Tipo II):** Casos em que o modelo previu incorretamente a classe negativa.
- **VN (Verdadeiro Negativo):** Casos em que o modelo previu corretamente a classe negativa.

Entrada:

A entrada consiste em uma string composta por:  $n$ , representando o número de matrizes de confusão, seguido dos valores que compõem as  $n$  matrizes.

Cada matriz consiste em quatro valores, onde os dois primeiros representam a primeira linha da matriz, composta por verdadeiros positivos (VP) e falsos positivos (FP); os dois últimos valores representam a segunda linha, que é composta por falsos negativos (FN) e verdadeiros negativos (VN). As duas linhas e os valores que as compõem estão separados por vírgulas.

Saída:

O resultado esperado inclui o valor do índice, acurácia e precisão (arredondada para duas casas decimais) da matriz com melhor desempenho com base no cálculo dessas métricas.

### Exemplos

A tabela abaixo apresenta exemplos com alguns dados de entrada e suas respectivas saídas esperadas. Certifique-se de testar seu programa com esses exemplos e com outros casos possíveis.

Entrada	Saída
3 50,10,5,85 20,5,8,67 30,12,4,88	Índice: 1 Acurácia: 0.9 Precisão: 0.83
4 70,15,8,78 60,20,10,80 45,5,3,92 80,7,15,98	Índice: 3 Acurácia: 0.94 Precisão: 0.9
2 100,0,0,50 80,10,2,98	Índice: 1 Acurácia: 1.0 Precisão: 1.0

## CÓDIGO INCOMPLETO - PROMPT DESAFIO 5

Preciso que analise o código abaixo em python e faça correção no que for preciso para ter as devidas entradas e saídas solicitadas. O desafio é criar um algoritmo que receba "n" matrizes de confusão e retorne o índice, precisão e acurácia da matriz que apresenta o melhor desempenho com base no cálculo dessas métricas. O código usando as regras impostas com os exemplos de entrada e saída.

```
n = int(input())
matrices = []

for n in range(0, n):
    matrix = input()
    matrices.append(matrix.split(','))

# TODO: Create a function to calculate accuracy and precision metrics

# TODO: Create a function to find the matrix index with the best combined accuracy and precision
def best_performance(matrices):
    best_index = 0
    best_accuracy = 0
    best_precision = 0
    # TODO: Define Loop through each matrix to calculate metrics
    for index, matrix in enumerate(matrices):
        # TODO: Define tp, fp fn and tn

        # TODO: Calculate accuracy and precision

        # TODO: Update best metrics if found

    return best_index, best_accuracy, best_precision

# Print the results
```

### {REGRAS}

> Considere que Acurácia é calculada pela fórmula:  $(VP + VN) / (VP + FP + FN + VN)$

> Considere que Precisão é calculada pela fórmula:  $VP / (VP + FP)$

> VP (Verdadeiro Positivo): Casos em que o modelo previu corretamente a classe positiva.

> FP (Falso Positivo ou Erro Tipo I): Casos em que o modelo previu incorretamente a classe positiva.

> FN (Falso Negativo ou Erro Tipo II): Casos em que o modelo previu incorretamente a classe

> VN (Verdadeiro Negativo): Casos em que o modelo previu corretamente a classe negativa.

> Entrada, a entrada consiste em uma string composta por: "n", representando o número de matrizes de confusão, seguido dos valores que compõem as "n" matrizes.

Cada matriz consiste em quatro valores, onde os dois primeiros representam a primeira linha da matriz, composta por verdadeiros positivos (VP) e falsos positivos (FP); os dois últimos valores representam a segunda linha, que é composta por falsos negativos (FN) e verdadeiros negativos (VN). As duas linhas e os valores que as compõem estão separados por vírgulas.

> Saída, o resultado esperado inclui o valor do índice, acurácia e precisão (arredondada para duas casas decimais) da matriz com melhor desempenho com base no cálculo dessas métricas.

> Exemplos de entrada e saída 01: Entrada,

3

50,10,5,85

20,5,8,67

30,12,4,88.

Saída,

Índice: 1

Acurácia: 0.9

Precisão: 0.83

> Exemplos de entrada e saída 02: Entrada,

4

70,15,8,78

60,20,10,80

45,5,3,92

80,7,15,98

Saída,

Índice: 3

Acurácia: 0.94

Precisão: 0.9

> Exemplos de entrada e saída 03: Entrada,

2

100,0,0,50

80,10,2,98.

Saída,

Índice: 1

Acurácia: 1.0

Precisão: 1.0



## RETORNO DA CORREÇÃO DA IA PELO CHATGPT

```
def calcular_metricas(matriz):
    # Extrair VP, FP, FN, VN da matriz
    vp, fp, fn, vn = map(int, matriz)

    # Calcular acurácia e precisão
    acuracia = (vp + vn) / (vp + fp + fn + vn)
    precisao = vp / (vp + fp) if (vp + fp) != 0 else 0 # Prevenção de divisão por zero

    return acuracia, precisao

def melhor_desempenho(matrizes):
    melhor_indice = -1
    melhor_acuracia = -1
    melhor_precisao = -1

    for indice, matriz in enumerate(matrizes):
        acuracia, precisao = calcular_metricas(matriz)

        # Verificar se essa matriz tem o melhor desempenho até agora
        if (acuracia > melhor_acuracia) or (acuracia == melhor_acuracia and precisao >
melhor_precisao):
            melhor_indice = indice
            melhor_acuracia = acuracia
            melhor_precisao = precisao

    return melhor_indice, round(melhor_acuracia, 2), round(melhor_precisao, 2)

if __name__ == "__main__":
    # Leitura das entradas
    n = int(input())
    matrizes = []

    for _ in range(n):
        matriz = input()
        matrizes.append(matriz.split(','))

    # Encontrar a matriz com o melhor desempenho
    indice, acuracia, precisao = melhor_desempenho(matrizes)

    # Imprimir o resultado
    print(f"Índice: {indice + 1}")
    print(f"Acurácia: {acuracia}")
    print(f"Precisão: {precisao}")
```