**NOME FACULDADE Y**

SEU NOME

**Atividade:** Rankeando Filmes com um sistema Fuzzy

Cidade XXXXX/UF XXXXX

2025

SEU NOME

**Atividade:** Rankeando Filmes com um sistema Fuzzy

Trabalho de ……

Professor: XXXXXX

Cidade XXXXX/UF XXXXX

2025

Dedico este trabalho meus pais que me ajudaram e me guiaram

**AGRADECIMENTOS**

Ao Profº XXXXXXX por XXXXXXX

A sabedoria suprema é ter sonhos bastante grandes para não se perderem de vista enquanto

os perseguimos.

(FAULKNER, William, 1929)

**INTRODUÇÃO**

Quando pensamos em tomar decisões no dia a dia, como avaliar se um filme é "bom" ou "ruim", nem sempre conseguimos usar regras rígidas. Por exemplo, imagine que um filme tem uma média de votos de 6.9 numa escala de 0 a 10. Ele é automaticamente "ruim" porque não chegou a 7? Ou o que acontece se ele tem muitos votos e uma popularidade enorme, mas a média é apenas razoável? A lógica tradicional, baseada em sim ou não (0 ou 1), não consegue capturar essas nuances de forma natural. É aí que entra o sistema fuzzy, uma ferramenta poderosa para lidar com incertezas e decisões mais próximas do pensamento humano.

Um sistema fuzzy é baseado na lógica fuzzy (ou lógica difusa), uma ideia proposta pelo matemático Lotfi Zadeh na década de 1960. Diferente da lógica binária, que só aceita valores absolutos (verdadeiro ou falso), a lógica fuzzy trabalha com graus de pertinência. Isso significa que algo pode pertencer parcialmente a um conjunto. Por exemplo, um filme pode ser considerado 70% "bom" e 30% "regular" ao mesmo tempo, dependendo dos critérios que usamos para avaliá-lo.

Mas como isso funciona na prática? Um sistema fuzzy segue três etapas principais:

* **Fuzzificação:** Aqui, pegamos valores numéricos exatos (como a média de votos de um filme) e os transformamos em graus de pertinência a conjuntos fuzzy. Esses conjuntos têm nomes como "baixo", "médio" ou "alto". Por exemplo, uma média de 6 pode ser 40% "média" e 60% "baixa", dependendo de como definimos os limites.
* **Inferência:** Nesta etapa, usamos regras fuzzy para combinar as entradas e decidir o resultado. Essas regras são escritas em linguagem natural, como "se a popularidade é alta e a média de votos é alta, então o filme é muito bom". O sistema avalia todas as regras para chegar a uma conclusão fuzzy.
* **Defuzzificação:** Por fim, transformamos o resultado fuzzy (que ainda é vago) em um número concreto. Por exemplo, o sistema pode calcular um "score" final entre 0 e 1 para indicar o quão "bom" é o filme.

**SUMÁRIO**

[**1. OBJETIVOS 8**](#_heading=h.gjdgxs)

[**2. METODOLOGIA 8**](#_heading=h.tyjcwt)

[2.1. Base de dados e ferramentas 8](#_heading=h.skx2gmpx8ndw)

[2.2. Pré-processamento de dados 9](#_heading=h.xs7umqsgxp4)

[2.3. Processamento de dados 10](#_heading=h.n2owqapro4d1)

[**3. RESULTADOS 15**](#_heading=h.1upzp21laz2a)

[**4. REPOSITÓRIO 15**](#_heading=h.vsb0vt7oved3)

[**5. VÍDEO TUTORIAL 15**](#_heading=h.fd0oglereal3)

[**REFERÊNCIAS 16**](#_heading=h.39gy2b3w3qt6)

[**GLOSSÁRIO 17**](#_heading=h.2s8eyo1)

# 1. OBJETIVOS

Criar um sistema de regras fuzzy que crie um score para so filmes. O score tem que ser aderente ao gosto dos filmes da equipe que está criando o código. Os filmes mais bem avaliados serão melhores rankeados.

Devido a grande quantidade de gostos, consideramos a avaliação baseada em critérios mais objetivos, lançando mão da famosa “sabedoria popular”, consideramos as colunas “popularity”, “vote\_average”, “vote\_count” e “revenue”. Além disso, consideramos também a coluna “title” para fins de visualização

# 2. METODOLOGIA

## 2.1. Base de dados e ferramentas

**Fonte de Dados:** Arquivo CSV

**Ferramentas:**

* **Python:** Linguagem principal para processamento e modelagem.
* **Bibliotecas:** Pandas (manipulação de dados), Scikit-fuzzy (lógica fuzzy), numpy (cálculos númericos) e o ctrl para criar os sistemas de controle fuzzy (entradas, saídas e regras)
* Testes realizados utilizando um notebook em um ambiente serveless da Google Colab.

## 2.2. Pré-processamento de dados

**python**

**df = df[['title', 'popularity', 'vote\_average', 'vote\_count', 'revenue']].dropna()**

****

* **Substituição de Valores:**
  + **Remoção de valores nulos:** O método .dropna() elimina qualquer linha que tenha valores faltantes (NaN) em alguma dessas colunas. Isso garante que os dados estejam completos para o processamento.
  + **Seleção de colunas:** Pegamos apenas as colunas que nos interessam: 'title' (título do filme), 'popularity' (popularidade), 'vote\_average' (média de votos), 'vote\_count' (quantidade de votos) e 'revenue' (receita)
* **Normalização dos Dados:**

**python**

****

**df\_normalized = df.copy()**

**cols\_to\_normalize = ['popularity', 'vote\_average', 'vote\_count', 'revenue']**

**df\_normalized[cols\_to\_normalize] = (df[cols\_to\_normalize] - df[cols\_to\_normalize].min()) / (df[cols\_to\_normalize].max() - df[cols\_to\_normalize].min())**

****

Os valores de popularidade, média de votos, quantidade de votos e receita podem estar em escalas muito diferentes (ex.: receita em milhões, média de votos entre 0 e 10). Para usar lógica fuzzy, precisamos que todos estejam na mesma escala, entre 0 e 1

* + **df.copy():** Criamos uma cópia do DataFrame original para não alterá-lo diretamente.
  + **Fórmula de normalização:** 
    - Para cada valor em uma coluna, subtraímos o menor valor dessa coluna (.min()) e dividimos pela diferença entre o maior (.max()) e o menor valor.

Exemplo: Se a popularidade varia de 0 a 100, um valor de 50 se torna (50 - 0) / (100 - 0) = 0.5.

O resultado é armazenado em df\_normalized, mas a coluna 'title' não é normalizada, pois é texto.

## 2.3. Processamento de dados

* **Definindo Variáveis Fuzzy:**

**python**

****

**popularity = ctrl.Antecedent(np.arange(0, 1.1, 0.1), 'popularity')**

**vote\_avg = ctrl.Antecedent(np.arange(0, 1.1, 0.1), 'vote\_avg')**

**vote\_count = ctrl.Antecedent(np.arange(0, 1.1, 0.1), 'vote\_count')**

**revenue = ctrl.Antecedent(np.arange(0, 1.1, 0.1), 'revenue')**

**score = ctrl.Consequent(np.arange(0, 1.1, 0.1), 'score')**

****

Na lógica fuzzy, variáveis têm "graus de pertinência" a certos conjuntos (ex.: "baixo", "médio", "alto"). Aqui definimos as entradas (antecedentes) e a saída (consequente).

1. **ctrl.Antecedent:** Define as variáveis de entrada (popularity, vote\_avg, vote\_count, revenue). Cada uma usa np.arange(0, 1.1, 0.1), que cria um intervalo de 0 a 1 com passos de 0.1 (0, 0.1, 0.2, ..., 1.0).
2. **ctrl.Consequent:** Define a variável de saída, chamada score, que será o resultado do sistema fuzzy (o "score" do filme).

Todas estão no intervalo [0, 1] porque os dados foram normalizados.

* **Definindo Funções de Pertinência:**

**python**

****

**for var in [popularity, vote\_avg, vote\_count, revenue]:**

**var.automf(3, names=['low', 'medium', 'high'])**

**score.automf(5, names=['very\_low', 'low', 'medium', 'high', 'very\_high'])**

****

Elas dizem o quanto um valor pertence a um conjunto fuzzy (ex.: "popularidade alta").

1. **var.automf(3, names=['low', 'medium', 'high']):** Para cada variável de entrada, criamos automaticamente 3 conjuntos fuzzy: "baixo", "médio" e "alto". O método automf divide o intervalo [0, 1] em três partes iguais e associa funções triangulares a cada uma.
2. **score.automf(5, ...):** Para a saída score, criamos 5 conjuntos: "muito baixo", "baixo", "médio", "alto" e "muito alto". Isso dá mais granularidade ao resultado.

* **Criando Regras Fuzzy:**

**python**

****

**rules = [**

**ctrl.Rule(popularity['high'] & vote\_avg['high'], score['very\_high']),**

**ctrl.Rule(vote\_count['high'] & revenue['high'], score['very\_high']),**

**ctrl.Rule(popularity['medium'] & vote\_avg['medium'], score['medium']),**

**ctrl.Rule(popularity['low'] | vote\_avg['low'], score['low']),**

**ctrl.Rule(revenue['low'] & vote\_count['low'], score['very\_low'])**

**]**

****

São as "regras de decisão" do sistema, baseadas em lógica humana. Elas conectam as entradas à saída.

1. **popularity['high'] & vote\_avg['high']:** Se a popularidade é alta E a média de votos é alta, então o score é "muito alto".
2. **popularity['low'] | vote\_avg['low']:** Se a popularidade é baixa OU a média de votos é baixa, o score é "baixo".

Usamos & para "E" e | para "OU".

Aqui, temos 5 regras, mas em sistemas reais pode haver dezenas ou centenas.

* **Configurando o Sistema de Controle:**

**python**

****

**scoring\_system = ctrl.ControlSystem(rules)**

**scoring = ctrl.ControlSystemSimulation(scoring\_system)**

****

1. **ctrl.ControlSystem(rules):** Cria o sistema fuzzy com base nas regras definidas.
2. c**trl.ControlSystemSimulation:** Prepara uma simulação para calcular os scores com base nos dados de entrada.

* **Calculando os Scores Fuzzy:**

**python**

****

**scores = []**

**for idx, row in df\_normalized.iterrows():**

**scoring.input['popularity'] = row['popularity']**

**scoring.input['vote\_avg'] = row['vote\_average']**

**scoring.input['vote\_count'] = row['vote\_count']**

**scoring.input['vote\_revenue'] = row['vote\_revenue']**

**scoring.compute()**

**scores.append(scoring.output['score'])**

****

1. **Loop pelos dados:** Para cada linha do df\_normalized (cada filme):
   * Definimos os valores das entradas (popularity, vote\_avg, etc.) com os valores normalizados da linha.
   * **scoring.compute():** Calcula o score fuzzy com base nas regras e funções de pertinência.
   * O resultado (scoring.output['score']) é adicionado à lista scores.

* **Adicionando os Scores ao DataFrame:**

**python**

****

**df['fuzzy\_score'] = scores**

****

Adicionamos uma nova coluna chamada 'fuzzy\_score' ao DataFrame original (df), contendo os scores calculados

* **Ordenando os Filmes:**

**python**

****

**df\_ranked = df.sort\_values(by='fuzzy\_score', ascending=False)**

****

Ordenamos o DataFrame pelo 'fuzzy\_score' em ordem decrescente (ascending=False), para que os filmes com maiores scores apareçam primeiro.

# 3. RESULTADOS

* **Exibindo os Resultados:**

**python**

****

**print(df\_ranked[['title', 'fuzzy\_score']].head(10))**

****

Mostramos os 10 primeiros filmes (.head(10)) com as colunas 'title' e 'fuzzy\_score', ou seja, os filmes com os melhores scores fuzzy.

# 4. REPOSITÓRIO

Link para o respositório do Github:

# 5. VÍDEO TUTORIAL

Link para a vídeo explicação:

# REFERÊNCIAS

BRASIL. Associação Brasileira de Normas Técnicas. NBR 6023:2018: *Informação e documentação - Referências - Elaboração*. Rio de Janeiro, 2018.

GERON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2. ed. Sebastopol: O'Reilly Media, 2019.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. ed. New York: Springer, 2009.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research,* v. 12, p. 2825-2830, 2011. Disponível em: https://jmlr.org/papers/v12/pedregosa11a.html. Acesso em: 06 mar. 2025.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research,* v. 16, p. 321-357, 2002. Disponível em: https://www.jair.org/index.php/jair/article/view/10302. Acesso em: 06 mar. 2025.

# GLOSSÁRIO

**Acurácia:** Medida que representa a proporção de previsões corretas em relação ao total de previsões realizadas por um modelo de classificação. É expressa como um valor entre 0 e 1 ou em percentual.

**Classificação:** Técnica de machine learning que consiste em atribuir uma categoria ou rótulo (como A, B, C) a um conjunto de dados com base em suas características, utilizando algoritmos como RandomForest.

**Feature Engineering:** Processo de criação de novas variáveis (features) ou transformação de dados existentes para melhorar o desempenho de um modelo de machine learning, como a criação de potencia\_por\_peso.

**Machine Learning:** Campo da inteligência artificial que permite que sistemas aprendam a partir de dados sem serem explicitamente programados, abrangendo técnicas como regressão e classificação.