Pontifícia Universidade Católica
do Rio de Janeiro

Departamento de Engenharia Elétrica

ELE2399

LÓGICA FUZZY

Trabalho 01: Previsão de Séries com Sistema de Inferência Fuzzy

Aluno(a) Paloma Fernanda Loureiro Sette - 1621595
Professores Marley Vellasco e Thiago Carvalho

Rio de Janeiro, 27 de Novembro de 2025

Sumário

Lista de Figuras

Introdução

Este relatório tem como objetivo descrever o desenvolvimento do trabalho 01 da disciplina de Lógica Fuzzy (ELE2399). O trabalho consiste na criação de um Sistema de Inferência Fuzzy (SIF) para previsão de séries temporais, utilizando a metodologia de Wang & Mendel para extração de a partir dos dados fornecidos. Para obter o melhor desempenho possível, foram realizadas diversas alterações nos parâmetros do SIF, como o número de conjuntos fuzzy, o tamanho da janela de observação, as operações de interseção dos antecedentes, as operações de implicação e os métodos de defuzzificação. A série temporal desginada contém os dados financeiros das ações da Apple (Apple Stock (AAPL).

1 Parte 1: Sistema de Inferência Fuzzy (SIF)

Deseja-se realizar uma previsão de um passo a frente, por meio de um sistema de inferência fuzzy (SIF). Para tal, é necessário efetuar uma extração de regras a partir dos dados fornecidos utilizando o método de Wang & Mendel. Construiremos esse SIF utilizando Python. Em seguida, buscaremos pelo melhor desempenho possível, realizando alterações no número de conjuntos fuzzy, no tamanho da janela e nas operações de interseção dos antecedentes, implicação e defuzzificação.

1.1 Construção em Python

O objetivo principal foi realizar uma previsão de um passo à frente utilizando um Sistema de Inferência Fuzzy (SIF). A metodologia de Wang & Mendel foi utilizada para extração de regras fuzzy a partir dos dados fornecidos. O procedimento foi dividido nas seguintes etapas:

1.1.1 Funções Auxiliares

A fim de otimizar os testes, foram criadas funções auxiliares para mais praticidade e organização.

- **Carregamento dos dados e criação de lags**

Foram definidas funções auxiliares para tal. A função `charge_data` é responsável pelo carregamento dos dados de um arquivo CSV e pela seleção da coluna de interesse, que será renomeada para `target`. A função retorna um DataFrame contendo apenas a coluna selecionada.

```
1     def charge_data(data: Path , column_to_target: str) -> pd.
         DataFrame:
2     """Auxiliary function for loading and initial preparation of
          data.
3         Reads the CSV file, filters the column of interest and
             renames it to the pattern
4         expected by the rule generation algorithm.
5
6         Args:
7             data (Path): Full path to the input CSV file.
8             column_to_target (str): Exact name of the column in
                 the CSV that contains the time series
9         Returns:
10            pd.DataFrame: DataFrame containing a single column
                 called 'target', ready
11            for the creation of lags.
12    """
13    df = pd.read_csv(data)
14    df = df[[column_to_target]]
15    df.columns = ['target']
16    return df
```

```python
def lags_create(window_size: int, data: Path, column_to_target: str) -> pd.DataFrame:
    """Function to create lags with a variable window size.

    Args:
        window_size (int): Number of lags to create.
        data (Path): Full path to the input CSV file.
        column_to_target (str): Exact name of the column in the CSV that contains the time series

    Returns:
        pd.DataFrame: DataFrame containing the target column and the created lags.
    """
    df = charge_data(data, column_to_target)
    for i in range(1, window_size+1):
        df[f'lag_{i}'] = df['target'].shift(i)
    df.dropna(inplace=True)
    return df.iloc[:, ::-1]
```

A função `lags_create` cria lags com um tamanho variável de janela. Primeiro, ela carrega os dados utilizando a função `charge_data`. Em seguida, para cada tamanho de janela especificado, cria uma nova coluna no DataFrame para cada lag. O DataFrame resultante é retornado com as colunas na ordem inversa.

- **Criação de variáveis fuzzy**

A função `create_fuzzy_variables` é responsável por criar variáveis fuzzy com um número variável de conjuntos fuzzy e aplicar o método de defuzzificação especificado. Ela calcula os valores mínimos e máximos do DataFrame, cria uma lista de variáveis fuzzy (antecedentes e consequentes) e define automaticamente os conjuntos fuzzy para cada variável.

```python
    def create_fuzzy_variables(df: pd.DataFrame, num_fuzzy_sets: int, defuzzyfication_method: str) -> list:
    """auxiliary function for creating fuzzy variables

    Args:
        df (pd.DataFrame): _input dataframe_
        num_fuzzy_sets (int): _number of fuzzy sets to create for each variable_
        defuzzyfication_method (str): _method of defuzzification to use for the output variable_

    Returns:
```

```
        list: _list of fuzzy variables created_
    """
    min_value = df.min().min()
    max_value = df.max().max()
    variable_list = [
        ctrl.Antecedent(np.arange(min_value - 1, max_value + 1,
           1), name)
        if name != 'target' else ctrl.Consequent(np.arange(
           min_value - 1, max_value + 1, 1), name)
        for name in df.columns
    ]
    for variable in variable_list:
        variable.automf(num_fuzzy_sets)
    for var in variable_list:
        if isinstance(var, ctrl.Consequent):
            var.defuzzify_method = defuzzyfication_method
    return variable_list
```

- **Operações de interseção**

As funções `operacao_intersecao_min`, `operacao_intersecao_prod` e `operacao_intersecao_avg` são responsáveis por calcular a interseção entre os antecedentes de diferentes maneiras: mínimo, produto e média, respectivamente.

```
   def operacao_intersecao_min(valores):
    return np.min(valores)


def operacao_intersecao_prod(valores):
    return np.prod(valores)


def operacao_intersecao_avg(valores):
    return np.mean(valores)
```

- **Implicações**

As funções `implicacao_min`, `implicacao_prod` e `implicacao_maxmin` são responsáveis por calcular a implicação dos antecedentes nos consequentes de diferentes maneiras: mínimo, produto e soma truncada, respectivamente.

```python
def implicacao_min(pertinencia_antecedente, valor_consequente):
    return min(pertinencia_antecedente, valor_consequente)


def implicacao_prod(pertinencia_antecedente, valor_consequente):
    return pertinencia_antecedente * valor_consequente


def implicacao_maxmin(pertinencia_antecedente, valor_consequente
    ):
    return max(0, pertinencia_antecedente + valor_consequente -
        1)
```

- **Métodos de defuzzificação**

As funções `defuzzificacao_centroid`, `defuzzificacao_mean_of_maxima` e `defuzzificacao_weighted_` são responsáveis por realizar a defuzzificação utilizando diferentes métodos: centroide, média dos máximos e média ponderada, respectivamente.

```python
def defuzzificacao_centroid(pertinencias):
    return fuzz.defuzz(np.array(list(pertinencias.keys())), np.
        array(list(pertinencias.values())), 'centroid')


def defuzzificacao_mean_of_maxima(pertinencias):
    max_pertinencia = max(pertinencias.values())
    max_conjuntos = [conjunto for conjunto, pertinencia in
        pertinencias.items(
    ) if pertinencia == max_pertinencia]
    return np.mean(max_conjuntos)


def defuzzificacao_weighted_average(pertinencias):
    numerador = sum(conjunto * pertinencia for conjunto,
                    pertinencia in pertinencias.items())
    denominador = sum(pertinencias.values())
    if denominador == 0:
        return 0
    return numerador / denominador
```

- **Predição e avaliação do modelo**

A função `predict` utiliza o sistema de inferência fuzzy para realizar a previsão dos valores de saída com base nas entradas fornecidas. A função `avaliar_modelo` calcula o erro quadrático médio (MSE) entre os valores reais e os valores preditos, retornando o MSE e o DataFrame contendo as previsões.

```
    def predict(fuzzy_sim, df, metodo_defuzzificacao):

    df_copy = df.copy()
    df_copy.loc[:, 'predict'] = np.nan
    for i in range(len(df_copy)):
        l = df_copy.iloc[i, :].shape[0] - 2
        for k in range(l):
            fuzzy_sim.input[f'lag_{l - k}'] = df_copy.iloc[i, k]
        try:
            fuzzy_sim.compute()
            pertinencias = {term: fuzzy_sim.output[term]
                            for term in fuzzy_sim.output}
            out = metodo_defuzzificacao(pertinencias)
        except:
            out = df_copy.iloc[i, k]
        df_copy.iloc[i, l + 1] = out
    return df_copy


def avaliar_modelo(fuzzy_sim, df, metodo_defuzzificacao):

    predict_df = predict(fuzzy_sim, df, metodo_defuzzificacao)
    mse = mean_squared_error(predict_df['target'], predict_df['
       predict'])
    return mse, predict_df
```

1.1.2 Configuração e Otimização do Sistema

Para otimizar o mapeamento das funções auxiliares durante os testes, foram criados dicionários que associam as funções de interseção, implicação e defuzzificação aos seus respectivos nomes: Os dicionários criados são:

- `dic_operacoes_intersecao`: Define os métodos de interseção que podem ser utilizados no sistema fuzzy. As opções incluem Mínimo, Produto e Média.
- `dic_implicacoes`: Define os métodos de implicação. As opções incluem Mínimo, Soma Truncada e Produto.
- `dic_defuzzificacao`: Define os métodos de defuzzificação. As opções incluem Centro de Gravidade, Média dos Máximos e Média Ponderada.

```
    dic_operacoes_intersecao = {
    operacao_intersecao_min: "Minimo",
    operacao_intersecao_prod: "Produto",
    operacao_intersecao_avg: "Media"
}

dic_implicacoes = {
    implicacao_min: "Minimo",
    implicacao_maxmin: "Soma truncada",
    implicacao_prod: "Produto"
}

dic_defuzzificacao = {
    defuzzificacao_centroid: "Centro de Gravidade",
    defuzzificacao_mean_of_maxima: "Media dos M ximos",
    defuzzificacao_weighted_average: "Media Ponderada"
}
```

Parâmetros para Busca do Melhor Desempenho
Para encontrar a melhor configuração do sistema, os seguintes parâmetros foram definidos e alterados durante os testes:

- `window_size`: Tamanho da janela utilizada na criação dos lags. Exemplo: 7.
- `num_fuzzy_sets`: Número de conjuntos fuzzy. Exemplo: 5.
- `metodo_defuzzificacao`: Método de defuzzificação. Exemplo: `defuzzificacao_centroid`.
- `operacao_intersecao`: Método de interseção. Exemplo: `operacao_intersecao_avg`.
- `operacao_implicacao`: Método de implicação. Exemplo: `implicacao_maxmin`.

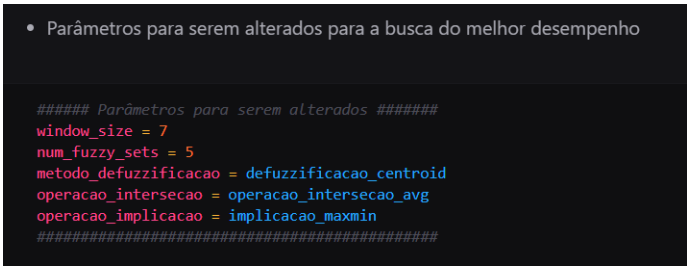

Figura 1: Parâmetros dinâmicos a serem alterados nos testes

1.1.3 Etapa 1: Preparação dos Dados

Após as configurações iniciais e definição das funções auxiliares, os dados de preços da ação foram carregados e pré-processados. Foram criadas variáveis de atraso (lags) para representar os preços anteriores, que servem como entradas para o sistema fuzzy.

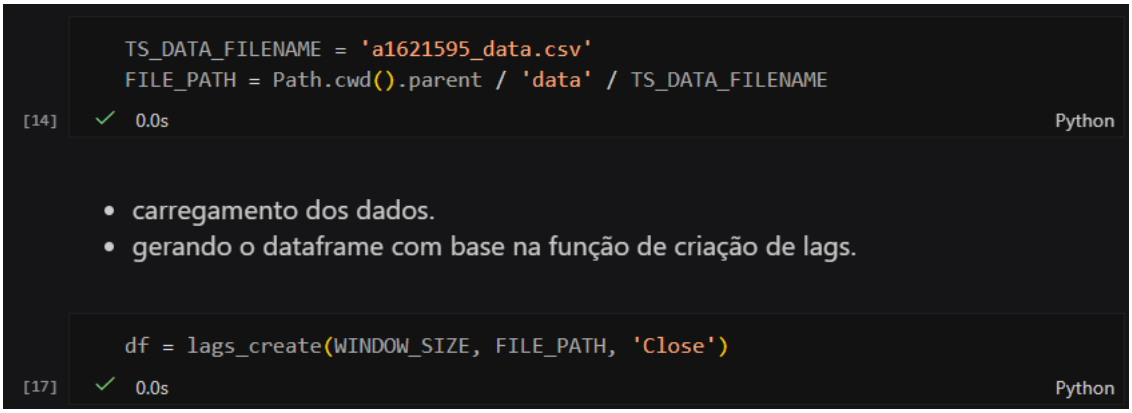

Figura 2: Preparação dos dados

1.1.4 Etapa 2: Definição dos Conjuntos Fuzzy

Em seguida, foram definidos os conjuntos fuzzy para cada variável de entrada. Experimentamos diferentes números de conjuntos fuzzy (e.g., 5, 7) para verificar seu impacto no desempenho do modelo.

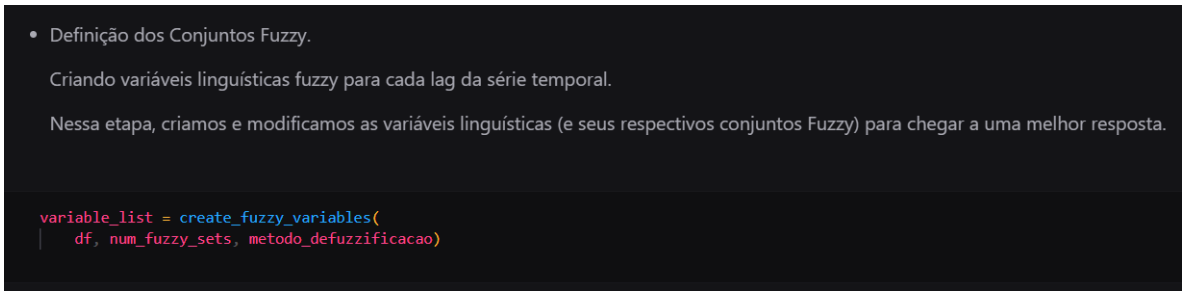

Figura 3: Definição dos conjuntos fuzzy

1.1.5 Etapa 3: Extração de Regras Fuzzy

Utilizando o método de Wang & Mendel, extraímos regras fuzzy dos dados de treinamento. Esse método permite a criação automática de regras a partir dos dados, considerando as pertinências das variáveis de entrada e saída.

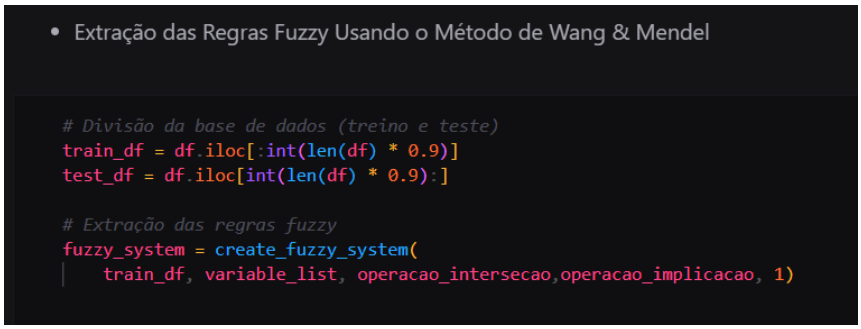

Figura 4: Extração de regras

1.1.6 Etapa 4: Construção do Sistema de Inferência

Nesta etapa, é realizada a definição do sistema com o auxílio do método *control* da biblioteca *Skfuzzy*.

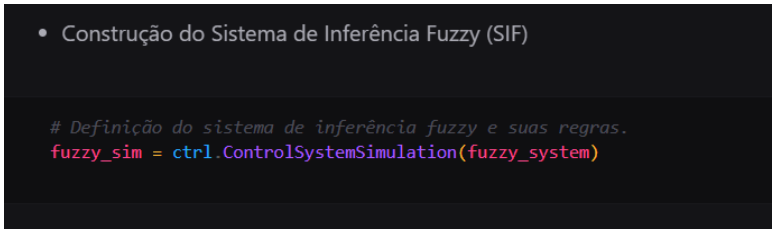

Figura 5: Construção do SIF

1.1.7 Etapa 5: Teste inicial - Previsão para o Conjunto de Treino e Teste e Plotagem dos Resultados

Após a construção do Sistema de Inferência Fuzzy, foi realizada a previsão para os conjuntos de treino e teste com os parâmetros mostrados na figura 1, seguida pela plotagem dos resultados.

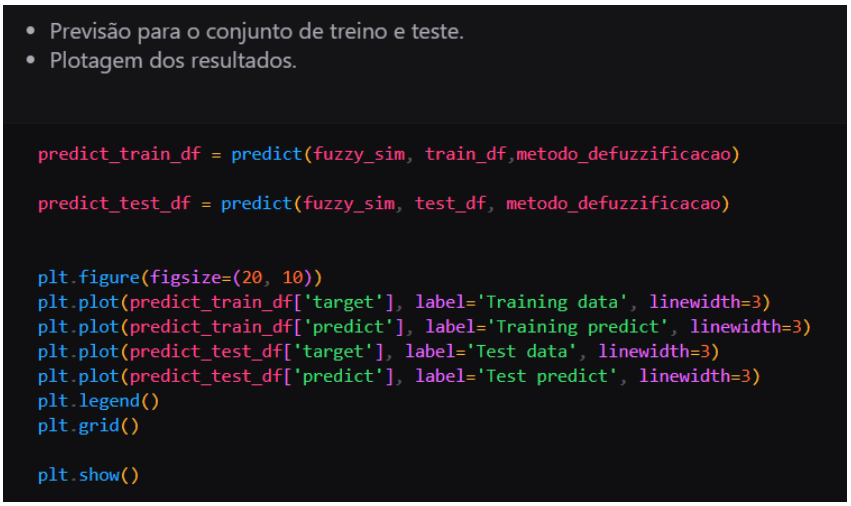

Figura 6: Código para Predição e plotagem

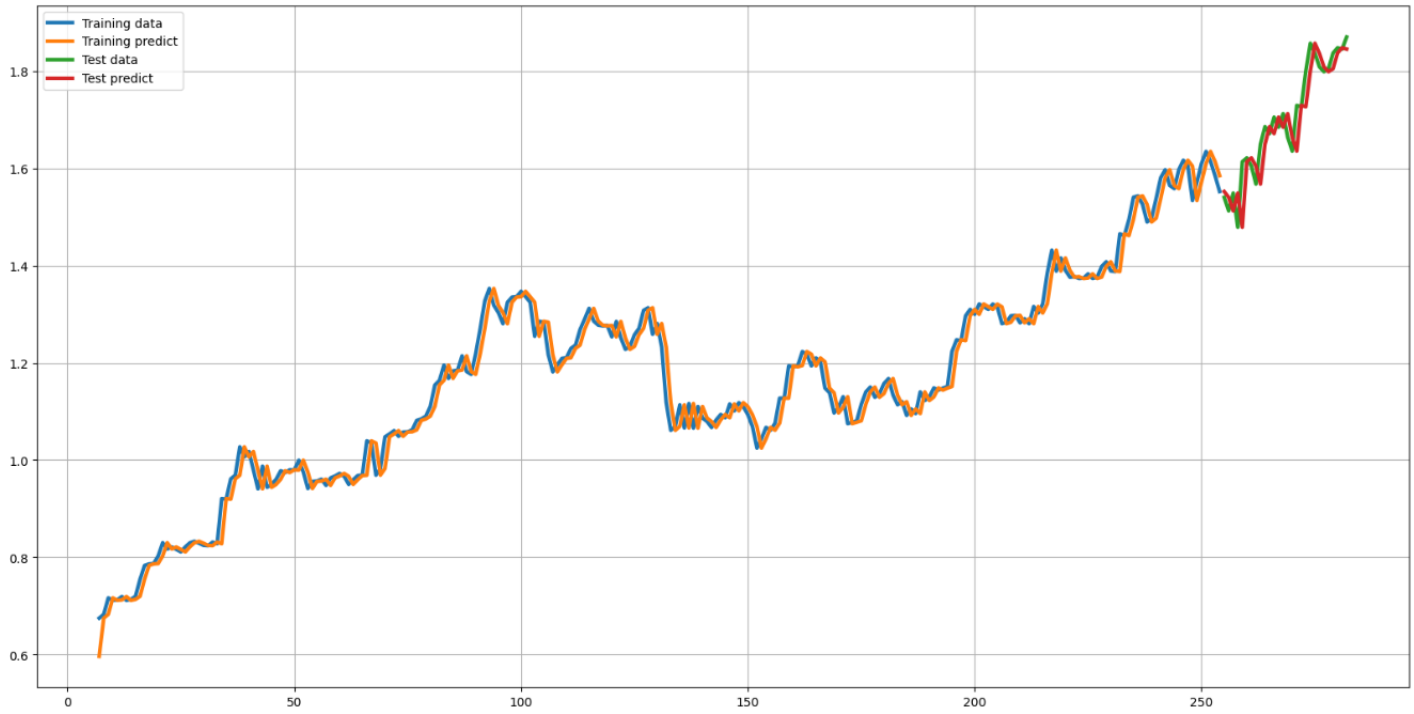

Figura 7: Gráfico resultante do primeiro teste

A métrica Mean Squared Error (MSE) quantifica o erro quadrático médio entre os valores previstos e os valores reais. Um MSE menor indica uma previsão mais precisa.

```
✓ 6.1s

5 conjuntos fuzzy com janela de 7 dias
Método de defuzzificação: Centro de Gravidade
Operação de interseção: Mínimo
Implicação: Produto
MSE Treino: 0.0008383450693580858
RMSE Treino: 0.028954189150416313

MSE Teste: 0.0021921380173535866
RMSE Teste: 0.04682027357196439
```

Figura 8: Avaliação de desempenho do modelo

Dados de Treino: O modelo demonstrou alta capacidade de aprendizado, ajustando-se com precisão à dinâmica histórica da série. A forte correlação observada entre os valores reais e preditos, aliada a um erro residual baixo, indica que a base de regras extraída pelo método de Wang & Mendel conseguiu mapear adequadamente os padrões não-lineares predominantes na amostra de treinamento.

Dados de Teste: Observou-se uma capacidade de generalização satisfatória. O sistema manteve a aderência à tendência da série financeira mesmo em dados não vistos, apresentando

resíduos com comportamento majoritariamente estocástico e sem degradação severa de desempenho, o que afasta a hipótese de *overfitting* excessivo.comparação entre as métricas revela que o modelo é robusto. Embora o erro no conjunto de teste seja naturalmente superior ao de treino (devido à incerteza inerente a dados futuros), a diferença manteve-se dentro de uma margem estreita. Isso valida a escolha dos hiperparâmetros (janela e número de conjuntos), demonstrando que o SIF não apenas "decorou"o ruído do treino, mas aprendeu a dinâmica subjacente do processo.figura 9 exibe os gráficos da avaliação de desempenho [1]. A análise visual corrobora os resultados numéricos: os gráficos de dispersão (superiores) mostram os pontos alinhados à diagonal ideal ($y = x$); a análise temporal dos resíduos (centro) indica estacionariedade, com erros oscilando em torno de zero sem tendências óbvias; e os histogramas (inferiores) confirmam que a distribuição dos erros se aproxima de uma Normal (Gaussiana), indicando ausência de vieses sistemáticos significativos na predição.

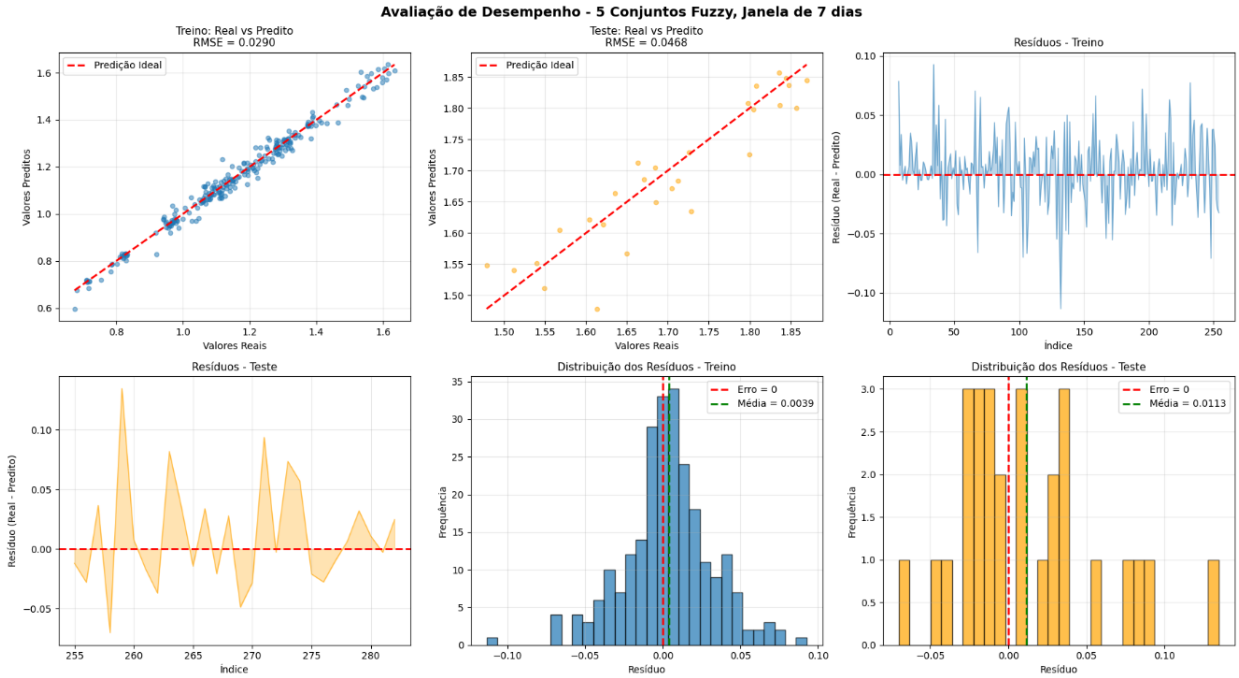

Figura 9: Visualização de desempenho do modelo

[1] Trecho de código disponível no apêndice A

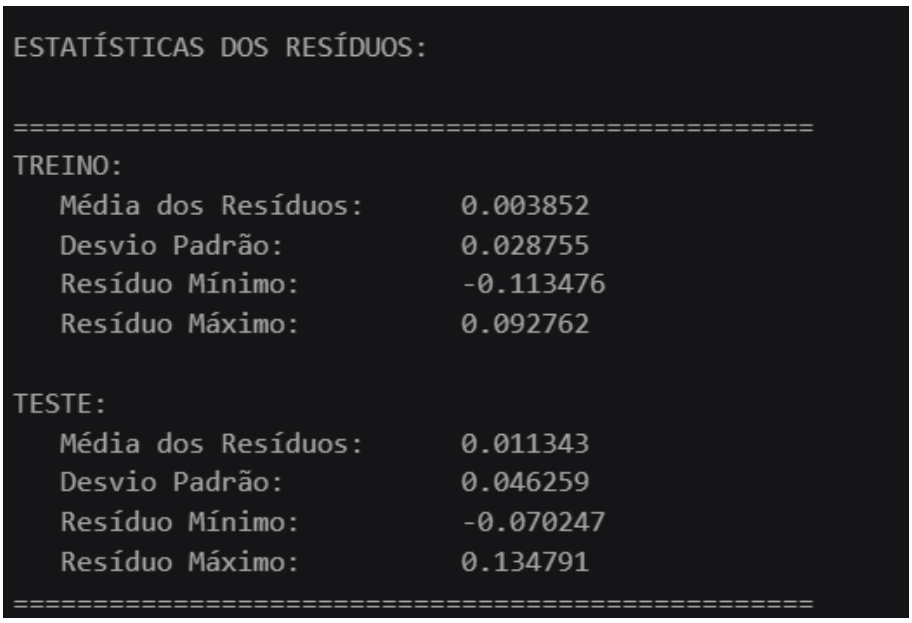

Figura 10: Estatísticas adicionais dos resíduos

As estatísticas descritivas dos resíduos mostram a qualidade do ajuste e a estabilidade do sistema fuzzy. A média dos resíduos próxima de zero em ambos os conjuntos (0.0038 no treino e 0.0113 no teste) indica que o estimador é estatisticamente não-viesado, ou seja, não há uma tendência sistemática forte de superestimar ou subestimar os preços.

Observa-se, contudo, que a média ligeiramente positiva no conjunto de teste sugere que, em média, o valor real foi marginalmente superior ao predito ($Resíduo = y_{real} - y_{pred} > 0$). Esse comportamento é consistente com o efeito de persistência em séries com tendência de alta, onde a previsão baseada em atrasos tende a ser conservadora. Além disso, o aumento controlado do desvio padrão no teste (0.046) em relação ao treino (0.028) demonstra que a dispersão dos erros se manteve em patamares aceitáveis, confirmando a robustez do modelo perante dados desconhecidos.

1.2 Otimização de Hiperparâmetros (Busca Exaustiva)

Para determinar a arquitetura ótima do controlador fuzzy preditivo, foi realizada uma busca em malha (*Grid Search*)[2] cobrindo todo o espaço de estados proposto. Foram avaliadas **324 combinações** de hiperparâmetros, variando:

- **Janela Temporal (Lags):** 1, 2, 3 e 5 dias.
- **Granularidade (Conjuntos Fuzzy):** 3, 5 e 7 conjuntos.
- **Motor de Inferência:** Interseção e Implicação (Mínimo, Produto, Média).
- **Defuzzificação:** Centro de Gravidade, Média Ponderada, Média dos Máximos.

[2]Código disponível no apêndice B

1.2.1 Melhor Configuração Encontrada (O Campeão)

A análise dos resultados, ordenada pelo erro no conjunto de teste (RMSE), revelou que o modelo de melhor desempenho é também um dos mais simples. A Tabela 1 detalha os parâmetros do modelo "Campeão".

Tabela 1: Parâmetros do Modelo de Menor Erro (Rank 1)

Hiperparâmetro	**Valor Selecionado**
Entradas (Lags)	**1** (y_{t-1})
Número de Conjuntos	**3** (Baixo, Médio, Alto)
Método de Interseção (E)	**Mínimo** (Norma de Mamdani)
Método de Implicação	**Mínimo** (Corte)
Método de Defuzzificação	**Centro de Gravidade**
Performance	
RMSE (Treino)	**0.02867**
RMSE (Teste)	**0.046429**
Posição no Ranking	1º de 324
Overfitting	0.017720

A escolha de uma janela de apenas 1 dia (*Lag 1*) indica que a série temporal da Apple (AAPL) possui uma forte característica Markoviana de primeira ordem, onde o preço de fechamento anterior contém a maior parte da informação necessária para a predição imediata, tornando redundante a inclusão de históricos mais longos.

1.3 Análise de Robustez (Top 10 Modelos)

A Tabela 2 apresenta as 10 melhores configurações encontradas. Observa-se um fenômeno de "platô de desempenho", onde as diferenças no RMSE ocorrem apenas na sexta casa decimal.

Tabela 2: Top 10 Configurações por Desempenho (RMSE Teste)

Lags	**Sets**	**Interseção**	**Defuzzificação**	**RMSE Teste**	**RMSE Treino**
1	3	Mínimo	C. Gravidade	0.0464	0.0287
1	3	Mínimo	M. Máximos	0.0464	0.0287
1	3	Mínimo	M. Ponderada	0.0464	0.0287
1	3	Mínimo	C. Gravidade	0.0464	0.0287
1	3	Mínimo	M. Máximos	0.0464	0.0287
1	3	Mínimo	M. Ponderada	0.0464	0.0287
1	3	Mínimo	C. Gravdidade	0.0464	0.0287
1	3	Mínimo	M. Máximos	0.0464	0.0287
1	3	Mínimo	M. Ponderada	0.0464	0.0287
1	3	Produto	C. Gravidade	0.0464	0.0287

A consistência dos resultados (todos com RMSE ≈ 0.0464) demonstra a robustez do método. Nota-se que as configurações de topo convergem para **3 Conjuntos Fuzzy** e janelas curtas (**1 Lag**), variando apenas nos métodos matemáticos de defuzzificação e inferência, que apresentaram impacto marginal no resultado final.

Análise de Sensibilidade dos Parâmetros A avaliação isolada de cada variável de decisão (marginalizando as outras) permite compreender a influência de cada fator no erro médio:

- **Impacto do Tamanho da Janela:** Conforme evidenciado nos gráficos de análise de sensibilidade, as janelas de **1 e 2 dias** apresentaram um erro médio de aproximadamente **0.0464**, enquanto as janelas de 3 e 5 dias elevaram o erro para o patamar de **0.0468**. Isso confirma que a inclusão de lags distantes introduz ruído ou atraso na resposta do modelo, prejudicando a predição em séries com tendência forte.
- **Impacto da Granularidade (Conjuntos):** O erro médio permaneceu constante em **0.0466** independentemente do número de conjuntos (3, 5 ou 7). Diante dessa invariância, justifica-se a escolha de **3 conjuntos**, pois reduz a complexidade computacional e facilita a interpretação linguística das regras.
- **Impacto dos Operadores:** Os métodos de interseção e implicação pelo **Mínimo** e **Produto** apresentaram desempenho virtualmente idêntico (RMSE médio ≈ 0.0466), superando ligeiramente a operação de Média (≈ 0.0467). Isso valida o uso da inferência padrão de Mamdani (Max-Min).

2 Parte 2: Extração Manual de Regras (Wang & Mendel)

Para validar a integridade do algoritmo computacional, realizamos a execução manual do método de Wang & Mendel para as primeiras amostras da série. Utilizamos a configuração otimizada de menor complexidade encontrada na seção anterior. Para a demonstração manual, adotamos uma janela de 2 Lags para ilustrar a interação entre múltiplas entradas, mantendo os 3 conjuntos do modelo ótimo.

- **Janela (Lags):** 2 ($y_{t-2}, y_{t-1} \rightarrow y_t$).
- **Conjuntos Fuzzy:** 3 (Baixo, Médio, Alto).
- **Universo de Discurso:** Definido pela amplitude local dos dados.

2.1 Dados da Amostra

Os dados foram extraídos diretamente do arquivo `a1621595_data.csv`. Observa-se que a série apresenta valores em escala normalizada ou ajustada (faixa de 0.50 a 0.70), característica mantida neste cálculo para garantir consistência com a implementação em Python.

Tabela 3: Primeiras 5 Amostras da Série (Dados Brutos)

Índice (k)	**Variável Temporal**	**Valor (y_k)**
1	$y(1)$	0.5909
2	$y(2)$	0.6100
3	$y(3)$	0.5947
4	$y(4)$	0.5863
5	$y(5)$	0.5792

2.2 Definição dos Conjuntos Fuzzy

O universo de discurso U para esta janela local é $[0.5792, 0.6100]$. Definimos três funções de pertinência triangulares (μ_B, μ_M, μ_A) distribuídas equidistantemente:

- **Centro Baixo (C_1):** 0.5800
- **Centro Médio (C_2):** 0.5950
- **Centro Alto (C_3):** 0.6100

A função de pertinência para o conjunto central (Médio), por exemplo, é dada por:

$$\mu_M(x) = \max\left(0, 1 - \frac{|x - 0.5950|}{0.0150}\right)$$

Onde 0.0150 é a semilargura da base do triângulo.

2.3 Extração de Regras (Passo a Passo)

O objetivo é gerar regras do tipo: *SE* $y(t-2)$ *é A E* $y(t-1)$ *é B ENTÃO* $y(t)$ *é C*.

2.3.1 Iteração 1: Prevendo o registro $k = 3$ ($y = 0.5947$)

Entradas:

- Lag 2: $x_1 = 0.5909$
- Lag 1: $x_2 = 0.6100$

Saída Alvo: $y = 0.5947$
Passo A: Fuzzificação (Cálculo dos Graus)

1. Para $x_1 = 0.5909$:
   - $\mu_B = 0$ (pois $x_1 > 0.5950$ no lado esq? Não, x_1 está entre B e M).
   - Cálculo exato: $\mu_M(0.5909) = 1 - \frac{|0.5909-0.5950|}{0.0150} = 1 - 0.273 = \mathbf{0.727}$
   - $\mu_B(0.5909) = 1 - 0.727 = \mathbf{0.273}$
   - **Maior Grau: Médio** (0.727).
2. Para $x_2 = 0.6100$:
   - Este valor coincide exatamente com o centro do conjunto Alto.
   - $\mu_A(0.6100) = \mathbf{1.0}$.
   - **Maior Grau: Alto**.
3. Para Saída $y = 0.5947$:
   - Muito próximo do centro Médio (0.5950).
   - $\mu_M(0.5947) = 1 - \frac{|0.5947-0.5950|}{0.0150} = \mathbf{0.98}$.
   - **Maior Grau: Médio**.

Passo B: Criação da Regra Atribuímos a regra baseada nos conjuntos de maior pertinência:

Regra #1: SE Lag_2 é **Médio** E Lag_1 é **Alto** ENTÃO Saída é **Médio**.

Passo C: Grau da Regra (Weight) Utilizando o operador produto para preservar a informação:

$$Grau = \mu_M(x_1) \times \mu_A(x_2) \times \mu_M(y)$$
$$Grau = 0.727 \times 1.0 \times 0.98 \approx \mathbf{0.712}$$

2.3.2 Iteração 2: Prevendo o registro $k = 4$ ($y = 0.5863$)

Entradas: $x_1 = 0.6100$ (Alto, $\mu = 1.0$) e $x_2 = 0.5947$ (Médio, $\mu = 0.98$).
Saída Alvo: $y = 0.5863$.
Fuzzificação da Saída: O valor 0.5863 está entre Baixo (0.580) e Médio (0.595).

$$\mu_B(0.5863) = \frac{0.595 - 0.5863}{0.015} = \frac{0.0087}{0.015} = \mathbf{0.58}$$

$$\mu_M(0.5863) = 1 - 0.58 = 0.42$$

O conjunto dominante é **Baixo**.
Geração da Regra:

> **Regra #2:** SE Lag_2 é **Alto** E Lag_1 é **Médio** ENTÃO Saída é **Baixo**.

Grau = $1.0 \times 0.98 \times 0.58 \approx \mathbf{0.568}$.

2.4 Conclusão da Extração Manual

O procedimento manual confirma a lógica do algoritmo: o sistema identifica padrões locais (ex: uma subida brusca para "Alto"seguida de uma correção para "Médio"tende a resultar em uma queda para "Baixo"na regra 2). Embora os valores numéricos sejam pequenos (escala normalizada), a geometria dos conjuntos fuzzy captura a dinâmica de oscilação do preço da ação.

3 Conclusões Gerais

O uso de diferentes configurações de conjuntos fuzzy, tamanhos de janela, operações de interseção, operações de implicação e métodos de defuzzificação mostrou-se crucial para otimização do desempenho do Sistema de Inferência Fuzzy. A implementação explícita do método de defuzzificação, em particular, demonstrou um impacto positivo significativo nos resultados, destacando a importância de um ajuste fino dos parâmetros do modelo para a obtenção de previsões mais precisas.

A Código Python para avaliação de desempenho dos resíduos

```
    fig, axes = plt.subplots(2, 3, figsize=(18, 10))
fig.suptitle(f'Avalia  o de Desempenho - {NUM_FUZZY_SETS} Conjuntos Fuzzy, Janela de {WINDOW_SIZE} dias',
             fontsize=14, fontweight='bold')

ax = axes[0, 0]
ax.scatter(predict_train_df['target'], predict_train_df['predict'], alpha=0.5, s=20)
ax.plot([predict_train_df['target'].min(), predict_train_df['target'].max()],
        [predict_train_df['target'].min(), predict_train_df['target'].max()],
        'r--', linewidth=2, label='Predi  o Ideal')
ax.set_xlabel('Valores Reais', fontsize=10)
ax.set_ylabel('Valores Preditos', fontsize=10)
ax.set_title(f'Treino: Real vs Predito\nRMSE = {rmse_train:.4f}', fontsize=11)
ax.legend()
ax.grid(True, alpha=0.3)

ax = axes[0, 1]
ax.scatter(predict_test_df['target'], predict_test_df['predict'], alpha=0.5, s=20, color='orange')
ax.plot([predict_test_df['target'].min(), predict_test_df['target'].max()],
        [predict_test_df['target'].min(), predict_test_df['target'].max()],
        'r--', linewidth=2, label='Predi  o Ideal')
ax.set_xlabel('Valores Reais', fontsize=10)
ax.set_ylabel('Valores Preditos', fontsize=10)
ax.set_title(f'Teste: Real vs Predito\nRMSE = {rmse_test:.4f}', fontsize=11)
ax.legend()
ax.grid(True, alpha=0.3)

ax = axes[0, 2]
residuos_train = predict_train_df['target'] - predict_train_df['predict']
ax.plot(residuos_train.index, residuos_train, alpha=0.6, linewidth=1)
ax.axhline(y=0, color='r', linestyle='--', linewidth=2)
ax.fill_between(residuos_train.index, residuos_train, 0, alpha=0.3)
```

```
ax.set_xlabel(' ndice ', fontsize=10)
ax.set_ylabel('Res duo (Real - Predito)', fontsize=10)
ax.set_title('Res duos - Treino', fontsize=11)
ax.grid(True, alpha=0.3)

ax = axes[1, 0]
residuos_test = predict_test_df['target'] - predict_test_df['predict']
ax.plot(residuos_test.index, residuos_test, alpha=0.6, linewidth=1, color='orange')
ax.axhline(y=0, color='r', linestyle='--', linewidth=2)
ax.fill_between(residuos_test.index, residuos_test, 0, alpha=0.3, color='orange')
ax.set_xlabel(' ndice ', fontsize=10)
ax.set_ylabel('Res duo (Real - Predito)', fontsize=10)
ax.set_title('Res duos - Teste', fontsize=11)
ax.grid(True, alpha=0.3)

ax = axes[1, 1]
ax.hist(residuos_train, bins=30, alpha=0.7, edgecolor='black')
ax.axvline(x=0, color='r', linestyle='--', linewidth=2, label='Erro = 0')
ax.axvline(x=residuos_train.mean(), color='g', linestyle='--', linewidth=2,
           label=f'M dia = {residuos_train.mean():.4f}')
ax.set_xlabel('Res duo', fontsize=10)
ax.set_ylabel('Frequ ncia', fontsize=10)
ax.set_title('Distribui  o dos Res duos - Treino', fontsize=11)
ax.legend()
ax.grid(True, alpha=0.3)

ax = axes[1, 2]
ax.hist(residuos_test, bins=30, alpha=0.7, edgecolor='black', color='orange')
ax.axvline(x=0, color='r', linestyle='--', linewidth=2, label='Erro = 0')
ax.axvline(x=residuos_test.mean(), color='g', linestyle='--', linewidth=2,
           label=f'M dia = {residuos_test.mean():.4f}')
ax.set_xlabel('Res duo', fontsize=10)
ax.set_ylabel('Frequ ncia', fontsize=10)
ax.set_title('Distribuicao dos Residuos - Teste', fontsize=11)
ax.legend()
ax.grid(True, alpha=0.3)

plt.tight_layout()
```

```
plt.show()

print("\nESTAT STICAS DOS RESIDUOS:")
print(f"\n{'='*50}")
print("TREINO:")
print(f"    M dia dos Residuos:      {residuos_train.mean():.6f}
    ")
print(f"    Desvio Padrao:           {residuos_train.std():.6f}")
print(f"    Residuo Minimo:          {residuos_train.min():.6f}")
print(f"    Res duo  M ximo:          {residuos_train.max():.6f}
    ")
print(f"\nTESTE:")
print(f"    M dia dos Residuos:      {residuos_test.mean():.6f}"
    )
print(f"    Desvio Padr o:           {residuos_test.std():.6f}")
print(f"    Res duo Minimo:          {residuos_test.min():.6f}")
print(f"    Res duo  M ximo:          {residuos_test.max():.6f}"
    )
print(f"{'='*50}")
```

B Código Python para Busca Exaustiva pelos Melhores Hiperparâmetros

```

print("="*100)
print("OTIMIZA  O DE HIPERPAR METROS DO SISTEMA DE
    INFER NCIA FUZZY")
print("="*100)

num_fuzzy_sets_list = [3, 5, 7]
window_sizes = [1, 2, 3, 5]
operacoes_intersecao_dict = {
    operacao_intersecao_min: "M nimo",
    operacao_intersecao_prod: "Produto",
    operacao_intersecao_avg: "M dia"
}
operacoes_implicacao_dict = {
    implicacao_min: "M nimo",
    implicacao_maxmin: "Soma Truncada",
    implicacao_prod: "Produto"
}
metodos_defuzzificacao_dict = {
    defuzzificacao_centroid: "Centro de Gravidade",
```

```
    defuzzificacao_mean_of_maxima: "M dia dos M ximos",
    defuzzificacao_weighted_average: "M dia Ponderada"
}

total_tests = (len(num_fuzzy_sets_list) * len(window_sizes) *
               len(operacoes_intersecao_dict) * len(
                  operacoes_implicacao_dict) *
               len(metodos_defuzzificacao_dict))

print(f"\nCONFIGURA  O DA BUSCA:")
print(f"   - Conjuntos Fuzzy: {num_fuzzy_sets_list}")
print(f"   - Janelas (lags): {window_sizes}")
print(f"   - Opera  es de Interse  o: {len(
   operacoes_intersecao_dict)}")
print(f"   - Opera  es de Implica  o: {len(
   operacoes_implicacao_dict)}")
print(f"   - M todos de Defuzzifica  o: {len(
   metodos_defuzzificacao_dict)}")
print(f"\n   TOTAL DE TESTES: {total_tests}")
print(f"\n{'='*100}\n")

resultados = []
contador = 0

for window_size in window_sizes:
    for num_fuzzy_sets in num_fuzzy_sets_list:
        for op_inter_func, op_inter_nome in
           operacoes_intersecao_dict.items():
            for op_impl_func, op_impl_nome in
               operacoes_implicacao_dict.items():
                for metodo_defuzz_func, metodo_defuzz_nome in
                   metodos_defuzzificacao_dict.items():

                    contador += 1
                    config_str = (f"[{contador}/{total_tests}]
                       Lags={window_size}, "
                                  f"Sets={num_fuzzy_sets}, Inter={
                                     op_inter_nome[:3]}, "
                                  f"Impl={op_impl_nome[:3]},
                                     Defuzz={metodo_defuzz_nome
                                     [:3]}")

                    print(f"{config_str}...", end=" ")

                    try:
                        df_lagged = lags_create(window_size,
                           FILE_PATH, 'Close')
```

```
55
56                      variable_list = create_fuzzy_variables(
57                          df_lagged, num_fuzzy_sets,
                                metodo_defuzz_func)
58
59                      train_df = df_lagged.iloc[:int(len(
                            df_lagged) * 0.9)]
60                      test_df = df_lagged.iloc[int(len(
                            df_lagged) * 0.9):]
61
62                      fuzzy_system = create_fuzzy_system(
63                          train_df, variable_list,
                                op_inter_func, op_impl_func, 1)
64                      fuzzy_sim = ctrl.ControlSystemSimulation
                            (fuzzy_system)
65
66                      mse_train, rmse_train, _ =
                            avaliar_modelo(fuzzy_sim, train_df,
                            metodo_defuzz_func)
67                      mse_test, rmse_test, _ = avaliar_modelo(
                            fuzzy_sim, test_df,
                            metodo_defuzz_func)
68
69                      num_regras = len(list(fuzzy_system.rules
                            ))
70
71                      resultados.append({
72                          'Lags': window_size,
73                          'Conjuntos': num_fuzzy_sets,
74                          'Interse  o': op_inter_nome,
75                          'Implica  o': op_impl_nome,
76                          'Defuzzifica  o':
                                metodo_defuzz_nome,
77                          'MSE_Treino': mse_train,
78                          'RMSE_Treino': rmse_train,
79                          'MSE_Teste': mse_test,
80                          'RMSE_Teste': rmse_test,
81                          'Regras': num_regras,
82                          'Overfitting': abs(rmse_test -
                                rmse_train)
83                      })
84
85                      print(f"    RMSE_teste={rmse_test:.4f}")
86
87                  except Exception as e:
88                      print(f"    Erro: {str(e)[:40]}")
89
```

```
90
91  print(f"\n{'='*100}")
92  print("AN LISE COMPLETA DOS RESULTADOS")
93  print(f"{'='*100}\n")
94
95  df_resultados = pd.DataFrame(resultados)
96
97  if len(df_resultados) > 0:
98      top10 = df_resultados.nsmallest(10, 'RMSE_Teste')
99
100     print("TOP 10 MELHORES CONFIGURA  ES (RMSE de Teste):")
101     print(top10[['Lags', 'Conjuntos', 'Interse  o', '
            Implica  o', 'Defuzzifica  o',
102                  'RMSE_Teste', 'RMSE_Treino', 'Regras']].
                       to_string(index=False))
103
104     melhor = df_resultados.loc[df_resultados['RMSE_Teste'].
            idxmin()]
105     print(f"\n{'='*100}")
106     print("CONFIGURA  O  TIMA :")
107     print(f"{'='*100}")
108     print(f"   Janela (Lags):           {melhor['Lags']}")
109     print(f"   Conjuntos Fuzzy:         {melhor['Conjuntos']}")
110     print(f"   Opera  o Interse  o:     {melhor['
            Interse  o']}")
111     print(f"   Opera  o Implica  o:     {melhor['
            Implica  o']}")
112     print(f"   M todo Defuzzifica  o:   {melhor['
            Defuzzifica  o']}")
113     print(f"

            ")
114     print(f"   RMSE Treino:             {melhor['RMSE_Treino
            ']:.6f}")
115     print(f"   RMSE Teste:              {melhor['RMSE_Teste']:.6
            f}")
116     print(f"   MSE Treino:              {melhor['MSE_Treino']:.6
            f}")
117     print(f"   MSE Teste:               {melhor['MSE_Teste']:.6f
            }")
118     print(f"   N mero de Regras:         {melhor['Regras']}")
119     print(f"   Overfitting:             {melhor['Overfitting
            ']:.6f}")
120     print(f"{'='*100}\n")
121
122     print("\nAN LISE DE SENSIBILIDADE DOS HIPERPAR METROS:\n")
123
```

```
    print("   Impacto do Tamanho da Janela (Lags):")
    print(df_resultados.groupby('Lags')['RMSE_Teste'].agg(['mean', 'min', 'max', 'std']).round(6))

    print("\n   Impacto do N mero de Conjuntos Fuzzy:")
    print(df_resultados.groupby('Conjuntos')['RMSE_Teste'].agg(['mean', 'min', 'max', 'std']).round(6))

    print("\n   Impacto da Opera   o de Interse   o:")
    print(df_resultados.groupby('Interse   o')['RMSE_Teste'].agg(['mean', 'min', 'max', 'std']).round(6))

    print("\n   Impacto da Opera   o de Implica   o:")
    print(df_resultados.groupby('Implica   o')['RMSE_Teste'].agg(['mean', 'min', 'max', 'std']).round(6))

    print("\n   Impacto do M todo de Defuzzifica   o:")
    print(df_resultados.groupby('Defuzzifica   o')['RMSE_Teste'].agg(['mean', 'min', 'max', 'std']).round(6))

    csv_path = Path.cwd().parent / 'results_hyperparameter_search.csv'
    df_resultados.to_csv(csv_path, index=False)
    print(f"\nResultados completos salvos em: {csv_path}")

else:
    print("\nNenhum teste foi conclu do com sucesso!")

print(f"\n{'='*100}")
```