

Preprocessing: Biblioteca de Pré-Processamento de Dados em Python

Autores: Fátima Pereira Santos Pinho
Ismael Rodrigues de Oliveira Neto
Rebeca Helen Batista Amorim

Agenda

Construção de uma biblioteca de pré-processamento de dados em Python, utilizando apenas recursos nativos da linguagem para resolver os desafios.

1. Introdução:

Contextualiza a importância do pré-processamento dos dados.

2. Fundamentação Teórica:

Apresenta um resumo e exemplos de cada técnica.

3. Resultados:

Exibe os resultados após a execução do código.

4. Considerações finais:

Apresenta um balanço do projeto, com as conquistas, desafios e próximos passos.

Introdução

O pré-processamento de dados é a etapa inicial da análise. Ele limpa e prepara dados brutos, que geralmente contêm inconsistências. Essa limpeza e transformação são cruciais para modelos de classificação, garantindo que o algoritmo aprenda padrões verdadeiros e faça previsões mais precisas.

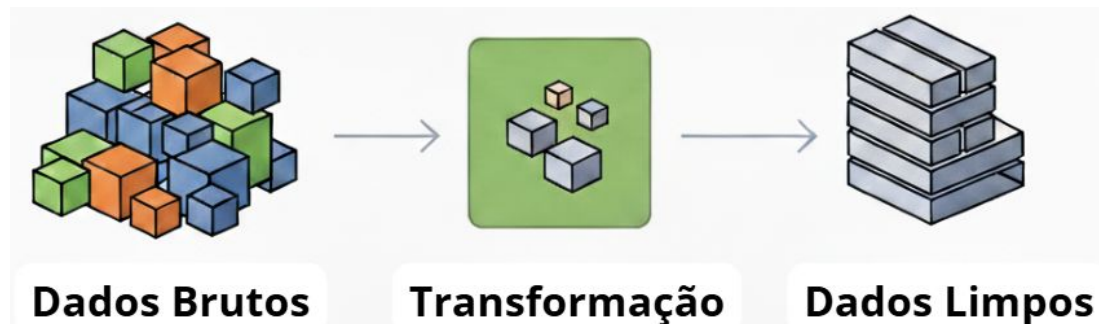


Figura 1: Fluxo simplificado do pré-processamento de dados.
Fonte: Google (2025).

Tratamento de valores ausentes

Fillna com mean (imputação por média):

Substitui valores ausentes (nulos) em uma coluna pela média de todos os outros valores existentes.

```
ALGORITMO ImputarPorMedia(coluna)
INÍCIO
    valores_validos ← []
    PARA CADA valor em coluna FAÇA
        SE valor não é nulo ENTÃO
            Adicionar valor a valores_validos
        FIM SE
    FIM PARA

    SE valores_validos não está vazio ENTÃO
        media ← Somar(valores_validos) / Tamanho(valores_validos)
        PARA CADA posição em coluna FAÇA
            SE coluna[posição] é nulo ENTÃO
                coluna[posição] ← media
            FIM SE
        FIM PARA
    FIM SE
FIM
```

Tratamento de valores ausentes **unex**

Fillna com mean (imputação por média):

Exemplo •

Dataset de notas com valores ausentes

```
notas_estudantes = {  
    'matematica': [8.5, 7.2, None, 9.1, 6.8],  
    'portugues': [7.8, None, 8.3, 8.7, 7.1]  
}
```

Aplicando imputação por média em matemática:

Valores válidos: [8.5, 7.2, 9.1, 6.8]

Média = $(8.5 + 7.2 + 9.1 + 6.8) / 4 = 7.9$

Resultado: [8.5, 7.2, 7.9, 9.1, 6.8]

Detecção e remoção de outliers

Método Z-Score

Transforma os dados para que tenham média 0 e desvio padrão 1, medindo a distância de cada ponto em relação à média

```
ALGORITMO DetectarOutliersZScore(dados, limiar=3.0)
INÍCIO
    media ← CalcularMedia(dados)
    desvio_padrao ← CalcularDesvioPadrao(dados)
    outliers ← []

    PARA CADA valor em dados FAÇA
        z_score ← ABS(valor - media) / desvio_padrao
        SE z_score > limiar ENTÃO
            Adicionar valor a outliers
        FIM SE
    FIM PARA

    RETORNAR outliers
FIM
```

Codificação de dados



Label Encoding

Converte dados categóricos (como "Bom", "Ruim") em números inteiros (0, 1), para que algoritmos processem essas informações.

```
// 1. DADOS ORIGINAIS: A coluna 'avaliacao_servico' com valores em texto.  
COLUNA 'avaliacao_servico' (ANTES):  
["Bom", "Ruim", "Ótimo", "Bom", "Regular"]  
  
// 2. REGRAS DE TRANSFORMAÇÃO: Definimos a regra para cada nível de satisfação.  
SE valor é "Ruim"      ENTÃO substituir por 1  
SE valor é "Regular"   ENTÃO substituir por 2  
SE valor é "Bom"       ENTÃO substituir por 3  
SE valor é "Ótimo"     ENTÃO substituir por 4  
  
// 3. PROCESSO: O algoritmo percorre a coluna aplicando as regras.  
"Bom"      -> se torna -> 3  
"Ruim"     -> se torna -> 1  
"Ótimo"    -> se torna -> 4  
"Bom"      -> se torna -> 3  
"Regular"  -> se torna -> 2  
  
// 4. RESULTADO FINAL: A coluna 'avaliacao_servico' agora é numérica e ordinal.  
COLUNA 'avaliacao_servico' (DEPOIS):  
[3, 1, 4, 3, 2]
```

Transformação de escalas



Min-Max scaler

Ajusta a escala dos dados para que todos os valores fiquem dentro de um intervalo específico, geralmente entre 0 e 1.

```
FUNÇÃO normalizar_min_max(valor, valor_min, valor_max):  
    // Retorna o valor normalizado aplicando a fórmula min-max.  
    RETORNO (valor - valor_min) / (valor_max - valor_min)  
FIM FUNÇÃO
```


Transformação de escalas

```
def minMax_scaler(self, columns: Set[str] = None): 5 usages (2 dynamic)  Fátima Pinho *
    """Aplica a normalização Min-Max ( $X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$ )
    nas colunas especificadas. Modifica o dataset. """
    target_columns = self._get_target_columns(columns)
    for column_name in target_columns:
        if not self._validate_numeric_data(column_name):
            continue

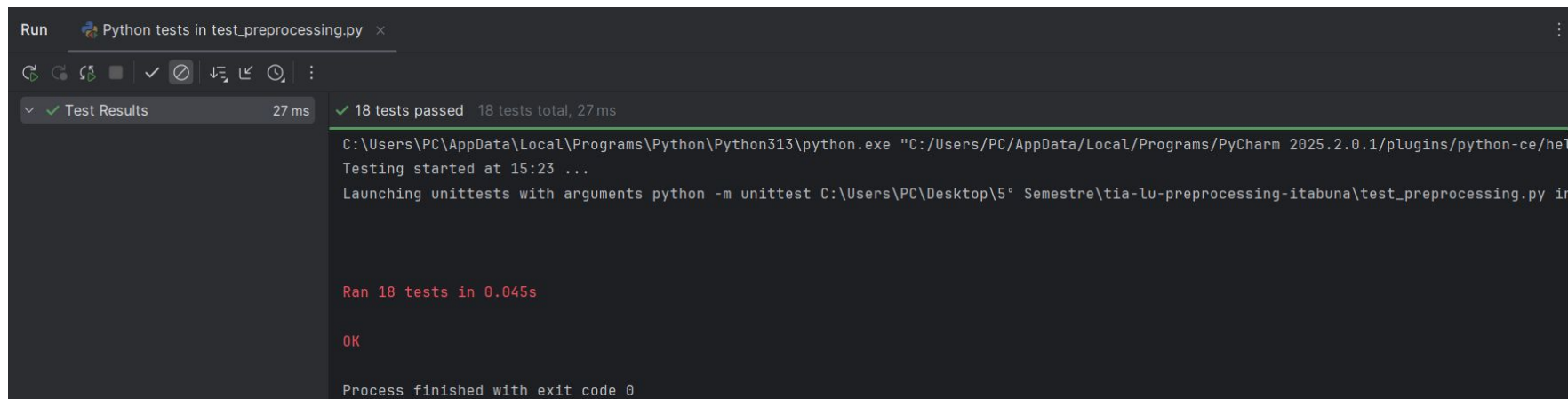
        column_data = self.dataset[column_name]
        if not column_data:
            continue

        min_value = min(column_data)
        max_value = max(column_data)
        value_range = max_value - min_value

        if value_range == 0:
            for index in range(len(column_data)):
                self.dataset[column_name][index] = 0.0
        else:
            for index in range(len(column_data)):
                normalized_value = (self.dataset[column_name][index] - min_value) / value_range
                self.dataset[column_name][index] = float(normalized_value)
```

Resultados

Todos os testes executados tiveram um retorno positivo conforme o esperado.



```
Run Python tests in test_preprocessing.py x
✓ 18 tests passed 18 tests total, 27 ms
C:\Users\PC\AppData\Local\Programs\Python\Python313\python.exe "C:/Users/PC/AppData/Local/Programs/PyCharm 2025.2.0.1/plugins/python-ce/he1
Testing started at 15:23 ...
Launching unittests with arguments python -m unittest C:\Users\PC\Desktop\5º Semestre\tia-lu-preprocessing-itabuna\test_preprocessing.py ir

Ran 18 tests in 0.045s

OK

Process finished with exit code 0
```

Considerações Finais



De modo geral, conseguimos criar com sucesso uma biblioteca de pré-processamento de dados funcional usando apenas Python nativo.

Principal desafio:

- Garantir a robustez da biblioteca em cenários extremos, tratando casos de borda como a divisão por zero (**Scaler**) e a integração com outras bibliotecas para lidar com dados vazios (**Statistics**).

Próximos Passos:

- Aprimorar a classe **Encoder** para tratar de forma resiliente valores ausentes (**None**) e dados mistos, aumentando sua aplicabilidade em cenários de dados do mundo real.

Referências



- ASTERA. O que é pré-processamento de dados? Definição, conceitos, importância, ferramentas. [Internet]. 13 mar. 2025. Disponível em: <https://www.astera.com/pt/type/blog/data-preprocessing/>. Acesso em: 16 set. 2025.
- IBM. O que é um pipeline de aprendizado de máquina? [Internet]. [s.d.]. Disponível em: <https://www.ibm.com/br-pt/think/topics/machine-learning-pipeline/>. Acesso em: 16 set. 2025.
- DATACAMP. Pré-processamento de dados: Um guia completo com exemplos em Python. DataCamp Blog, 2025. Disponível em: <https://www.datacamp.com/pt/blog/data-preprocessing>. Acesso em: 22 set. 2025.
-