Centro Universitário de Excelência
Sistemas de Informação

Pré-processamento de dados

Autor: <Rodrigo Souza Guimarães>
<César Filipe Gomes da Silva>
<Valnei Sousa Conceicao>

Feira de Santana, 2025

Agenda

- Apresentar a importância do pré-processamento no processo de KDD.
- Ilustrar e mostrar o código desenvolvido para tratar dados
- Relacionar teoria e prática sobre os processos realizado.

1. **Introdução ao Pré-processamento no KDD:**
Breve definição ao KDD e menção a etapa de pré-processamento

2. **Tratamento de Valores Ausentes:**
Inclui os tópicos e imagens sobre as principais funções que realizam o tratamento de valores ausentes

3. **Escalonamento de Dados:**
Definição e inclui os tópicos das funções que correspondem ao escalonamento de dados

4. **Codificação de Variáveis Categóricas:**
Definição e inclui os tópicos das funções que correspondem a codificação de variáveis categóricas

5. **Considerações finais:**
Breve detalhamento da importância da etapa(pré-processamento) e algoritmos utilizados

Introdução ao Pré-processamento no KDD

- O KDD é um processo iterativo para extrair conhecimento útil.
- O **pré-processamento** é a etapa que garante a **qualidade dos dados**.
- Sem dados consistentes, os resultados das análises podem ser incorretos.

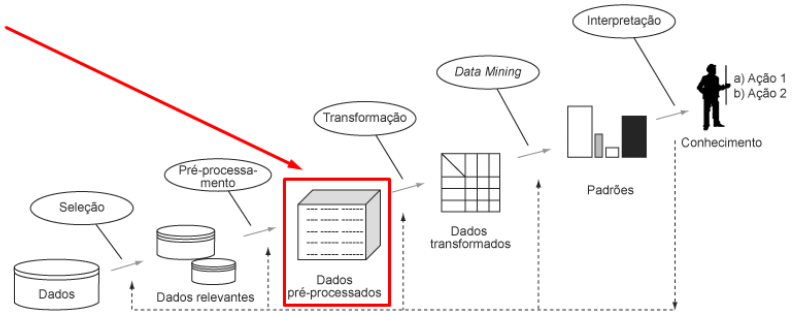


Figura 1. Etapas do processo *KDD* (Fayyad et al. (1996)).

Tratamento de Valores Ausentes

Objetivo

- Identificar(isna, notna) e lidar com dados nulos/None.

Estratégias de tratamento

- Dropna
- Fillna (média, mediana, moda ou valor padrão).

Questões importantes

- Qual é o impacto de simplesmente remover linhas com valores nulos?
- Em que situações usar média/mediana/moda?

```python
def isna(self, columns: Set[str] = None) -> Dict[str, List[Any]]:
    colunas = self._get_target_columns(columns)
    colunas_na = {coluna: [] for coluna in self.dataset}

    for linha in range(self._pegar_total_linhas(self.dataset)):
        if any(self.dataset[coluna][linha] is None for coluna in colunas):
            for coluna in colunas:
                colunas_na[coluna].append(self.dataset[coluna][linha])

    return colunas_na
```

```python
def notna(self, columns: Set[str] = None) -> Dict[str, List[Any]]:
    colunas = self._get_target_columns(columns)
    colunas_notna = {coluna: [] for coluna in self.dataset}

    for linha in range(self._pegar_total_linhas(self.dataset)):
        if all(self.dataset[coluna][linha] is not None for coluna in colunas):
            for coluna in colunas:
                colunas_notna[coluna].append(self.dataset[coluna][linha])

    return colunas_notna
```

```python
def dropna(self, columns: Set[str] = None):
    self.dataset = self.notna(columns)
```

Função fillna:

```python
def fillna(self, columns: Set[str] = None, method: str = 'mean', default_value: Any = 0):

    colunas = self._get_target_columns(columns)

    if method == 'mode':
        for coluna in colunas:
            moda = Statistics(self.dataset).mode(coluna)
            for i in range(self._pegar_total_linhas(self.dataset)):
                if self.dataset[coluna][i] is None:
                    self.dataset[coluna][i] = moda[0]
    elif method == 'mean':
        for coluna in colunas:
            media = Statistics(self.dataset).mean(coluna)
            for i in range(self._pegar_total_linhas(self.dataset)):
                if self.dataset[coluna][i] is None:
                    self.dataset[coluna][i] = round(media, 7)
    elif method == 'median':
        for coluna in colunas:
            mediana = Statistics(self.dataset).median(coluna)
            for i in range(self._pegar_total_linhas(self.dataset)):
                if self.dataset[coluna][i] is None:
                    self.dataset[coluna][i] = round(mediana, 7)
    elif method == 'default_value':
        for coluna in colunas:
            for i in range(self._pegar_total_linhas(self.dataset)):
                if self.dataset[coluna][i] is None:
                    self.dataset[coluna][i] = default_value
```

Escalonamento de Dados

Objetivo

- Necessário para algoritmos que são sensíveis à escala
- Ajusta a escala das variáveis para mesma faixa

Estratégia de tratamento

- **Min-Max Scaler**
- **Standard Scaler (Z-score)**

Questões importantes

- Em que contextos Min-Max é preferível ao Z-score?
- Como o escalonamento pode mudar os resultados de modelos de ML?

Codificação de Variáveis Categóricas

Objetivo

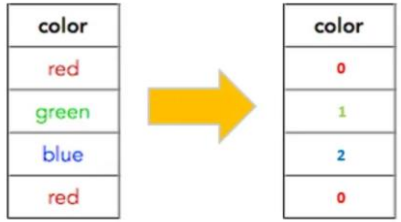
- Transformar categorias em valores numéricos

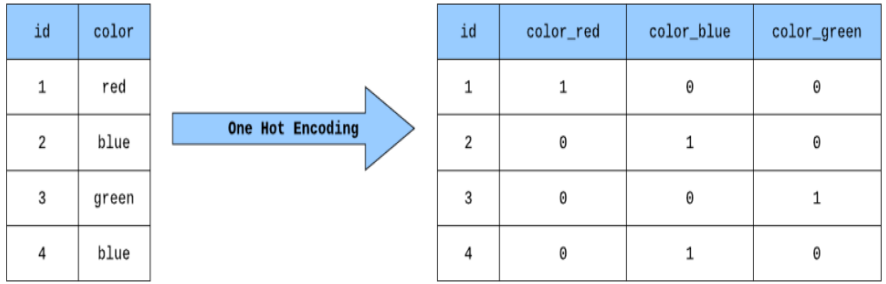
Estratégias de tratamento

- **Label Encoding**
- **One-Hot Encoding**

Questões importantes

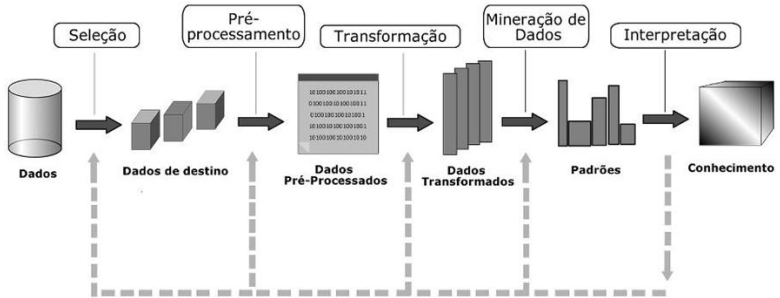
- Quando o **Label Encoding** pode introduzir vieses?
- O **One-Hot** pode aumentar muito a dimensionalidade?





Considerações finais

- O pré-processamento é **fundamental para qualidade** no KDD. Pois realiza o tratamento de valores ausentes, escalonamento e codificação tornam os dados **mais consistentes e utilizáveis**. Essa etapa prepara o código para análises estatísticas robustas e algoritmos de mineração de dados.



Referências

- https://iacomcafe.com.br/entendendo-label-encoding-python/. Acesso em: 27 set. 2025.
- https://www.devmedia.com.br/mineracao-de-dados-educacionais-usando-kdd-parte-1/28968. Acesso em: 27 set. 2025.
- https://www.scielo.br/j/gp/a/gC9RkgLD8B8FffPsNhBYWkB/?format=html&lang=pt. Acesso em: 27 set. 2025.