CURSO SUPERIOR DE ADS

Preparação de Dados e Pré Processamento de Dados

Prof. Fernando Marlon Soares Figueiredo

Disciplina: Ciência de Dados e Bigdata

SECTRAS

ADS

Preparação de Dados e Pré Processamento de Dados

O que é Preparação de Dados?

- **Definição:** A preparação de dados envolve todas as etapas necessárias para transformar dados brutos em um formato adequado para a análise e modelagem.
- **Importância:** Dados reais raramente estão prontos para serem usados diretamente em modelos de aprendizado de máquina. Sem uma preparação adequada, os modelos podem produzir resultados imprecisos, ser menos eficazes, ou mesmo falhar completamente.

1.2. Passos Principais na Preparação de Dados:

- **Coleta de Dados:** Obtenção dos dados de diversas fontes (bancos de dados, APIs, arquivos CSV, etc.).
- **Entendimento dos Dados:** Compreensão das características dos dados, como tipos de variáveis, distribuição e relações entre as variáveis.
- **Limpeza de Dados:** Remoção de valores faltantes, outliers e dados inconsistentes.
- **Transformação de Dados:** Conversão de dados em formatos apropriados, normalização, padronização, etc.
- **Redução de Dimensionalidade:** Seleção e extração de características relevantes para melhorar a eficiência e a precisão dos modelos.

Limpeza de Dados

2. Limpeza de Dados

- **2.1. Identificação e Tratamento de Valores Faltantes**
- **Métodos Comuns:**
  - **Remoção:** Excluir linhas ou colunas com muitos valores faltantes.
  - **Imputação:** Substituir valores faltantes por média, mediana, moda, ou usar técnicas mais avançadas, como imputação KNN.

Exemplo Prático:

```python
import pandas as pd

# Carregar um DataFrame de exemplo
df = pd.DataFrame({
    'A': [1, 2, None, 4, 5],
    'B': [5, None, None, 8, 10],
    'C': [10, 20, 30, 40, 50]
})

# Imputar valores faltantes com a média
df['A'].fillna(df['A'].mean(), inplace=True)
df['B'].fillna(df['B'].median(), inplace=True)
print(df)
```

2.2. Detecção e Tratamento de Outliers

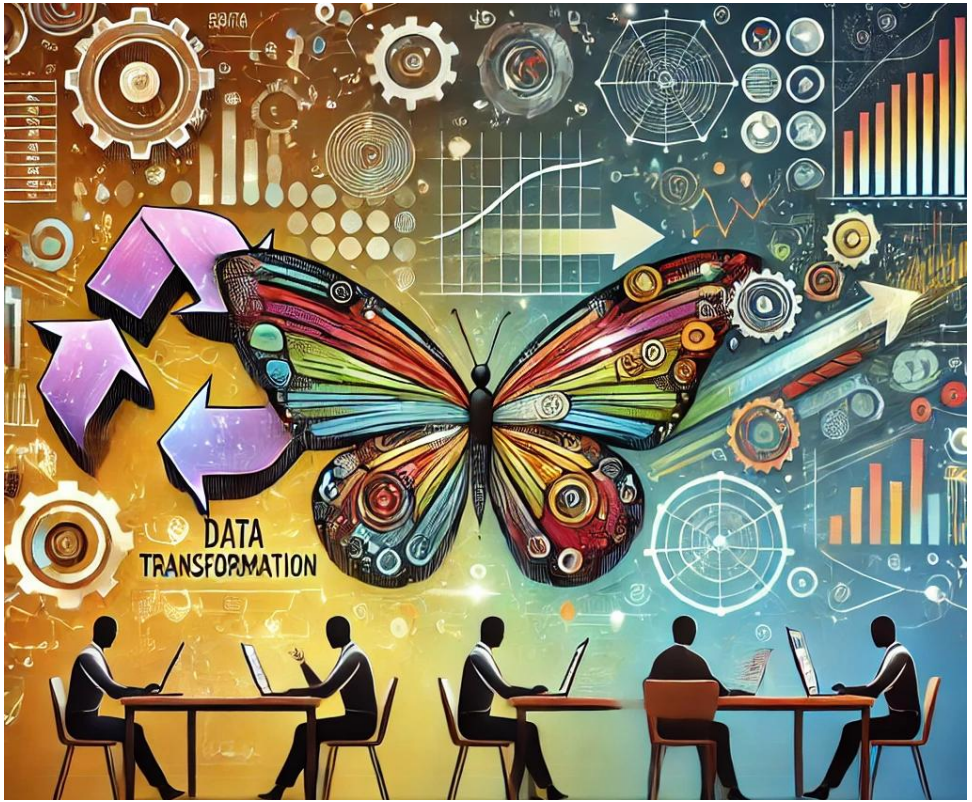
- **O que são Outliers?** Dados que são significativamente diferentes do restante dos dados.
- **Técnicas de Detecção:**
  - **Z-score:** Valores com Z-score > 3 ou < -3 são considerados outliers.
  - **IQR (Interquartile Range):** Dados fora de 1.5x o IQR acima do terceiro quartil ou abaixo do primeiro quartil.

2.2. Detecção e Tratamento de Outliers

```
# Detecção de outliers usando IQR
Q1 = df['C'].quantile(0.25)
Q3 = df['C'].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df['C'] < (Q1 - 1.5 * IQR)) | (df['C'] > (Q3 + 1.5 * IQR))]
print(outliers)
```

Transformação de Dados



3. Transformação de Dados

- **3.1. Normalização**
- **Normalização:** Escala os dados para um intervalo entre 0 e 1.
- O objetivo da normalização é mudar os valores das colunas numéricas no conjunto de dados para usar uma escala comum, sem distorcer as diferenças nos intervalos de valores nem perder informações. A normalização também é necessária para alguns algoritmos para modelar os dados corretamente

3.1. Normalização

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df[['A', 'B', 'C']] = scaler.fit_transform(df[['A', 'B', 'C']])
print(df)
```

3.1. Padronização

- **Padronização:** Consiste em subtrair a média e dividir pelo desvio padrão é uma técnica conhecida como normalização z-score ou padronização z-score.
- Transformar os dados para que tenham uma **média de 0** e um **desvio padrão de 1**.
- Tornar as variáveis comparáveis, especialmente quando estão em diferentes escalas.
- Melhorar o desempenho de algoritmos que dependem da escala dos dados, como regressão linear, SVM e redes neurais.

3.1. Padronização

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df[['A', 'B', 'C']] = scaler.fit_transform(df[['A', 'B', 'C']])
print(df)
```

3.2. Codificação de Variáveis Categóricas

- **3.2. Codificação de Variáveis Categóricas**
- **Codificação One-Hot:** Converte variáveis categóricas em colunas binárias.
    - **Exemplo Prático**

```
df = pd.DataFrame({
    'Categoria': ['A', 'B', 'A', 'C', 'B']
})

df_encoded = pd.get_dummies(df, columns=['Categoria'])
print(df_encoded)
```

3.2. Codificação de Variáveis Categóricas

- A **codificação One-Hot** é uma técnica comum usada para transformar variáveis categóricas em uma forma que pode ser usada por algoritmos de aprendizado de máquina.
- Muitos algoritmos não conseguem trabalhar diretamente com variáveis categóricas, especialmente aquelas que são não-ordenadas (ou **nominais**), então é necessário convertê-las em variáveis numéricas.
- O **One-Hot Encoding** resolve esse problema criando novas variáveis binárias (0 ou 1) para cada categoria.

3.2. Codificação de Variáveis Categóricas

- **Objetivo do One-Hot Encoding**
- **Converter variáveis categóricas em representações numéricas** que possam ser usadas por algoritmos de aprendizado de máquina.
- Evitar a atribuição de valores numéricos arbitrários que possam sugerir uma relação ordinal ou quantitativa entre categorias, quando não existe

3.2. Codificação de Variáveis Categóricas

- **Como Funciona o One-Hot Encoding**
- O One-Hot Encoding transforma uma variável categórica com nnn categorias em nnn variáveis binárias (também chamadas de **dummies**), onde cada variável binária indica a presença (1) ou ausência (0) de uma categoria.
- **Passos:**
- **Identificar as categorias únicas** da variável categórica.
- **Criar uma nova coluna para cada categoria**.
- **Atribuir 1** para a coluna correspondente à categoria presente em cada registro e **0** para as demais colunas.

3.2. Codificação de Variáveis Categóricas

- **Como Funciona o One-Hot Encoding**
- O One-Hot Encoding transforma uma variável categórica com nnn categorias em nnn variáveis binárias (também chamadas de **dummies**), onde cada variável binária indica a presença (1) ou ausência (0) de uma categoria.
- **Passos:**
- **Identificar as categorias únicas** da variável categórica.
- **Criar uma nova coluna para cada categoria**.
- **Atribuir 1** para a coluna correspondente à categoria presente em cada registro e **0** para as demais colunas.

3.2. Codificação de Variáveis Categóricas

- **Passos:**
- **Identificar as categorias únicas** da variável categórica.
- **Criar uma nova coluna para cada categoria**.
- **Atribuir 1** para a coluna correspondente à categoria presente em cada registro e **0** para as demais colunas.

3. Vantagens do One-Hot Encoding

- **Mantém a natureza nominal das categorias**: O One-Hot Encoding não atribui nenhum peso ou ordem às categorias. Cada categoria é tratada de forma igual.
- **Funciona bem com a maioria dos algoritmos** de aprendizado de máquina, como árvores de decisão, redes neurais, regressão logística, etc., que precisam de variáveis numéricas para fazer cálculos.

4. Desvantagens do One-Hot Encoding

- **Alta dimensionalidade**: Se uma variável categórica tiver muitas categorias únicas (por exemplo, milhares), o One-Hot Encoding resultará em uma grande quantidade de colunas. Isso pode aumentar muito a dimensionalidade do conjunto de dados, o que pode ser um problema para certos algoritmos.
- **Efeito de esparsidade**: Muitos dos valores nos dados resultantes serão 0, o que pode tornar os dados esparsos, impactando a eficiência computacional e a performance do modelo.

5. Situações em que o One-Hot Encoding é Necessário

- Quando se lida com **variáveis categóricas nominais**, onde as categorias não têm uma ordem natural (como cores, tipos de produtos, etc.).
- Em modelos que **não conseguem lidar com variáveis categóricas diretamente**, como regressão linear ou redes neurais, que esperam entradas numéricas.

6. Evitar a Colinearidade

- Quando se usa o One-Hot Encoding, se todas as categorias forem representadas por suas próprias colunas, pode ocorrer um problema de **colinearidade** (as novas variáveis criadas são interdependentes). Para evitar isso, é comum **remover uma coluna** do conjunto One-Hot.
- A categoria ausente é implicitamente representada quando todas as outras colunas têm valor 0.

- IMPLEMENTE

Considere um problema de classificação de tipos de animais com uma variável categórica "Espécie" com valores como "Cachorro", "Gato", e "Pássaro". Para que um algoritmo como uma rede neural possa usar esses dados, você aplicaria o One-Hot Encoding para representar essas categorias como

Cachorro	Gato	Pássaro
1	0	0
0	1	0
0	0	1

7. **Ferramentas para Implementar One-Hot Encoding**

A maioria das bibliotecas de ciência de dados tem métodos embutidos para realizar o One-Hot Encoding:

- **Pandas (Python):**
  - Função `pd.get_dummies()`.
- **Scikit-learn (Python):**
  - Função `OneHotEncoder()`.

```python
import pandas as pd

# Dados de exemplo
df = pd.DataFrame({'Cor': ['Vermelho', 'Azul', 'Verde', 'Vermelho', 'Azul']})

# Aplicando One-Hot Encoding
df_encoded = pd.get_dummies(df, columns=['Cor'])
print(df_encoded)
```

3.2. Codificação de Variáveis Categóricas

- **8. Alternativas ao One-Hot Encoding**
- **Label Encoding**: Transforma cada categoria em um número inteiro. No entanto, isso só deve ser usado se houver uma relação ordinal entre as categorias (por exemplo, "pequeno", "médio", "grande").
- **Target Encoding**: Substitui as categorias pelos valores médios de uma variável alvo associada a elas. Essa técnica é útil em conjuntos de dados com muitas categorias únicas.

4. Redução de Dimensionalidade

4. Redução de Dimensionalidade

- **4.1. Seleção de Características**
- **Seleção de Características**

A seleção de características (ou *feature selection*) é uma técnica usada para reduzir o número de variáveis de entrada em um modelo de aprendizado de máquina, mantendo apenas as mais importantes. Isso ajuda a melhorar o desempenho do modelo, reduzir o tempo de treinamento e prevenir o problema de **overfitting**.

Overfitting

- **Overfitting** (ou sobreajuste) ocorre quando um modelo de aprendizado de máquina se ajusta **muito bem aos dados de treinamento**, a ponto de capturar **não apenas os padrões reais**, mas também o **ruído e as variações aleatórias** dos dados. Como resultado, o modelo se torna extremamente específico aos dados de treinamento, perdendo a capacidade de **generalizar** para novos dados (dados de teste ou do mundo real).

Característica Principal do Overfitting:

1. Um modelo que sofre de overfitting tende a ter **alta precisão nos dados de treinamento**, mas **desempenho ruim** em dados que ele nunca viu antes (dados de teste ou novos dados).
2. Isso acontece porque o modelo fica "viciado" nos dados de treinamento e se ajusta muito ao que aprendeu nesses dados, capturando até mesmo as flutuações e irregularidades que são específicas desse conjunto, em vez de aprender os padrões gerais.

Causa do Overfitting:

- **Modelo muito complexo**: Ocorre principalmente quando o modelo é muito complexo em relação à quantidade de dados disponíveis. Modelos com muitos parâmetros, como redes neurais profundas ou modelos de árvore de decisão com muitas ramificações, são mais suscetíveis ao overfitting.
- **Poucos dados**: Se você tiver um conjunto de dados pequeno, o modelo pode acabar "memorizando" os dados de treinamento em vez de generalizar.
- **Demasiadas características**: Se o conjunto de dados tem muitas variáveis (features) irrelevantes ou correlacionadas, o modelo pode aprender a depender de ruídos em vez de padrões reais.

2. Métodos de Seleção de Características

- **Método de Filtro**
- O método de filtro envolve o uso de técnicas estatísticas para selecionar as variáveis mais relevantes, de forma independente do modelo de aprendizado de máquina que será usado.
- **Como Funciona**: As características são classificadas com base em suas correlações com a variável alvo (ou algum outro critério estatístico, como teste de qui-quadrado ou análise de variância - ANOVA). Apenas as características que têm uma forte relação com o alvo são selecionadas.
- **Exemplos de Técnicas**:
    - Correlação entre as características e a variável alvo (por exemplo, coeficiente de Pearson).
    - Testes de significância estatística, como *chi-squared* para variáveis categóricas.
    - Seleção de características baseadas na variância (mantenha as que têm maior variação).

Método Wrapper

- O método wrapper envolve testar diferentes combinações de variáveis e verificar o impacto de cada combinação na performance do modelo. Este método "envolve" o algoritmo de aprendizado de máquina para encontrar o subconjunto ideal de características.
- **Como Funciona**: Aqui, você faz várias execuções do modelo com diferentes combinações de variáveis e avalia o desempenho em cada uma delas. O objetivo é identificar a combinação que gera o melhor resultado de acordo com algum critério de performance, como precisão, recall ou F1-score.

- **Técnicas Comuns**:**Backward Elimination**: Começa com todas as variáveis e remove gradualmente aquelas que contribuem menos para o desempenho do modelo.
- **Forward Selection**: Começa sem nenhuma variável e vai adicionando as que mais aumentam a performance do modelo.
- **Recursão (Recursive Feature Elimination - RFE)**: Utiliza um algoritmo para treinar o modelo várias vezes, removendo as variáveis menos importantes em cada iteração.

Comparação entre Filtro e Wrapper

Método de Filtro: É rápido e independente do modelo de aprendizado, mas pode não capturar interações entre variáveis.

Método Wrapper: Fornece melhores resultados por testar o desempenho diretamente com o modelo, mas é computacionalmente caro, especialmente quando há muitas variáveis.

4.2. PCA (Principal Component Analysis)

- **O que é PCA?** Uma técnica de redução de dimensionalidade que transforma as variáveis em componentes principais, mantendo a maior parte da variação nos dados.

```
from sklearn.decomposition import PCA


pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
print(X_pca)
```

RESUMINDO...

Limpeza de Dados: A remoção de valores faltantes e outliers é crucial para garantir que os dados sejam de alta qualidade e representativos.

Transformação de Dados: A normalização, padronização, e codificação são passos essenciais para preparar os dados para modelos de aprendizado de máquina.

Redução de Dimensionalidade: Técnicas como Seleção de Características e PCA ajudam a simplificar os dados, mantendo a informação mais relevante.

REFERÊNCIAS

- HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** New York: Springer, 2009.
- KIMBALL, R.; CASERTA, J. **The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data.** Indianapolis: Wiley, 2004.
- GERON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.** 2. ed. Sebastopol: O'Reilly Media, 2019.
- DASU, T.; JOHNSON, T. **Exploratory Data Mining and Data Cleaning.** Hoboken: Wiley-Interscience, 2003.
- JOLLIFFE, I. T. Principal Component Analysis. 2. ed. New York: Springer, 2002.

REFERÊNCIAS

- Christopher., Chatfield, (1995). Problem solving : a statistician's guide 2nd ed. London: Chapman & Hall. ISBN 0412606305. OCLC 32881624
- Tukey, John W. (1962). «The Future of Data Analysis». The Annals of Mathematical Statistics (em inglês). 33 (1): 1–67. ISSN 0003-4851. doi:10.1214/aoms/1177704711