

Inteligência Artificial

Aula 10 – Manipulação de dados



Objetivo

- Preparar dados com a biblioteca Pandas para alimentar modelos de classificação usando o Scikit-learn, cobrindo:
 - Tratamento de valores ausentes
 - Codificação de variáveis categóricas
 - Normalização de atributos numéricos

Preparação

- Vamos usar um dataset fictício de exemplo:

```
import numpy as np

data = {
    'idade': [25, 30, np.nan, 45, 22],
    'salario': [50000, 60000, 52000, None, 42000],
    'genero': ['Feminino', 'Masculino', 'Feminino', np.nan, 'Feminino'],
    'comprou': ['sim', 'não', 'sim', 'não', 'sim']
}
```

Dados ausentes

- Em datasets do mundo real, é comum que algumas informações estejam ausentes
- Também chamados de *missing values*, podem comprometer análise estatística e desempenho dos modelos

Dados ausentes

- Modelos do Scikit-learn **não aceitam dados com valores nulos**
- Além disso, valores ausentes podem introduzir viés ou perda de informação

Dados ausentes

Técnicas comuns:

- **Remoção de dados:**

- Remover linhas (dropna) ou colunas inteiras com muitos valores faltantes

- **Preenchimento de valores (imputação):**

- **Numéricos:** usar média, mediana ou interpolação
 - **Catégoricos:** usar a moda (valor mais frequente)
 - **Modelos mais avançados:** imputação com algoritmos (ex: KNN Imputer)

Exemplo

- Conversão em DataFrame

```
import pandas as pd
```

```
df = pd.DataFrame(data)
```

```
print(df)
```

- Visualizar valores faltantes

```
print(df.isnull().sum())
```

Exemplo

- Estratégias comuns:
 - Preenchimento com média/mediana/moda

```
df['idade'].fillna(df['idade'].mean(), inplace=True)
df['salario'].fillna(df['salario'].median(), inplace=True)
df['genero'].fillna(df['genero'].mode()[0], inplace=True)
```

- Remoção de linhas (com cuidado!)

```
df.dropna(inplace=True)
```


Codificação de variáveis

- Alguns atributos nos dados são categóricos, ou seja, representam qualidades ou categorias, como “sexo”, “cor”, “estado civil”
- Esses dados precisam ser transformados em valores numéricos para que os algoritmos possam interpretá-los

Codificação de variáveis

- Vários algoritmos de machine learning não conseguem trabalhar diretamente com strings ou texto
- É necessário representar essas informações de forma numérica

Codificação de variáveis

Técnicas comuns:

- **Label Encoding (Codificação Ordinal):**

- Atribui um número inteiro a cada categoria
- Ex: ['baixo', 'médio', 'alto'] → [0, 1, 2]
- Útil quando existe **ordem** nas categorias

- **One-Hot Encoding:**

- Cria uma nova coluna para cada categoria, com 0 ou 1
- Ex: sexo → ['Masculino', 'Feminino'] vira duas colunas: sexo_Masculino, sexo_Feminino
- Ideal para categorias **sem ordem** (nominais)

Codificação de variáveis

One-Hot Encoding

datagy.io

Island		Biscoe	Dream	Torgensen
Biscoe	→	1	0	0
Torgensen		0	0	1
Dream		0	1	0

Exemplo

- Codificação ordinal
 - Ex: `LabelEncoder`
 - Útil para classes com ordem ou para target (y):

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
df['comprou'] = le.fit_transform(df['comprou'])
```

```
# 'sim' vira 1, 'não' vira 0
```

Exemplo

- Codificação nominal
 - Ex: OneHotEncoder ou get_dummies()

```
df = pd.get_dummies(  
    df,  
    columns=['genero'],  
    drop_first=True)
```

Normalização

- Processo de escalar os valores numéricos de forma que fiquem dentro de uma faixa comum
- Evita que uma variável com valores maiores domine as outras no modelo

Normalização

- Alguns algoritmos são sensíveis à escala dos dados, como KNN, SVM e Redes Neurais
- Variáveis em escalas diferentes podem causar problemas de performance ou interpretação incorreta das distâncias entre dados

Normalização

Técnicas comuns:

- **StandardScaler (Padronização):**
 - Transforma os dados para terem **média 0 e desvio padrão 1**
 - Preserva a forma da distribuição dos dados
- **MinMaxScaler:**
 - Escala os dados para o intervalo **[0, 1]**
 - Útil quando a distribuição não é normal e queremos preservar proporcionalidade

Exemplo

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

```
scaler = MinMaxScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

Exercício: Dataset Titanic

Dataset clássico para tarefas de classificação: prever se uma pessoa sobreviveu ou não com base em dados como idade, sexo, classe, etc.

Dataset disponível em <https://www.kaggle.com/c/titanic/data> (use o arquivo train.csv)

- Visualize o dataset e os tipos de dados, verifique colunas com valores nulos
- Trate: Age, Embarked, Cabin (média/moda/drop?)
- Remova colunas desnecessárias
- Transforme variáveis categóricas em numéricas
- Normalize colunas numéricas
- Aplicar os dados preparados em um modelo de classificação (Regressão Logística)