# PREPARAÇÃO DOS DADOS

A ordem das tarefas de preparação de dados pode variar dependendo do conjunto de dados específico e dos objetivos da análise ou modelagem. No entanto, a limpeza de dados é geralmente uma das primeiras etapas que deve ser realizada, pois os dados geralmente precisam ser limpos antes de serem transformados ou normalizados.

## LIMPEZA

A limpeza de dados pode incluir tarefas como:

-> Remoção de dados duplicados

-> Tratamento de valores faltantes

-> Correção de erros de digitação

-> Formatação de dados (por exemplo, convertendo data e hora para um formato padronizado)

Uma vez que os dados estejam limpos, então é possível fazer a transformação e a normalização dos dados. Transformação pode incluir tarefas como agregar ou dividir colunas, mudando o formato de data e hora, e criando novas variáveis. Normalização pode incluir tarefas como reescalonamento de valores ou colocando-os em uma escala comum.

É importante notar que essas tarefas podem ser realizadas iterativamente, ou seja, é possível precisar retornar e limpar novamente os dados após a transformação ou normalização.

### Código 1

import pandas as pd

**# carrega o conjunto de dados**

df = pd.read\_csv("data.csv")

**# remove linhas duplicadas**

df.drop\_duplicates(inplace=True)

**# preenche valores faltantes com a média**

df.fillna(df.mean(), inplace=True)

**# corrige erros de digitação nas colunas "name" e "email"**

df["name"] = df["name"].str.lower()

df["email"] = df["email"].str.replace("gmial", "gmail")

## TRANSFORMAÇÃO

A transformação de dados é o processo de modificar o formato ou a estrutura dos dados de entrada para torná-los compatíveis com os algoritmos de modelagem ou para facilitar a análise. Existem várias técnicas comuns de transformação de dados, como normalização, escalonamento, codificação one-hot, agregação e discretização.

Um exemplo simples de transformação de dados em Python seria o escalonamento de características numéricas em um conjunto de dados. Isso pode ser feito usando a biblioteca scikit-learn, que fornece uma classe chamada MinMaxScaler para isso:

### Código 1 - Escala

from sklearn.preprocessing import MinMaxScaler

import numpy as np

**# exemplo de conjunto de dados**

data = np.array([[1, 2], [3, 4], [5, 6]])

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

print(scaled\_data)

[[0. 0. ]

[0.5 0.5 ]

[1. 1. ]]

from sklearn.preprocessing import StandardScaler

data = np.array([[1, 2], [3, 4], [5, 6]])

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(data)

print(scaled\_data)

**A saída seria algo próximo de [[-1.22474487 -1.22474487] [0. 0. ] [1.22474487 1.22474487]]**

**Note que essas são apenas algumas das técnicas de transformação de dados disponíveis. As necessidades específicas de seus dados e modelo determinarão qual técnica é a mais adequada a ser utilizada.**

### RAIZ QUADRADA E LOGARITMO

Uma das transformações de dados mais comuns é aplicar a raiz quadrada ou o logaritmo em colunas numéricas do conjunto de dados para aliviar a distorção de escala. Isso é útil em casos em que algumas características possuem valores muito maiores do que outras, o que pode afetar negativamente a performance do modelo.

Aqui está um exemplo de como aplicar a raiz quadrada e o logaritmo em colunas específicas do conjunto de dados usando o pandas e numpy:

### Código 2 – SQ e LOG

import pandas as pd

import numpy as np

**# exemplo de conjunto de dados**

data = {

'feature1': [1, 3, 5, 7],

'feature2': [2, 4, 6, 8],

'feature3': [9, 27, 81, 243]

}

df = pd.DataFrame(data)

**# Aplicando a raiz quadrada**

df['feature1'] = np.sqrt(df['feature1'])

df['feature2'] = np.sqrt(df['feature2'])

df['feature3'] = np.sqrt(df['feature3'])

**#Aplicando o logaritmo**

df['feature1'] = np.log(df['feature1'])

df['feature2'] = np.log(df['feature2'])

df['feature3'] = np.log(df['feature3'])

print(df)

feature1 feature2 feature3

0 0.000000 1.414214 3.000000

1 1.732051 2.000000 5.196152

2 2.236281 2.449490 4.656613

3 2.645751 2.828427 5.567764

**É importante observar que nem sempre é uma boa ideia utilizar essas transformações em todas as colunas ou em todas as ocasiões, pois elas podem afetar na interpretabilidade dos resultados. Além disso, alguns modelos (como os baseados em árvores de decisão) são menos sensíveis às distorções de escala, então essa transformação pode ser desnecessária.**

### Código 3 - Outros

import pandas as pd

**# carrega o conjunto de dados**

df = pd.read\_csv("data.csv")

**# cria uma nova coluna com o ano de uma data**

df["year"] = pd.to\_datetime(df["date"]).dt.year

**# agrupa os dados por ano e calcula a média**

df\_agg = df.groupby("year").mean()

**# divide uma coluna em duas**

df[["first\_name", "last\_name"]] = df["name"].str.split(" ", expand=True)

## NORMALIZAÇÃO

A normalização dos dados é importante porque muitos algoritmos de aprendizado de máquina e outros métodos de análise são sensíveis à escala dos dados. Isso significa que se os dados não estiverem na mesma escala, os resultados podem ser afetados. Por exemplo, se você estiver trabalhando com um conjunto de dados que inclui informações sobre preços de imóveis e salários, os preços de imóveis provavelmente estarão em uma escala muito maior do que os salários, o que pode afetar a precisão de um modelo de aprendizado de máquina treinado nesses dados.

A normalização dos dados é uma técnica para colocar os dados em uma escala comum. Isso geralmente é feito escalando os dados para que eles estejam entre 0 e 1 ou para que a média seja 0 e o desvio padrão seja 1.

Aqui está um exemplo de como normalizar dados usando a biblioteca scikit-learn do Python:

### Código 1

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

**# carrega o conjunto de dados**

df = pd.read\_csv("data.csv")

**# seleciona as colunas a serem normalizadas**

data = df[["col1", "col2", "col3"]]

**# cria o objeto de normalização**

scaler = MinMaxScaler()

**# aplica a normalização aos dados**

data\_normalized = scaler.fit\_transform(data)

**Neste exemplo, o conjunto de dados é carregado usando o pandas, em seguida, as colunas "col1", "col2" e "col3" são selecionadas para serem normalizadas. O objeto MinMaxScaler é criado e usado para normalizar os dados, escalando-os para estarem entre 0 e 1.**

## Observações

### ChatGPT:

**Esses exemplos fornecem uma idéia geral de como realizar tarefas de limpeza, transformação e normalização de dados usando o Python e bibliotecas populares como o pandas e sklearn. É importante notar que essas tarefas podem ser mais complexas dependendo do conjunto de dados específico e dos objetivos da análise ou modelagem.**

**É importante notar que existem várias técnicas de normalização, como normalização Z-score, Log normalização e etc, que podem ser usadas dependendo do conjunto de dados específico e dos objetivos da análise ou modelagem.**

### Co-Autor:

**IMPORTANTE reparar que a informação do ChatGPT quanto à ordem ou uma suposta ordem na PREPARAÇÃO DOS DADOS, não implica necessariamente em uma obrigação. Veja que a TRANSFORMAÇÃO e a NORMALIZAÇÃO se confundem, uma vez que a NORMALIZAÇÃO implica na transformação dos dados. Também é importante compreender que, mesmo após a finalização da TRANSFORMAÇÃO / NORMALIZAÇÃO dos dados, talvez seja preciso “refazer” a LIMPEZA DOS DADOS e, por consequência, uma nova NORMALIZAÇÃO.**

**Outro ponto importante é entender a natureza da análise antes de começar qualquer TRANSFORMAÇÃO, as necessidades impostas tecnicamente e os objetivos a serem atingidos (Responder uma pergunta, fazer uma comparação, descobrir uma tendência, etc).**