CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

PRÉ-PROCESSAMENTO DE DADOS

Prof. Anderson França

OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

OBJETIVO

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

Capacitar os alunos a compreender e aplicar **técnicas essenciais de pré-processamento de dados** para garantir que um conjunto de dados esteja adequado para modelagem preditiva, com foco em **regressão**.

Durante a aula, os alunos irão:

- **Identificar e tratar valores ausentes**, escolhendo estratégias adequadas para minimizar impactos na análise.
- **Converter variáveis categóricas em numéricas** utilizando técnicas como **One-Hot Encoding e Dummy Encoding**, garantindo compatibilidade com modelos de regressão.
- **Aplicar normalização e padronização de variáveis numéricas**, avaliando quando usar **MinMaxScaler ou StandardScaler**.
- **Garantir a integridade dos dados** após todas as transformações, preparando-os para a fase de modelagem.

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

PRÉ-PROCESSAMENTO DE DADOS

OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

PRÉ-PROCESSAMENTO

O **pré-processamento de dados** é uma etapa fundamental para treinar qualquer modelo de machine learning. Essa etapa garante que os dados estejam em um formato adequado para o algoritmo, melhorando a precisão e o desempenho do modelo.

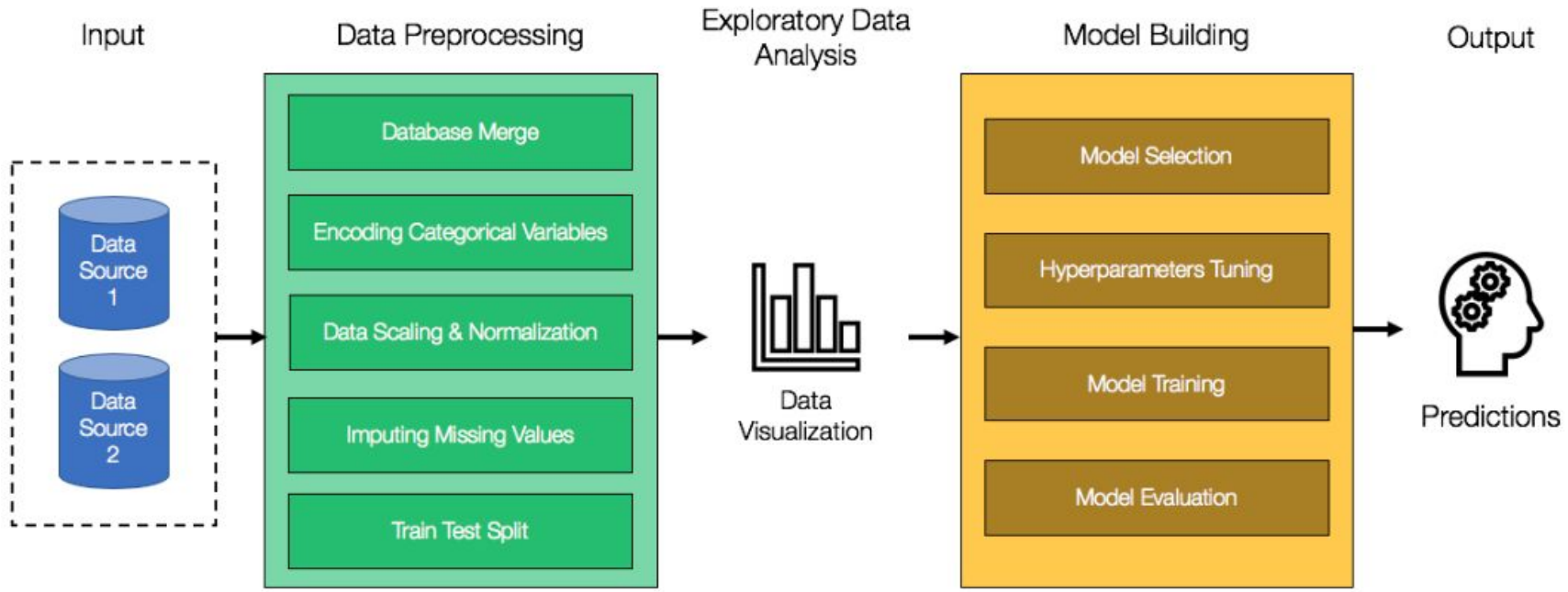
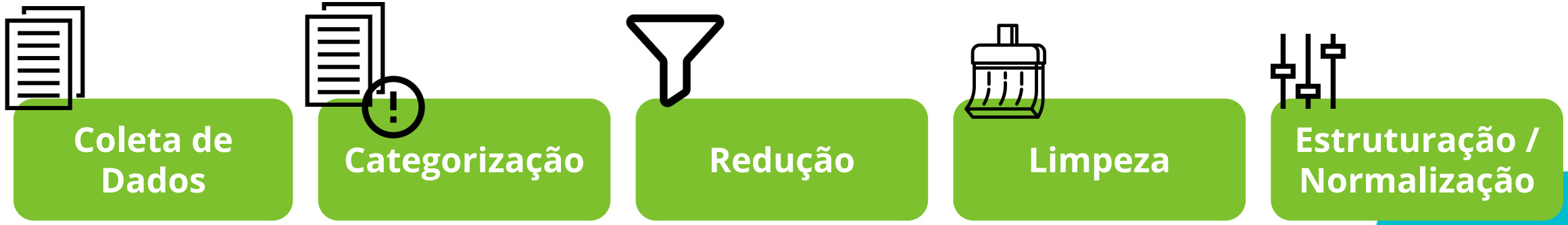


Imagem: [The machine learning workflow]

PRINCIPAIS OBJETIVOS DO PRÉ-PROCESSAMENTO

- **Melhorar a qualidade dos dados**: Corrigir erros, preencher valores ausentes e remover ruído.
- **Aumentar a eficiência dos modelos**: Garantir que os dados estejam em um formato adequado para o aprendizado.
- **Evitar viés e overfitting**: Manter a integridade dos dados e garantir que não haja padrões falsos.
- **Adaptar os dados ao algoritmo**: Ajustar escalas e formatos de acordo com a necessidade do modelo escolhido.



PRINCIPAIS ETAPAS

- **Limpeza dos dados:** Remoção ou imputação de valores ausentes e tratamento de duplicatas e inconsistências
- **Escalonamento e padronização:** Normalizar ou padronizar as variáveis numéricas para manter a coerência nas métricas.
- **Codificação de variáveis categóricas:** Converter variáveis não-numéricas em valores numéricos (e.g., One-Hot Encoding).
- **Tratamento de outliers:** Identificação e ajuste de valores extremos que possam influenciar o modelo.
- **Divisão dos dados:** Separação dos dados em conjuntos de treinamento e teste, garantindo que o modelo seja validado corretamente.

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

VALORES FALTANTES

OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

TRATAMENTO DE VALORES FALTANTES

O tratamento de valores ausentes é uma **etapa essencial do pré-processamento de dados**, uma vez que dados incompletos podem comprometer a performance dos nossos modelos.

A ausência de valores pode ocorrer por **falhas na coleta, erros humanos** ou **limitações de sistemas** e precisa ser tratada com cuidado para evitar distorções nos resultados.

Imagem: qresearch software

TRATAMENTO DE VALORES FALTANTES

Antes de iniciar o tratamento dos dados faltantes, é preciso **identificar o motivo desses valores faltantes** e se há algum valor determinado para esses casos.

Uma vez que já sabemos um pouco mais sobre esses valores faltantes, é necessário decidir se serão substituídos ou não, e qual o valor que deverá ser adicionado. Aqui estão algumas alternativas para trabalhar com esses valores faltantes:

- Valores como **“Não disponível”** ou **“NA”** podem ser utilizados para substituir os valores ausentes.
- **Substituir os valores manualmente**, o que é inviável para grandes conjuntos de dados.
- Utilizar o valor **médio/mediano** da variável.
- **Utilizar um algoritmo** (ex. Árvore de Decisão) para definir o valor mais provável

É importante lembrar que a escolha da técnica de tratamento de valores faltantes deve ser baseada nas características do conjunto de dados e no tipo de análise que será realizada. Além disso, é recomendável que os dados tratados sejam avaliados para verificar se a técnica escolhida não gerou novos problemas, como distorção ou perda de informação.

IDENTIFICAÇÃO DE VALORES AUSENTES

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

Antes de começar a tratar os valores faltantes, é necessário avaliar onde estão os dados ausentes. Podemos identificá-los da seguinte forma:

```
base_dados.isnull().sum()
```

filhos	1186
fumante	2208
regiao	0
custos_medicos	0
doencas_cronicas	1259

EXCLUIR VALORES FALTANTES

Linhas ou colunas com muitos valores ausentes não significativos podem ser excluídas com a função **dropna** dos pandas. Os parâmetros mais importantes da função são:

- axis: 0 para linhas, 1 para colunas
- inplace: atualizar os dados

```
#Excluir TODAS as observações que contenham NA
base_dados.dropna()
```

Note que a função retira todas as variáveis que continham valores faltantes. Para excluir uma coluna que possui valores faltantes, basta alterar o argumento ***axis***

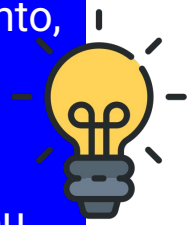
```
#Excluir Colunas com NA
base_dados.dropna(axis=1)
```

Após a substituição dos valores faltantes, agora é só atualizar o valor do index utilizando a função **reset_index**

ATENÇÃO: Remover os valores faltantes pode levar à perda de informação e, portanto, deve ser usada com cautela.

Além disso, é recomendável avaliar se a quantidade de dados removidos não afetou significativamente a análise dos dados.

Para alguns casos, pode ser mais adequado preencher os valores faltantes com medidas estatísticas ou usar técnicas de imputação de dados.



IMPUTAÇÃO DE VALORES FALTANTES

Existem muitas técnicas de imputação de dados, entre elas:

- **Utilizar a média, mediana ou moda**: Nesse método, substitui-se os valores faltante por alguma métrica de posição de cada variável. É uma técnica simples e rápida de ser implementada, mas pode gerar distorções nos dados se os valores ausentes estiverem concentrados em uma determinada faixa de valores ou se houver um volume muito alto de valores ausentes.
- **Imputação por regressão**: Utiliza-se a regressão linear para estimar os valores ausentes com base nas demais variáveis do conjunto de dados. É uma técnica mais complexa do que o preenchimento por média, mediana ou moda, mas pode gerar estimativas mais precisas, especialmente se houver relação linear entre as variáveis.
- **K-vizinhos mais próximos (K-NN)**: essa técnica consiste em preencher os valores ausentes utilizando os vizinhos mais próximos em um conjunto de dados. É uma técnica que leva em conta a relação entre as observações, mas pode gerar estimativas imprecisas se o conjunto de dados for muito grande ou se houver muitos valores ausentes.
- **Imputação por modelo**: Utiliza-se modelos de aprendizado de máquina para estimar os valores faltantes. É uma técnica que pode gerar estimativas precisas se o modelo utilizado for adequado, mas também pode ser computacionalmente intensiva e requerer um conjunto de dados grande o suficiente para treinar o modelo.

IMPUTAÇÃO SIMPLES

Para substituir valores faltantes (NaN) por zero (0), ou qualquer outro valor já definido, podemos utilizar o método .fillna():

```
#Substituir valores faltante por 0

base_dados['filhos'] = base_dados['filhos'].fillna(0)
```

IMPUTAÇÃO DE VALORES CATEGÓRICOS

Para preencher valores faltantes em variáveis categóricas, utilize a função fillna() especificando o valor desejado. Inclua o argumento inplace=True para aplicar a modificação diretamente no DataFrame, sem a necessidade de criar uma nova variável.

```
# Substituição por um valor definido em cada coluna
base_dados = base_dados.fillna({'fumante': 'nao'})
```

IMPUTAÇÃO COM MÉDIA/MEDIANA

Para fazer a imputação utilizando a média, basta usar a função fillna() passando a **média da coluna** como argumento e definindo inplace=True para que a alteração seja feita diretamente no DataFrame.

```
media_filhos = base_dados['filhos'].mean()
base_dados['filhos'] = base_dados['filhos'].fillna(media_filhos)
```

Antes de optar pela média, é importante analisar a distribuição dos dados e verificar se essa é a medida mais adequada para a imputação. Se houver muitos outliers ou uma distribuição assimétrica, considere alternativas como a mediana ou técnicas mais avançadas

EXECUTANDO VÁRIAS OPERAÇÕES

É possível substituir múltiplos valores ausentes ao mesmo tempo em um único comando utilizando um dicionário dentro da função fillna(). Basta especificar, para cada coluna, o valor desejado para a substituição.

```
base_dados = base_dados.fillna({'fumante': 'nao',
                                'filhos': 0,
                                'doencas_cronicas':0
                                })
```

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

TRANSFORMAÇÃO DO DADOS

OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

TRANSFORMAÇÃO DOS DADOS

A Transformação de dados tem como objetivo modificar a distribuição ou formato dos dados para torná-los mais adequados para análise. As transformações de dados podem ser necessárias por vários motivos, incluindo:

- Reduzir a variabilidade nos dados e torná-los mais homogêneos;
- Melhorar a normalidade dos dados, o que é importante para muitas análises estatísticas;
- Facilitar a interpretação dos dados;
- Tornar os dados mais adequados para um determinado modelo de análise.

NORMALIZAÇÃO E PADRONIZAÇÃO

A **normalização e a padronização** são duas técnicas de escalonamento de dados, utilizadas para ajustar a escala das variáveis numéricas. Essas técnicas garantem que todas as variáveis tenham a mesma importância, evitando que uma variável com valores maiores domine as demais.

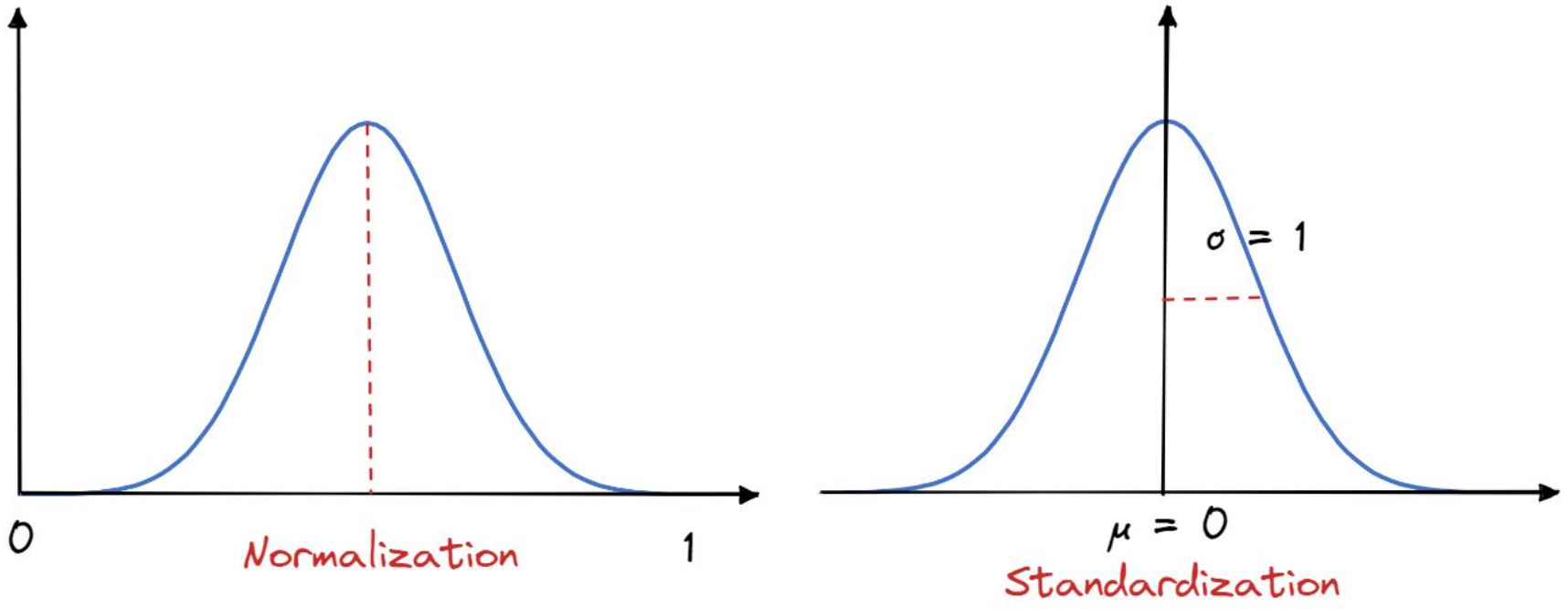


Imagem: AlgoDaily - Standardization & Normalization

APLICADA AO MACHINE LEARNING

Em modelos supervisionados (como regressão linear, regressão logística, e árvores de decisão), o escalonamento pode melhorar o desempenho e a estabilidade do modelo.

- **Regressão Linear:** A regressão linear é sensível à escala porque o modelo atribui coeficientes às variáveis. Variáveis em diferentes escalas podem influenciar de maneira desigual os coeficientes, gerando um **modelo enviesado**.

  Como a regressão linear assume uma distribuição normal para variáveis contínuas, a **padronizar os dados** garante que todas as variáveis tenham a mesma média (0) e desvio-padrão (1).

- **Regressão Logística:** Assim como a regressão linear, a escala pode afetar a otimização do modelo. Como a regressão logística é otimizada com base em gradientes, dados não **padronizados** podem prejudicar a convergência do modelo.

- **Árvores e Random Forest:** Apesar de não ser necessário padronizar os dados, se o modelo fizer parte de um *pipeline* com outros algoritmos sensíveis a escala (como alguns métodos de *ensemble*), o escalonamento pode ser útil.

NORMALIZAÇÃO

O objetivo da normalização é **escalar os dados para um intervalo específico**, geralmente entre [0, 1], garantindo que todas as variáveis tenham a mesma importância e evitando que aquelas com valores maiores dominem o modelo.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Quando Usar: Recomendado para algoritmos que exigem que os dados estejam em uma escala limitada (e.g., Redes Neurais e KNN).

NORMALIZAÇÃO NO PYTHON

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

Para realizar a **normalização** das variáveis numéricas, vamos utilizar a função **MinMaxScaler** da biblioteca **scikit-learn**. A normalização transforma os valores para um intervalo específico, geralmente entre **0 e 1**, mantendo a relação proporcional entre os dados.

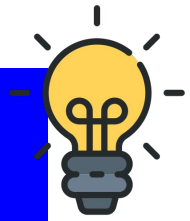
```
#Utilizando o sklearn para o escalonamento
from sklearn.preprocessing import MinMaxScaler

# Escolher colunas
colunas_escolhidas = ['custos_medicos','dieta_saudavel','IMC']

# Aplicar StandardScaler apenas nas colunas escolhidas
scaler = MinMaxScaler() #Definir os parâmetros da função
dados_padronizados = scaler.fit_transform(base_dados2[colunas_escolhidas])

base_dados2[['custos_medicos','dieta_saudavel','IMC']] = dados_padronizados
```

PADRONIZAÇÃO

O objetivo da **padronização** é transformar os dados para que tenham **média 0 e desvio-padrão 1**, alinhando-os a uma distribuição normal. Isso garante que todas as variáveis contribuem de forma equilibrada no modelo, sendo essencial em algoritmos que utilizam gradientes ou assumem uma distribuição gaussiana, como **SVM e Regressão Linear**.

$$x' = \frac{x - \mu}{\sigma}$$

Onde μ é a média e σ é o desvio-padrão da variável.



Quando Usar: Recomendado para algoritmos que assumem que os dados têm distribuição gaussiana, como **Regressão Linear** e **SVM**.

PADRONIZAÇÃO NO PYTHON

Para realizar a **padronização** das variáveis numéricas, você pode utilizar o **StandardScaler** da biblioteca **scikit-learn**. A padronização transforma os dados para que tenham **média 0** e **desvio-padrão 1**, o que é útil quando os dados têm diferentes escalas, mas precisam ser comparáveis.

```
#Utilizando o sklearn para a padronização
from sklearn.preprocessing import StandardScaler

# Escolher colunas
colunas_escolhidas = ['custos_medicos','dieta_saudavel','IMC']

# Aplicar StandardScaler apenas nas colunas escolhidas
scaler = StandardScaler()
dados_padronizados = scaler.fit_transform(base_dados2[colunas_escolhidas])

#salvar na base de dados original
base_dados2[['custos_medicos','dieta_saudavel','IMC']] = dados_padronizados
```

DIFERENÇA ENTRE OS DOIS MÉTODOS

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

Critério	Padronização (StandardScaler)	Escalonamento (MinMaxScaler)
Intervalo	Média 0, desvio padrão 1	Normalmente [0,1] ou [-1,1]
Afeta a distribuição?	Não altera	Sim, comprime ou expande os dados
Melhor para dados	Distribuição normal	Distribuição qualquer
Usado em	Regressão Linear, SVM, KNN, Redes Neurais	KNN, Redes Neurais, Modelos que usam distância
Sensível a outliers?	Menos sensível	Sim, pode ser afetado

DICAS

- Dados com **distribuição normal**: Use StandardScaler (padronização).
- Dados com **escalas muito diferentes ou sem distribuição normal**: Use MinMaxScaler (escalonamento).
- **Muitos outliers:** StandardScaler é mais robusto, pois o desvio padrão reduz o impacto.
- **Se estiver usando KNN, Redes Neurais ou SVM:** Prefira MinMaxScaler para evitar que valores grandes dominem o modelo.

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

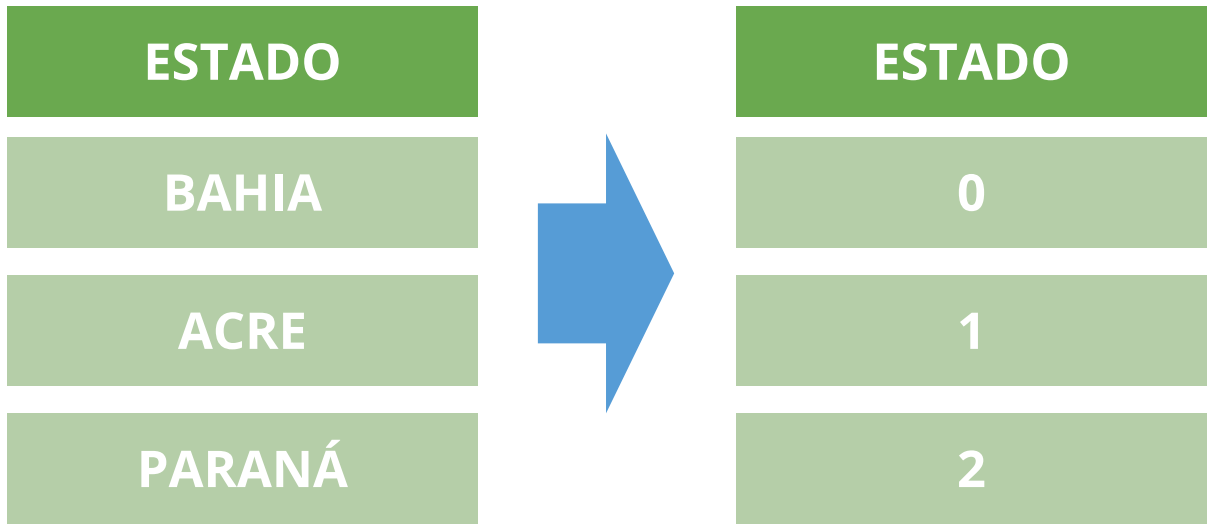
VARIÁVEIS CATEGÓRICAS

OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

LABEL ENCODING

Técnica de codificação que converte variáveis categóricas em valores numéricos inteiros. Cada categoria recebe um número exclusivo.

Essa técnica é simples e eficiente, ideal para variáveis com poucas categorias, porém, pode induzir uma ordem artificial entre as categorias, o que pode impactar algoritmos que consideram a relação entre os valores (como Regressão Linear).



LABEL ENCODING NO PYTHON

Para realizar o **Label Encoding** em variáveis categóricas, podemos utilizar a classe **LabelEncoder** da biblioteca **scikit-learn**. Essa técnica transforma cada categoria em um **valor numérico** inteiro, mas é importante lembrar que ela pode induzir **ordem artificial**, o que pode ser problemático em alguns algoritmos.

```
# Carregar Biblioteca do Label Encoder
from sklearn.preprocessing import LabelEncoder

# Carregar codificador
le_fumante = LabelEncoder()

# Aplicar label encoder em uma nova coluna
base_dados2['fumante_encoded'] =
le_fumante.fit_transform(base_dados2['fumante'])
```

	fumante	fumante_encoded
0	sim	1
1	nao	0
2	nao	0
3	nao	0
4	nao	0

DICA

Para visualizar os valores atribuídos a cada categoria após a codificação, é possível criar um **dicionário de mapeamento**, que permite reverter a transformação e entender quais números correspondem a cada classe original.

```
# Criar o dicionário mapeando os valores das categorias originais
dicionario_label = dict(enumerate(le_fumante.classes_))
print(dicionario_label)
```

```
{0: 'nao', 1: 'sim'}
```

CODIFICAÇÃO ORDINAL

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

Utilizada para converter variáveis categóricas em valores numéricos que respeitam uma ordem lógica entre as categorias.

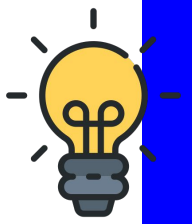
Cada categoria é mapeada para um número de acordo com sua hierarquia e isso preserva a relação de ordem entre as categorias, sendo ideal para variáveis com significado ordinal, como níveis de prioridade ou satisfação. Só devemos nos atentar na ordem, pois atribuir números arbitrários pode induzir uma relação que não existe.

ALTURA	ESTADO
BAIXO	1
MÉDIO	2
ALTO	3

LABEL ENCODING ORDINAL

Para realizar **Label Encoding Ordinal** (codificação com **ordem lógica**) em variáveis categóricas usando **OrdinalEncoder** do **scikit-learn**, podemos seguir o exemplo abaixo. Esse método é útil para variáveis com uma **ordem natural**, como "Baixo", "Médio", "Alto".

```
# Carregar biblioteca do Ordinal Encoder
from sklearn.preprocessing import OrdinalEncoder


# Definir ordem das categorias
oe_regiao = OrdinalEncoder(categories=[['sudeste','sudoeste','noroeste','nordeste']])

# Aplicar label encoder em uma nova coluna
base_dados2[['regiao_encoded']] = oe_regiao.fit_transform(base_dados2[['regiao']])
```



- **Ordem Correta:** Verifique se a ordem definida faz sentido no contexto do problema.
- **Categorias ausentes:** Se houver categorias nos dados que não estão na lista de **categorias ordenadas**, ocorrerá um erro. Use handle_unknown='use_encoded_value' para lidar com esse caso.

ONE-HOT ENCODING

É uma técnica de codificação usada para converter variáveis categóricas em **colunas binárias**, onde, para cada categoria única, é criada uma nova coluna que recebe 1 quando a categoria está presente e 0 caso contrário.

Essa abordagem evita atribuir ordens artificiais às categorias, mas pode resultar em **explosão dimensional** quando há muitas categorias. É muito utilizada em algoritmos que precisam de entradas numéricas.

ESTADO
BAHIA
ACRE
PARANÁ

ACRE	BAHIA	PARANÁ
0	1	0
1	0	0
0	0	1

ONE-HOT ENCODING COM ONEHOTENCODER (SCIKIT-LEARN)


```
# Gerar o one-hot encoding
base_dados2 = pd.get_dummies(base_dados2,
                             columns=['sexo'],
                             dtype=int)
```

sexo_feminino	sexo_masculino
1	0
0	1
0	1
0	1
0	1

VARIÁVEIS *DUMMY*

Técnica de **codificação binária** usada para representar variáveis categóricas em modelos estatísticos, especialmente em regressão linear e logística. O objetivo é converter categorias não numéricas em valores binários (0 e 1), facilitando o uso da variável em modelos que exigem dados numéricos.

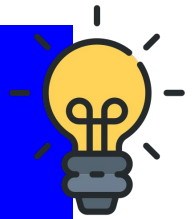
Para evitar redundância ou multicolinearidade nas análises, o método exclui uma das colunas (**drop first**), eliminando a coluna de referência. No exemplo abaixo, quando Acre e Bahia forem 0, infere-se que a categoria é Paraná (a excluída).

ESTADO
BAHIA
ACRE
PARANÁ

ACRE	BAHIA
0	1
1	0
0	0

One-hot: cria colunas para todas as categorias, muito útil em algoritmos como Redes Neurais e KNN.

Dummy: Exclui uma categoria pra evitar redundância (essencial para regressão linear e modelos estatísticos).



CRIAR *DUMMY*


```
# Criar uma cópia da base de dados
base_dados2 = base_dados.copy()

# Gerar variável Dummy
base_dados2 = pd.get_dummies(base_dados2,columns=['sexo'],
                             drop_first=True,
                             dtype=int)
```

1. **drop_first=False:** Mantém todas as colunas geradas pelo One-Hot Encoding. Se desejar remover uma categoria para evitar multicolinearidade (como em regressão linear), use **drop_first=True**.
2. **dtype=int:** Garante que as colunas geradas tenham valores inteiros **0** e **1**.

One-hot Encoding vs Dummy Encoding

A escolha entre One-Hot Encoding e Dummy Encoding vair depender do tipo de modelo que será utilizado e do tipo dos dados.

- **One-Hot Encoding** é ideal para modelos que não assumem relações lineares entre variáveis, como KNN, Redes Neurais e SVM, pois representa cada categoria de forma independente. No entanto, pode aumentar a dimensionalidade do conjunto de dados, o que pode ser um problema para bases muito grandes.
- **Dummy Encoding** é uma alternativa eficiente para modelos estatísticos, como Regressão Linear e Logística, pois evita multicolinearidade ao remover uma das categorias e usá-la como referência. Isso melhora a estabilidade dos coeficientes do modelo.

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

PARTICIONAMENTO DOS DADOS

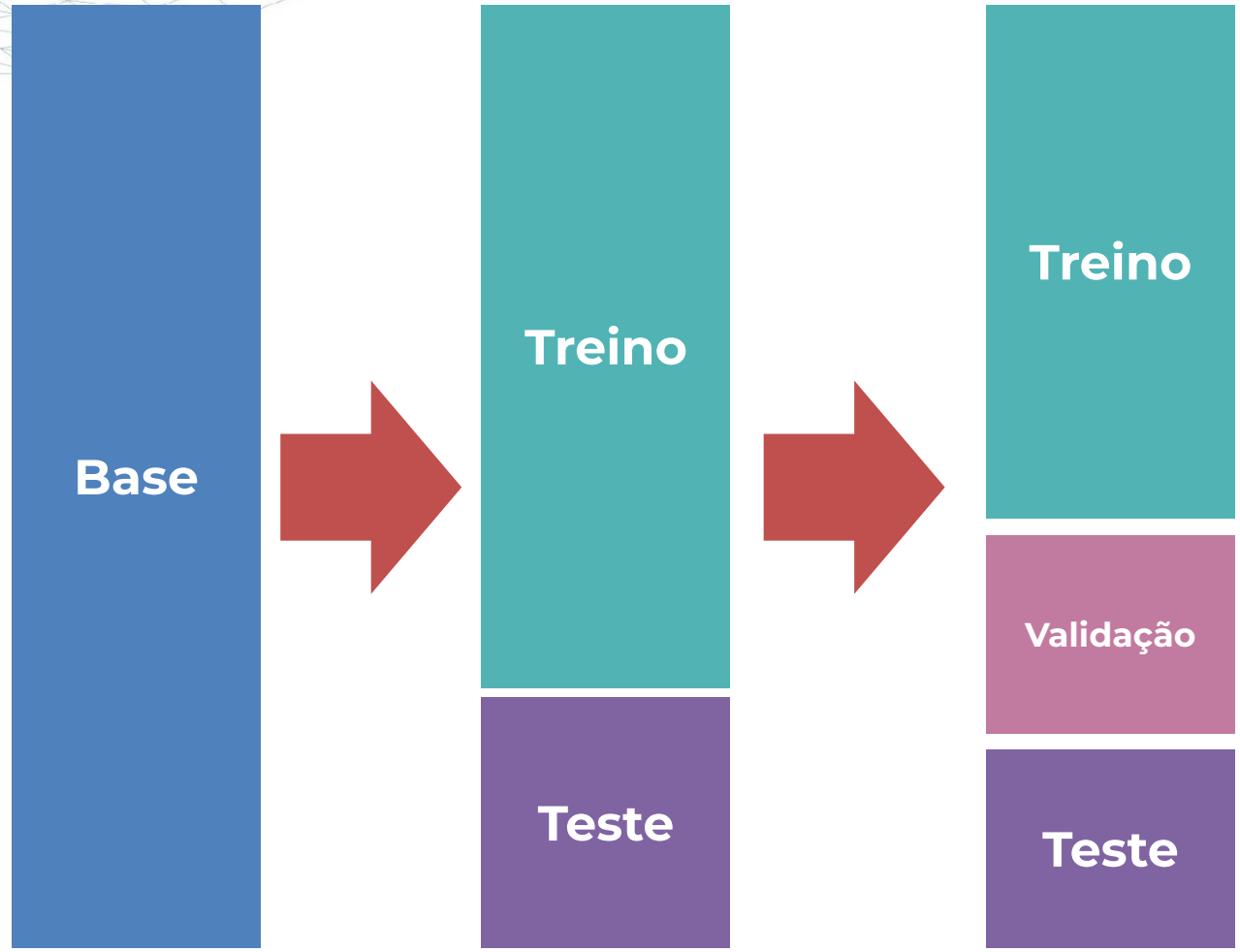
OSWALDO CRUZ HOSPITAL ALEMÃO
ANVISA Agência Nacional de Vigilância Sanitária
PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde
BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA
SUS
MINISTÉRIO DA SAÚDE
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

PARTICIONAMENTO DOS DADOS

A divisão dos dados é uma etapa fundamental em Machine Learning, garantindo que o modelo seja **treinado** e **avaliado** corretamente, evitando overfitting e promovendo uma generalização adequada.

- **Treinamento:** Ensinar o modelo com uma parte dos dados (dados de treinamento) para que ele aprenda os padrões.
- **Teste:** Avaliar o desempenho do modelo com dados não vistos (dados de teste) para medir sua capacidade de generalização.

HOLD-OUT (DIVISÃO SIMPLES)



Uma das formas mais utilizadas para a divisão da base de dados é separar em treinamento e teste, onde podemos atribuir 2/3 da base ao treinamento e o restante para a base de teste. Com isso, treinamos os modelos utilizando a base de treinamento e avaliamos o modelo na base de teste.

Se o resultado da base de treinamento for alta e o resultado na base de teste for baixa, muito provavelmente o modelo está super ajustado ou com overfitting

HOLD-OUT (DIVISÃO SIMPLES)

A função **train_test_split** da biblioteca **scikit-learn** é usada para dividir o dataset em dois conjuntos: **treinamento** e **teste**. Isso é essencial para avaliar o desempenho do modelo de forma justa, garantindo que ele seja validado em dados que **não foram usados durante o treinamento**.

```
from sklearn.model_selection import train_test_split

# Definindo X (variáveis independentes) e y (target)
X = base.drop('Inadimplente', axis=1)
y = base['Inadimplente']

# Dividindo o dataset em 80% treino e 20% teste
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=42)
```

stratify: Se a variável target for desbalanceada, use o argumento **stratify** para garantir que a proporção de classes seja mantida nos conjuntos de treino e teste:

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

ATIVIDADE

OSWALDO CRUZ HOSPITAL ALEMÃO · ANVISA Agência Nacional de Vigilância Sanitária · PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde · BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA · SUS · MINISTÉRIO DA SAÚDE · GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

INVESTIMENTO EM SAÚDE

A gestão da saúde pública enfrenta desafios na alocação de recursos devido à variabilidade nos custos médicos dos pacientes. Fatores como idade, hábitos de vida, condições preexistentes e tabagismo podem influenciar significativamente as despesas com atendimento médico. Para otimizar os investimentos em prevenção e tratamento, é fundamental entender quais características impactam mais os custos e como prever esses valores com precisão.

OBJETIVO

Nesta atividade, você deverá realizar **o pré-processamento completo dos dados** de uma base contendo informações sobre pacientes e seus respectivos gastos médicos. O objetivo é aplicar técnicas adequadas para tratar problemas comuns nos dados brutos, preparando-os para modelagem.

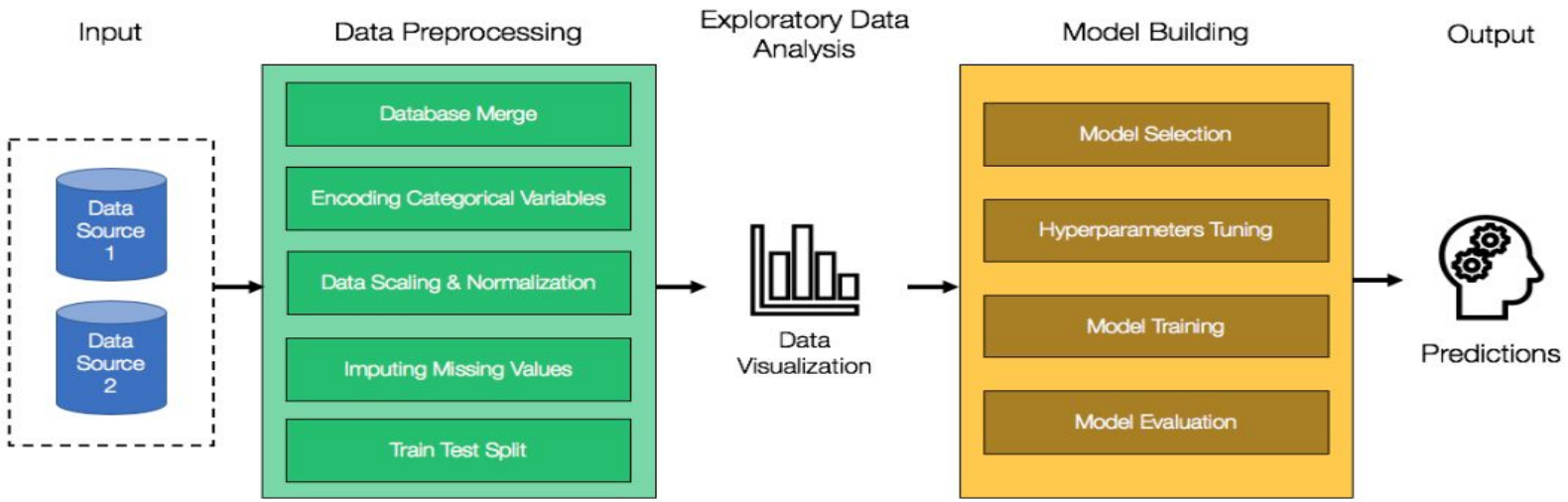


Imagem: The machine learning workflow

SUGESTÕES

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

- **Análise de valores ausentes**: Identifique quais colunas contêm dados faltantes e escolha uma estratégia apropriada para tratá-los.
- **Tratamento de variáveis categóricas**: Converta variáveis qualitativas para um formato numérico utilizando **One-Hot Encoding ou Dummy Encoding**, justificando a escolha.
- **Escalonamento e Padronização**: Decida se as variáveis numéricas precisam de **normalização (MinMaxScaler)** ou **padronização (StandardScaler)**, considerando o impacto nos modelos.

Lembre-se, o objetivo desta base de dados é passar por um **processo completo de pré-processamento** para garantir que esteja adequada para a **modelagem preditiva baseada em regressão**. Todas as transformações devem ser feitas considerando que as variáveis estarão estruturadas para um **modelo de regressão**, assegurando que os dados estejam limpos, numéricos e escalonados corretamente para a melhor performance do modelo.

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

REFERÊNCIAS

OSWALDO CRUZ HOSPITAL ALEMÃO · ANVISA Agência Nacional de Vigilância Sanitária · PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde · BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA · SUS · MINISTÉRIO DA SAÚDE · GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

REFERÊNCIAS

- Scikit-learn. Pré-processamento de dados. January, 2025. User Guide: https://scikit-learn.org/stable/modules/preprocessing.html
- Tuker, J. W. Exploratory Data Analysis. Pearson; 1st edition (January 1, 1977)
- McKinney, W. Python for Data Analysis - Data Wrangling with Pandas, NumPy & Jupyter. 3rd Edition - Open Access: https://wesmckinney.com/book/
- Google Colab - https://colab.research.google.com/
- Pandas - https://pandas.pydata.org/docs/index.html
- Numpy - https://numpy.org/

CIÊNCIA DE DADOS
E INTELIGÊNCIA ARTIFICIAL

OSWALDO CRUZ HOSPITAL ALEMÃO

ANVISA Agência Nacional de Vigilância Sanitária

PROADI-SUS Programa de Apoio ao Desenvolvimento Institucional do Sistema Único de Saúde

BRASIL BEM CUIDADO MAIS SAÚDE PARA QUEM MAIS PRECISA

SUS

MINISTÉRIO DA SAÚDE

GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO