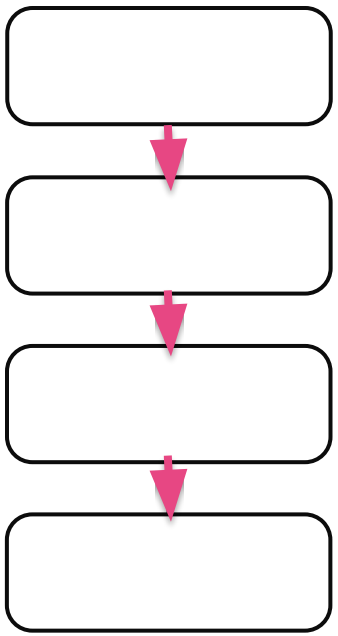
Python para dados

Pipeline de dados

WOMAKERS CODE

O que é?



- Coleta
- Transformação
- Carregamento/ Armazenamento
- Monitoramento

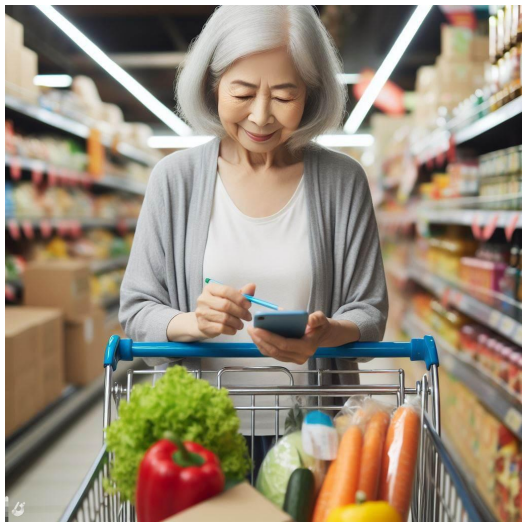
Automação

Pipeline de dados em português, é um sistema que facilita a coleta, processamento e movimentação de dados de uma fonte para um destino específico.Uma pipeline de dados, em termos gerais, é uma série de processos ou etapas interligadas que são usadas para mover e transformar dados de um ponto de origem para um destino específico. As pipelines de dados são comumente usadas em projetos de ciência de dados, engenharia de dados, análise de dados e em geral em ambientes onde é necessário lidar com grandes volumes de dados de maneira eficiente e automatizada.

Qual a diferença entre data wrangling e pipelines?

Data Wrangling

Data Pipeline



Data Wrangling

A MANIPULAÇÃO DE DADOS TORNÁ-LOS ADEQUADOS À RECEITA

Data Pipeline

COLETANDO/EXTRAINDO

TRANSFORMANDO

CARREGAMENTO

Data Wrangling

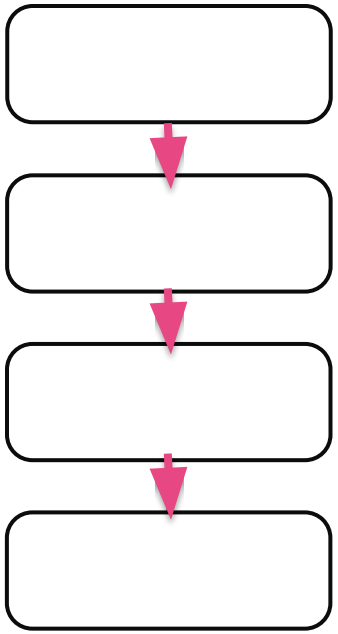
Suponha que você tenha um conjunto de dados com informações sobre clientes, mas as datas estão em um formato bagunçado. Data wrangling seria o processo de organizar essas datas para que elas possam ser facilmente compreendidas e utilizadas, como convertê-las para um formato padrão.

A manipulação de dados é como organizar e preparar esses ingredientes para torná-los adequados à receita. Isso envolve limpar os ingredientes, cortá-los do jeito certo, misturar quando necessário, e garantir que tudo esteja pronto para ser usado.

Data Pipeline

Vamos dizer que você tenha dados de vendas em diferentes lojas e deseje analisar o desempenho total. A ETL envolveria extrair os dados de cada loja, transformá-los para garantir consistência (por exemplo, unificar formatos de datas, moedas), e carregá-los em um único local para análise.

A ETL é como o processo de preparação de uma refeição completa, desde a escolha dos ingredientes (extração), o preparo (transformação) até colocar a comida na mesa (carregamento). No contexto de dados, ETL é o processo de mover dados de um lugar para outro, transformá-los para atender às necessidades específicas e, finalmente, carregá-los em um local onde possam ser utilizados. manipulação de dados é como organizar e preparar esses ingredientes para torná-los adequados à receita. Isso envolve limpar os ingredientes, cortá-los do jeito certo, misturar quando necessário, e garantir que tudo esteja pronto para ser usado.



- **Coleta**
- Transformação
- Carregamento/ Armazenamento
- Monitoramento

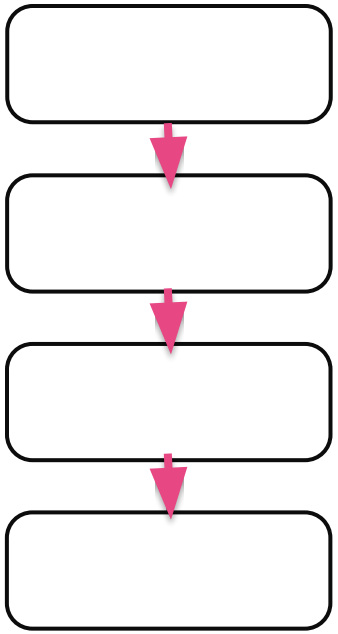
Automação

Coleta

Coletando dados

Coletamos dados de diferentes fontes e é necessário checá-las antes. Normalmente em um projeto nós temos a fonte de dados disponível, mas em outros ainda temos que buscar essa informação e checar a qualidade desse dado, como a fonte.

Exploração de Dados: Verificação dos tipos de dados das características, valores únicos e descrição dos dados.



- Coleta
- **Transformação**
- Carregamento/ Armazenamento
- Monitoramento

Automação

Transformação

Separamos limpeza e transformação em data wrangling, agora vamos por ambos no mesmo momento.

O que é?

Processando e transformando os dados de acordo com as necessidades do projeto, incluindo limpeza, normalização, agregação, etc.

Imagine que você está preparando uma receita que requer diferentes temperos para realçar o sabor do prato. A transformação de dados, nesse contexto, seria como ajustar a quantidade, a mistura e a preparação desses temperos para obter o sabor desejado.

O que é?

Remoção de Impurezas (Limpeza de Dados): Assim como você lavaria e removeria qualquer sujeira dos legumes antes de cortá-los, na limpeza de dados, você removeria valores nulos, duplicatas ou outliers que possam afetar a qualidade dos dados.

Combinação de Sabores (Feature Engineering): Misturar diferentes ervas e especiarias para criar uma mistura exclusiva que complementa o prato é como realizar feature engineering nos dados. Você está criando novas características (sabores) que agregam valor ao prato final.

Mudança na Forma (Transformação de Dados): Se a receita original pede alho picado, mas você prefere alho em pó, fazer essa substituição é uma forma de transformação de dados. Você está alterando a forma dos dados (alho) para melhor atender às suas preferências ou requisitos específicos.

Exemplo de limpeza

Processando e transformando os dados de acordo com as necessidades do projeto, incluindo limpeza, normalização, agregação, etc.

Neste exemplo: utilizamos fillna para preencher os valores nulos na coluna 'Preco' com a média dos valores existentes.Removemos outliers definindo um limite superior (nesse caso, 1000) para a coluna 'Preco'.

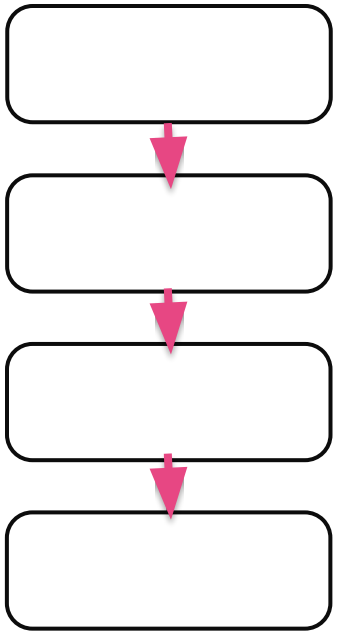
```python
import pandas as pd

# Criando um DataFrame de exemplo
data = {'Produto': ['A', 'B', 'C', 'D', 'E'],
        'Preco': [100.0, 120.0, None, 150.0, 5000.0]}  # Incluindo um valor nulo e um outlier
df = pd.DataFrame(data)

# Exibindo o DataFrame original
print("DataFrame Original:")
print(df)

# Limpeza de dados: tratando valores nulos e outliers na coluna 'Preco'
# Substituindo valores nulos pela média e removendo outliers (valores acima de 1000)
df['Preco'].fillna(df['Preco'].mean(), inplace=True)
df = df[df['Preco'] < 1000]

# Exibindo o DataFrame após a limpeza
print("\nDataFrame após Limpeza:")
print(df)
```

Exemplo de transformação

Processando e transformando os dados de acordo com as necessidades do projeto, incluindo limpeza, normalização, agregação, etc.

Neste exemplo simples, a transformação envolve a conversão da coluna **'Data'** para o formato de data, a criação de uma nova coluna **'DiaDaSemana'** e a aplicação de One-Hot Encoding a essa coluna. Essas transformações são específicas para o contexto do conjunto de dados e das análises desejadas.

```
import pandas as pd

# Supondo que 'df' seja o DataFrame com os dados de vendas
df['Data'] = pd.to_datetime(df['Data'])  # Convertendo para o formato de data

# Criando uma nova coluna com o dia da semana
df['DiaDaSemana'] = df['Data'].dt.day_name()

# One-Hot Encoding para a coluna 'DiaDaSemana'
df = pd.get_dummies(df, columns=['DiaDaSemana'])

# Exibindo as primeiras linhas do DataFrame após a transformação
print(df.head())
```



- Coleta
- Transformação
- **Carregamento/Armazenamento**
- Monitoramento

Automação

Carregamento/Armazenamento

O que é?

Armazenando os dados transformados em um local de destino, como um banco de dados, um data warehouse, ou mesmo um arquivo.

Imagine que você está preparando uma receita especial e precisa de diferentes ingredientes que estão em diferentes lugares. O carregamento de dados seria como ir até a despensa, à geladeira e à despensa novamente para pegar todos os ingredientes necessários.

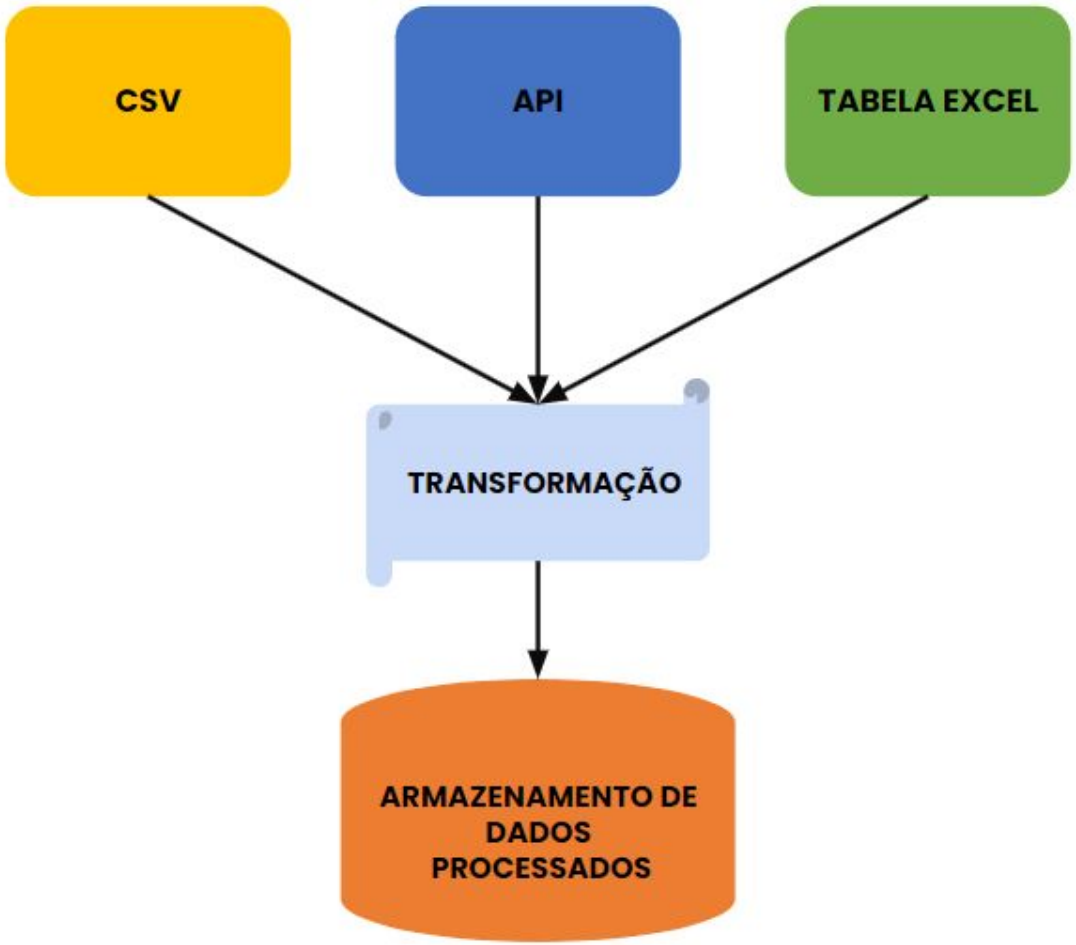
O que é?

Carregamento dos Dados: Você, como chef de dados, precisa coletar todos esses ingredientes de suas respectivas fontes. Você vai até a geladeira para pegar os ovos, à despensa para a farinha e ao armário para o açúcar.

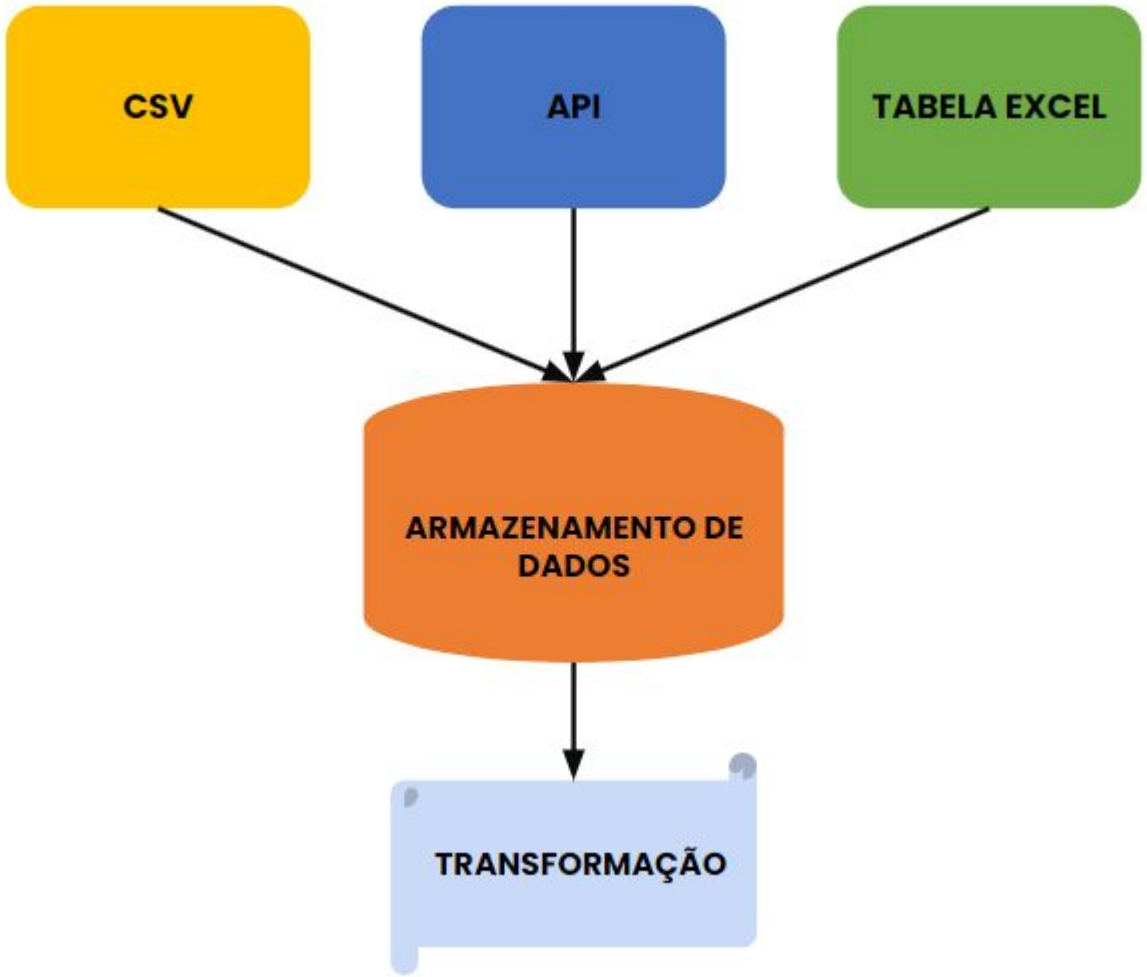
Destino (Prato Final): Depois de coletar todos os ingredientes necessários (carregamento de dados), você os reúne na bancada para começar a preparar seu prato final (resultado desejado).

Uso dos Ingredientes (Análise/Relatórios): Agora que você tem todos os ingredientes no local de preparo, você pode começar a usá-los para criar sua obra-prima culinária (análises, relatórios ou outras ações que você deseja realizar com os dados).

Carregamento

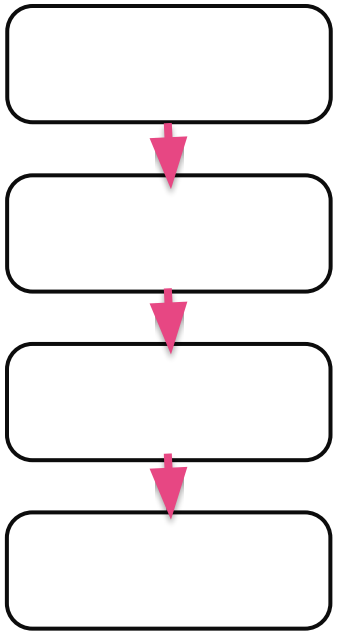


Carregamento



Carregamento

```python
import pandas as pd
from sqlalchemy import create_engine

# Suponhamos que temos dados em um arquivo CSV, em um banco de dados SQL e em uma API.

# Carregando dados de um arquivo CSV
df_csv = pd.read_csv('dados_csv.csv')

# Conectando ao banco de dados SQL (SQLite neste exemplo)
engine = create_engine('sqlite:///banco_sql.db')
query_sql = 'SELECT * FROM tabela_sql'
df_sql = pd.read_sql_query(query_sql, engine)

# Supondo uma API fictícia (você precisa da URL real da API e as credenciais, se necessário)
api_url = 'https://api.exemplo.com/dados'
df_api = pd.read_json(api_url)

# Agora, temos os dados de diferentes fontes. Vamos transformá-los.

# Transformação: Por exemplo, adicionar uma nova coluna aos DataFrames
df_csv['Nova_Coluna'] = df_csv['Coluna_A'] + df_csv['Coluna_B']
df_sql['Nova_Coluna'] = df_sql['Coluna_C'] * 2
df_api['Nova_Coluna'] = df_api['Coluna_X'] - df_api['Coluna_Y']

# Agora, vamos concatenar (ou combinar) os DataFrames em um único DataFrame

df_final = pd.concat([df_csv, df_sql, df_api], ignore_index=True)

# Por fim, vamos carregar o DataFrame final em um arquivo CSV ou em um banco de dados.

# Salvando em um arquivo CSV
df_final.to_csv('dados_final.csv', index=False)

# Salvando no mesmo banco de dados SQL
df_final.to_sql('tabela_final', engine, if_exists='replace', index=False)
```



- Coleta
- Transformação
- Carregamento/ Armazenamento
- **Monitoramento**

Automação

Monitoramento

O que é?

Monitorar o desempenho da pipeline, gerenciar falhas e lidar com situações excecionais são partes críticas da administração de uma pipeline de dados.

Imagine que você está preparando uma receita especial e precisa de diferentes ingredientes que estão em diferentes lugares. O carregamento de dados seria como ir até a despensa, à geladeira e à despensa novamente para pegar todos os ingredientes necessários.

O que é?

Tempo de Cozimento (Monitoramento de Desempenho): Ao preparar diferentes pratos, você monitora o tempo de cozimento para garantir que cada componente seja cozido corretamente. Na pipeline de dados, isso seria como monitorar o tempo que cada etapa leva para ser concluída.

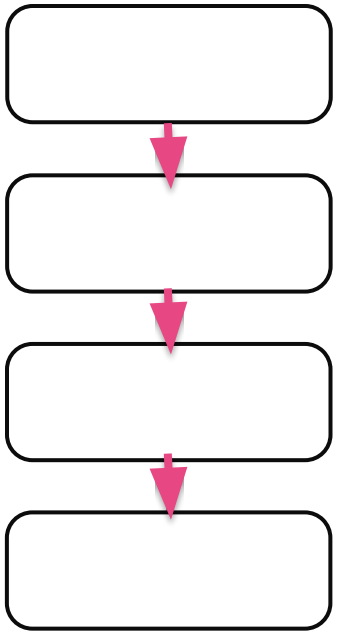
Degustação (Monitoramento de Qualidade): Você faz degustações ao longo do processo para ajustar o tempero e garantir que o sabor esteja no ponto certo. No contexto de uma pipeline de dados, seria equivalente a verificar a qualidade dos dados em cada etapa.

Controle de Inventário (Monitoramento de Recursos): Você verifica se há ingredientes suficientes e se tudo está em ordem no inventário. Da mesma forma, em uma pipeline de dados, você monitoraria o uso de recursos, como capacidade de armazenamento e poder de processamento.

Diferentes tecnologias para monitoramento de dados

- Apache Airflow
- Prometheus
- Grafana
- Datadog
- Azure Application Insights
- Python usando a biblioteca logging

```python
import pandas as pd
import logging

# Configurando o logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Função de Extração
def extract_data(file_path):
    logging.info("Iniciando extração de dados...")
    df = pd.read_csv(file_path)
    logging.info("Extração de dados concluída.")
    return df
```



- Coleta
- Transformação
- Carregamento/ Armazenamento
- Monitoramento

- **Automação**

Automação

O que é?

Automação em uma pipeline de dados refere-se à capacidade de realizar tarefas e processos de forma programada e sem intervenção manual. Isso é essencial para garantir eficiência, consistência e confiabilidade em todo o fluxo de dados. Aqui estão alguns aspectos-chave da automação em uma pipeline de dados:

- Agendamento de Tarefas
- Orquestração de Fluxo de Trabalho
- Gestão de Dependências
- Monitoramento e Notificação
- Tratamento de Erros e Retentativas
- Gerenciamento de Configuração
- Atualizações Automáticas
- Integração Contínua e Implantação Contínua (CI/CD)
- Escalonamento Automático

HORA DA PRÁTICA

WOMAKERS® CODE