Relatório Detalhado do Processo de ETL

Análise Preditiva da Pesquisa de Orçamentos Familiares (POF) 2018

Vinícius de Paula R Carvalho

12 de outubro de 2025

Sumário

Capítulo 1

Introdução

Este relatório descreve em detalhes o processo de Extração, Transformação e Carga (ETL) implementado para preparar os dados da Pesquisa de Orçamentos Familiares (POF) de 2018 para fins de análise preditiva. O objetivo central deste processo é consolidar, limpar, transformar e enriquecer os dados brutos, originários de múltiplas visões de um banco de dados PostgreSQL, em um único conjunto de dados coeso, robusto e pronto para ser utilizado em modelos de machine learning.

O processo foi encapsulado em um script Python (‘ETL.py‘) e orquestrado por um notebook Jupyter (‘01_ETL.ipynb‘), garantindo modularidade e reprodutibilidade. As etapas abordam desde a conexão com o banco de dados e extração de dados, passando por complexas operações de junção, agregação, limpeza, tratamento de valores ausentes e outliers, até a carga final em um arquivo CSV.

Capítulo 2

Arquitetura e Ferramentas

O sistema de ETL foi construído utilizando a linguagem de programação Python, com o apoio de um conjunto de bibliotecas especializadas para manipulação de dados e acesso a bancos de dados.

2.1 Bibliotecas Utilizadas

As seguintes bibliotecas foram essenciais para a execução do processo:

- **psycopg2:** Utilizada como adaptador para a comunicação e execução de consultas no banco de dados PostgreSQL.
- **pandas:** A principal ferramenta para manipulação de dados. Foi utilizada para criar DataFrames, realizar junções, agregações, limpeza e todas as transformações subsequentes.
- **numpy:** Forneceu suporte para operações numéricas e a representação de valores ausentes (‘np.nan‘).
- **math:** Utilizada especificamente para a função ‘ceil‘ no tratamento de outliers.

2.2 Ambiente de Execução

O processo é iniciado a partir de um notebook Jupyter, que configura os parâmetros de conexão com o banco de dados e invoca a função principal do ETL. Isso permite uma execução interativa e um ambiente claro para a passagem de configurações, como credenciais de banco de dados e caminhos de arquivo.

Capítulo 3

Etapa 1: Extração (Extract)

A fase de extração consiste em conectar-se ao banco de dados PostgreSQL e carregar os dados de diversas visões relacionadas à POF 2018 em DataFrames do pandas.

3.1 Conexão com o Banco de Dados

A conexão é estabelecida pela biblioteca ‘psycopg2‘ utilizando os parâmetros fornecidos (host, porta, nome do banco, usuário e senha). Cada função de leitura de dados encapsula sua própria lógica de conexão e fechamento para garantir o gerenciamento adequado dos recursos.

3.2 Fontes de Dados

Oito fontes de dados distintas, representadas como "Views"no schema ‘POF_2018‘, foram extraídas. A função principal ‘lerDados‘ orquestra a chamada a sub-funções, cada uma responsável por ler uma visão específica:

1. **View_Despesa_Coletiva:** Despesas coletivas dos domicílios.
2. **View_Despesa_Individual:** Despesas individuais dos membros do domicílio.
3. **View_Condições_Vida:** Informações sobre a percepção das condições de vida.
4. **View_Caracteristica_Dieta:** Características da dieta dos moradores. *(Nota: embora extraída, esta tabela não foi utilizada nas etapas subsequentes de transformação).*
5. **View_Caderneta_Coletiva:** Registros de despesas da caderneta coletiva.
6. **View_Aluguel_Estimado:** Contém a estimativa de aluguel para os domicílios. Durante a extração, valores sentinela (‘9999999.99‘ e ‘99999.00‘) na coluna ‘v8000‘ foram imediatamente convertidos para ‘NaN‘ (Not a Number) para representar dados ausentes.
7. **View_Rendimento_Trabalho:** Rendimentos provenientes de trabalho. As colunas de valores foram convertidas para o tipo ‘float‘ e todas as colunas foram renomeadas para nomes descritivos durante esta fase.

8. **View_Domicilio:** Tabela central com características gerais do domicílio. Serviu como base para a integração dos demais conjuntos de dados. Nesta etapa, as colunas foram renomeadas e a variável ‘Situação do Domicílio‘ foi decodificada de ‘’1‘‘/‘’2‘‘ para ‘’Urbano‘‘/‘’Rural‘‘.

Capítulo 4

Etapa 2: Transformação (Transform)

Esta é a fase mais complexa do processo, envolvendo a integração dos múltiplos DataFrames, limpeza de dados, engenharia de atributos, e tratamento de inconsistências.

4.1 Integração e Agregação de Dados

A partir do DataFrame principal, 'df_pof_domicilio', os demais DataFrames foram integrados utilizando junções do tipo 'left join' e agregações. As chaves de junção padrão foram '['cod_upa', 'num_dom']', que identificam unicamente cada domicílio.

- **Aluguel Estimado:** O valor do aluguel ('v8000') foi diretamente juntado ao DataFrame principal e renomeado para 'Aluguel Estimado'.
- **Caderneta Coletiva:** Os valores de despesa ('v8000_defla') foram primeiramente somados por domicílio ('.groupby().sum()') e, em seguida, o resultado foi juntado, criando a coluna 'Valor em reais (R)$dedespesarealizada$'.
- **Condições de Vida:** As colunas de rendimento mínimo ('v6102', 'v6103') foram agregadas pela média ('.groupby().mean()') por domicílio antes da junção.
- **Despesa Individual:** Similarmente à caderneta, os valores ('v8000_defla') foram somados por domicílio e depois integrados na coluna 'Valor em reais (R$) de despesa individual'.
- **Despesa Coletiva:** O mesmo processo de agregação por soma foi aplicado a esta tabela, gerando a coluna 'Valor em reais (R$) de despesa coletiva'.
- **Rendimentos do Trabalho:** Múltiplas colunas de rendimento e deduções foram agregadas por soma e juntadas ao DataFrame final.

Após cada operação de junção, o DataFrame correspondente era removido da memória ('del df_...') para otimizar o uso de recursos.

4.2 Conversão de Tipos de Dados

A função 'conversoes' foi responsável por garantir a consistência dos tipos de dados.

- **Colunas de Inteiros:** Colunas como ‘Qtd de cômodos‘ e ‘Qtd de banheiros‘ foram convertidas para o tipo ‘Int64‘. Este tipo de dado específico do pandas foi escolhido por sua capacidade de suportar valores ausentes (‘NaN‘), o que é crucial, pois a conversão inicial usou ‘pd.to_numeric‘ com o parâmetro ‘errors=’coerce’‘, que transforma valores não numéricos em ‘NaN‘.

- **Colunas de Ponto Flutuante:** Colunas monetárias e de rendimento foram convertidas para ‘float‘ utilizando ‘pd.to_numeric(..., errors=’coerce’)‘.

4.3 Engenharia de Atributos e Limpeza Final

A função ‘featuresEnginer‘ executou as transformações finais para preparar o dataset para modelagem.

4.3.1 Tratamento de Valores Ausentes (NaN)

Diferentes estratégias de imputação foram aplicadas:

1. **Remoção de Linhas:** Todas as linhas onde ‘Aluguel Estimado‘, ‘Valor em reais (R$) de despesa individual‘ ou ‘Valor em reais (R$) de despesa coletiva‘ eram nulos foram removidas. Esta foi uma decisão crítica que definiu o escopo final do conjunto de dados.

2. **Imputação por Constante:** A coluna ‘Qtd de banheiros de uso comum‘ teve seus valores nulos preenchidos com ‘0‘.

3. **Imputação por Mediana Agrupada:** Uma técnica mais sofisticada foi usada para ‘Valor em reais (R$) de despesa realizada‘ e ‘Valor em reais (R$) do rendimento bruto‘. Os valores ausentes foram preenchidos com a mediana de seus respectivos grupos, definidos pela coluna ‘Tipo do domicílio‘. Isso garante que a imputação seja sensível ao contexto (urbano/rural) do domicílio.

4.3.2 Criação de Novos Atributos

Uma nova coluna categórica foi criada:

- **Aluguel Estimado (Faixa):** A variável contínua ‘Aluguel Estimado‘ foi discretizada em 5 faixas (quantis) usando a função ‘pd.qcut‘. As faixas foram nomeadas de ’1 - Muito Baixo’ a ’5 - Muito Alto’, transformando uma variável numérica em uma categórica ordinal.

4.3.3 Remoção de Atributos

As colunas relacionadas a deduções de imposto de renda e previdência foram removidas (‘.drop()‘) para simplificar o modelo, focando-se nos valores brutos de rendimento e despesa.

4.3.4 Tratamento de Outliers

Um tratamento robusto de outliers foi aplicado a **todas** as colunas numéricas do DataFrame. O método utilizado foi o do Intervalo Interquartil (IQR). Para cada variável, o processo foi:

1. Calcular o primeiro quartil ($Q1$).
2. Calcular o terceiro quartil ($Q3$).
3. Calcular o Intervalo Interquartil: $IQR = Q3 - Q1$.
4. Definir um limite superior para outliers: $Limite_{Superior} = Q3 + 1.5 \times IQR$.
5. Todos os valores na coluna que excediam este $Limite_{Superior}$ foram substituídos pelo próprio valor do limite (técnica conhecida como *capping* ou *winsorizing*).

Este procedimento assegura que valores extremos não distorçam as análises ou o treinamento de modelos preditivos futuros.

```
# Loop para tratar cada vari vel
for variavel in df_pof_domicilio.select_dtypes(include=np.number).
    columns:

    # Calcular Q1 e Q3
    Q1 = df_pof_domicilio[variavel].quantile(0.25)
    Q3 = df_pof_domicilio[variavel].quantile(0.75)

    # Calcular o IQR
    IQR = Q3 - Q1

    # Calcular o Limite Superior
    limite_superior = Q3 + (1.5 * IQR)
    limite_superior = math.ceil(limite_superior)

    # Substituir outliers pelo limite superior
    df_pof_domicilio.loc[df_pof_domicilio[variavel] > limite_superior,
    variavel] = limite_superior
```

Listing 4.1: Lógica de tratamento de outliers aplicada a todas as colunas numéricas.

Capítulo 5

Etapa 3: Carga (Load)

A etapa final do processo de ETL é a persistência do DataFrame transformado. A função ‘ETL‘ encapsula todo o fluxo e, ao final, invoca o método ‘to_csv()‘ do pandas. O DataFrame resultante, limpo e processado, é salvo em um arquivo de formato CSV (‘Comma-Separated Values‘) no local especificado pelo parâmetro ‘output_path‘. A opção ‘index=False‘ é utilizada para evitar que o índice do DataFrame seja escrito como uma coluna no arquivo final.

Capítulo 6

Conclusão

O processo de ETL implementado transformou com sucesso um conjunto de dados brutos, distribuídos em múltiplas tabelas, em um único dataset analítico. As etapas de transformação foram abrangentes, incluindo a integração de oito fontes de dados, conversão rigorosa de tipos, imputação inteligente de valores ausentes, criação de um novo atributo categórico e um método robusto para tratamento de outliers em todas as variáveis numéricas.

O resultado é um arquivo CSV limpo, consistente e estruturado, que serve como uma base sólida e confiável para as próximas fases de análise exploratória de dados e desenvolvimento de modelos preditivos. Os avisos (‘SettingWithCopyWarning‘) observados durante a execução no notebook Jupyter indicam oportunidades de refatoração no código para utilizar acessores ‘.loc‘ de forma mais explícita em atribuições, garantindo que as operações modifiquem o DataFrame original de maneira inequívoca.