Data Exploration and Feature Engineering Report

1. Introduction

This report summarizes the initial phase of the healthcare fraud detection project found in the notebook `01_data_exploration_and_feature_engineering.ipynb`. The primary objective of this phase was to understand the raw data structure, perform exploratory data analysis (EDA), and transform granular claim-level and beneficiary-level data into an aggregated **provider-level dataset** suitable for machine learning models.

2. Data Loading and Setup

The analysis began by importing standard data science libraries (`pandas`, `numpy`, `matplotlib`, `seaborn`) and loading the raw datasets from the `../data` directory.

Datasets Loaded

1. **`train_labels`**: Contains the target variable `PotentialFraud` (Yes/No) for each `Provider`.
2. **`train_beneficiary`**: Demographic and health condition data for beneficiaries (e.g., DOB, Gender, Chronic Conditions).
3. **`train_inpatient`**: Claims data for hospital admissions.
4. **`train_outpatient`**: Claims data for outpatient services.

A helper function, `inspect_dataframe`, was defined and utilized to print the shape, data types, and missing value counts for each dataframe, establishing a baseline understanding of data quality.

3. Data Cleaning and Preprocessing

Significant preprocessing was required to clean the data and extract meaningful features before aggregation. This process involved dropping columns with excessive missing values and transforming existing features.

Handling Missing Values and Dropping Columns

A heatmap visualization of missing values revealed significant gaps in certain columns, leading to the following cleaning decisions:

- **Inpatient Data Drops**: Several columns in the inpatient dataset had over 80% missing data or were deemed redundant for the initial model. The following were dropped:
  - `OtherPhysician` and `OperatingPhysician`: High missingness suggested these fields were often not applicable or not recorded.
  - `ClmProcedureCode_1` through `ClmProcedureCode_6`: These procedure codes had extensive missing values.
  - `ClmDiagnosisCode_4` through `ClmDiagnosisCode_10`: Secondary and tertiary diagnosis slots were frequently empty.
- **Outpatient Data Drops**: Similar to inpatient data, the outpatient dataset required cleaning of sparse columns:
  - `ClmProcedureCode_1` through `ClmProcedureCode_6`.
  - `ClmDiagnosisCode_1` through `ClmDiagnosisCode_10` (specifically codes 3-10).
  - `OtherPhysician`, `OperatingPhysician`, and `ClmAdmitDiagnosisCode`.
- **Date Columns**: After extracting necessary duration features (see below), the original date columns `ClaimStartDt`, `ClaimEndDt`, and `DOB` were dropped to prevent leakage.
- **Beneficiary Data Cleaning**: The `DOD` (Date of Death) column was dropped after being used to create the `isDead` flag.

Feature Transformations and Engineering

- **Beneficiary Data**:
  - **Age Calculation**: A new feature, **`Age`**, was derived by calculating the time difference between `DOB` and a reference date (Dec 1, 2009).
  - **Mortality Flag**: A boolean feature, **`isDead`**, was created to indicate if a beneficiary has a recorded Date of Death.
  - **Chronic Conditions**: Binary columns for chronic conditions (e.g., Diabetes, Ischemic Heart Disease) were originally encoded as `1` (Yes) and `2` (No). The code explicitly remaps these to a standard **`0` (No) and `1` (Yes)** format to facilitate calculation of prevalence rates.
  - **Renal Disease**: The `RenalDiseaseIndicator` was converted from 'Y'/'0' string format to a binary **1/0** integer format.
- **Claims Data**:
  - **Date Standardization**: All claim-related dates (`ClaimStartDt`, `ClaimEndDt`, `AdmissionDt`, `DischargeDt`) were converted to standard datetime format.
  - **Duration**: A new feature, **`ClaimDuration`**, was calculated as the number of days between the claim start and end dates.
  - **Claim Type Flag**: A `ClaimType` column was added to distinguish between 'Inpatient' and 'Outpatient' records before merging them into a single dataset.

4. Feature Engineering and Aggregation

The core of this notebook involved aggregating the granular data up to the **Provider** level.

Merging Strategy

1. **Combine Claims**: Inpatient and outpatient claims were concatenated into a single `all_claims` DataFrame.
2. **Link Information**: This combined dataset was merged with **Provider Labels** and **Beneficiary Data** to create a unified view.

Aggregation Logic

A custom aggregation function, `aggregate_provider_data`, was implemented to group the data by `Provider` and calculate:

- **Volume Metrics**: `TotalClaims`, `UniqueBeneficiaries`, `UniquePhysicians`.
- **Financial Metrics**: Sum, Mean, and Standard Deviation of `InscClaimAmtReimbursed` and `DeductibleAmtPaid`.
- **Patient Demographics & Health**: Mean `Age`, coverage duration, and prevalence rates for all chronic conditions.

5. Test Data Processing

The notebook ensures consistency by applying the **exact same preprocessing pipeline** to the test datasets (`test_beneficiary`, `test_inpatient`, `test_outpatient`).

- Beneficiary features (Age, Chronic Conditions remapping) were generated identically.
- Claims were combined and tagged with `ClaimType`.
- The `aggregate_provider_data` function was reused to generate `test_provider_features`.

6. Conclusion and Outputs

The notebook successfully transformed the raw relational tables into a flat, provider-centric feature matrix ready for modeling.

- **Training Data**: Saved as **`train_provider_features.csv`**.
- **Test Data**: Saved as **`test_provider_features.csv`**.