2.LOSFeatures

December 5, 2025

1 Length of Stay (LOS) Feature Script

1.1 Description

This script calculates the total hospital length of stay for each admission using admission and discharge timestamps from the `admissions` table.

1.2 Clinical Justification for HAPI Research

Length of Stay is a primary cumulative risk factor for pressure injuries: * **Cumulative Exposure:** The longer a patient remains in the hospital bed, the greater the cumulative time explicitly exposed to pressure and shear forces. * **Hospital-Acquired Status:** HAPI is defined by injuries occurring *after* admission; longer stays provide a wider window for these complications to develop.

1.3 Inputs & Outputs

- **Input:** `admissions.csv`
- **Output:** `los_feat.csv`
- **Key Features:**
  - `LOS_HOURS`: Total length of stay in hours.
  - `LOS_DAYS`: Total length of stay in days (standard clinical metric).

[1]:
```
import pandas as pd
import os
```

[2]:
```
# Configuration

BASE_DIR = r"D:\School\5141"

# Path to admissions table
ADMISSIONS_PATH = os.path.join(BASE_DIR, "admissions.csv", "admissions.csv")

# Output file for LOS features
OUTPUT_PATH = os.path.join(BASE_DIR, "los_feat.csv")
```

[3]:
```
# Load Admissions Data
def load_data():
    """
    Load the admissions table and keep only fields needed to calculate
```

```
    length of stay: admission time and discharge time.
    """
    df = pd.read_csv(
        ADMISSIONS_PATH,
        usecols=["hadm_id", "admittime", "dischtime"],
        low_memory=False
    )
    df["hadm_id"] = df["hadm_id"].astype("Int64")
    return df
```

[4]:
```
# Process LOS Features
def process_data(df):
    """
    Calculate Length of Stay (LOS) for each hospital admission.
    Produces:
      - LOS_HOURS: total length of stay in hours
      - LOS_DAYS:  total length of stay in days
    """
    # Convert columns to datetime
    df["admittime"] = pd.to_datetime(df["admittime"], errors="coerce")
    df["dischtime"] = pd.to_datetime(df["dischtime"], errors="coerce")

    # LOS in hours
    df["los_hours"] = (
        df["dischtime"] - df["admittime"]
    ).dt.total_seconds() / 3600.0

    # Eliminate negative values caused by bad timestamps
    df["LOS_HOURS"] = df["los_hours"].clip(lower=0)

    # Convert to days
    df["LOS_DAYS"] = df["LOS_HOURS"] / 24.0

    # Return only the feature columns
    return df[["hadm_id", "LOS_HOURS", "LOS_DAYS"]]
```

[5]:
```
# Save
def save_data(df):
    df.to_csv(OUTPUT_PATH, index=False)
    print(f"Saved {len(df)} rows to {OUTPUT_PATH}")

# Execute
if __name__ == "__main__":
    df = load_data()
    feat = process_data(df)
    save_data(feat)
```

```
Saved 546028 rows to D:\School\5141\los_feat.csv
```