# 3.TransferFeatures

December 5, 2025

## 1 ICU Transfer Feature Script

### 1.1 Description

This script extracts ICU-related features from the MIMIC-IV `transfers` table. It identifies specific ICU stays and aggregates duration data to quantify critical care exposure.

### 1.2 Clinical Justification for HAPI Research

Patients in the ICU are at significantly higher risk for Hospital-Acquired Pressure Injuries (HAPI) due to: * **Immobility:** ICU patients are often sedated or mechanically ventilated, preventing independent repositioning. * **Device Exposure:** Critical care environments involve high usage of medical devices (tubes, lines) that create pressure points. * **Severity of Illness:** Time spent in the ICU serves as a proxy for overall patient acuity and hemodynamic instability.

### 1.3 Inputs & Outputs

- **Input:** `transfers.csv`
- **Output:** `icu_transfer.csv`
- **Key Features:**
  - `ICU_LOS_HOURS`: Total hours spent in any ICU location.
  - `NUM_ICU_TRANSFERS`: Number of distinct ICU segments (indicates care complexity).
  - `ICU_FLAG`: Binary indicator (1 = Ever in ICU, 0 = Non-ICU).

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

[2]:
```
# Congiguration

# Base directory for MIMIC-IV data files
BASE_DIR = r"D:\School\5141"

# Path to MIMIC-IV patient transfers data
TRANSFERS_PATH = os.path.join(BASE_DIR, "transfers.csv", "transfers.csv")

# Output file for transfer features
OUTPUT_PATH = os.path.join(BASE_DIR, "icu_transfer.csv")

# List of care units that are ICU locations
```

```
ICU_UNITS = [
    "Medical Intensive Care Unit (MICU)",
    "Surgical Intensive Care Unit (SICU)",
    "Cardiac Vascular Intensive Care Unit (CVICU)",
    "Neuro Surgical Intensive Care Unit (NSICU)",
    "Coronary Care Unit (CCU)"
]
```

[ ]:
```
# Load Transfer Data
def load_data():
    """
    Load the patient transfer table .
    Only keep columns needed for ICU duration and counting transfers.
    These ICU features will later be merged with HAPI labels
    as potential risk factors.
    """
    # Read in transfer data
    df = pd.read_csv(
        TRANSFERS_PATH,
        usecols=["hadm_id", "careunit", "intime", "outtime"],
        low_memory=False
    )
    df["hadm_id"] = df["hadm_id"].astype("Int64")
    return df
```

[4]:
```
# Process ICU Stays
def process_data(df):
    """
    Identify ICU stays and compute:
      - total ICU hours per admission
      - number of ICU transfers
      - a binary flag indicating whether the patient ever entered an ICU
    """

    # Convert timestamps
    df["intime"] = pd.to_datetime(df["intime"], errors="coerce")
    df["outtime"] = pd.to_datetime(df["outtime"], errors="coerce")

    # Flag rows where the patient was in an ICU unit
    df["is_icu"] = df["careunit"].isin(ICU_UNITS)

    # Keep only ICU rows
    icu_df = df[df["is_icu"]].copy()

    # If none found, return an empty table with correct columns
    if icu_df.empty:
        print("Warning: No ICU stays found.")
```

```
        return pd.DataFrame(columns=["hadm_id", "ICU_LOS_HOURS",␣
↪"NUM_ICU_TRANSFERS", "ICU_FLAG"])

    # Duration (in hours) for each ICU segment
    icu_df["segment_hours"] = (
        icu_df["outtime"] - icu_df["intime"]
    ).dt.total_seconds() / 3600.0

    # Aggregate to admission-level features
    agg = icu_df.groupby("hadm_id").agg(
        ICU_LOS_HOURS=("segment_hours", "sum"),      # Total hours in ICU
        NUM_ICU_TRANSFERS=("careunit", "count")      # Number of ICU visits/
↪segments
    ).reset_index()

    # Add binary flag: 1 = patient was ever in ICU
    agg["ICU_FLAG"] = 1

    return agg
```

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

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

```
Saved 59151 rows to D:\School\5141\icu_transfer.csv
```