6.POEfeatures

December 5, 2025

1 POE Detail (Interventions) Feature Script

1.1 Description

This script extracts detailed nursing orders and interventions from the `poe_detail` table, specifically focusing on equipment and protocols used to prevent pressure injuries.

1.2 Clinical Justification for HAPI Research

To accurately predict HAPI, the model must account for preventative measures already in place: * **Pressure Redistribution:** "Specialty mattresses" and "Heel protectors" actively reduce interface pressure. * **Shear Reduction:** "Foam dressings" (like Mepilex) are used prophylactically to reduce shear forces on the sacrum. * **Process Compliance:** "Turning schedules" (e.g., Turn q2h) are the gold standard for prevention. * **Modeling Context:** Presence of these orders often indicates the clinical team already identified the patient as "High Risk."

1.3 Inputs & Outputs

- **Inputs:** `poe.csv`, `poe_detail.csv`
- **Output:** `poe_detail_feat.csv`
- **Key Features:**
  - `has_specialty_mattress`
  - `has_turning_schedule`
  - `has_foam_dressing`

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

[2]:
```
#Configurations
BASE_DIR = r"D:\School\5141"

POE_PATH        = os.path.join(BASE_DIR, "poe.csv", "poe.csv")
POE_DETAIL_PATH = os.path.join(BASE_DIR, "poe_detail.csv", "poe_detail.csv")
OUTPUT_PATH     = os.path.join(BASE_DIR, "poe_detail_feat.csv")
```

[3]:
```
# Prevention Intervention Keywords
# These lists capture specific interventions ordered by providers to mitigate␣
 ↪HAPI risk.
```

```
# Special Mattress (Pressure Redistribution)
# Reduces pressure on bony prominences to prevent pressure ulcers.
SPECIALTY_MATTRESS_KW = [
    "air mattress", "low air loss", "specialty mattress",
    "pressure relieving mattress", "p500", "low-air-loss"
]

# HEEL PROTECTION
# The heel is the second most common site for Deep Tissue Injury (DTI).
# These devices "float" the heel to remove all pressure.
HEEL_PROTECTOR_KW = [
    "heel protector", "heel boot", "heel boots",
    "heel suspension", "heel offloading"
]

# Prophylactic Sacral Foam Dressings
# Foam dressings (e.g., Mepilex) absorb moisture and reduce shear forces on the␣
 ↪sacrum.
FOAM_DRESSING_KW = [
    "foam dressing", "mepilex", "sacral foam",
    "bordered foam", "foam sacral"
]

# Advanced Wound Care
# Negative Pressure Wound Therapy indicates an existing, severe wound is likely␣
 ↪present.
WOUND_VAC_KW = [
    "wound vac", "vac dressing", "negative pressure wound",
    "npwt", "vac therapy"
]

# Repositioning/Turning Schedule
# "Turn every 2 hours" is the standard of care for immobility.
TURNING_SCHEDULE_KW = [
    "turn q2", "turn q3", "turn q4",
    "reposition q2", "reposition q3", "reposition q4",
    "turn every 2", "turn every 3", "turn every 4"
]
```

[4]:
```
#Loader Functions
def load_poe(path: str):
    """
    Load poe.csv with only the columns needed for mapping:
      - poe_id
      - hadm_id
    """
    df = pd.read_csv(path, low_memory=False)

    # Keep just mapping columns if present
    keep_cols = [c for c in ["poe_id", "hadm_id"] if c in df.columns]
    df = df[keep_cols].copy()

    if "hadm_id" in df.columns:
        df["hadm_id"] = df["hadm_id"].astype("Int64")

    return df


def load_poe_detail(path: str):
    """
    Load poe_detail.csv and create a unified lowercase text column detail_text␣
  ↪from field_name & field_value.
    Poe_detail has columns: combine the last two
        ['poe_id', 'poe_seq', 'subject_id', 'field_name', 'field_value']
    """

    df = pd.read_csv(path, low_memory=False)

    required_cols = ["poe_id", "field_name", "field_value"]
    for c in required_cols:
        if c not in df.columns:
            raise ValueError(
                f"Expected column '{c}' in poe_detail, but it is missing. "
                f"Columns: {list(df.columns)}"
            )

    # Combine field_name and field_value into one searchable text field
    df["detail_text"] = (
        df["field_name"].astype(str) + " " + df["field_value"].astype(str)
    ).str.lower()

    return df
```

[5]:
```
#Build Features Function
def build_poe_detail_features(poe: pd.DataFrame, detail: pd.DataFrame) :
    """
    Join poe_detail to poe on poe_id to get hadm_id.
    Then aggregate detail-level signals to admission-level features.
    """
    if "poe_id" not in detail.columns or "poe_id" not in poe.columns:
        raise ValueError("Both poe and poe_detail must have 'poe_id' to merge.")

    merged = detail.merge(
        poe,
        on="poe_id",
        how="left",
        validate="m:1"  # many detail rows per one poe row
    )

    # Drop rows where we couldn't find hadm_id
    merged = merged[merged["hadm_id"].notna()].copy()
    merged["hadm_id"] = merged["hadm_id"].astype("Int64")


    # Total number of detail items per admission
    num_detail = (
        merged.groupby("hadm_id")["poe_id"]
        .size()  # count rows; each = detail item
        .rename("num_detail_items")
    )

    # Helper to create flag by keyword list
    def flag_by_keywords(keywords, col_name):
        if not keywords:
            return pd.Series(dtype="Int64", name=col_name)
        mask = merged["detail_text"].str.contains("|".join(keywords), na=False)
        flagged = (
            merged[mask]
            .groupby("hadm_id").size()
            .gt(0)
            .astype("Int64")
            .rename(col_name)
        )
        return flagged

    # Flag specialty mattress orders
    spec_mattress_flag = flag_by_keywords(
        SPECIALTY_MATTRESS_KW, "has_specialty_mattress"
    )

    # Flag heel protectors
    heel_flag = flag_by_keywords(
        HEEL_PROTECTOR_KW, "has_heel_protector"
    )

    # Flag foam dressings
    foam_flag = flag_by_keywords(
        FOAM_DRESSING_KW, "has_foam_dressing"
    )
```

```
    # Flag wound VAC / NPWT
    vac_flag = flag_by_keywords(
        WOUND_VAC_KW, "has_wound_vac"
    )

    # Flag turning / repositioning schedule
    turn_flag = flag_by_keywords(
        TURNING_SCHEDULE_KW, "has_turning_schedule"
    )

    # Combine all features
    feat = (
        num_detail.to_frame()
        .join(spec_mattress_flag, how="left")
        .join(heel_flag, how="left")
        .join(foam_flag, how="left")
        .join(vac_flag, how="left")
        .join(turn_flag, how="left")
        .reset_index()
        .fillna(0)
    )

    # Ensure Int64 dtypes
    feat["num_detail_items"] = feat["num_detail_items"].astype("Int64")
    for col in [
        "has_specialty_mattress",
        "has_heel_protector",
        "has_foam_dressing",
        "has_wound_vac",
        "has_turning_schedule",
    ]:
        feat[col] = feat[col].astype("Int64")

    print("POE_DETAIL features built:")
    print(feat.head())

    return feat
```

[6]:
```
#Execute
if __name__ == "__main__":
    poe = load_poe(POE_PATH)
    poe_detail = load_poe_detail(POE_DETAIL_PATH)

    feat = build_poe_detail_features(poe, poe_detail)

    print(f"Saving to: {OUTPUT_PATH}")
    feat.to_csv(OUTPUT_PATH, index=False)
    print("Done.")
```

```
POE_DETAIL features built:
    hadm_id  num_detail_items  has_specialty_mattress  has_heel_protector  \
0  20000019                 8                       0                   0
1  20000024                13                       0                   0
2  20000034                 5                       0                   0
3  20000041                 6                       0                   0
4  20000045                88                       0                   0

   has_foam_dressing  has_wound_vac  has_turning_schedule
0                  0              0                     0
1                  0              0                     0
2                  0              0                     0
3                  0              0                     0
4                  0              0                     0
Saving to: D:\School\5141\poe_detail_feat.csv
Done.
```