Machine Learning-Derived Temperature Trajectories Identify Coagulation Phenotypes in Sepsis

1.preprocess_temperature_data

```
def preprocess_temperature_data(df):
    # 标准化：以每个患者前3次体温测量的均值为基线
    df['temp_normalized'] = df.groupby('patient_id')['temperature'].transform(
        lambda x: x / x.head(3).mean()
    )
    # 立方样条插值处理缺失值
    df['temp_interpolated'] = df.groupby('patient_id')['temp_normalized'].transform(
        lambda x: x.interpolate(method='cubic')
    )
```

2.cluster_temperature_trajectories)

```
def cluster_temperature_trajectories(trajectories):
    # DTW距离矩阵计算
    for i in range(n):
        alignment = dtw(trajectories[i], trajectories[j])
        dtw_matrix[i,j] = alignment.distance

    # GMM聚类 + BIC选择最优簇数
    gmm = GaussianMixture(n_components=optimal_clusters)
    labels = gmm.fit_predict(dtw_matrix)
```

3.analyze_coagulation_by_phenotype

```
results[phenotype] = {
    'platelet_count': subset['platelet'].mean(),  # 血小板计数
    'pt_ratio': subset['pt_ratio'].mean(),      # 凝血酶原时间比值
    'd_dimer': subset['d_dimer'].mean(),        # D-二聚体水平
    'mortality_rate': subset['mortality'].mean() # 死亡率
```

Full data:

```
import numpy as np
import pandas as pd
from sklearn.mixture import GaussianMixture
from dtw import dtw
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler

def preprocess_temperature_data(df):
    """Preprocess temperature data as described in Methods section"""
    # Normalize to baseline (mean of first 3 measurements)
    df['temp_normalized'] = df.groupby('patient_id')['temperature'].transform(
        lambda x: x / x.head(3).mean()
    )

    # Handle missing data with cubic spline interpolation
    df['temp_interpolated'] = df.groupby('patient_id')['temp_normalized'].transform(
        lambda x: x.interpolate(method='cubic')
    )

    return df

def cluster_temperature_trajectories(trajectories, max_clusters=5):
    """Cluster temperature trajectories using DTW and GMM"""
    # Calculate DTW distance matrix
    n = len(trajectories)
    dtw_matrix = np.zeros((n, n))

    for i in range(n):
        for j in range(i+1, n):
            alignment = dtw(trajectories[i], trajectories[j])
            dtw_matrix[i,j] = alignment.distance
```

```
            dtw_matrix[j,i] = alignment.distance

    # Find optimal cluster number using BIC
    bic_scores = []
    silhouette_scores = []

    for n_clusters in range(2, max_clusters+1):
        gmm = GaussianMixture(n_clusters=n_clusters)
        labels = gmm.fit_predict(dtw_matrix)

        bic_scores.append(gmm.bic(dtw_matrix))
        silhouette_scores.append(silhouette_score(dtw_matrix, labels))

    optimal_clusters = np.argmin(bic_scores) + 2  # +2 because range starts at 2

    # Final clustering with optimal clusters
    gmm = GaussianMixture(n_components=optimal_clusters)
    cluster_labels = gmm.fit_predict(dtw_matrix)

    return cluster_labels, optimal_clusters, silhouette_scores[optimal_clusters-2]

def analyze_coagulation_by_phenotype(data, labels):
    """Analyze coagulation parameters by phenotype"""
    results = {}

    for phenotype in np.unique(labels):
        subset = data[labels == phenotype]

        results[phenotype] = {
            'platelet_count': subset['platelet'].mean(),
            'platelet_slope': subset['platelet_slope'].mean(),
            'pt_ratio': subset['pt_ratio'].mean(),
```

```
            'd_dimer': subset['d_dimer'].mean(),
            'mortality_rate': subset['mortality'].mean()
        }

    return pd.DataFrame.from_dict(results, orient='index')

# Example usage:
if __name__ == "__main__":
    # Load sample data (replace with actual data loading)
    data = pd.read_csv('sepsis_data.csv')

    # Preprocess temperature data
    processed_data = preprocess_temperature_data(data)

    # Extract trajectories (assuming data is in long format)
    trajectories = processed_data.groupby('patient_id')['temp_interpolated'].apply(np.array).values

    # Cluster trajectories
    labels, n_clusters, silhouette = cluster_temperature_trajectories(trajectories)
    print(f"Identified {n_clusters} phenotypes with silhouette score {silhouette:.2f}")

    # Analyze coagulation parameters by phenotype
    phenotype_analysis = analyze_coagulation_by_phenotype(data, labels)
    print(phenotype_analysis)
```