Lucas_Cell_Line_Alignment

May 13, 2025

[5]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.metrics.pairwise import euclidean_distances
import warnings
warnings.filterwarnings('ignore')

def impute_missing_values(df, column):
    value_dist = df[column].value_counts(normalize=True)
    missing_indices = df[column].isna()

    if missing_indices.any():
        imputed_values = np.random.choice(
            value_dist.index,
            size=missing_indices.sum(),
            p=value_dist.values
        )
        df.loc[missing_indices, column] = imputed_values


    return df

def preprocess(merged_df, harvard_df):

    merged = merged_df.copy()
    harvard = harvard_df.copy()

    if 'Age' in merged.columns:
        merged['Age'] = merged['Age'].fillna(merged['Age'].mean())
    if 'Age' in harvard.columns:
        harvard['Age'] = harvard['Age'].fillna(harvard['Age'].mean())

    harvard = harvard.rename(columns={
        'Age': 'Donor Age',
        'Race': 'Donor Ethnicity'
    })
```

```
    for col in ['Donor Ethnicity', 'T Stage']:
        if col in merged.columns:
            merged[col] = merged[col].fillna(merged[col].mode()[0] if not␣
 ↪merged[col].mode().empty else 'Unknown')
        if col in harvard.columns:
            harvard[col] = harvard[col].fillna(harvard[col].mode()[0] if not␣
 ↪harvard[col].mode().empty else 'Unknown')
    numerical_features = []
    if 'Age' in merged.columns:
        numerical_features.append('Age')
        harvard['Age'] = harvard['Donor Age']

    categorical_features = []
    if all(col in merged.columns and col in harvard.columns for col in ['Donor␣
 ↪Ethnicity', 'T Stage']):
        categorical_features.extend(['Donor Ethnicity', 'T Stage'])

    if not numerical_features and not categorical_features:
        raise ValueError("No valid features found for preprocessing")

        # Convert T Stage to string type to handle mixed numeric/string values
    if 'T Stage' in categorical_features:
        merged['T Stage'] = merged['T Stage'].astype(str)
        harvard['T Stage'] = harvard['T Stage'].astype(str)

    preprocessor = ColumnTransformer(
        transformers=[
            ('num', StandardScaler(), numerical_features),
            ('cat', OneHotEncoder(handle_unknown='ignore',␣
 ↪sparse_output=False), categorical_features)
        ])

    merged_processed = preprocessor.fit_transform(merged)
    harvard_processed = preprocessor.transform(harvard)

        # Clean up temporary column
    if 'Age' in harvard.columns:
        harvard.drop('Age', axis=1, inplace=True)

    return merged_processed, harvard_processed


def similarity(merged_processed, harvard_processed):
    if hasattr(merged_processed, 'toarray'):
        merged_processed = merged_processed.toarray()
    if hasattr(harvard_processed, 'toarray'):
```

```
        harvard_processed = harvard_processed.toarray()

    merged_processed = np.nan_to_num(merged_processed, nan=0.0)
    harvard_processed = np.nan_to_num(harvard_processed, nan=0.0)

    distances = euclidean_distances(merged_processed, harvard_processed)
    similarity_matrix = 1 / (1 + distances)

    return similarity_matrix

def best_matches(similarity_matrix, merged_df, harvard_df, top_n=3):
    results = []

    for i, patient_id in enumerate(merged_df['Patient ID']):
        top_indices = np.argsort(similarity_matrix[i])[-top_n:][::-1]

        matches = []
        for idx in top_indices:
            cell_line = harvard_df.iloc[idx]['HMS LINCS ID']
            similarity = similarity_matrix[i][idx]
            matches.append({
                'cell_line': cell_line,
                'similarity_score': similarity,
                'age': harvard_df.iloc[idx]['Donor Age'],
                'race': harvard_df.iloc[idx]['Donor Ethnicity'],
                't_stage': harvard_df.iloc[idx]['T Stage']
            })

        results.append({
            'patient_id': patient_id,
            'matches': matches
        })

        return results

def cellline_alignment():

    merged_df = pd.read_csv('final_merged.csv')
    harvard_df = pd.read_csv('data/HarvardCellLines.csv')

    merged_processed, harvard_processed = preprocess(merged_df, harvard_df)
    similarity_matrix = similarity(merged_processed, harvard_processed)
    results = best_matches(similarity_matrix, merged_df, harvard_df)

    output_data = []
    for result in results:
        for match in result['matches']:
```

```
            output_data.append({
                'Patient_ID': result['patient_id'],
                'Cell_Line': match['cell_line'],
                'Similarity_Score': match['similarity_score'],
                'Cell_Line_Age': match['age'],
                'Cell_Line_Race': match['race'],
                'Cell_Line_T_Stage': match['t_stage']
            })

    output_df = pd.DataFrame(output_data)
    output_df.to_csv('cell_line_matches.csv', index=False)

cellline_alignment()
```

[ ]: