Turmerik
ML/SWE Take Home Assignment

Clinical Trial Eligibility Matching Algorithm: Step-by-Step Implementation Documentation

Table of Contents

1. Project Setup

1.1 Environment Setup

Install required libraries using the following command:

- pip install pandas requests beautifulsoup4 openpyxl

1.2 Directory Structure

Create the following directory structure to keep your files organized:

```
/ClinicalTrialMatching
    ├── data/
    ├── output/
    ├── src/
    ├── README.md
    └── requirements.txt
```

1.3 Download Data

Download the Synthea sample data from https://syntheticahealth.github.io/synthea-sample-data/downloads/latest/synthea_sample_data_csv_latest.zip and place the extracted CSV files in the data/ directory.

---

2. Load Patient Data

2.1 Import Libraries

In Turmerik.ipynb, import the necessary libraries:

```
[203]: import pandas as pd
       import zipfile
       import requests
       import os
```

2.2 Load Patient CSV

Load the patient data CSV into a Pandas DataFrame:

Download the zip file of data from URL: https://syntheticahealth.github.io/synthea-sample-data/downloads/latest/synthea_sample_data_csv_latest.zip and these are data of patients with their information such as age, sex, medication and etc.

[205]:
```
# Download and extract patient data
url = "https://synthetichealth.github.io/synthea-sample-data/downloads/latest/synthea_sample_data_csv_latest.zip"
response = requests.get(url)

with open("synthea_sample_data.zip", "wb") as f:
    f.write(response.content)

with zipfile.ZipFile("synthea_sample_data.zip", 'r') as zip_ref:
    zip_ref.extractall("synthea_data")

#load patient data
patient_data = pd.read_csv("synthea_data/patients.csv")
```

---

3. Calculate Patient Age

3.1 Define Age Calculation Function

Create a function to calculate age based on the birthdate:

[207]:
```
# Function to calculate age from birthdate
from datetime import datetime
def calculate_age(birthdate):
    if pd.isna(birthdate):
        return None
    birth_date = pd.to_datetime(birthdate)
    today = datetime.today()
    return today.year - birth_date.year - ((today.month, today.day) < (birth_date.month, birth_date.day))
```

This function checks if the birthdate is valid, converts it to a datetime object, and calculates the age by comparing it to the current date.

3.2 Apply Age Calculation

Add an AGE column to the DataFrame:

[209]:
```
# Add an AGE column to the patient DataFrame
patient_data['AGE'] = patient_data['BIRTHDATE'].apply(calculate_age)
```

This line applies the calculate_age function to each birthdate in the DataFrame, creating a new column for ages.

---

4. Load Additional Data

4.1 Load Conditions, Medications, Observations

Load other relevant CSV files:

[211]:
```
# Load other CSVs
conditions_data = pd.read_csv('synthea_data/conditions.csv')
medications_data = pd.read_csv('synthea_data/medications.csv')
observations_data = pd.read_csv('synthea_data/observations.csv')
```

These lines read in the conditions, medications, and observations data from their respective CSV files.

4.2 Load Clinical Trials Data

Load and normalize the clinical trials data:

[15]:
```
import pandas as pd

# Load trial data from Google Sheets as a CSV
trials_data = pd.read_csv('https://docs.google.com/spreadsheets/d/1ntxXSP5uOoOAlgf4N6SmLbPdxmaCAQZfuEZWJ0ULS8o/export?format=csv')
trials_data = trials_data.rename(columns={
    'NCT Number': 'nct_number',
    'Study Title': 'study_title',
    'Age': 'age',
    'Conditions': 'conditions'
})
```

The trial data is read from a CSV file which is take from google link (data is scraped from Clinical.gov, and columns are renamed for consistency.

---

5. Create Patient Profiles

5.1 Initialize List for Profiles

Create a list to store patient profiles:

[213]:
```
# Create a list to store patient profiles
#Create Patient Profiles: Extract relevant attributes into a structured format
patient_profiles = []
```

This initializes an empty list that will hold the profiles for each patient.

5.2 Iterate Through Patients

Extract relevant attributes for each patient:

Extract and integrate all attributes from different csv into together a single list patient_profiles list as shown below.

[14]:
```python
# Create a list to store patient profiles
#Create Patient Profiles: Extract relevant attributes into a structured format
patient_profiles = []

# Iterate through each patient
for _, patient in patient_data.iterrows():
    patient_id = patient['Id']

    # Get patient attributes
    profile = {
        'patientId': patient_id,
        'age': patient['AGE'],
        'gender': patient['GENDER'],
        'conditions': conditions_data[conditions_data['PATIENT'] == patient_id]['DESCRIPTION'].tolist(),
        'labResults': observations_data[observations_data['PATIENT'] == patient_id][['DESCRIPTION', 'VALUE', 'UNITS']].dropna().values.tolist(),
        'medications': medications_data[medications_data['PATIENT'] == patient_id][['DESCRIPTION', 'REASONDESCRIPTION']].dropna().values.tolist()
    }

    patient_profiles.append(profile)
```

This loop creates a profile for each patient, including their ID, age, gender, conditions, lab results, and medications, and appends it to the patient_profiles list.

---

6. Check Eligibility for Trials

6.1 Define Eligibility Check Function

Create a function to check eligibility based on age and conditions:

[36]:
```python
# Define a function to check eligibility for a trial
def check_eligibility(patient, trial):
    criteria_met = []

    # Check age criteria
    if isinstance(trial['age'], str):
        age_range = trial['age'].split(' to ')
        min_age = int(age_range[0]) if age_range[0].isdigit() else None
        max_age = int(age_range[1]) if len(age_range) > 1 and age_range[1].isdigit() else None

        if (min_age is None or patient['age'] >= min_age) and (max_age is None or patient['age'] <= max_age):
            criteria_met.append("Inclusion Criteria")

    # Check conditions
    trial_conditions = trial['conditions'].split(',') if isinstance(trial['conditions'], str) else []
    if not any(condition in trial_conditions for condition in patient['conditions']):
        criteria_met.append("Exclusion Criteria")

    return criteria_met
```

This function checks if a patient meets the age and condition requirements for a trial, returning a list of criteria that were met.

Inclusion criteria: Compare patient age against trial age range

Exclusion criteria: Ensure patient doesn't have any excluded conditions or medications

---

7. Match Patients to Trials

7.1 Initialize Results List

[42]:
```
#Initialize Results List
results= [ ]
```

Create a list to store the eligibility results:

This initializes an empty list that will hold the results of eligible trials for each patient.

7.2 Iterate Through Patients and Trials

Check each patient's eligibility for each trial:

[19]:
```
def match_patients_to_trials(patient_profiles, trials_data):
    results = []
    for patient in patient_profiles:
        patient_results = {"patientId": patient['patientId'], "eligibleTrials": []}

        for _, trial in trials_data.iterrows():
            eligibility = check_eligibility(patient, trial)
            if eligibility:
                trial_data = {
                    "trialId": trial['nct_number'],
                    "trialName": trial['study_title'],
                    "eligibilityCriteriaMet": eligibility
                }
                patient_results["eligibleTrials"].append(trial_data)

        results.append(patient_results)
    return results

# Assuming trials_data is already loaded and formatted
results = match_patients_to_trials(patient_profiles, trials_data)
```

This nested loop checks each patient's eligibility against all trials and stores the results, including trial ID, name, and met criteria.

---

8. Output Results

8.1 Save to JSON File

Write the results to a JSON file:

[215]:
```
# Save results to JSON file
import json

with open(r'D:\Project\Turmerik\eligible_trials.json', 'w') as json_file:
    json.dump(results, json_file, indent=4)
```

This code saves the eligibility results to a JSON file for easy sharing and access.

8.2 Save to Excel File

Convert results into a DataFrame and save as Excel:

```
[217]: # Save results to Excel file
       results_df = pd.DataFrame(results)
       results_df.to_excel(r'D:\Project\Turmerik\eligible_trials.xlsx', index=False)

       print("Eligibility check complete and results saved.")
```

This code saves the eligibility results to a excel file for easy sharing and access.

9. Testing and Validation

Implemented tests to validate individual functions.

Tested the complete flow using sample data to ensure the algorithm works as expected.

---

10. Documentation and Cleanup

10.1 Write Comments

Added comments throughout the code to clarify complex sections and explain the code part wherever required in the file.

10.2 Create README File

Drafted a README file including:

- Overview of the project
- Instructions for setup and usage
- Example outputs

---

11. Conclusion

This document outlines the implementation of a clinical trial eligibility matching algorithm. By following these steps, you can create a system to determine patient eligibility for clinical trials based on their characteristics.