Annex A: Code for AI-Assisted Data Extraction and Analysis

import pandas as pd

import numpy as np

from bs4 import BeautifulSoup

import requests

import re

import spacy

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.metrics import f1\_score

# Web scraping and data preprocessing

def scrape\_and\_preprocess(url):

response = requests.get(url)

soup = BeautifulSoup(response.content, 'html.parser')

text = soup.get\_text()

text = re.sub(r'\s+', ' ', text) # Remove extra whitespace

text = text.lower() # Convert to lowercase

return text

# Named Entity Recognition

nlp = spacy.load("en\_core\_web\_sm")

def perform\_ner(text):

doc = nlp(text)

entities = [(ent.text, ent.label\_) for ent in doc.ents]

return entities

# Custom keyword-based classification

policy\_keywords = ['social assistance', 'cash transfer', 'labor market', 'unemployment benefit']

def keyword\_classification(text):

return [keyword for keyword in policy\_keywords if keyword in text]

# Machine Learning Classification

vectorizer = TfidfVectorizer(max\_features=5000)

def prepare\_data(texts, labels):

X = vectorizer.fit\_transform(texts)

return train\_test\_split(X, labels, test\_size=0.2, random\_state=42)

def train\_and\_evaluate\_models(X\_train, X\_test, y\_train, y\_test):

nb\_model = MultinomialNB()

svm\_model = SVC(kernel='linear')

nb\_model.fit(X\_train, y\_train)

svm\_model.fit(X\_train, y\_train)

nb\_pred = nb\_model.predict(X\_test)

svm\_pred = svm\_model.predict(X\_test)

nb\_f1 = f1\_score(y\_test, nb\_pred, average='weighted')

svm\_f1 = f1\_score(y\_test, svm\_pred, average='weighted')

print(f"Naive Bayes F1 Score: {nb\_f1}")

print(f"SVM F1 Score: {svm\_f1}")

# Perform cross-validation

nb\_cv\_scores = cross\_val\_score(nb\_model, X\_train, y\_train, cv=5)

svm\_cv\_scores = cross\_val\_score(svm\_model, X\_train, y\_train, cv=5)

print(f"Naive Bayes CV Scores: {nb\_cv\_scores.mean()}")

print(f"SVM CV Scores: {svm\_cv\_scores.mean()}")

return nb\_model, svm\_model

# Main execution

if \_\_name\_\_ == "\_\_main\_\_":

# Example usage

url = "https://example.com/social\_protection\_policy"

text = scrape\_and\_preprocess(url)

entities = perform\_ner(text)

print("Named Entities:", entities)

keywords = keyword\_classification(text)

print("Keywords found:", keywords)

# Assuming we have a dataset of texts and labels

texts = ["example text 1", "example text 2", ...]

labels = [0, 1, ...]

X\_train, X\_test, y\_train, y\_test = prepare\_data(texts, labels)

nb\_model, svm\_model = train\_and\_evaluate\_models(X\_train, X\_test, y\_train, y\_test)