import streamlit as st  
import helper  
import pickle  
  
model = pickle.load(open('model.pkl', 'rb'))  
  
st.header('duplicate question Pairs')  
  
q1 = st.text\_input('Enter question 1')  
q2 = st.text\_input('Enter question 2')  
  
if st.button('Find'):  
 query = helper.query\_point\_creator(q1, q2)  
 result = model.predict(query)[0]  
  
 if result:  
 st.header('Duplicate')  
 else:  
 st.header('Not Duplicate')

above file is main.py file

import re  
from bs4 import BeautifulSoup  
import distance  
from fuzzywuzzy import fuzz  
import pickle  
import numpy as np  
  
cv = pickle.load(open('cv.pkl','rb'))  
  
  
def test\_common\_words(q1,q2):  
 w1 = set(map(lambda word: word.lower().strip(), q1.split(" ")))  
 w2 = set(map(lambda word: word.lower().strip(), q2.split(" ")))  
 return len(w1 & w2)  
  
def test\_total\_words(q1,q2):  
 w1 = set(map(lambda word: word.lower().strip(), q1.split(" ")))  
 w2 = set(map(lambda word: word.lower().strip(), q2.split(" ")))  
 return (len(w1) + len(w2))  
  
  
def test\_fetch\_token\_features(q1, q2):  
 SAFE\_DIV = 0.0001  
  
 STOP\_WORDS = pickle.load(open('stopwords.pkl','rb'))  
  
 token\_features = [0.0] \* 8  
  
 # Converting the Sentence into Tokens:  
 q1\_tokens = q1.split()  
 q2\_tokens = q2.split()  
  
 if len(q1\_tokens) == 0 or len(q2\_tokens) == 0:  
 return token\_features  
  
 # Get the non-stopwords in Questions  
 q1\_words = set([word for word in q1\_tokens if word not in STOP\_WORDS])  
 q2\_words = set([word for word in q2\_tokens if word not in STOP\_WORDS])  
  
 # Get the stopwords in Questions  
 q1\_stops = set([word for word in q1\_tokens if word in STOP\_WORDS])  
 q2\_stops = set([word for word in q2\_tokens if word in STOP\_WORDS])  
  
 # Get the common non-stopwords from Question pair  
 common\_word\_count = len(q1\_words.intersection(q2\_words))  
  
 # Get the common stopwords from Question pair  
 common\_stop\_count = len(q1\_stops.intersection(q2\_stops))  
  
 # Get the common Tokens from Question pair  
 common\_token\_count = len(set(q1\_tokens).intersection(set(q2\_tokens)))  
  
 token\_features[0] = common\_word\_count / (min(len(q1\_words), len(q2\_words)) + SAFE\_DIV)  
 token\_features[1] = common\_word\_count / (max(len(q1\_words), len(q2\_words)) + SAFE\_DIV)  
 token\_features[2] = common\_stop\_count / (min(len(q1\_stops), len(q2\_stops)) + SAFE\_DIV)  
 token\_features[3] = common\_stop\_count / (max(len(q1\_stops), len(q2\_stops)) + SAFE\_DIV)  
 token\_features[4] = common\_token\_count / (min(len(q1\_tokens), len(q2\_tokens)) + SAFE\_DIV)  
 token\_features[5] = common\_token\_count / (max(len(q1\_tokens), len(q2\_tokens)) + SAFE\_DIV)  
  
 # Last word of both question is same or not  
 token\_features[6] = int(q1\_tokens[-1] == q2\_tokens[-1])  
  
 # First word of both question is same or not  
 token\_features[7] = int(q1\_tokens[0] == q2\_tokens[0])  
  
 return token\_features  
  
  
def test\_fetch\_length\_features(q1, q2):  
 length\_features = [0.0] \* 3  
  
 # Converting the Sentence into Tokens:  
 q1\_tokens = q1.split()  
 q2\_tokens = q2.split()  
  
 if len(q1\_tokens) == 0 or len(q2\_tokens) == 0:  
 return length\_features  
  
 # Absolute length features  
 length\_features[0] = abs(len(q1\_tokens) - len(q2\_tokens))  
  
 # Average Token Length of both Questions  
 length\_features[1] = (len(q1\_tokens) + len(q2\_tokens)) / 2  
  
 strs = list(distance.lcsubstrings(q1, q2))  
 length\_features[2] = len(strs[0]) / (min(len(q1), len(q2)) + 1)  
  
 return length\_features  
  
  
def test\_fetch\_fuzzy\_features(q1, q2):  
 fuzzy\_features = [0.0] \* 4  
  
 # fuzz\_ratio  
 fuzzy\_features[0] = fuzz.QRatio(q1, q2)  
  
 # fuzz\_partial\_ratio  
 fuzzy\_features[1] = fuzz.partial\_ratio(q1, q2)  
  
 # token\_sort\_ratio  
 fuzzy\_features[2] = fuzz.token\_sort\_ratio(q1, q2)  
  
 # token\_set\_ratio  
 fuzzy\_features[3] = fuzz.token\_set\_ratio(q1, q2)  
  
 return fuzzy\_features  
  
  
def preprocess(q):  
 q = str(q).lower().strip()  
  
 # Replace certain special characters with their string equivalents  
 q = q.replace('%', ' percent')  
 q = q.replace('$', ' dollar ')  
 q = q.replace('₹', ' rupee ')  
 q = q.replace('€', ' euro ')  
 q = q.replace('@', ' at ')  
  
 # The pattern '[math]' appears around 900 times in the whole dataset.  
 q = q.replace('[math]', '')  
  
 # Replacing some numbers with string equivale…

Above flie is helper.py file

For developing website I have used pycharm .

Here I have used streamlit to created website.