# Department of Computing

**CS 471: Machine Learning**

**BESE: 11A**

**Lab 12: Deploying ML model using Flask**

**Date: 5th May, 2023**

**Time: 02:00 PM-05:00 PM**

**Instructor: Dr. Seemab Latif**

Name: Raja Haider Ali

CMS ID: 346900

**Lab 12: Deploying ML model using Flask**

**Introduction:**

Flask is a popular Python web framework that allows developers to create web applications easily. It provides a simple and flexible way to build APIs, and is often used for deploying machine learning models as APIs. In this lab, we will learn how to deploy a trained machine learning model as a web API using Flask and later deploy on our own model.

**Objectives:**

The main objectives of this lab are:

1. To learn how to deploy a machine learning model using Flask.
2. To learn how to create a web API using Flask.
3. To learn how to send data to the API and receive predictions.

**Tools:**

Python 3.x, Flask, scikit-learn, NumPy, Pandas, Tensorflow, Anaconda, Jupyter Notebook or Google Colab.

**Description:**

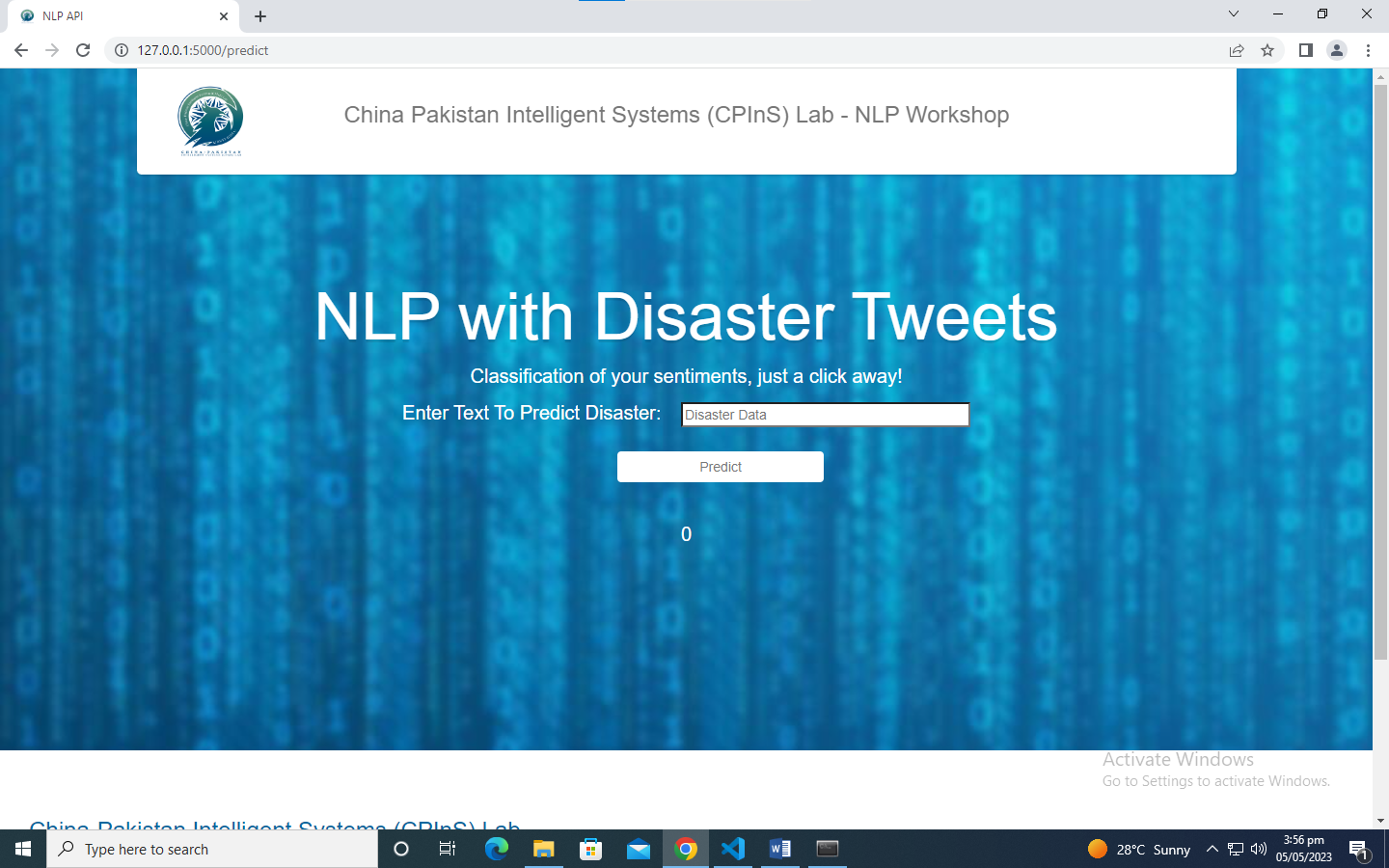
As a sample, in this lab we will deploy a machine learning model that predicts the sentiment of tweets whether it is about a disaster or not. We will create a Flask application that exposes an API endpoint that can receive requests containing the text of a tweet, and returns a predicted sentiment either 0 or 1. You will follow the same steps to deploy your own ML model using Flask.

**Disaster Tweets Dataset:**

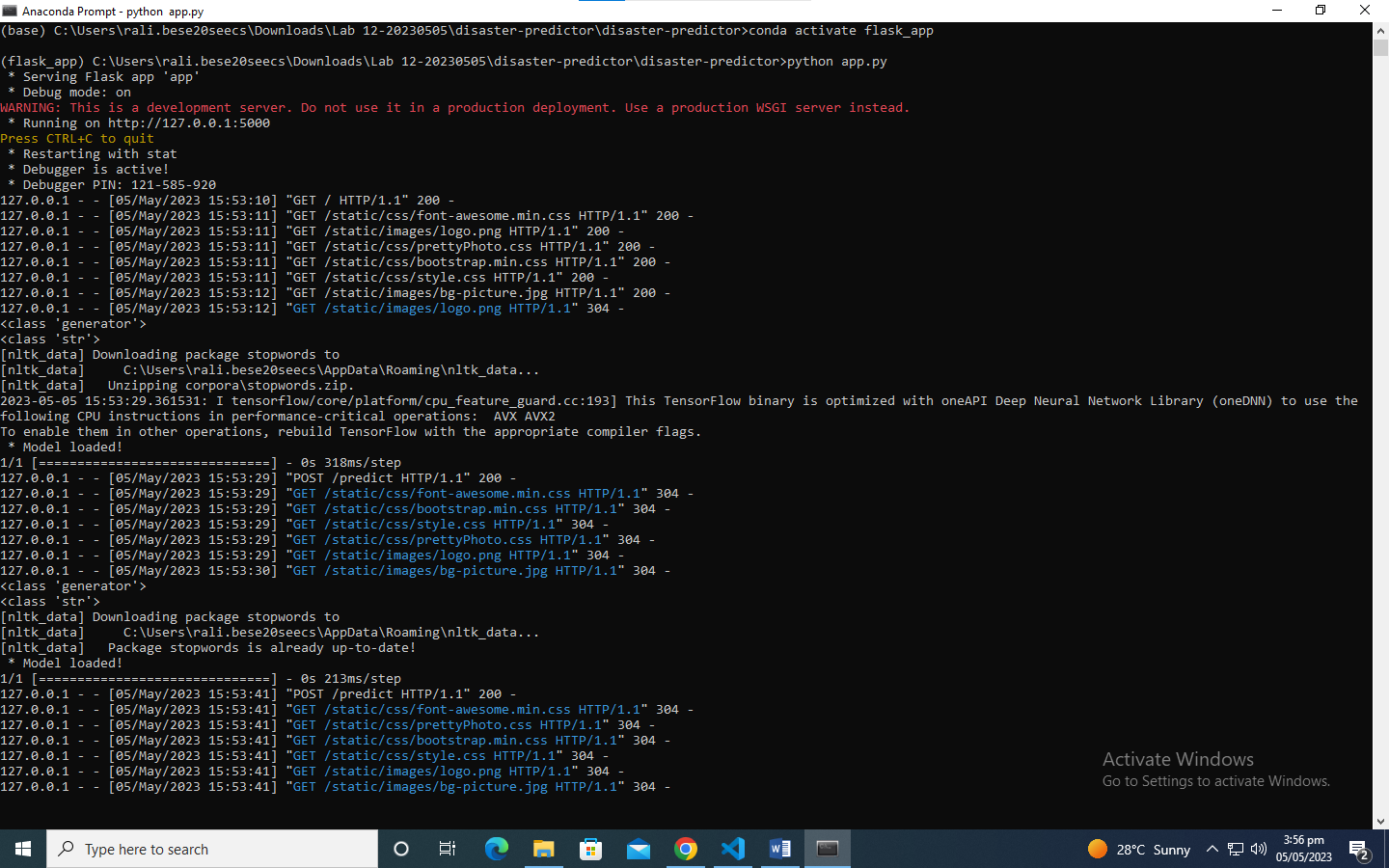
The [dataset](https://www.kaggle.com/competitions/nlp-getting-started/data) contains tweets about real disasters. Each sample in the train and test set has the following information:

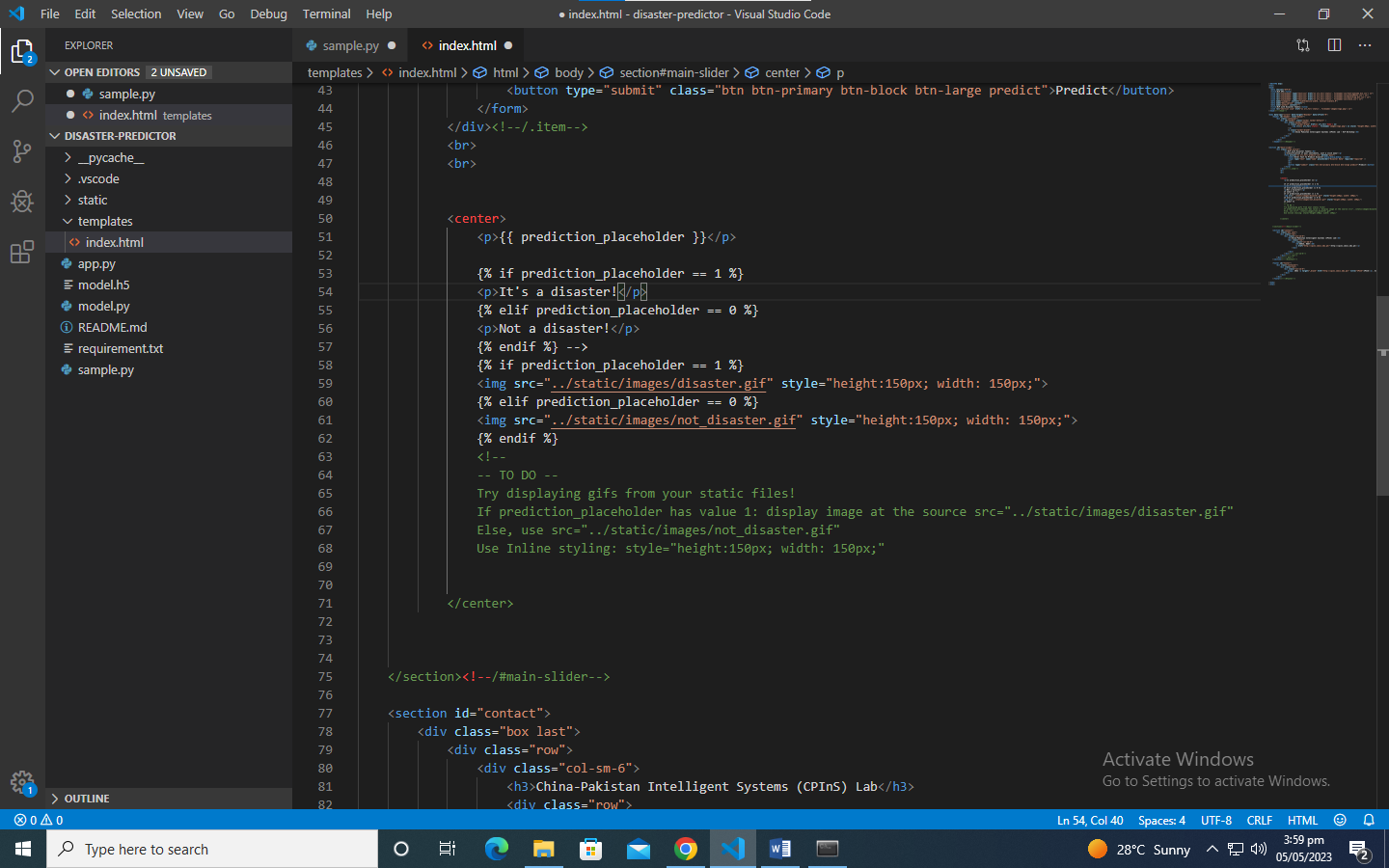
1. id - a unique identifier for each tweet
2. text - the text of the tweet
3. location - the location the tweet was sent from (may be blank)
4. keyword - a particular keyword from the tweet (may be blank)
5. target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)

**Task 1: Deploy LSTM trained on Disaster Tweets data using Flask.**









**Step 1:** Install Anaconda (if not already installed). Set up your virtual environment in Anaconda with python version 3.9

* Create a virtual environment named ‘flask\_app’

**conda create –n flask\_app python=3.9**

* Activate the conda environment

**conda activate flask\_app**

* Navigate to the project folder and run the requirements file to download the required libraries:

**pip install -r requirement.txt**

**Step 2:** Design your HTML homepage. You can use the template given in ‘static’ folder.

**Step 3:** Create the flask app and set route.

* Import the required libraries such as, Tensorflow, nltk, etc.
* Create a Flask application object named *‘app’* using Flask's *Flask()* constructor.
* Define a function to set the route for homepage using *'@app.route'* decorator for the root URL ('/') to define an API endpoint. This function will render the 'index.html' template using the *render\_template()* function and return the HTML page.

**Step 4:** Define a pre-processing function named *‘preprocess\_data’* to clean your data.

**Step 5:** Create a function *‘get\_model’* to load your trained.h5 model. This function returns your trained model.

**Step 6:** Create a function ‘predict’ to receive and process input data

* Define another function named 'predict' with the *'@app.route'* decorator for the '/predict' URL, which will handle the prediction request. This function will
  + Extract the text input from the HTML form using the *request.form.values()* function.
  + Preprocess the text input by calling the *'preprocess\_data'* function and tokenizing it using the Tokenizer object.
  + Call the *'get\_model()'* function to load the model.
  + Call the *'predict'* method of the pre-trained model to predict the sentiment of the input text.
  + Display the result by rendering the 'index.html' template using the *render\_template()* function and passing the predicted value as a parameter. i.e. return the predicted output from the API endpoint to the user.

**Step 7:** Change the output according to your own requirements.You can display a simple text, an image, or a fun gif!

**Task 2:** Deploy your own Semester Project model using Flask. You can use the templates in practice task.

***Happy learning! ☺***

**Deliverable:**

Zip & Submit the practice task files and your own project files on LMS before deadline.