**How to run my code ?**

**1. Backend Setup**

**Step 1: Install Python and Virtual Environment**

* First, ensure you have **Python 3.x** installed on your machine.
* Create a virtual environment in the backend folder to manage dependencies. This isolates your project dependencies and avoids conflicts with other Python projects.

**Step 2: Install Dependencies**

* In the backend folder, there is a file called requirements.txt that lists all the necessary Python packages.
* Install the dependencies by running the command that reads from this file. This will set up everything needed for the backend to work, including libraries like Flask (for web framework), scikit-learn (for machine learning), and others for data handling.

**Step 3: Backend Functionality**

* The backend runs a **Flask web server**. When it's started, it exposes an API that the frontend can interact with.
* The backend includes a function that connects to **Twitter API** to fetch tweets based on a search query.
* It uses a **pre-trained machine learning model** to analyze the sentiment or emotion of those tweets. This emotion could be classified into categories like positive, negative, or neutral.

**Step 4: Start the Backend Server**

* To run the backend, you simply start the Flask server by running a Python script. This allows it to listen for requests (like those coming from the frontend) on a particular port (usually localhost:5000).
* When the server is running, it waits for the frontend to send data (such as the search term for tweets) and processes it accordingly.

**2. Frontend Setup**

**Step 1: Create the HTML Interface**

* The frontend is essentially a **web page**. It consists of an HTML file that provides the layout and structure for user interaction.
* The page contains input fields for entering a search query (what to search for on Twitter) and the number of tweets to fetch.
* There's also a button to trigger the analysis and a section to display the results, such as the tweets and their corresponding emotions.

**Step 2: Styling the Page**

* A **CSS file** is used to style the page, making it look visually appealing and easy to use. You can customize colors, spacing, fonts, and layout to suit your design preferences.

**Step 3: JavaScript for Interaction**

* The **JavaScript file** enables the frontend to interact with the backend. It listens for when the user clicks the "Analyze" button.
* Upon clicking, JavaScript gathers the user input (search query and tweet count), then sends this data to the backend server via an **HTTP request**.
* After the backend processes the data, JavaScript will receive the response and display the tweets along with their emotions on the page.

**3. Running the Project**

**Step 1: Start the Backend**

* Open a terminal/command prompt in the backend folder and run the Flask app.
* This will start the backend server, which will run on localhost:5000 by default. It is now ready to accept requests from the frontend.

**Step 2: Open the Frontend**

* Open the index.html file (located in the frontend folder) in a **web browser**. This is your user interface.
* You can double-click the index.html file or open it directly in a browser of your choice.

**Step 3: User Interaction**

* On the frontend page, enter a **search query** (e.g., "Customer service" or "Product feedback") in the input box.
* Specify how many tweets you want to analyze (e.g., 10).
* Click the "Analyze" button to trigger the process.

**4. Backend-Frontend Communication**

* When you click "Analyze," the frontend sends a **POST request** to the backend with the user-provided data (search query and tweet count).
* The backend then fetches tweets from Twitter based on the search query and uses the pre-trained **emotion analysis model** to determine the emotion of each tweet.
* After processing, the backend sends the tweets along with their corresponding emotion labels back to the frontend.
* The frontend then displays the results (tweets and their emotions) in a list format on the web page.

**5. Troubleshooting and Testing**

* Make sure the backend server is running before you try to interact with the frontend. If the server isn’t running, the frontend won’t be able to send requests to it.
* If the frontend is not displaying results or if there are errors, check the **browser's developer console** (right-click and choose "Inspect" then go to the "Console" tab) for any JavaScript errors or failed HTTP requests.
* If the backend is not responding, ensure that it is running correctly and check the terminal where the Flask app is running for any error messages.

**6. Final Test**

Once everything is set up:

1. **Start the backend** by running the Flask app in the backend folder.
2. **Open the frontend** in a browser.
3. Enter a search query (for example, "customer feedback") and select the number of tweets.
4. Press "Analyze" and check if the tweets and their corresponding emotions are displayed on the page.

**Note :**I attempted to implement both the backend and frontend, but I couldn't achieve the desired integration and interface. While the backend works perfectly fine, I faced challenges with the frontend implementation, especially with correctly integrating it with the backend and handling CORS-related issues, which I couldn’t fully resolve.

**Example JSON ouput :** "predictions": [

    {

      "text": "im feeling rather rotten so im not very ambitious right now",

      "prediction": 0,

      "emotion\_scores": {

        "sadness": 0.9988346695899963,

        "joy": 0.00021661062783095986,

        "love": 0.00022842823818791658,

        "anger": 0.00028547539841383696,

        "fear": 0.00021835765801370144,

        "surprise": 0.00021647124958690256

      },

      "predicted\_emotion": "sadness"

    },