app.py

Import Statements:

The application imports necessary modules and libraries such as Flask, NLTK (Natural Language Toolkit), VaderSentiment (for sentiment analysis), Pandas, NumPy, Pickle (for loading machine learning model), and the camera module (for capturing video frames).

Flask Application Setup:

The Flask application is created with Flask(__name__, template_folder='templates'), specifying the name of the application and the folder where HTML templates are stored.

Data Preparation and Loading:

- The application loads a machine learning model (model_pkl.pkl) from a file using Pickle. This model is likely trained to predict depression levels based on some input features.
- Precautions and links are defined as dictionaries to provide guidance and resources based on the predicted depression level.
- A DataFrame (df1) is initialized, possibly containing music recommendations.

Speech Analysis Function:

The speech_analysis function tokenizes input text, removes stop words and punctuation, lemmatizes words, and computes metrics such as speaking rate, average word length, and count of unique words.

Routes:

The application defines several routes using the @app.route() decorator.

1. **Index Route (/):**
    - Renders the index.html template.
2. **Mental Health Route (/mental_health):**
    - Renders the mental_health.html template.
    - Handles POST requests to predict depression levels based on user input and returns appropriate guidance and resources.
3. **Stress Emotion Route (/stress_emotion):**
    - Renders the stress_emotion.html template and passes data from df1 (music recommendations) to the template.
4. **Video Feed Route (/video_feed):**
    - Streams video frames captured from the camera using the VideoCamera class.
5. **Voice Route (/voice):**
    - Serves the voice.html file from the static directory.
6. **Analyze Sentiment Route (/analyze):**
    - Handles POST requests containing text data for sentiment analysis.
    - Uses the VaderSentiment library to analyze sentiment and the speech_analysis function to analyze speech.
    - Returns JSON response containing sentiment analysis results.

camera.py

Imports:

The script imports necessary libraries such as NumPy, OpenCV (cv2), PIL (Python Imaging Library), TensorFlow (for deep learning), Pandas, datetime, Thread (for multi-threading), and time.

Cascade Classifier and Model Setup:

- The script loads a pre-trained face cascade classifier (haarcascade_frontalface_default.xml) for face detection.
- It defines a convolutional neural network (CNN) model for emotion detection. The model architecture seems to be designed for recognizing emotions from facial expressions.

Global Variables:

Global variables like last_frame1, cap1, and show_text are initialized for use across different functions.

Classes for FPS Calculation and Video Streaming:

- FPS class calculates frames per second (FPS) during video streaming.
- WebcamVideoStream class initializes a video stream from the webcam and handles reading frames from the stream in a separate thread for performance optimization.

VideoCamera Class:

- VideoCamera class reads frames from the webcam, detects faces, predicts emotions using the CNN model, and recommends music based on the detected emotion.
- It uses OpenCV to capture frames, detect faces, and draw rectangles around detected faces.
- Emotion prediction is done by passing cropped facial images through the CNN model.
- Music recommendation is based on the detected emotion, with different CSV files containing music metadata for each emotion.

music_rec Function:

This function retrieves music recommendations based on the currently detected emotion.

spotify.py

Imports:

The script imports necessary modules from the Spotipy library and Pandas for data manipulation.

Authentication:

The client_id and client_secret are provided to authenticate with the Spotify API using the SpotifyClientCredentials authentication manager.

Spotify API Setup:

An instance of the Spotify class is created with the provided authentication manager (auth_manager).

Helper Functions:

- **getTrackIDs(user, playlist_id):** This function takes a Spotify user ID and a playlist ID as input and returns a list of track IDs from the specified playlist.
- **getTrackFeatures(id):** This function takes a track ID as input, retrieves track information from the Spotify API, and returns a list containing the track's name, album, and artist.

Global Variables:

- **emotion_dict:** A dictionary mapping emotion indices to corresponding emotional states.
- **music_dist:** A dictionary mapping emotion indices to playlist IDs on Spotify.

utils.py

Initialization:

- The __init__ method initializes the WebcamVideoStream object.
- It takes an optional argument src, which represents the index of the webcam device to be used. By default, it's set to 0, indicating the default webcam.
- It initializes a video capture object self.stream using cv2.VideoCapture, which captures video from the specified source.
- It reads the first frame from the video stream and stores it in self.frame.
- It initializes self.stopped to False, indicating that the video stream is currently active.

Threaded Video Streaming:

- The start method starts a new thread by calling the update method as the target.
- Inside the update method, the video stream is continuously read in a loop.
- If the self.stopped flag is set to True, the loop exits, and the thread terminates.
- Otherwise, it continuously reads frames from the video stream by calling self.stream.read().

Reading Frames:

- The read method returns the latest frame captured from the video stream (self.frame).
- This method can be called from outside to retrieve frames from the video stream.

Stopping the Stream:

- The stop method sets the self.stopped flag to True, indicating that the video stream should stop.
- This method can be called to terminate the video streaming thread.

train.py

Import Statements:

The script imports necessary modules from Keras for building and training the CNN model.

Data Preparation:

- Training and validation directories (train_dir and val_dir) are specified.
- Image data generators (train_datagen and val_datagen) are created to preprocess the image data by rescaling pixel values.

Data Generators:

- Image data generators (train_generator and val_generator) are instantiated to flow images from directories in batches during model training.
- Images are resized to (48, 48) pixels and converted to grayscale.
- Class mode is set to categorical, indicating one-hot encoded labels.

CNN Model Architecture:

- A Sequential model (emotion_model) is created.
- Convolutional layers with ReLU activation and max pooling are added for feature extraction.
- Dropout layers are added for regularization to prevent overfitting.
- Flatten layer is added to convert 2D feature maps into a 1D feature vector.
- Fully connected dense layers with ReLU activation are added for classification.
- The output layer uses softmax activation for multi-class classification (7 emotions).

Model Compilation:

The model is compiled with categorical cross-entropy loss, Adam optimizer with a learning rate of 0.0001 and decay rate of 1e-6, and accuracy as the metric.

Model Training:

- The fit_generator method is used to train the model.
- Training data is provided using train_generator, with the number of steps per epoch calculated based on the batch size and the total number of training samples.
- Validation data is provided using val_generator, with similar calculation for the number of validation steps.
- The training process is run for 75 epochs.

Model Saving:

After training, the model weights are saved to a file named 'model.h5'.

Index.html

Head Section:

- <head>: Contains metadata and links to external resources.
- <meta charset="UTF-8">: Defines the character encoding of the document as UTF-8.
- <meta name="viewport" content="width=device-width, initial-scale=1.0">: Sets the viewport properties for responsive design.
- <link rel="stylesheet" href="/static/index.css">: Links an external CSS file (index.css) for styling.
- <title>Home</title>: Sets the title of the webpage to "Home".

Body Section:

- <body>: Contains the visible content of the webpage.
- <header>: Represents the header section of the webpage.
- <h1>: Displays the main heading "Welcome to Our Website".
- <p>: Provides a subheading "Empowering Your Well-being".
- : Acts as a container for the main content sections.
- <section class="feature">: Represents a feature section.
- <img>: Displays an image representing mental health.
- <h2>: Displays the heading "Mental Health".
- <p>: Provides a brief description of mental health resources.
- <a href="/mental_health">: Links to the mental health page using a button labeled "Explore Now".

(Similar sections exist for "Stress and Emotion Detection" and "Speech Analysis for Stress" with respective images, headings, descriptions, and buttons.)

Footer Section:

- <footer>: Contains information at the bottom of the webpage.
- Two <p> elements provide copyright information ("© 2024 All rights reserved.") and contact details ("Contact: pambillaa@gmail.com").

Mental_health.html

Head Section:

- <head>: Contains metadata and links to external resources.
- <meta charset="UTF-8">: Defines the character encoding of the document as UTF-8.
- <meta http-equiv="X-UA-Compatible" content="IE=edge">: Specifies the version of Internet Explorer to render the webpage.
- <meta name="viewport" content="width=device-width, initial-scale=1.0">: Sets the viewport properties for responsive design.
- External CSS files are linked for styling, including style.css, mobile_style.css, and firebase.css.
- Bootstrap CSS and JavaScript files are linked for additional styling and functionality.

JavaScript Functions:

- JavaScript functions are defined within <script> tags.
- fun(): Displays an alert message and toggles the visibility of form elements.
- slide(): Toggles the visibility of form elements and updates the anchor link.
- validateForm(): Validates the form input fields for email, age, and gender.
- Title:
- <title>Mental Health Prediction</title>: Sets the title of the webpage.

Body Section:

- <body>: Contains the visible content of the webpage.
- Divisions (`win1, win2, footer) and form elements are used to structure the content. - A heading (h1) with a span is used for the main title, "Mental Health". - A button (‘button’) with an onclick event triggers the slide()function to toggle the visibility of form elements. – A ‘div’ with the id ‘res’ displays the result and precaution messages if they exist. - A form (‘form’) with the id form-target is used for submitting user responses. It calls ‘thevalidateForm()` function on submission.

Form Section:

- The form contains a series of questions related to mental health, each with multiple choice radio buttons for responses.
- Each question is presented within a div with the class card, containing the question text and radio button options.
- The questions range from feelings of hopelessness to thoughts of self-harm, with response options ranging from "I don't feel like this" to "I feel this almost every day".

Footer Section:

- The footer contains contact information and a form for users to fill out.
- It includes fields for name, email, age, and gender.
- A button labeled "Send" submits the form data, triggering the fun() function.
- Additional text explains why users should fill out the form and how their data will be used.

Stress_face.html

Title and Styling:

- The page title is "Mood Music Recommender."
- External stylesheets and JavaScript libraries like Bootstrap and jQuery are imported for styling and functionality.

Body Content:

- The body has a container with a div having the id body.

Inside the body container, there are two main sections floated side by side:

1. **Stress and Emotion Detector Section:**
   - Includes a title "Stress and Emotion Detector" styled with a specific font.
   - Contains an image element (img) sourced from a URL generated by the Flask url_for function, which likely serves a video feed for emotion detection.
2. **Song Recommendations Section:**
   - Includes a title "Song Recommendations" styled similarly to the previous section.

Result Area:

- A div with the id ResultArea serves as a container for dynamically updating content.
- It has an outer-shadow class applied to it.

JavaScript:

- The JavaScript section contains code to periodically fetch JSON data from a Flask route (/t) using AJAX.
- This data is used to dynamically create an HTML table (table) inside the ResultArea div, displaying the name, album, and artist of songs.
- The CreateHtmlTable() function constructs the HTML table based on the received JSON data.

Voice.html

Body Content:

- The body contains a div with the class container, which serves as the main content area.
- Inside the container, there's an h1 element displaying the title "Speech Sentiment Analysis."
- Below the title, there's a div with the id result where the analysis results will be displayed.
- A button with the id recordButton is provided to initiate the recording of speech.

JavaScript:

- The JavaScript section handles the functionality of recording speech and analyzing sentiment.
- When the recordButton is clicked, it triggers the click event listener.
- Within the event listener, a webkitSpeechRecognition object is created to recognize speech.
- The recognition settings are configured, including language, maximum duration, and result handling.
- When speech is recognized (onresult event), the transcript is extracted and sent to the server for sentiment analysis via a POST request to the /analyze endpoint.
- The response from the server is then displayed in the resultDiv, showing the transcript, sentiment, and polarity scores (negative, neutral, positive).
- Error handling is also implemented to handle recognition errors or server communication errors.

Models used

Model_pkl.pkl – trained model used in mental health detection (random forest)

Model.h5 – trained model for emotion detection using CNN

Haarcascade_frontalface_Detection – pre-trained model for face detection.