Prototype Documentation:

Objective:
The objective of this prototype is to develop a Python script that enables users to perform audio analysis by combining emotion detection and talk ratio detection. This prototype guides users through the process of inputting audio files, conducting two distinct analyses, and visualizing the results.

Approach:

1. Input:

User-Provided Audio Files: Users are able to upload their own audio files for analysis.
Dataset Selection: The script provides the flexibility to select a suitable dataset from available options on Hugging Face. If no appropriate dataset is found, users have the liberty to choose any other dataset.

2. Emotion Detection:

Feature Extraction: We extract relevant audio features for emotion detection using the `librosa` library. These features may include Mel-frequency cepstral coefficients (MFCCs) and pitch.
Machine Learning Model: We employ `scikit-learn` to build a basic machine learning model. This model is trained using the extracted features to classify audio as either "happy" or "sad."

3. Talk Ratio Detection:

Research and Integration: Comprehensive research is conducted to identify the most suitable solution for talk ratio detection. This may involve integrating third-party APIs or open-source libraries to analyze audio files and compute the talk ratio.

4. Printing Results:

Emotion Detection Results:
- Predicted emotions ("happy" or "sad") are printed in tabular format. Column A contains the Audio File name, and Column B displays the predicted emotion. The prediction is also visualized in a smooth line chart, with the positive y-axis representing "happy," the negative y-axis representing "sad," and the x-axis denoting time.

Talk Ratio Detection Results:
- Predicted Talk Ratios are presented in tabular format with appropriate columns. They are also illustrated in smooth line charts with relevant axes.

User Manual:

1. Inputting Audio Files:

- Users can provide their audio files by utilizing the provided upload feature in the script. The script accepts audio files in common formats such as MP3 and WAV.

2. Running the Script:

- Users need to execute the Python script, which will guide them through the entire process.

3. Viewing Results:

- After processing, users can review the results on the script's interface.

Accepted Audio File Formats:

- The script supports audio files in formats like MP3 and WAV.

Challenges Faced:

- Challenges encountered included preprocessing and feature extraction from audio files. The selection of relevant features for emotion detection was a crucial task. Moreover, integrating a third-party solution for talk ratio detection and ensuring compatibility with the chosen dataset was a significant challenge.

User Manual:

Below is a step-by-step guide on how to use the prototype:

Step 1: Installation

1. Ensure you have Python installed on your system, which can be downloaded from [Python's official website](https://www.python.org/downloads/).

2. Install the required libraries using pip:
   ```
   pip install librosa scikit-learn numpy pandas matplotlib
   ```

Step 2: Running the Script

1. Open a terminal or command prompt.

2. Navigate to the directory where you have the Python script.

3. Run the script using the following command:
   ```
   python script.py
   ```

Step 3: Uploading Audio Files

1. After running the script, a web interface will open in your default web browser.

2. Use the provided file upload feature to select and upload audio files. Supported formats are MP3 and WAV.

Step 4: Viewing Results

1. The script will process the uploaded audio files for emotion detection and talk ratio.

2. You can view the results in the web interface. The predicted emotion ("happy" or "sad") will be displayed in tabular format with columns for Audio File name and predicted emotion. Additionally, the emotion prediction will be shown in a smooth line chart.

3. The predicted Talk Ratio will be presented as a table with appropriate columns, and it will be visualized in a smooth line chart.

Note: This is a prototype, and for a complete implementation, you would need to develop the Python script and user interface for uploading audio files and displaying results.