1. Introduction

This application is designed to automatically assess call verification processes in customer service environments using machine learning techniques. It processes audio files of calls, transcribes them, and predicts whether proper verification procedures were followed.

2. System Requirements

2.1 Hardware Requirements

- Processor: Intel Core i5 or equivalent (i7 recommended for faster processing)
- RAM: Minimum 8GB (16GB or more recommended)
- Storage: At least 1GB of free disk space for the application and its dependencies
- GPU: NVIDIA GPU with CUDA support (optional, but recommended for faster audio transcription)
    - Minimum NVIDIA GeForce GTX 1060 or equivalent
    - 6GB of VRAM or more

2.2 Software Requirements

- Operating System:
    - Windows 10 (64-bit) or later
    - macOS 10.14 or later
    - Ubuntu 18.04 or later (or other Linux distributions with equivalent specifications)
- Python: Version 3.7 or higher
- Required Python libraries:
    - whisper: For audio transcription
    - nltk: For natural language processing
    - pandas: For data manipulation
    - torch: For machine learning operations
    - tkinter: For the graphical user interface
    - dill: For model serialization
- CUDA Toolkit: Version 11.0 or later (if using GPU acceleration)
- Audio Codecs: Support for WAV and MP3 file formats

2.3 Additional Software

- A text editor or IDE for viewing the CSV output (e.g., Microsoft Excel, Google Sheets, or any text editor)

3. Installation and Setup

3.1 Installation Steps

1. Install Python 3.7 or higher from the official Python website (https://www.python.org/downloads/).
2. Install the required Python libraries using pip:

   ```
   pip install whisper nltk pandas torch tkinter dill
   ```

3. If using GPU acceleration, install the appropriate CUDA Toolkit for your NVIDIA GPU from the NVIDIA website.
4. Download the application files:
    - finalcodegui2.1.py (main application script)
    - callcheck_model.pkl (trained logistic regression model)

    - token_vect.pkl (vectorizer)
    - tokens.pkl (token set)
5. Place all files in the same directory.

3.2 First-Time Setup

1. Run the application once to create necessary NLTK data:

   ```
   python finalcodegui2.1.py
   ```

2. If prompted, allow the application to download required NLTK data.

4. Launching the Application

To start the application, navigate to the directory containing the application files in your terminal or command prompt, then run:

```
python finalcodegui2.1.py
```

A graphical user interface (GUI) will appear.

5. Using the Application

5.1 Adding Audio Files

1. Click the "Add Audio Files" button.
2. In the file dialog that appears, select one or more audio files (WAV or MP3 format).
3. The number of selected files will be displayed on the interface.

5.2 Selecting Output File

1. Click the "Select Output File" button.
2. Choose a location and name for the output CSV file.
3. A confirmation message will appear showing the selected output file path.

5.3 Processing Files

1. After adding audio files and selecting an output file, click the "Transcribe and Verify" button.
2. The application will process each audio file:
    - Transcribing the audio
    - Preprocessing the text
    - Predicting the verification status
3. A progress message will be displayed in the console (not visible in the GUI).
4. Once complete, a message box will appear confirming that the results have been saved.

6. Understanding the Output

The application generates a CSV file with the following columns:

- Audio File: Name of the processed audio file
- Cleaned Transcription: The preprocessed transcription of the call
- Verification: The predicted verification status (3 for verified, 0 for not verified)

7. Troubleshooting

- If the application fails to start, ensure all required libraries are correctly installed.
- If file processing fails, check that the audio files are in a supported format (WAV or MP3).
- For optimal performance on systems with a CUDA-enabled GPU, ensure CUDA is properly set up.
- If you encounter "Out of Memory" errors, try processing fewer files at a time or upgrade your system's RAM.
- If the application is running slowly, consider using a machine with a CUDA-enabled GPU for faster processing.

8. Best Practices

- Process files in batches to manage system resources effectively.
- Regularly back up your output CSV files.
- Periodically validate the model's predictions against manual assessments to ensure continued accuracy.
- Keep the application and its dependencies updated to the latest stable versions.
- When processing large volumes of calls, consider running the application on a dedicated machine to avoid interfering with other tasks.