

PROJECT: Gender And Age Detector

EXECUTIVE SUMMARY

This report documents six machine learning projects designed to evaluate logic-building, problem-solving, and practical implementation skills. Each project combines computer vision, audio processing, or video analysis with graphical user interfaces (GUIs). The report provides detailed problem statements, methodologies, model architectures, and implementation guidelines for each task.

1. LONG HAIR IDENTIFICATION

1.1 Problem Statement

The objective is to build a computer vision model that infers a modified "gender" label from a single face image based on hair length and age constraints. The system must:

- For persons aged 20–30: classify long-haired persons as female and short-haired persons as male, regardless of their actual biological gender.
- For persons below 20 or above 30: predict gender normally, independent of hair length.

A graphical user interface (GUI) must allow image upload, run inference, and display the final label.

1.2 Dataset and Preprocessing

The project can be built by combining:

- A face age–gender dataset (e.g., IMDB-WIKI, UTKFace, or similar public sets) to train age and gender estimation.
- A hair attribute dataset or manual hair-length annotations for a subset of images created using bounding-box heuristics or by labeling existing face datasets.

Preprocessing steps:

- Detect and crop faces using Haar Cascades or a deep learning-based face detector.
- Resize faces to a fixed input size (e.g., 224×224).
- Normalize pixel values and apply basic augmentation (flip, rotation, brightness).

1.3 Model Architecture and Logic

A practical approach is to build a two-head CNN or use a pre-trained backbone (e.g., MobileNet, ResNet) with:

- Age head: multi-class or regression age prediction.
- Hair-length head: binary classification (long vs short).

Age range logic is implemented as a post-processing rule:

- If predicted age in [20, 30]:
 - If hair = long → output "female".
 - If hair = short → output "male".
- Else: use a separate gender head or a secondary gender classifier for normal gender estimation.

1.4 GUI Design

Using Tkinter or PyQt:

- Components: image upload button, image preview panel, "Predict" button, label outputs for age, hair length, and final displayed gender.
- On click, the GUI runs face detection, passes the cropped face through the model, applies the age/hair-length rule, and displays the result.

2. SENIOR CITIZEN IDENTIFICATION (VIDEO/WEBCAM)

2.1 Problem Statement

The goal is to process a CCTV-style video or webcam feed from a mall or store, detect multiple persons, estimate their age and gender, and label anyone older than 60 as "Senior Citizen" with gender. Age, gender, and visit timestamp must be saved to a CSV/Excel file for each detected person.

2.2 Dataset and Preprocessing

Age–gender models can be trained or fine-tuned using face datasets such as UTKFace or similar age–gender corpora. For real-time deployment, the model must be robust to low resolution and surveillance angles.

Pipeline:

- Frame capture from webcam/video at a fixed interval (e.g., every N frames).
- Person/face detection (e.g., YOLO or Haar-based face detection).
- Cropping detected faces and resizing to model input size.

2.3 Model and Senior Logic

Use a CNN-based age–gender estimator; for efficiency, use a light backbone or a pre-trained age–gender model.

- If predicted age > 60: mark as "Senior Citizen – Male/Female".
- Else: label with age and gender only.

Each detection generates a row in a CSV file: [timestamp, age, gender, is_senior (yes/no)]. Saving can be done using pandas or Python's csv module.

3. AGE AND EMOTION DETECTION THROUGH VOICE

3.1 Problem Statement

This task focuses on speech analysis. From a voice note, the model must:

- Accept only male voices. If the speaker is female, reject input and show "Upload male voice."
- If age > 60: output age and emotion (e.g., happy, sad, neutral, angry).
- If age ≤ 60: output only age.

A GUI is required to upload audio, trigger inference, and display results.

3.2 Dataset and Features

Possible datasets:

- Age and gender: corpora like Mozilla Common Voice or specialized speech corpora labelled with speaker age and gender.
- Emotion: CREMA-D, RAVDESS, or other emotion-labelled speech datasets.

Feature extraction:

- Convert audio to mono, fixed sample rate.
- Extract MFCCs, chroma, spectral features or log-mel spectrograms as input to the network.

3.3 Model and Logic Flow

A typical architecture:

- Gender classifier: CNN/LSTM over spectrograms to detect male vs female.
- Age estimator: age-classification network trained on male speech only.
- Emotion classifier: multi-class network trained on standard emotion categories.

Inference logic:

1. Run gender model; if predicted "female" → show message and stop.
2. If male: run age model.
3. If age > 60: run emotion model and output both.
4. If age ≤ 60: output only age.

3.4 GUI Design

GUI elements:

- Audio file selector, waveform or file name preview.
- Buttons: "Upload audio", "Predict".

- Text panels for: detected gender, age, emotion (or rejection message).

4. SIGN LANGUAGE DETECTION

4.1 Problem Statement

The aim is to recognize selected sign language words from either an uploaded image/video or live webcam feed. The model must be active only during a predefined time window (e.g., 6 PM to 10 PM); outside this window, the GUI should display a message that the service is unavailable.

4.2 Methodology

Sign language involves complex spatio-temporal patterns of hands and body. An effective approach is:

- Use a pose-estimation model (e.g., MediaPipe Hands or PoseNet) to extract keypoints from each frame.
- Feed sequences of keypoints into a sequence model such as an LSTM or temporal CNN for action recognition of specific signs.

Training data can be custom-recorded for a limited vocabulary (e.g., "Hello", "Yes", "No", "Thank you") to keep the scope manageable.

4.3 GUI and Time Logic

GUI requirements:

- Options for "Upload Image/Video" and "Start Camera".
- Preview window of the input stream.
- Text area showing predicted sign word.

Time logic:

- On every prediction attempt, check system time; allow recognition only between 18:00 and 22:00. If out of range, disable prediction buttons and show a dedicated status message.

5. CAR COLOUR DETECTION MODEL

5.1 Problem Statement

This project focuses on traffic scene analysis. The system must:

- Detect cars in traffic and predict each car's colour.

- Count total number of cars at a traffic signal.
- Draw a red rectangle for blue cars and a blue rectangle for all other car colours.
- Detect and count people at the signal.

A GUI should display input images and overlay the detections.

5.2 Methodology

A typical pipeline:

- Detect vehicles and persons using an object detector such as YOLOv8 or similar deep-learning model.
- For each detected car, extract the region of interest and estimate dominant colour using simple colour histograms or a small CNN trained for car colour classification.
- Apply drawing rules:
 - If car colour classification = blue → draw red rectangle.
 - Else → draw blue rectangle.

Person detection is obtained from the same detector class (e.g., "person") and counted per frame.

5.3 GUI Design

GUI elements:

- Image upload or video selection and preview pane.
- "Process" button to run detection and overlay bounding boxes with counts.
- Labels displaying: number of blue cars, other cars, total cars, and number of persons.

6. NATIONALITY DETECTION MODEL

6.1 Problem Statement

The goal is to build a model that:

- Predicts the nationality of a person from an uploaded face image.
- Predicts emotion for every person.
- If predicted nationality is:
 - Indian: additionally predict age and dress colour.
 - United States: predict age and emotion only.
 - African: predict emotion and dress colour only.
 - Other nationalities: output nationality and emotion only.

A GUI must show the input image preview and a structured output section.

6.2 Dataset and Approach

Nationality (or ethnicity/region) classification is a challenging and sensitive problem. For this educational project, a simplified approach can be:

- Use a dataset where faces are grouped into broad regions (e.g., "India / South Asia", "United States / North America", "Africa") plus an "Others" category.

- Train a CNN classifier on these region labels.
- Use a standard emotion recognition model trained on facial expression datasets (e.g., FER2013).
- Dress colour can be estimated by segmenting the upper-body region below the face and computing the dominant colour via clustering in HSV space.

6.3 Decision Logic and GUI

Decision logic:

- Predict nationality/region, emotion, and optionally age.
- Apply rule-based branching based on nationality with conditional outputs.

GUI:

- Left panel: input image preview.
- Right panel: text fields or table listing nationality, age (if applicable), emotion, and dress colour (if applicable).

7. COMMON IMPLEMENTATION DETAILS

7.1 Tools and Frameworks

Across all projects, typical technologies include:

- Python: Core programming language for all implementations.
- OpenCV: Image and video processing, computer vision utilities.
- TensorFlow/Keras or PyTorch: Deep learning model training and inference.
- Tkinter, PyQt, or Streamlit: Graphical user interface development.
- Pandas or csv module: Data logging and file management, especially for CSV/Excel output.
- MediaPipe or scikit-image: Specialized processing (pose estimation, segmentation).

7.2 Evaluation and Limitations

- Basic evaluation can be performed via accuracy, confusion matrices, or F1-score on held-out test sets.
- Ethical considerations: Tasks like gender, age, and nationality detection can be biased and should be treated purely as academic exercises with attention to fairness and privacy.
- Each model should be tested on diverse datasets to ensure robustness across different demographics and conditions.
- Edge cases (e.g., extreme lighting, occlusions, unusual angles) should be validated during testing.

7.3 Deployment Recommendations

- Optimize models for inference using quantization or lightweight architectures for real-time performance.
- Implement proper error handling and user feedback mechanisms in GUIs.
- Log predictions and timestamps for audit trails and further analysis.

- Respect privacy by not storing raw images permanently; consider deletion after processing.

CONCLUSION

This report outlines six comprehensive machine learning projects designed to develop practical skills in computer vision, audio processing, and video analysis. Each project combines theoretical knowledge with hands-on implementation, requiring:

- Model design and training on relevant datasets
- Implementation of complex conditional logic and decision trees
- Development of functional GUIs for user interaction
- Testing and evaluation across diverse scenarios

These projects encourage learners to embrace challenges as growth opportunities while maintaining ethical awareness in sensitive areas such as biometric detection. Through completion of these tasks, students will gain valuable experience in end-to-end machine learning pipeline development, from data preparation through deployment.

REPORT COMPLETED

Date Prepared: December 11, 2025

Total Projects: 6