Internship Report: Task 1 - Voice Gender Detection

Introduction:

The Voice Gender Detection project assigned to me involved creating a machine learning model to predict the gender of a speaker based on short audio clips. Throughout this task, I focused on audio data pre-processing, feature extraction, and the application of machine learning techniques. The main challenge addressed was handling variations in voice tones and pitches among speakers of the same gender.

Code Overview:

In my Python code, I completed the following key tasks:

Data Handling:

- Loaded the dataset from 'voice.csv' and extracted features (X) and labels (y).
- Employed label encoding to convert categorical labels into numerical format.

Data Splitting:

- Utilized the train_test_split function to divide the dataset into training and testing sets.

Feature Scaling:

- Applied feature scaling using StandardScaler to standardize the features.

Model Training:

- Chose the RandomForestClassifier and trained the model on the standardized training set.

Model Evaluation:

- Made predictions on the testing set, achieving an impressive accuracy of 98%.
- Provided a comprehensive classification report showcasing precision, recall, and F1-score metrics for both genders.

Feature Extraction Function:

- Implemented the extract_features function using the librosa library for extracting MFCC features from audio files.

Gender Prediction Function:

- Developed the predict_gender function, allowing users to input an audio file path for real-time gender prediction.

Sample Output:

Upon testing the model with a male voice sample (voicesample2.wav), the following output was generated:

Accuracy: 0.98

Classification Report:

	precision	**recall**	**f1-score**	**support**
0	0.97	0.99	0.98	297
1	0.99	0.98	0.99	337

accuracy			0.98	634
macro avg	0.98	0.98	0.98	634
weighted avg	0.98	0.98	0.98	634

Enter the path to the audio file: C:/Users/sufiy/Downloads/voicesample2.wav

Extracted Features: [feature values here]

Predicted gender: male

Conclusion:

The successful prediction of gender for the provided male voice sample demonstrates the efficacy of the model. The high accuracy and detailed classification report indicate the model's robustness in handling diverse voice characteristics. This report provides an overview of my work, emphasizing my contributions to data processing, model development, and real-time gender prediction based on user-provided audio samples.

Internship Report: Task 2 - Pedestrian Detector

Introduction:

The Pedestrian Detector project assigned to me involved developing a machine learning model capable of detecting and recognizing pedestrians in street images or videos. This task presented unique challenges, including scale, perspective, occlusion, and potential motion, requiring the model to operate reliably in various settings and lighting conditions.

Code Overview:

I implemented the Pedestrian Detector using Python, tkinter for the graphical user interface (GUI), and the YOLO (You Only Look Once) model for pedestrian detection. The key components include:

GUI Setup:

- Developed a tkinter-based GUI with buttons for selecting images and videos.
- Created an instance of the PedestrianDetector class, utilizing the YOLO model for image and video processing.

Image Processing:

- Implemented the select_image method to allow users to choose an image file.
- Utilized the YOLO-based PedestrianDetector class to detect pedestrians in the image.

- Drew bounding boxes around detected pedestrians and displayed the result in a new window.

Video Processing:

- Implemented the process_and_display_video method for selecting a video file.
- Processed the video frames using the YOLO-based PedestrianDetector for real-time pedestrian detection.
- Displayed the processed video with bounding boxes in a new window.

Video Saving:

- Implemented video saving functionality by writing processed frames with bounding boxes to a new video file.
- Utilized the VideoWriter class from OpenCV for video creation.

GUI for Video Display:

Created a new tkinter window for displaying the processed video with bounding boxes.

Sample Output:

Upon selecting an image or video, the Pedestrian Detector, powered by the YOLO model, successfully detected pedestrians and displayed bounding boxes around them. The output included the confidence value for each detection, typically showing a high confidence value of 1.0.

Conclusion:

The implementation of the YOLO model for pedestrian detection enhances the robustness and accuracy of the Pedestrian Detector project. The GUI-based approach facilitates user-friendly interaction, allowing users to test the YOLO-based model with ease. The code effectively addresses challenges such as scale, perspective, occlusion, and motion in detecting pedestrians. This report provides an overview of my work on the Pedestrian Detector task, emphasizing the integration of the YOLO model for pedestrian detection.

This concludes my report on the Pedestrian Detector task, summarizing my contributions and highlighting the successful implementation of the YOLO model for image and video processing.