CAIRO UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

Team 5

Speaker Gender Age Recognition

NN Project

ProjectReport

Name	Sec	ID
• Salma Muhammed Entsar Marzouk	1	9220364
• Mohammed Khaled Ahmed	2	9220681
• Abdelrahman Moustafa Gomaa	1	9220475
• Youssef Roushdy	2	9220985

Contact Info

salma.hassan03@eng-st.cu.edu.eg
mohamed.abdelwahed03@eng-st.cu.edu.eg
abdelrahman.abdeltawab03@eng-st.cu.edu.eg

3/05/2025

PROJECT OVERVIEW

This project focuses on developing a plant disease detection system using machine learning. We utilized a dataset from Kaggle, which consists of three classes: Healthy, Rust, and Powdery.

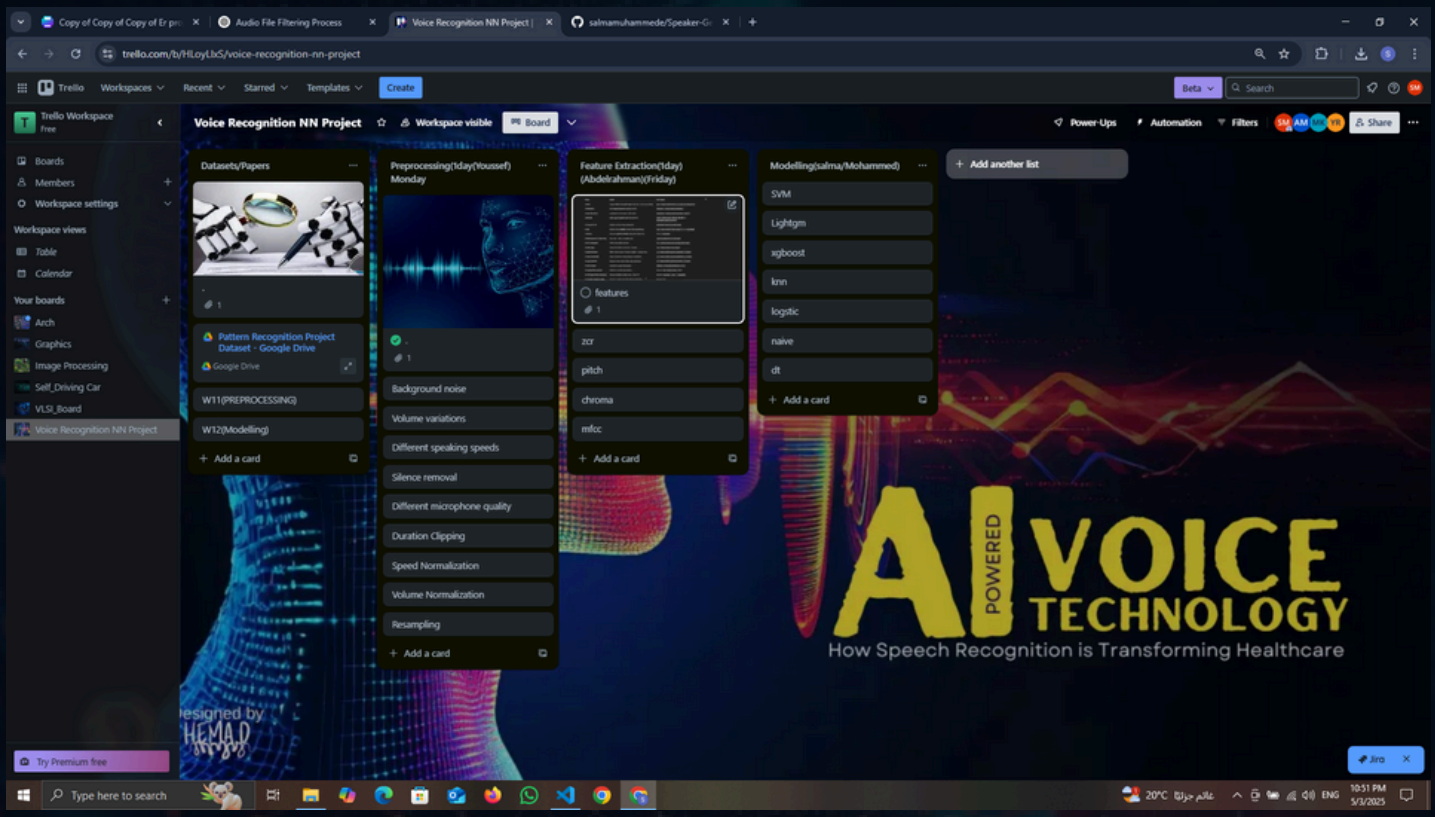
The initial step involved class imbalance checking, confirming that the dataset was balanced. To enhance model performance and generalization, we applied data augmentation on the training set. Additionally, we ensured the dataset was well-prepared by performing necessary enhancements and preprocessing.

For feature extraction, we applied four texture analysis methods: HSV, RGB, GLCM, and Gabor to extract meaningful features from the images.

We then applied four different machine learning models: XGBoost, KNN, Random Forest, and SVM, to classify the images into the respective disease categories. To improve the models' performance, we performed hyperparameter tuning. After evaluating the results, we selected the best-performing model.

Finally, the chosen model was deployed in a Flask-based web application, allowing users to upload images and receive predictions about the plant's health. Additionally, a pretrained model was applied to the images for comparison, and the application displays the results of both models for comparison.

WORKLOAD



Youssed: Filtering/ Preprocessing
Abdelrahman: Feature Extraction/ Docker
Salma/ Mohammed: Modelling

FILTERING/PREPROCESSING

Goal

Automatically clean a raw audio dataset by detecting and removing corrupted or invalid audio files.

🧠 What the Script Does

- Validates audio files (using librosa) based on length, silence, and load errors.
- Skips very small files (<1KB).
- Copies valid files to a clean output folder (../data_filtered), renaming duplicates if needed.

Tracks corrupted files for review.

📁 Outputs

- A cleaned dataset of valid audio files.
- Two log files:
- valid_files.txt and corrupted_files.txt listing all processed files.

🔍 Purpose

Clean and enhance speech audio files for ML tasks by reducing noise, normalizing volume, filtering frequencies, and removing silence.

🛠️ Core Steps

- Loads audio from .mp3 or .wav, converts to mono, resamples to 16kHz.
- Reduces background noise using noisereduce.
- Normalizes volume for consistent loudness.
- Applies bandpass filter (80–8000 Hz) to focus on speech frequencies.
- Removes silence from non-speaking segments.
- Saves output as cleaned .wav files in a new directory.

⚙️ Efficiency

- Uses parallel processing (joblib, multiprocessing) to handle large datasets fast.
- Supports multiple folder pairs for batch processing.

✅ Result

High-quality, trimmed, and clean speech audio files ready for training or analysis.

FEATURE EXTRACTION

🔧 Purpose:
Extract comprehensive audio features (time domain, frequency domain, and statistical) from audio files using a custom class (FeatureExtractor).
🏗️ Key Components:
✅ Initialization Parameters:

- sr: Sample rate (default 22050 Hz)
- frame_size: Window size for analysis (default 2048)
- hop_length: Step size between frames (default 512)

⚙️ Supporting Functions:

- Cached spectrograms to speed up computation
- Exception handling for edge cases (short audio, NaNs)
- Flattening complex features (MFCC, Chroma, etc.) into multiple columns

🎵 Features Extracted:
Time-Domain:

- ZCR: Zero Crossing Rate (signal roughness)

Advanced:

- MFCCs: Mel-frequency cepstral coefficients
-

Statistical/Music Features:

- Chroma: Pitch class profiles
- Pitch: Estimated pitch values

FEATURE EXTRACTION

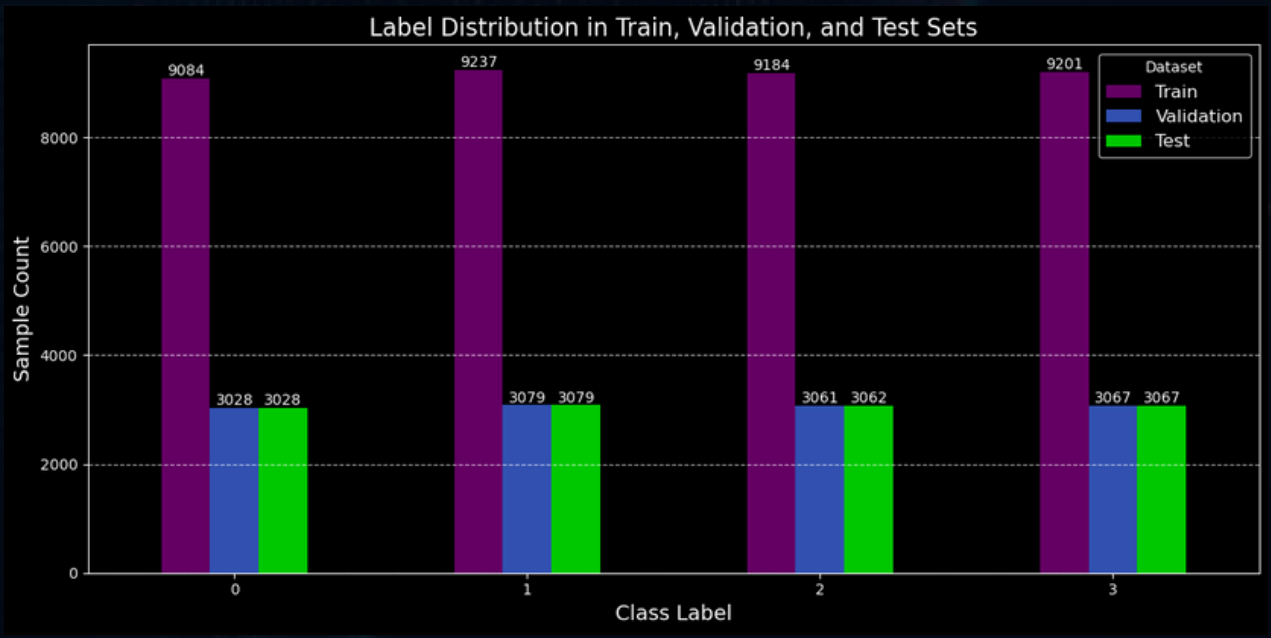
To ensure our model performs fairly across all demographic groups, we carefully handled the balance between classes throughout our dataset preparation process. We defined four distinct classes based on combinations of gender and age:

- Class 0: Male in their twenties
- Class 1: Female in their twenties
- Class 2: Male in their fifties
- Class 3: Female in their fifties

Each class was stored in its own CSV file, allowing us to control the data volume and labeling independently. We assigned numerical labels (0 to 3) and added auxiliary columns for gender and age to support further analysis or downstream tasks.

To ensure a balanced distribution, we split the data from each class individually into training (60%), validation (20%), and testing (20%) sets. This method guarantees that each split contains equal representation from all four classes.

After merging these subsets, we verified the distribution of labels using a bar plot. The results showed that all classes are evenly represented across the train, validation, and test sets. This balance is crucial—it prevents the model from being biased toward any specific demographic group and supports more reliable and fair performance evaluation.

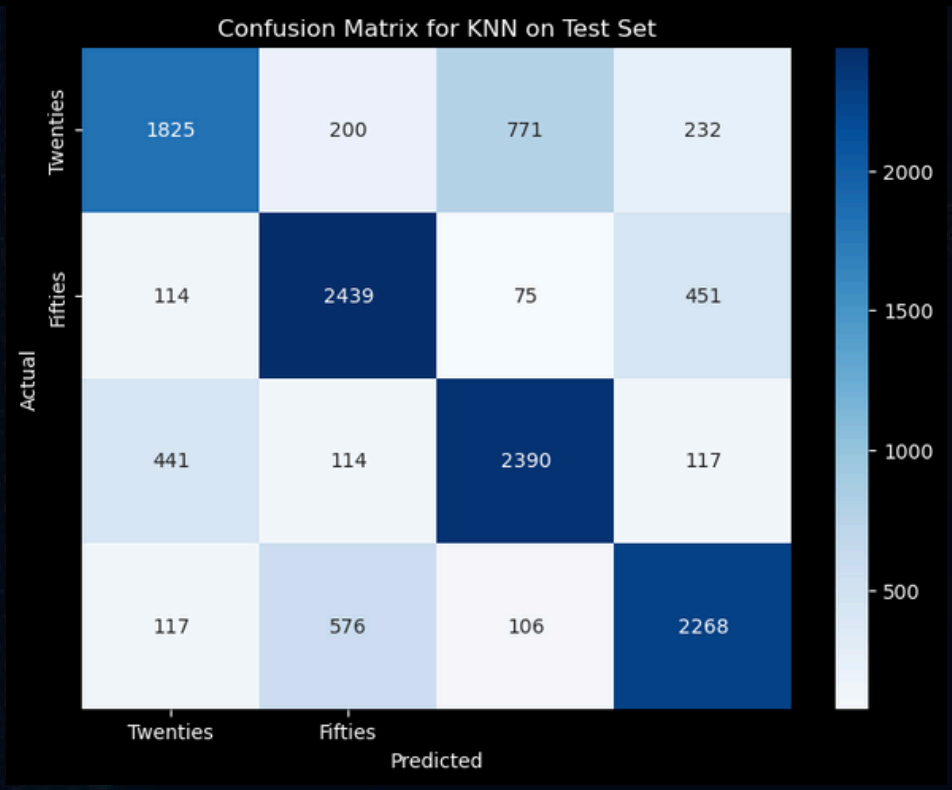


SVM

EWe performed hyperparameter tuning for an SVM classifier using Optuna to maximize validation accuracy. The search space included regularization strength (C), kernel type, gamma, and polynomial degree (when applicable). After 10 trials, we selected the best configuration: C=5.25, kernel='rbf', and gamma='scale'.
We then retrained the best SVM model on the combined training and validation data to maximize generalization. The model was evaluated on the test set, showing its performance through accuracy 76%, a classification report, and a confusion matrix.
Finally, we plotted a learning curve using only the original training data to visualize how the model's performance scales with more data.

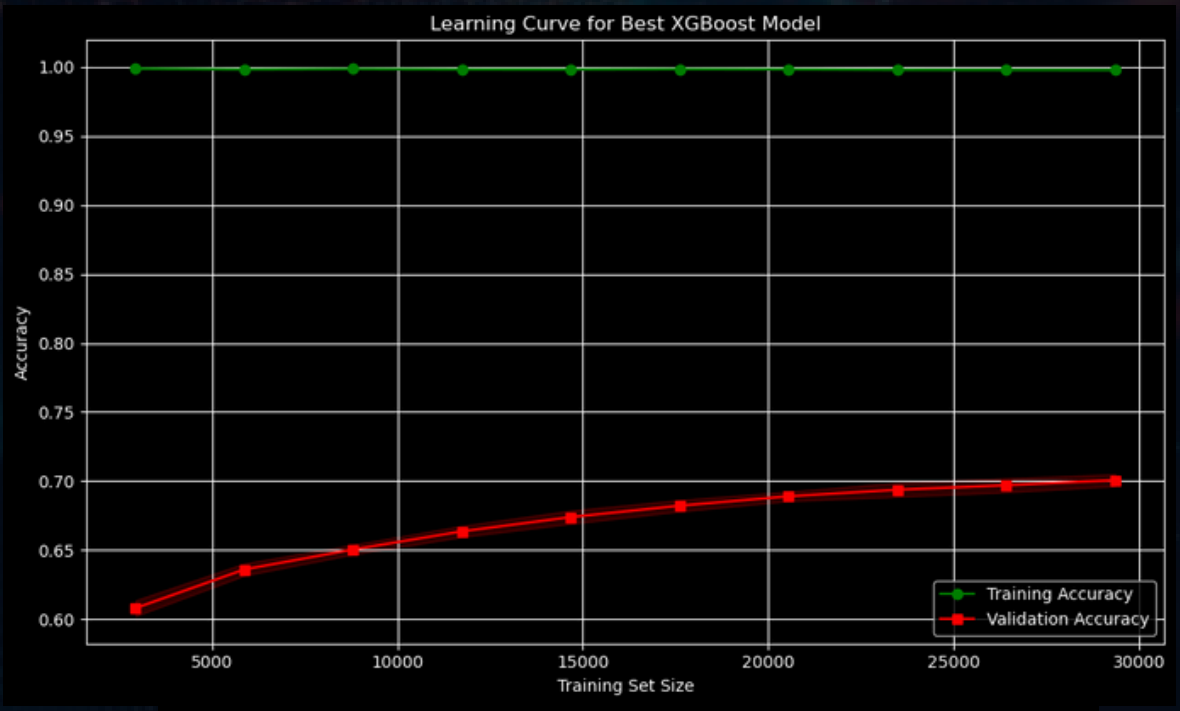
KNN

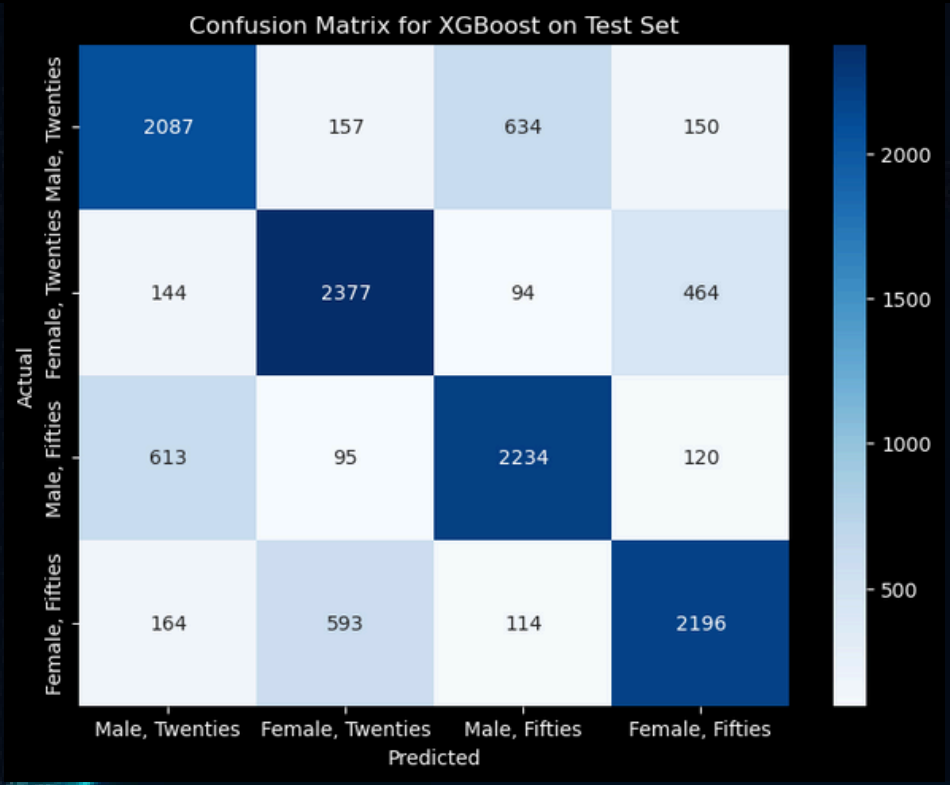
You applied Optuna to tune hyperparameters for a KNN classifier by optimizing validation accuracy over 50 trials. After identifying the best parameters, you retrained the model on the combined training and validation set, then evaluated it on the test set, achieving approximately 73% accuracy. You also visualized performance using a classification report and confusion matrix. Finally, you plotted a learning curve using 5-fold cross-validation on the original training set to analyze how the model's accuracy evolves with increasing training data.



XGBOOST

using Optuna for hyperparameter tuning of an XGBoost classifier, optimizing validation accuracy over 30 trials. After obtaining the best parameters, you retrain the model on the combined training and validation sets. The model is then evaluated on the test set, achieving a test accuracy of around 73%, along with a classification report and confusion matrix for further insights. Additionally, you visualize the model's learning curve using 5-fold cross-validation to assess its accuracy as a function of the training set size. This approach provides a thorough evaluation of XGBoost's performance on your dataset.





LIGHTGM

After tuning hyperparameters with Optuna, the XGBoost model achieved a test accuracy of 73%, while the LightGBM model performed slightly better with a test accuracy of 74%, indicating it handled the data structure and feature interactions more effectively. Both models were evaluated using learning curves and confusion matrices to understand their generalization performance, and the results showed stable training and validation trends, confirming that LightGBM slightly outperformed XGBoost in this classification task.
4o