OPEN PROJECTS (2025)

AI/ML PROBLEM STATEMENTS

1. Design and implement a robust pipeline to perform emotion classification on speech data

Objective

The objective of this project is to design and implement an end-to-end pipeline for emotion classification using speech data. The system will leverage audio processing techniques and machine learning/deep learning models to accurately identify and categorize emotional states conveyed in speech/song.

Dataset

The dataset description and the files are available in below given link.

Note: You don't have to download all files, filenames to be downloaded will be informed in the doubt groups.

Audio Dataset

Evaluation Criteria

Model's accuracy will be judged on validation data.

Conditions:

- Confusion matrix will be the judging criteria.
- F1 score should be greater than 80%.
- Accuracy of each class should be greater than 75%.
- Overall accuracy should be greater than 80%.

Apart from validation data we will be checking the accuracy on our custom test dataset which will not be shared publicly.

Deliverables

1. GitHub repository containing:

   a. ipynb notebook with full running code.

   b. Trained model.
   c. Python script file where model can be tested by feeding test data.
   d. Include a clear and detailed README.md including project description, pre-processing methodology, model pipeline and accuracy metrics.
   e. Demo video of around 2-minutes showing the use-case of web-app.

2. Fully functional web-app hosted using stream-lit which receives audio file as input and returns the classified emotion.

Group Link

2. Automated Meta Data Generation

Objective

This project aims to develop an automated metadata generation system. This system will enhance document discoverability, classification, and subsequent analysis by producing scalable, consistent, and semantically rich metadata.

Key Areas of Focus

The development of this system will involve:

- Automating metadata generation: The system must automatically generate metadata for various unstructured document types.
- Content extraction: Capabilities for extracting text content from diverse formats like PDF, DOCX, and TXT, incorporating optical character recognition (OCR) where necessary, are essential.
- Semantic content identification: The system needs to identify and leverage the most meaningful sections of a document to inform metadata generation.
- Structured metadata creation: The system will be responsible for generating structured metadata outputs.
- User interface development: An intuitive web interface for document upload and metadata viewing will be a core component.
- System deployment: The entire system will be deployed to ensure public accessibility and ease of use.

Deliverables

GitHub repository containing:

- ipynb notebook with full running code
- Include a clear and detailed Readme.md file.
- Demo video of around 2-minutes showing the use-case of web-app

Group Link

WEB DEVELOPMENT PROBLEM STATEMENTS

1. Collaborative whiteboard with real-time drawing

Objective

Design and develop a real-time collaborative whiteboard application that allows multiple users to draw, write, and interact simultaneously, replicating the experience of a physical whiteboard on the web.

Key Features

- Drawing Tools: Pen, shapes (rectangle, circle), text tool, eraser, color picker.
- Real-Time Sync: Real-time updates using WebSocket so all users see changes instantly.
- Multi-User Collaboration: Users can join a session via a shared link and collaborate live.
- Access Control: Create public or private rooms with permissions (edit/view- only).
- Save & Export: Option to save the whiteboard as an image or PDF.
- Canvas Management: Undo/Redo actions and clear canvas button.
- (Optional): Voice chat, or real-time comment/chat sidebar.

Tech Stack Suggestions

- Frontend: React.js or Next.js, HTML5 Canvas or Fabric.js
- Backend: Node.js + Socket.io
- Database: MongoDB or Firebase (for room/user/session data)
- Hosting: Vercel/Netlify (frontend), Render/Heroku (backend)

Submission Instructions

1. GitHub Repository
    a. Push the complete project code to a public GitHub repo.
    b. Include a clear and detailed README.md with:
        i. Project description
        ii. Features
        iii. Tech stack used
        iv. Setup instructions to run the project locally
        v. Deployed demo link

2. Demo Video
    a. Upload a screen-recorded video (5–10 mins) demonstrating all the core features working live.
    b. Upload it to Google Drive or YouTube and share the link

Group Link

2. Data Compression & Decompression Portal

Objective

Design and develop a web application that allows users to upload files and apply various data compression algorithms (such as Huffman coding, Run-Length Encoding, LZ77) to reduce file size, as well as decompress previously compressed files. The system should demonstrate the efficiency of different algorithms by providing compression ratios and allow users to download the processed files.

Key Features

- File Upload: Users can upload any file (text, image, binary).
- Multiple Compression Algorithms: Support for popular algorithms like Huffman Coding, Run-Length Encoding (RLE), LZ77, and so on (at least two).
- Compression & Decompression: Users can compress or decompress files using the chosen algorithm.
- Compression Statistics: Display compression ratio, original size, compressed size, and processing time.
- Download Processed Files: Allow users to download the compressed or decompressed files.
- Algorithm Explanation: Provide brief descriptions of the chosen compression algorithms to educate users.
- Error Handling: Proper feedback for unsupported files or decompression errors.
- Responsive UI: Intuitive frontend with React for easy file management and status updates.

Tech Stack Suggestions

Frontend

- **Framework**: React.js (for a responsive and dynamic UI)
- **Styling**: Tailwind CSS / Material-UI / Bootstrap (for responsive design and UI components)
- **File Handling**: JavaScript FileReader API or third-party libraries (to read and display file metadata)
- **Visualization** (optional for brownie points): Chart.js / D3.js (for compression statistics and visualizations)

Backend

- **Runtime**: Node.js (for efficient file handling and performance)
- **Framework**: Express.js (for creating RESTful APIs)
- **Compression Algorithms**:
    - Implement Huffman Coding, RLE, LZ77 in custom JavaScript modules or WebAssembly (for better performance)

    - Alternatively, use Python (Flask or FastAPI) if leveraging existing compression libraries or more advanced algorithms
- **File Handling**: multer (for file uploads), fs module (for file processing)

Database (optional, if you want to store user sessions/history)

- **MongoDB** (NoSQL, for storing user info, uploaded file metadata, compression logs)
- **Firebase** (if you want a quick, scalable serverless option)

Hosting

- **Frontend**: Vercel / Netlify (easy integration with React apps)
- **Backend**: Render / Heroku / Railway (for deploying Node.js or Python backend)

Submission Instructions

1. GitHub Repository
    a. Push the complete project code to a public GitHub repository.
    b. Include a clear and detailed README.md with:
        i. Project description
        ii. Features
        iii. Tech stack used
        iv. Setup instructions to run the project locally
        v. Deployed demo link

2. Demo Video
    a. Upload a screen-recorded video (5–10 mins) demonstrating all the core features working live.
    b. Upload it to Google Drive or YouTube and share the link.

Group Link

Plagiarism Notice

Submissions found to be copied from GitHub or other teams will be disqualified. All work must be original and done by you/your team. Feel free to use open-source tools/libraries but ensure your implementation and logic are your own.