Title: Internship Report – Data Science

Name: Manoj Dhanawade

Internship Duration: 30-06-2025 to 30-07-2025

Organization: Nullclass Edtech Private Limited

Project Tasks:

- Age Detection
- Animal Detection with GUI
- Emotion Detection through Voice with GUI

1. Introduction

During my Data Science internship at Nullclass Edtech Private Limited, I had the opportunity to work on a series of applied machine learning tasks that closely reflected real-world challenges. This internship was structured to strengthen my practical understanding of data-driven problem-solving using Python and key data science tools.

Over the one-month period, I was assigned three major tasks involving data analysis, machine learning model development, and deployment:

1. Building an age prediction model using facial image data and a pre-trained Convolutional Neural Network (CNN),
2. Designing an animal detection and classification system for both images and videos, with a user interface and special handling for carnivores,
3. Creating a voice-based emotion recognition model restricted to female voices, along with a GUI for audio recording and uploading.

These tasks required end-to-end development from collecting and preparing data to training models and evaluating their performance using metrics such as accuracy, precision, recall, and confusion matrix. In some cases, interactive graphical user interfaces (GUIs) were developed to enhance the usability of the models.

This internship helped bridge the gap between theoretical knowledge and hands-on application. It enabled me to gain valuable experience in areas such as image and audio processing, neural network training, model evaluation, and user interaction design. The skills and insights gained during this internship will serve as a strong foundation for my future work in data science and machine learning.

2. Background

Before starting this internship, I had a foundational understanding of key concepts in data science, including statistics, data preprocessing, machine learning algorithms, and model evaluation techniques. Through online courses and academic learning, I had gained exposure to Python programming, libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, and basic neural network architectures using TensorFlow and Keras.

However, most of this knowledge had been theoretical or limited to small-scale, guided projects. I had not yet applied these skills independently on complete, real-world tasks. This internship provided me the platform to work on full-fledged data science projects right from handling raw datasets to developing and evaluating models and integrating them with user interfaces where required.

Each assigned task brought a new learning curve. For instance:

- In the age detection project, I explored how to fine-tune pre-trained CNNs on custom datasets.
- The animal detection task introduced me to object detection models like YOLO and helped me understand model deployment with graphical interfaces.
- The emotion detection task required working with audio features, voice signal processing, and specialized constraints like detecting only female voices.

Overall, this internship enabled me to apply and expand my prior knowledge, while also pushing me to independently research solutions, troubleshoot errors, and understand the end-to-end workflow of a real data science project.

3. Learning Objectives

The primary objective of this internship was to strengthen my practical skills in the field of data science by working on real-time machine learning projects. While I had prior theoretical knowledge, this internship gave me the opportunity to apply those concepts in realistic problem settings, with minimal guidance—encouraging independent research and critical thinking.

Key Learning Goals:

1. Develop End-to-End Machine Learning Models
    a. Build, train, and evaluate models on diverse data types such as images and audio.
    b. Handle tasks like data preprocessing, feature extraction, and performance tuning.
2. Improve Model Evaluation and Interpretation Skills
    a. Go beyond accuracy by learning to analyze models using confusion matrices, precision, recall, and F1-score.
    b. Understand how to assess and improve model generalization and performance.
3. Explore Advanced Deep Learning Architectures
    a. Fine-tune pre-trained CNNs for facial age prediction.
    b. Use audio features like spectrograms for voice-based emotion recognition.
    c. Work with object detection frameworks like YOLO for multi-object recognition.
4. Build User-Interactive Applications
    a. Create user-friendly GUIs for image, video, and voice-based models using tools like Flask.
    b. Learn how to integrate trained models into applications that accept real user input (file uploads or recordings).
5. Strengthen Independent Problem Solving
    a. Tackle errors and unexpected challenges without relying on mentor support.
    b. Develop the habit of researching documentation, forums, and academic resources to find effective solutions.
6. Apply Version Control and Documentation Best Practices
    a. Organize codebases clearly and professionally.
    b. Maintain documentation via README files and internship reports for reproducibility.

By setting and meeting these objectives, I aimed to become more confident in handling practical data science challenges and deepen my readiness for industry-level Data Science/AI/ML roles.

4. Activities and Tasks

Task 1: Age Detection Using Pretrained CNN (UTKFace Dataset)

Objective:

The primary aim of this task was to create a deep learning-based system capable of predicting a person's age from facial images. A pre-trained CNN model was fine-tuned on the UTKFace dataset to enhance prediction accuracy and reduce training time. The goal was to achieve reliable performance using regression and classification metrics.

Problem Description:

Age estimation from facial images presents several challenges, such as variation in lighting, ethnicity, facial expressions, and image resolution. These factors influence the visual features available for model learning. A robust neural network capable of generalizing across such variations is essential to achieve accurate age predictions.

Dataset Overview:

- Name: UTKFace Dataset
- Description: Contains over 20,000 facial images labeled with age, gender, and ethnicity.
- Age Range Considered: 1 to 119 years
- Source: Kaggle - UTKFace Dataset

The dataset was cleaned to exclude incorrectly labeled or invalid images. It was split into:

- Training Set: 72.25%
- Validation Set: 12.75%
- Testing Set: 15%

Tools and Libraries Used:

- Language: Python 3.10+
- Framework: PyTorch
- Model: ConvNeXt-Base (via timm library)
- Augmentation: Albumentations
- Metrics: scikit-learn
- Development Environment: Jupyter Notebook

Model Development Approach:

1. Data Preprocessing:
    a. Images were resized and cropped to 224×224 resolution.
    b. Augmentations like horizontal flips, brightness-contrast changes, and dropout were applied.
    c. Pixel values were normalized using ImageNet mean and standard deviation.
2. Model Architecture:
    a. A ConvNeXt-Base CNN, pre-trained on ImageNet, was loaded and customized.
    b. The final classification head was replaced with a regression output layer.
    c. Mixed-precision training was enabled using torch.cuda.amp.
3. Training Configuration:
    a. Epochs: 25
    b. Batch Size: 32
    c. Loss Function: Mean Squared Error (MSE)
    d. Optimizer: Adam
    e. Scheduler: ReduceLROnPlateau (for adaptive learning rate)

Evaluation and Performance:

- Final Accuracy: 70.85%
- Regression Metrics:
    - MAE (Mean Absolute Error): [Insert Value]
    - MSE (Mean Squared Error): [Insert Value]
- Classification Metrics:
    - Confusion Matrix
    - Precision, Recall, F1-Score

The training and validation loss curves confirmed stable learning with minimal overfitting.

Challenges and Solutions:

- Data Imbalance: Age groups were unevenly distributed, which required careful shuffling and validation splitting.
- Memory Usage: The large model and batch size required GPU memory optimization using mixed precision training.
- Model Tuning: Hyperparameters like learning rate and dropout were fine-tuned to improve MAE and reduce overfitting.

Learning Outcomes:

- Gained hands-on experience with transfer learning using advanced architectures like ConvNeXt.
- Learned how to structure a robust training and evaluation pipeline for regression tasks.
- Understood the importance of using both classification and regression metrics for deeper model insight.

Conclusion:

The age detection model achieved the target benchmark by crossing 70% accuracy, along with a low error margin. This task offered valuable experience in applying deep learning to real-world data and laid the foundation for understanding facial analysis models.

Task 2: Animal Detection and Carnivore Classification Using YOLOv8 with GUI

Objective:

This task involved building a custom animal detection system using the YOLOv8 object detection framework. The model was trained to recognize 88 different animal classes from both images and videos. The system was also designed to identify carnivorous animals, highlight them in red bounding boxes, and display a pop-up message with their count. A user-friendly web-based GUI was developed to accept input files and visualize results.

Problem Description:

Monitoring animal populations is crucial for biodiversity studies, wildlife conservation, and forest surveillance. Manual analysis of visual data can be time-consuming and inconsistent. Additionally, distinguishing carnivores is often critical for understanding predator behavior and ensuring safety in surveillance systems. This project automates the process by detecting and classifying animals while specifically tagging carnivorous species for special attention.

Dataset Description:

- Dataset Type: Custom YOLOv8-compatible dataset
- Total Classes: 88 animal species
- Label Types: Bounding boxes
- Carnivorous Species Tracked: Lion, Tiger, Wolf, Leopard, Fox, Bear, Shark, Snake, Owl, Eagle, Dog, Cat, etc.
- Split:
    - Training Set: 80%
    - Validation Set: 10%
    - Testing Set: 10%

Tools and Technologies:

- Backend: Flask (Python)
- Frontend: HTML, CSS, JavaScript
- Model: YOLOv8 from Ultralytics
- Other Libraries: OpenCV, Pillow, NumPy, Torch, Matplotlib

System Architecture:

- Frontend:
    - Allows the user to upload images or videos
    - Displays the output with bounding boxes and carnivore count

- Backend:
  - Handles file upload and model inference
  - Draws bounding boxes using predefined class mappings
- Model Weights:
  - Stored as best.pt (YOLOv8 trained weights)

Methodology:

1. Model Training:
   a. YOLOv8 was fine-tuned on the annotated dataset for 88 classes.
   b. Used bounding box loss and confidence thresholding for multi-animal detection.
2. Carnivore Tagging:
   a. A dictionary of carnivorous species was predefined.
   b. During inference, any detected class that matched a carnivore was marked in red.
   c. A counter kept track of total carnivores in the frame.
3. Web Application (GUI):
   a. Built using Flask (backend) and HTML/CSS/JS (frontend).
   b. Users could upload an image or video, then receive:
      i. Annotated preview
      ii. Carnivores count in a browser pop-up
      iii. Saved output file (image/video with detections)

Challenges Faced:

- Managing real-time frame rendering with Flask for larger video files
- Distinguishing similar species (e.g., wolf vs. dog)
- Optimizing detection accuracy while maintaining speed for GUI rendering

Learning Outcomes:

- Gained in-depth experience in object detection with YOLOv8
- Developed skills in integrating ML models with web-based interfaces
- Understood how to manage frontend-backend communication for real-time file processing

Conclusion:

This project successfully implemented a custom animal detection system with a focus on identifying carnivores. The integration of a GUI enhanced usability, while the detection pipeline proved effective in processing both image and video inputs. The system has real-world applicability in wildlife tracking, surveillance, and conservation technology.

Task 3: Emotion Detection from Female Voice Using Deep Learning with GUI

Objective:

The purpose of this task was to develop an intelligent system capable of detecting emotions from audio input. The system begins by identifying the speaker's gender. If the input is from a female voice, it proceeds to classify the emotional state. Otherwise, it prompts the user to upload a valid female voice. The solution integrates gender detection, emotion recognition, and a user-friendly graphical interface for interaction.

Problem Description:

Most emotion recognition systems do not distinguish between male and female voices, even though vocal characteristics significantly differ across genders. This can reduce prediction accuracy. To address this, the system was designed to filter out non-female voices to maintain reliable emotion detection performance.

Scope of the Project:

- Accepts audio input via upload or live browser recording
- Implements gender classification using a CNN model
- Restricts processing to female voices only
- Uses a CRNN model to classify emotions like happy, sad, angry, neutral, etc.
- Displays results in a GUI pop-up, providing immediate feedback to users

Datasets Used:

1. Gender Detection
    a. Dataset: Gender Recognition by Voice
2. Emotion Detection
    a. Dataset: RAVDESS – Emotional Speech Audio

Technologies and Libraries:

- Programming Language: Python 3.10
- Framework: Flask (for backend and routing)
- Frontend: HTML5, CSS3, JavaScript
- Audio Processing: Librosa
- Deep Learning: TensorFlow, Keras
- Model Types:
    - CNN for gender detection

    - CRNN for emotion classification

System Architecture:

- Frontend:
    - index.html for user interaction
    - recorder.js for browser-based audio recording
    - style.css for design and responsiveness
- Backend:
    - Flask app to handle input and predictions
    - utils.py for audio feature extraction and model inference
    - Models saved as .keras files in the saved_models/ directory
- Models:
    - gender_model.keras for gender prediction
    - emotions_model.keras for emotion classification

Workflow / Methodology:

1. Audio Collection:
    a. Users upload or record audio through the browser.
2. Gender Detection:
    a. The CNN model analyzes the input audio.
    b. If male: a prompt appears requesting a female voice.
    c. If female: the process moves to the next stage.
3. Emotion Classification:
    a. A CRNN model predicts the emotional state.
    b. The result is shown to the user in a pop-up message box.
4. Model Output:
    a. Gender: Male/Female
    b. Emotion: Happy, Sad, Angry, Neutral, etc.

Challenges Faced:

- Differentiating emotions when audio quality was poor or noisy
- Ensuring real-time processing within the browser-recorded clips
- Integrating multiple models in a single pipeline with minimal latency

Learning Outcomes:

- Learned how to extract meaningful features from audio using MFCC/Spectrograms
- Gained experience in gender classification and CRNN-based emotion modeling
- Understood integration of ML models with full-stack web applications

Conclusion:

This project resulted in a working prototype that combines gender detection with emotion classification using voice input. The system ensures that only female voices are processed for emotion analysis, enhancing accuracy and clarity. The final implementation serves as a strong foundation for further applications in emotion-aware systems and voice-driven interfaces.

5. Skills and Competencies Developed

During the internship, I had the opportunity to work on real-world problems that significantly contributed to my technical and professional growth. Through independent work and consistent experimentation, I developed a wide range of valuable skills.

Technical Skills:

- Implemented and fine-tuned deep learning models using frameworks like TensorFlow, PyTorch, and Keras.
- Carried out data preprocessing for diverse data types—images, videos, and audio signals.
- Applied model evaluation techniques including MAE, MSE, precision, recall, F1-score, and mAP.
- Used libraries such as OpenCV and Librosa for visual and audio feature extraction.
- Integrated AI models into user-friendly interfaces using Flask, HTML, CSS, and JavaScript.

Project Development Skills:

- Designed modular project structures with clear separation between backend logic, frontend interface, and saved models.
- Built end-to-end pipelines, from dataset loading and model training to final predictions and GUI display.
- Practiced saving, reusing, and deploying trained models with proper version control.

Soft Skills:

- Strengthened independent learning and self-discipline through research-driven development.
- Enhanced critical thinking by solving implementation bugs and refining model performance.
- Developed time management habits by maintaining regular progress and meeting deadlines.

6. Feedback and Evidence

To ensure transparency and track progress throughout the internship, the following practices were consistently followed:

- Daily work updates were recorded using the provided reporting portal.
- All project code, training notebooks, and GUI files were organized into dedicated folders.

- Results were documented using metrics and graphs, such as confusion matrices and training loss plots.
- Trained model weights were saved and uploaded to Google Drive for evaluation.
- Screenshots of GUI outputs, inference results, and performance reports were retained as evidence.

All artifacts, including model files, source code, and documentation, have been submitted via GitHub and linked for review.

7. Challenges and Solutions

During the internship, I encountered a variety of technical and practical difficulties. Each challenge taught me something new and helped strengthen my problem-solving skills. Below are the key challenges I faced and how I addressed them:

- Challenge 1: Imbalanced Age Distribution in the Dataset
  The UTKFace dataset had a skewed age distribution, which affected the age prediction model's accuracy.
  Solution: I applied proper data shuffling and ensured a balanced validation set. I also used evaluation metrics like MAE and MSE instead of just accuracy.
- Challenge 2: Slow Video Processing in Animal Detection GUI
  Processing video files in real time caused noticeable lag in the GUI.
  Solution: I optimized the video input resolution and reduced the frame rate for smoother processing. The detection code was also optimized for better efficiency.
- Challenge 3: Background Noise in Audio Files
  Background noise in the uploaded or recorded voice clips negatively impacted gender and emotion classification.
  Solution: I used audio preprocessing techniques such as silence trimming and noise reduction using Librosa to improve input quality.
- Challenge 4: Incorrect Gender Classification Due to Voice Variation
  Some recordings led to misclassification of gender due to voice pitch and noise artifacts.
  Solution: I improved model robustness by training on clearer samples and applied normalization techniques during feature extraction.
- Challenge 5: GUI Freezing During Model Prediction
  In some cases, the interface became unresponsive during model execution, especially with video inputs.
  Solution: I separated backend processing from the GUI using optimized Flask routing and minimal frontend blocking to maintain a smooth user experience.

Each of these challenges provided an opportunity to think critically, search for solutions independently, and apply them effectively.

8. Outcomes and Impact

At the end of the internship, I was able to complete and deliver all three assigned tasks with fully working implementations and satisfactory accuracy levels. Each task resulted in a practical solution that meets user interaction, performance, and functionality expectations.

Results Achieved:

- All models met the minimum performance benchmarks (70% accuracy or higher).
- Functional GUIs were developed for both image/video and audio-based applications.
- Trained models were successfully saved, tested, and shared using Google Drive and GitHub.

Personal Growth:

- Improved understanding of deep learning workflows and deployment methods.
- Gained experience in building complete AI systems, from training to real-world interaction.
- Developed confidence to independently plan, execute, and document data science projects.

9. Conclusion

This internship offered a valuable opportunity to transform theoretical concepts into hands-on applications. Working on age prediction, animal detection, and voice-based emotion recognition taught me how to manage real-world challenges while applying machine learning techniques.

Beyond technical growth, I learned how to document my work effectively, evaluate models with appropriate metrics, and design usable interfaces. The internship has not only enhanced my portfolio but also prepared me for future roles in data science and AI development. I am grateful for this experience and look forward to applying these skills to more complex and impactful projects in the future.