A Capston Project report submitted

in partial fulfillment of requirement for the award of degree

**BACHELOR OF TECHNOLOGY**

in

**SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE**

by

**2203A52158 Mamidisetti Prabhu Kumar**

Under the guidance of

**Dr.Ramesh Dadi**

Assistant Professor, School of CS&AI.


SR University, Ananthsagar,Warangal,Telagnana-506371

## CONTENTS

# CHAPTER 1
# DATASET

**Project -1**

The Road Traffic Accident (RTA) dataset is derived from manually recorded accident reports collected between 2017 and 2020. All sensitive and personally identifiable information has been removed to ensure data privacy. The original dataset, saved as "RTA Dataset.csv," contains 32 features and 12,316 instances detailing various aspects of each accident, such as time, location, weather conditions, vehicle types, and causes of the accidents. After a thorough preprocessing phase—which involved cleaning missing data, encoding categorical values, and removing inconsistencies—a refined version named "cleaned.csv" was produced, making it suitable for machine learning tasks. The primary goal of this dataset is to identify and analyze the major causes of road traffic accidents using various classification algorithms, helping to uncover patterns that could inform safety measures and policy decisions.

**Project – 2**

This **weather conditions** dataset consists of **6,862 images** categorized into **11 distinct weather conditions**, making it ideal for developing and evaluating image-based **weather classification models**. The images are labeled according to weather types, including **dew, fog/smog, frost, glaze, hail, lightning, rain, rainbow, rime, sandstorm, and snow**, providing a diverse range of atmospheric scenarios. Each image captures visual characteristics unique to its respective weather condition, enabling models to learn and distinguish patterns based on texture, color, and other visual cues. This dataset serves as a valuable resource for computer vision tasks such as **image classification, environmental monitoring, and automated weather detection systems**.

**Project – 3**

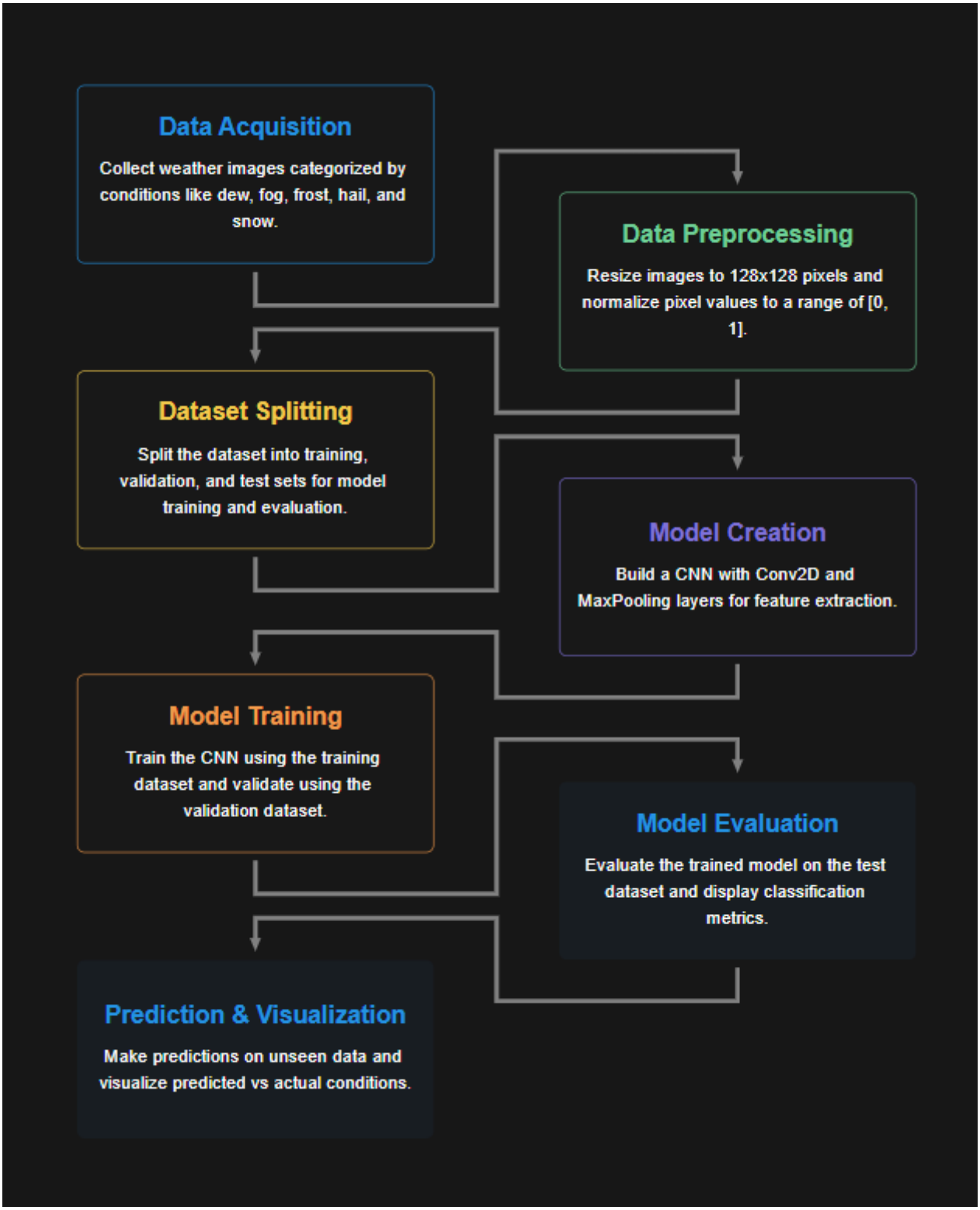
The Snoring Dataset is designed to support audio classification tasks by providing a balanced collection of **1,000 one-second audio clips**, evenly split between snoring and non-snoring sounds. The dataset is organized into two folders: **Folder 1** contains **500 snoring samples**, including **363 clean snoring clips** from children, adult men, and women, and **137 samples with background noise**, simulating more realistic acoustic environments. **Folder 0** includes **500 non-snoring samples** representing ambient sounds commonly found near a snorer. These are grouped into **10 distinct categories**—such as baby crying, clock ticking, door movement, silence, gadget vibrations, toilet flushing, emergency sirens, rain/thunderstorms, streetcars, conversation, and television news—with **50 samples per category**. This dataset is ideal for developing and evaluating snore detection systems and other audio-based health monitoring applications.

# CHAPTER 2

# FLOWCHART

**Project-1**

**Project – 2**


Data Acquisition
Collect weather images categorized by conditions like dew, fog, frost, hail, and snow.
Data Preprocessing
Resize images to 128x128 pixels and normalize pixel values to a range of [0, 1].
Dataset Splitting
Split the dataset into training, validation, and test sets for model training and evaluation.
Model Creation
Build a CNN with Conv2D and MaxPooling layers for feature extraction.
Model Training
Train the CNN using the training dataset and validate using the validation dataset.
Model Evaluation
Evaluate the trained model on the test dataset and display classification metrics.
Prediction & Visualization
Make predictions on unseen data and visualize predicted vs actual conditions.

**Project -3**


Data Collection
Gather audio files from two folders: snoring audio files and non-snoring audio files.
Feature Extraction
Extract Mel Frequency Cepstral Coefficients (MFCC) from each audio file for analysis.
Preprocessing
Normalize MFCC features and encode labels into numeric format for training.
Data Splitting
Split the dataset into training (80%) and testing (20%) sets for model evaluation.
Model Training
Train an LSTM model using the preprocessed features, setting appropriate layers and functions.
Model Evaluation
Evaluate the model using the test data, calculating accuracy and generating a confusion matrix.
Prediction
Use the trained model to classify new audio files as snoring or non-snoring after preprocessing.

# METHODOLOGY

## Project – 1

### 1. Data Acquisition

The dataset used for this study was sourced from manually recorded road traffic accident data spanning from 2017 to 2020. The raw data contained 32 features and 12,316 instances. Sensitive information was removed prior to analysis, ensuring compliance with privacy standards. The dataset was loaded from a CSV file named Road.csv.

### 2. Data Preprocessing

Data preprocessing was essential to convert the raw data into a structured and analyzable format:

Time Conversion: The 'Time' column was converted into datetime format, and the hour of the day was extracted for temporal analysis.

Handling Missing Values: All missing values were imputed using the mode of each column to maintain consistency in categorical features.

Categorical Encoding: Label Encoding was applied to convert categorical features into numerical representations for model compatibility.

### 3. Exploratory Data Analysis (EDA)

To understand the underlying patterns and distributions in the dataset, several exploratory analyses were performed:

Statistical Analysis: Computation of mean, standard deviation, skewness, kurtosis, and variance for numerical features.

Visualizations: Scatter plots for feature correlations, histograms and box plots for distributional insights, and a time series line plot for casualty trends by hour.

### 4. Outlier Detection and Removal

Outliers were identified and removed using the Interquartile Range (IQR) method:

This helped in reducing noise and improving model performance.

Only numerical features were considered for outlier detection to maintain categorical integrity.

### 5. Feature Scaling

StandardScaler was used to normalize the numerical features, ensuring uniform scaling across all features. This was particularly crucial for models sensitive to feature magnitudes, such as Support Vector Machines (SVM).

### 6. Machine Learning Model Training

Three classification models were implemented to predict the Accident Severity:

- Support Vector Machine (SVM)
- Random Forest Classifier
- Decision Tree Classifier

The dataset was split into training and testing subsets with an 80-20 ratio using stratified sampling.

**7. Model Evaluation**

Model performance was assessed using:

Accuracy Score to determine the percentage of correct predictions.

Classification Reports, which provided metrics such as precision, recall, and F1-score for each class.

Model evaluation was conducted both before and after outlier removal to examine its impact.

**8. Model Comparison and Selection**

Finally, all models were compared based on their accuracy and classification metrics. This helped in identifying the most suitable model for accident severity prediction based on the given data characteristics.

# Project -2

**1. Data Collection and Preprocessing**

a. Data Acquisition:

- The dataset contains weather-related images categorized by weather conditions such as dew, fog, frost, hail, lightning, rain, rainbow, rime, sandstorm, snow, and glaze.
- The images are stored in separate folders, each labeled with the corresponding weather condition.

b. Image Resizing and Normalization:

- Images are resized to a consistent size (e.g., 128x128 pixels) to ensure uniformity and optimize the input size for the model.
- Each image is normalized by scaling pixel values to a range of [0, 1] by dividing pixel values by 255. This normalization improves model convergence during training.

c. Label Encoding:

- Each weather condition is assigned a numerical label (e.g., 0 for 'dew', 1 for 'fog', etc.), which is used for classification tasks.

**2. Dataset Splitting**

Training Set: The largest portion of the dataset is used to train the model. It is essential that this data contains a variety of examples for each class to help the model generalize.

Validation Set: A smaller portion of the dataset is reserved for validation, helping to tune hyperparameters and detect overfitting during training.

Test Set: This set is used to evaluate the final model performance after training and tuning.

**3. Model Architecture**

Convolutional Neural Network (CNN): The model is based on CNNs, which are well-suited for image classification tasks. The architecture consists of:

Convolutional Layers: These layers use filters to extract features from the images.

Max Pooling Layers: These layers reduce the spatial dimensions (height and width) of the image while preserving important features.

Flatten Layer: After feature extraction, the 2D output from the convolutional layers is flattened into a 1D vector.

Dense Layers: Fully connected layers are added to make predictions based on the extracted features.

Output Layer: The final layer uses softmax activation for multi-class classification, which outputs the predicted weather condition.

**4. Model Training**

The model is trained on the training dataset using an Adam optimizer and a sparse categorical cross-entropy loss function, which is appropriate for multi-class classification tasks.

The model is trained for a fixed number of epochs (e.g., 10) with a batch size of 32.

The validation set is used to monitor the model's performance during training to avoid overfitting and adjust the model as needed.

**5. Model Evaluation**

After training, the model is evaluated on the test dataset to assess its accuracy and generalization.

Classification Metrics: Evaluation includes metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

A higher accuracy indicates better model performance, while the other metrics provide insight into how well the model handles class imbalances and misclassifications.

**6. Prediction and Visualization**

The trained model is used to predict the weather condition of new, unseen images.

Visualization: For a better understanding, a few sample predictions are visualized with the true label and predicted label displayed on the images.

**7. Comparison with Grayscale Images**

The model is re-trained using grayscale images to check whether reducing the image complexity affects performance.

The results of the grayscale model are compared to the RGB model to analyze if the color information is crucial for classification**.**

**8. Model Deployment**

After training, the models (both RGB and grayscale) are saved as .h5 files, which can be loaded for future predictions.

The model is deployed to predict weather conditions from user-uploaded images in a web or application interface. The user uploads an image, and the model predicts the weather condition based on the image content.

**9. Future Improvements**

Data Augmentation: To further improve the model's generalization, techniques like rotation, flipping, or adding noise can be used to augment the training data.

Advanced Architectures: Models such as ResNet or Inception can be tested for performance improvements.

Real-time Prediction: The model can be deployed for real-time predictions using webcam feeds or live image sources.

# Project – 3

**1. Data Collection:**

Data Source: The dataset is obtained from two distinct folders:

Snoring Audio Folder: Contains audio files of snoring sounds.

Non-Snoring Audio Folder: Contains audio files of non-snoring sounds.

The audio files are in .wav format and are used for the classification task.

**2. Feature Extraction:**

Mel Frequency Cepstral Coefficients (MFCC):

MFCCs are extracted from each audio file using the librosa library.

MFCCs represent the short-term power spectrum of sound and are commonly used in speech and audio classification tasks.

13 MFCC coefficients are extracted for each audio file.

The MFCCs are averaged across all frames of the audio to produce a single feature vector for each file.

**3. Data Preprocessing:**

Normalization: The MFCC features are normalized to scale them to a range of [0, 1], making them suitable for training machine learning models.

Label Encoding: The labels are encoded into numeric format using the LabelEncoder from scikit-learn:

Snoring → 1

Non-Snoring → 0

**4. Data Splitting:**

The dataset is split into training and testing sets:

Training Set (80%)

Testing Set (20%)

This allows the model to learn from one subset and evaluate performance on an unseen subset.

**5. Model Training:**

Model Selection: A Long Short-Term Memory (LSTM) neural network model is used to classify the audio data. LSTM is well-suited for sequential data like audio signals.

The model consists of:

An LSTM layer to capture the temporal dependencies in the MFCC features.

A Dropout layer for regularization to prevent overfitting.

A Dense layer for output with a sigmoid activation function (binary classification).

Compilation: The model is compiled using:

Adam optimizer for efficient gradient descent.

Binary cross-entropy loss function for binary classification.

Accuracy as the evaluation metric.

Training: The model is trained for 50 epochs, using batches of 32 samples.

**6. Model Evaluation:**

Test Set Evaluation: After training, the model is evaluated on the test set to measure its performance.

The model's accuracy is calculated, and it is printed.

Confusion Matrix: A confusion matrix is generated to assess the classification performance (True Positive, True Negative, False Positive, False Negative).

Classification Report: The model's precision, recall, and F1-score are computed to evaluate its classification performance.

Receiver Operating Characteristic (ROC) Curve: The ROC curve and Area Under the Curve (AUC) score are plotted to evaluate the model's ability to discriminate between the two classes.

**7. Prediction:**

Prediction on New Data: The trained model can be used to predict whether a new audio file is snoring or non-snoring.

Feature extraction is performed on the new audio file.

The feature vector is reshaped to match the input shape expected by the model, and then predictions are made.

Thresholding: If the output is greater than 0.5, the audio is classified as "snoring"; otherwise, it is classified as "non-snoring".

**8. Model Deployment:**

Model Saving: The trained model is saved as a .h5 file for later use.

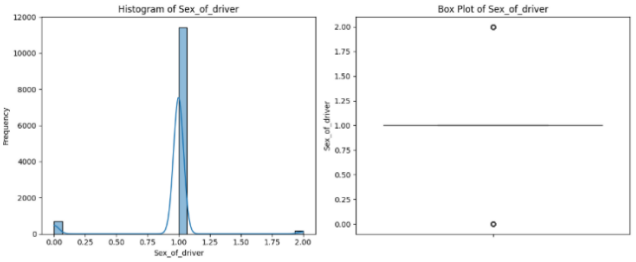
Model Loading: The saved model can be loaded to make predictions on new uploaded audio files.
User Interface: An interface is provided for users to upload audio files, which are then classified using the trained model.

# CHAPTER 3

# RESULTS

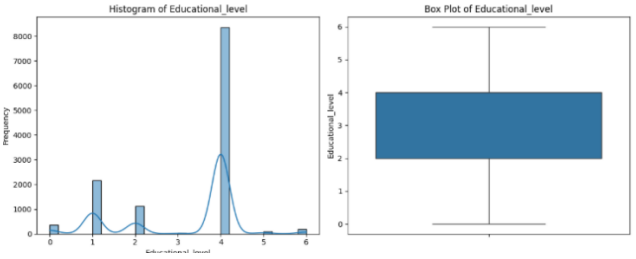
## Project – 1

Histograms and Box Plots in between features



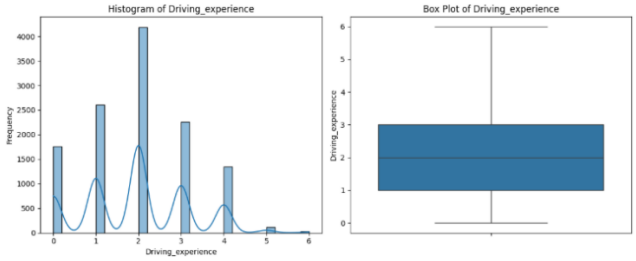






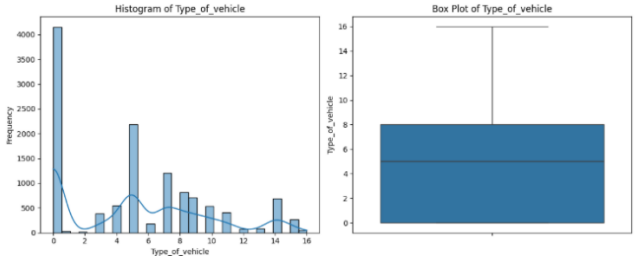






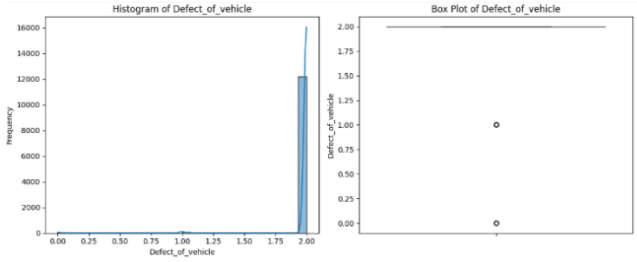






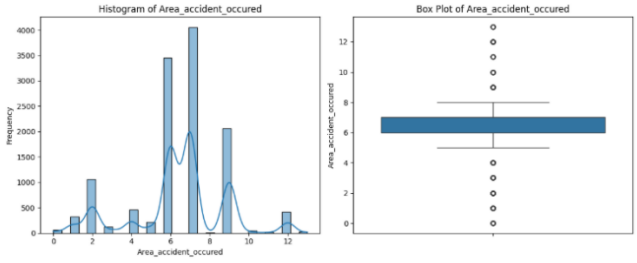






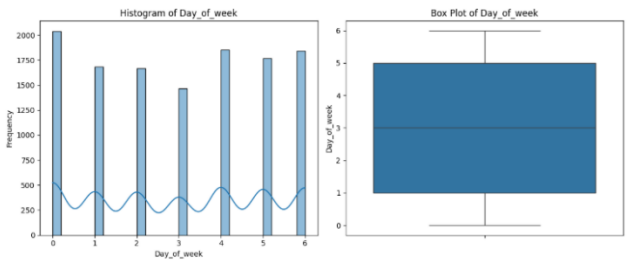






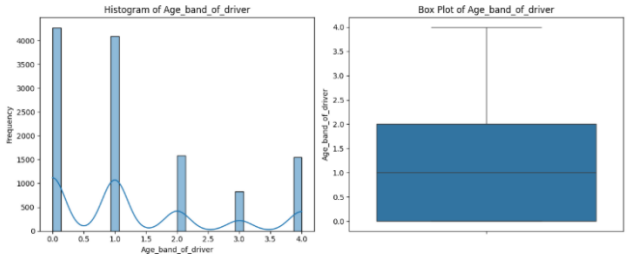






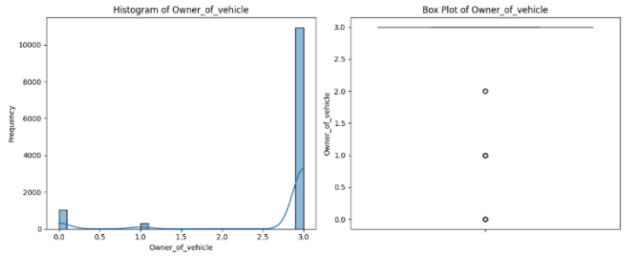






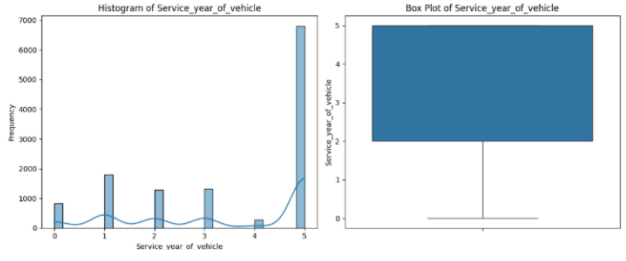






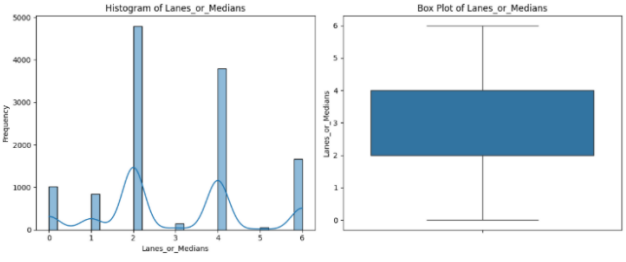






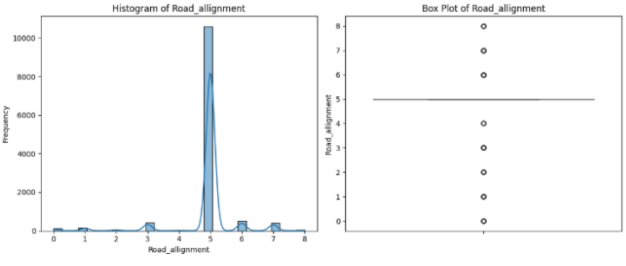


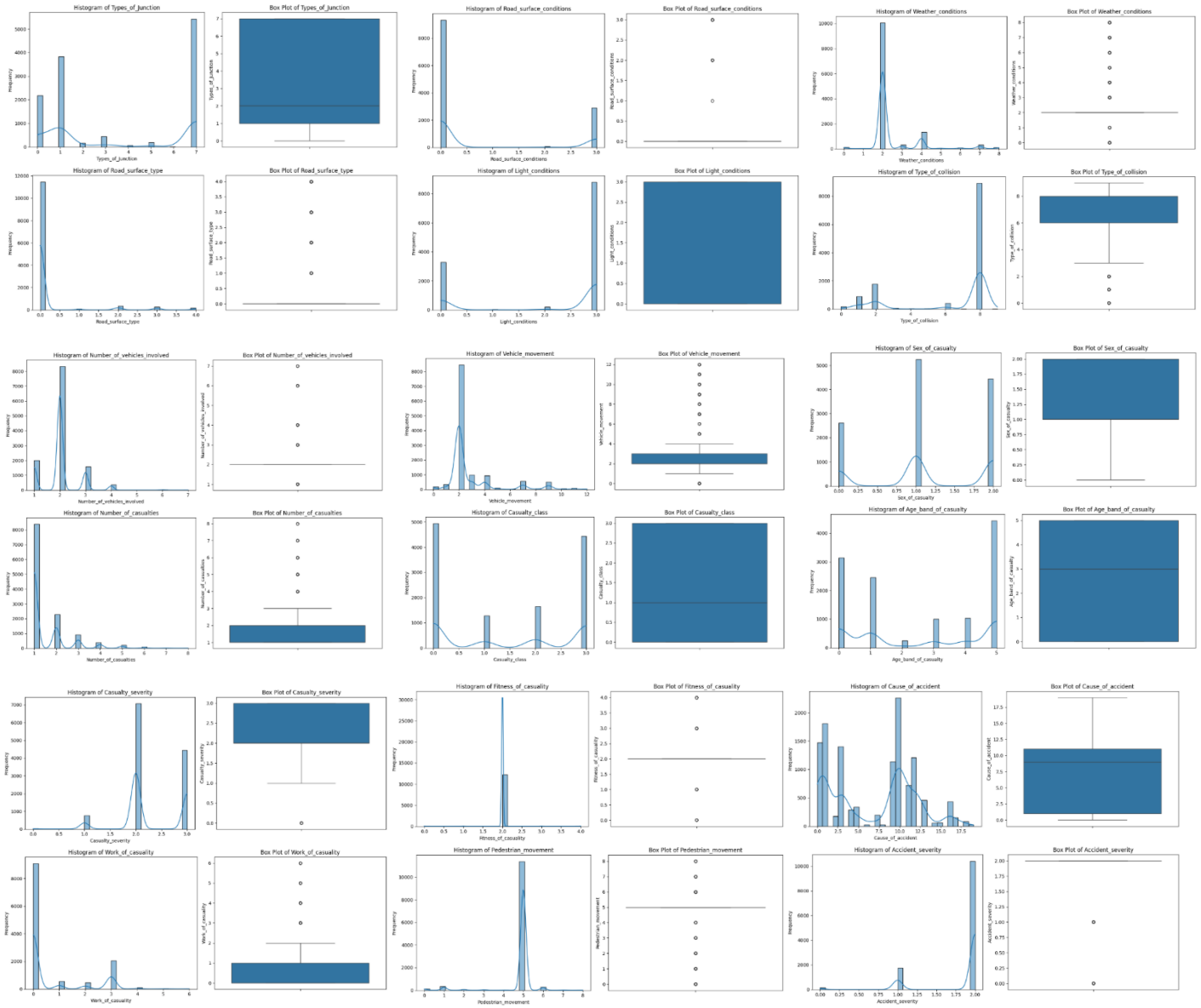




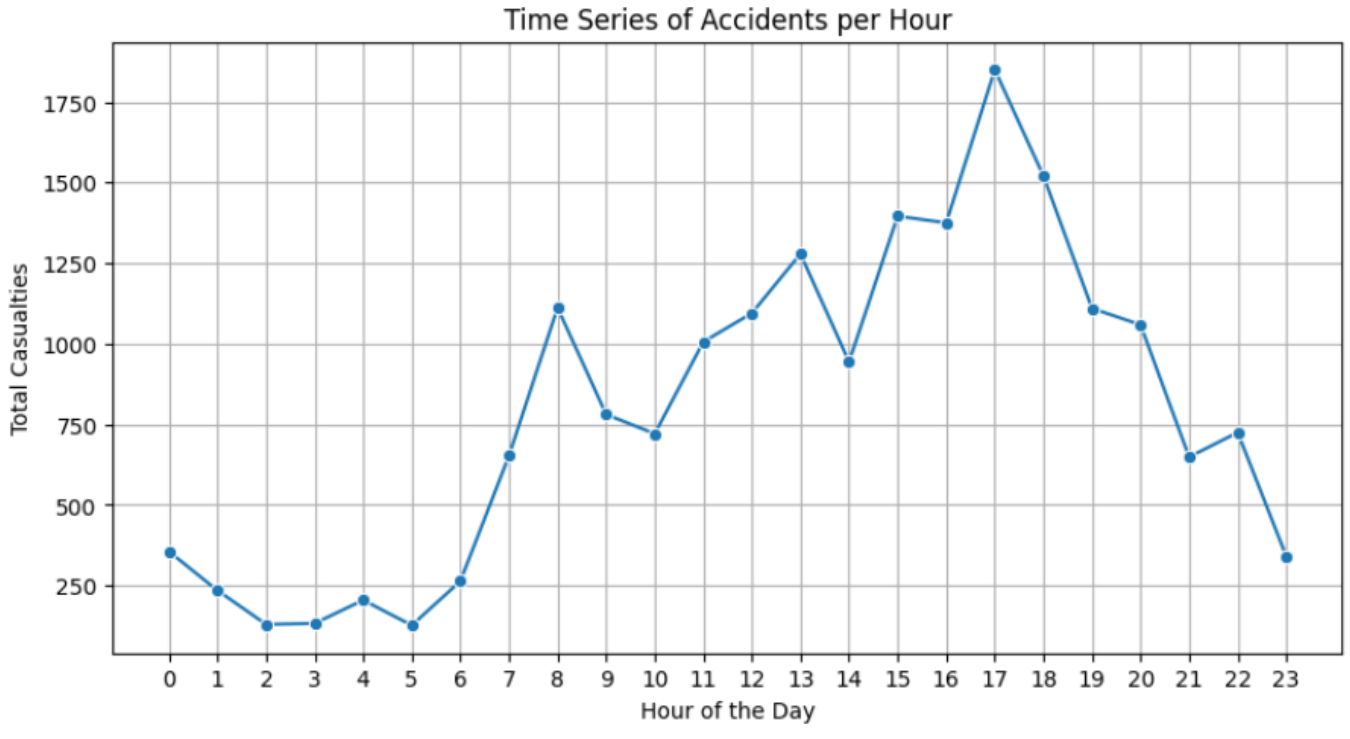









## Graph of Accidents per hour:



**SVM:**

| | precision | recall | F1-score | support |
|---|---|---|---|---|
| 0 | **0.00** | **0.00** | **0.00** | **37** |
| **1** | **0.00** | **0.00** | **0.00** | **363** |
| **2** | **0.84** | **1.00** | **0.91** | **2064** |
| **accuracy** | | | **0.84** | **2464** |
| **Macroavg** | **0.28** | **0.33** | **0.30** | **2464** |
| **Weighted avg** | **0.70** | **0.84** | **0.76** | **2464** |

The SVM model is heavily biased toward the dominant class (class 2) and fails to classify minority classes (0 and 1).

Accuracy is not a good measure in this imbalanced scenario.

Consider the following improvements:

Class balancing techniques (SMOTE, class weights, oversampling).

One-vs-Rest SVM approach.

Try different kernels or optimize hyperparameters.

**After removing Outliers**

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **0** | **0.00** | **0.00** | **0.00** | **2** |
| **1** | **0.00** | **0.00** | **0.00** | **21** |
| **2** | **0.88** | **1.00** | **0.94** | **172** |
| **Accuracy** | | | **0.88** | **195** |
| **Macro Avg** | **0.29** | **0.33** | **0.31** | **195** |
| **Weighted Avg** | **0.78** | **0.88** | **0.83** | **195** |

**Random Forest:**

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **0** | **0.00** | **0.00** | **0.00** | **37** |

| | | | | |
|---|---|---|---|---|
| **1** | **0.90** | **0.02** | **0.05** | **363** |
| **2** | **0.84** | **1.00** | **0.91** | **2064** |
| **Accuracy** | | | **0.84** | **2464** |
| **Macro Avg** | **0.58** | **0.34** | **0.32** | **2464** |
| **Weighted Avg** | **0.84** | **0.84** | **0.77** | **2464** |

The model performs exceptionally well on the majority class but performs poorly on minority classes (especially class 0).

You may need to apply techniques like class balancing (e.g., SMOTE, undersampling), cost-sensitive learning, or ensemble methods to improve performance across all classes.

**After removing Outliers**

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **0** | **0.00** | **0.00** | **0.00** | **2** |
| **1** | **0.00** | **0.00** | **0.00** | **21** |
| **2** | **0.88** | **1.00** | **0.94** | **172** |
| **Accuracy** | | | **0.88** | **195** |
| **Macro Avg** | **0.29** | **0.33** | **0.31** | **195** |
| **Weighted Avg** | **0.78** | **0.88** | **0.83** | **195** |

**Decision Tree:**

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **0** | **0.29** | **0.35** | **0.32** | **37** |
| **1** | **0.26** | **0.28** | **0.27** | **363** |
| **2** | **0.87** | **0.85** | **0.86** | **2064** |
| **Accuracy** | | | **0.76** | **2464** |
| **Macro Avg** | **0.47** | **0.49** | **0.48** | **2464** |
| **Weighted Avg** | **0.77** | **0.76** | **0.76** | **2464** |

The model shows high performance for the majority class (class 2) but fails to generalize well for the minority classes (0 and 1).

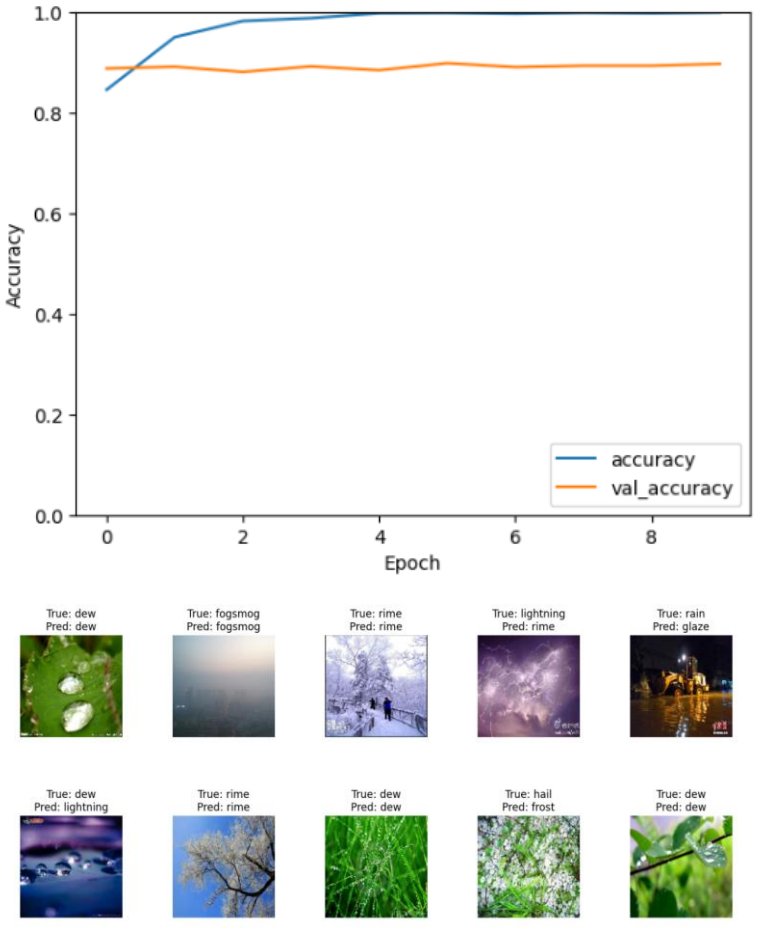
The data appears to be imbalanced, and techniques like oversampling, undersampling, or using class weights might improve model performance on the underrepresented classes.

Overall, while the accuracy seems decent, the macro metrics suggest the need for improvement in class-wise balance and fairness.

**After removing Outliers**

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0.00 | 0.00 | 0.00 | 2 |
| 1 | 0.04 | 0.05 | 0.04 | 21 |
| 2 | 0.88 | 0.85 | 0.86 | 172 |
| Accuracy | | | 0.76 | 195 |
| Macro Avg | 0.30 | 0.30 | 0.30 | 195 |
| Weighted Avg | 0.78 | 0.76 | 0.77 | 195 |

# Project – 2

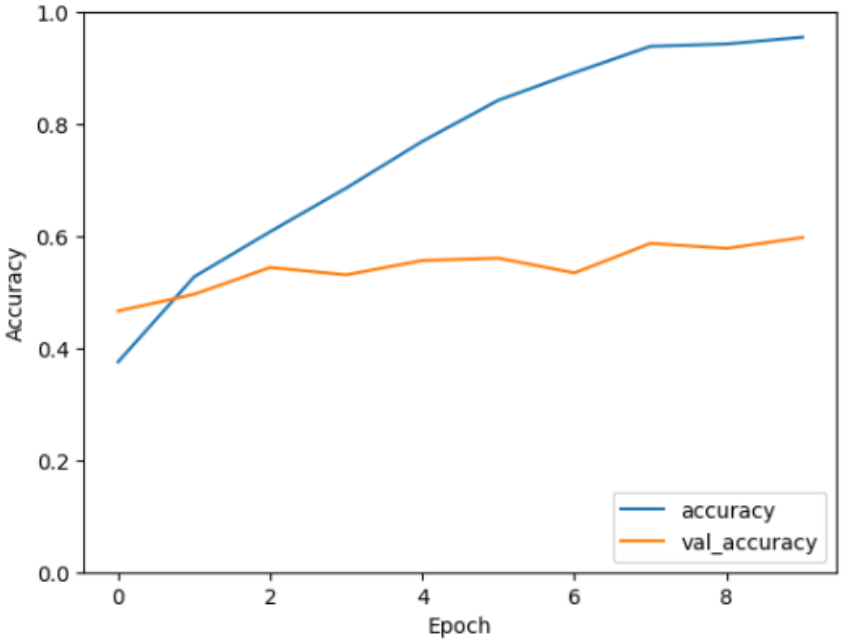


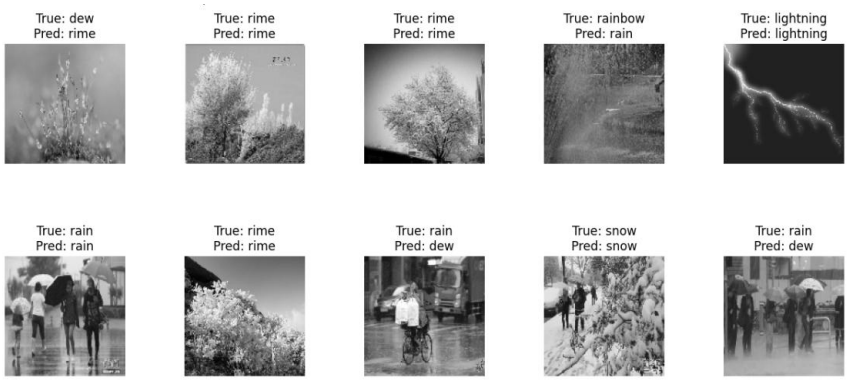

**(RGB)Accuracy: 0.9043**

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 126, 126, 32) | 896 |
| max_pooling2d (MaxPooling2D) | (None, 63, 63, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 61, 61, 64) | 18,496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 30, 30, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 28, 28, 128) | 73,856 |
| max_pooling2d_2 (MaxPooling2D) | (None, 14, 14, 128) | 0 |
| flatten (Flatten) | (None, 25088) | 0 |
| dense (Dense) | (None, 128) | 3,211,392 |
| dense_1 (Dense) | (None, 11) | 1,419 |

**Total params:** 3,306,059 (12.61 MB)
**Trainable params:** 3,306,059 (12.61 MB)
**Non-trainable params:** 0 (0.00 B)

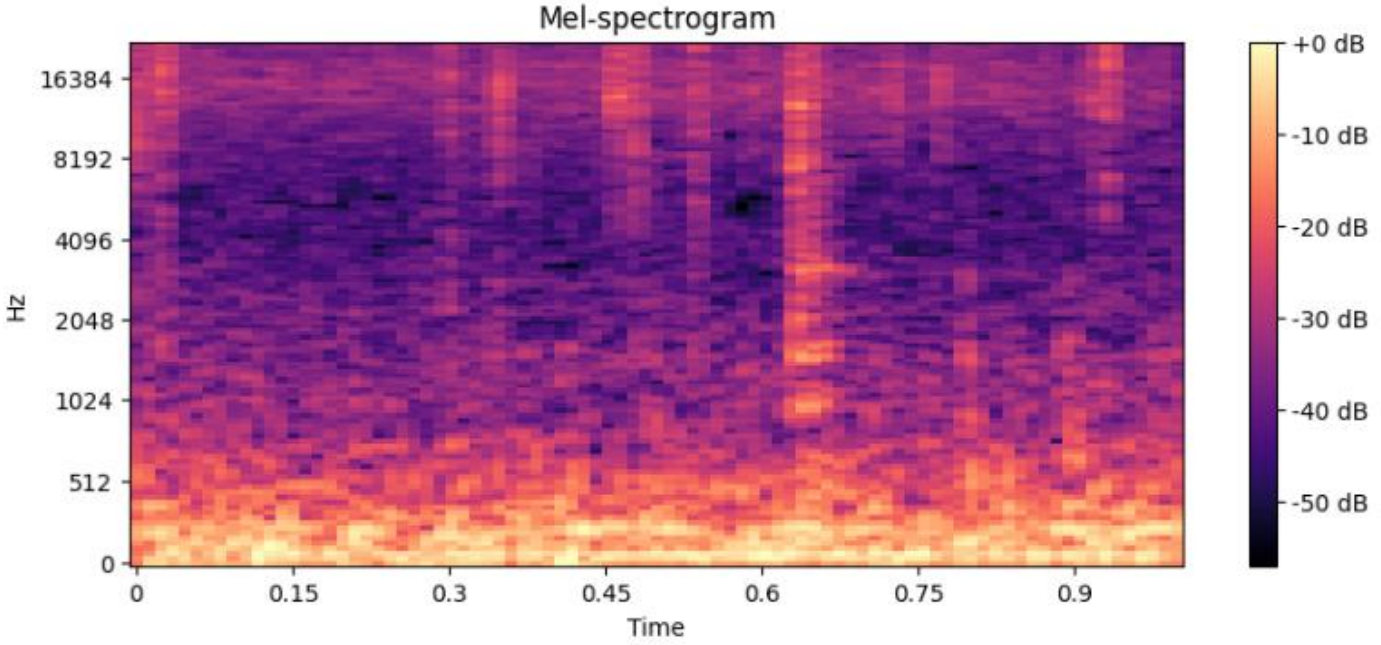



**(Grey Scale)Accuracy: 0.57**

## Deployment :


# Project-3

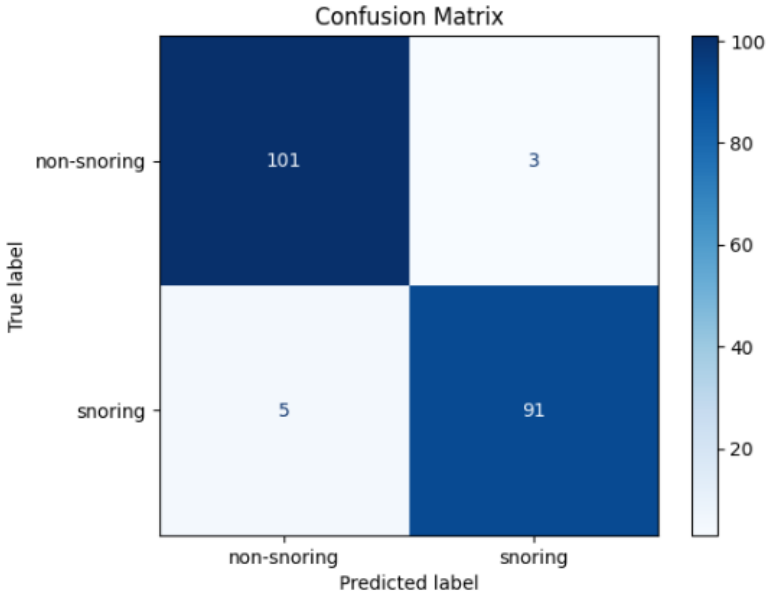


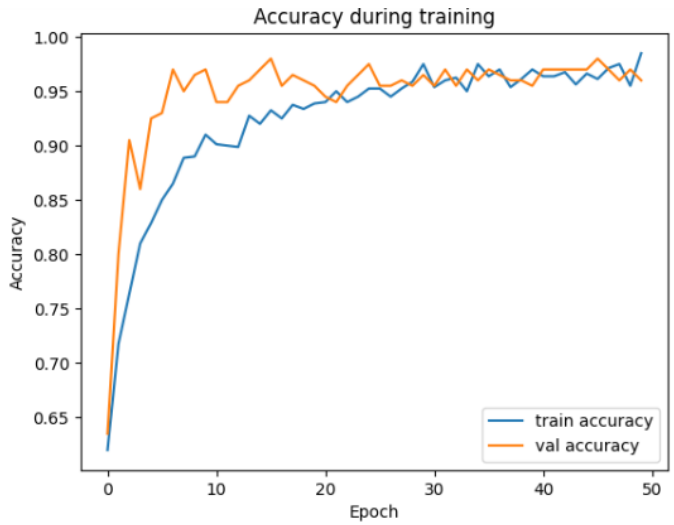

The top plot shows the raw audio signal's amplitude over time. The bottom plot shows the MFCCs, which are a time-frequency representation of the audio that is more perceptually relevant for human hearing and commonly used in audio analysis. Different patterns in the MFCC heatmap correspond to different sounds or phonetic elements present in the audio waveform.

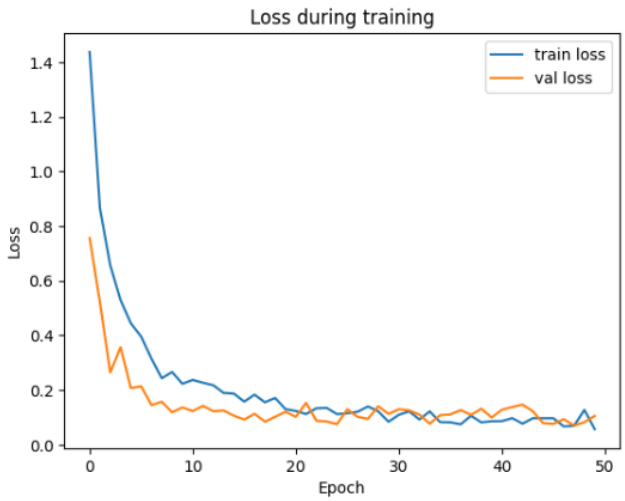
Classifiaction Report:

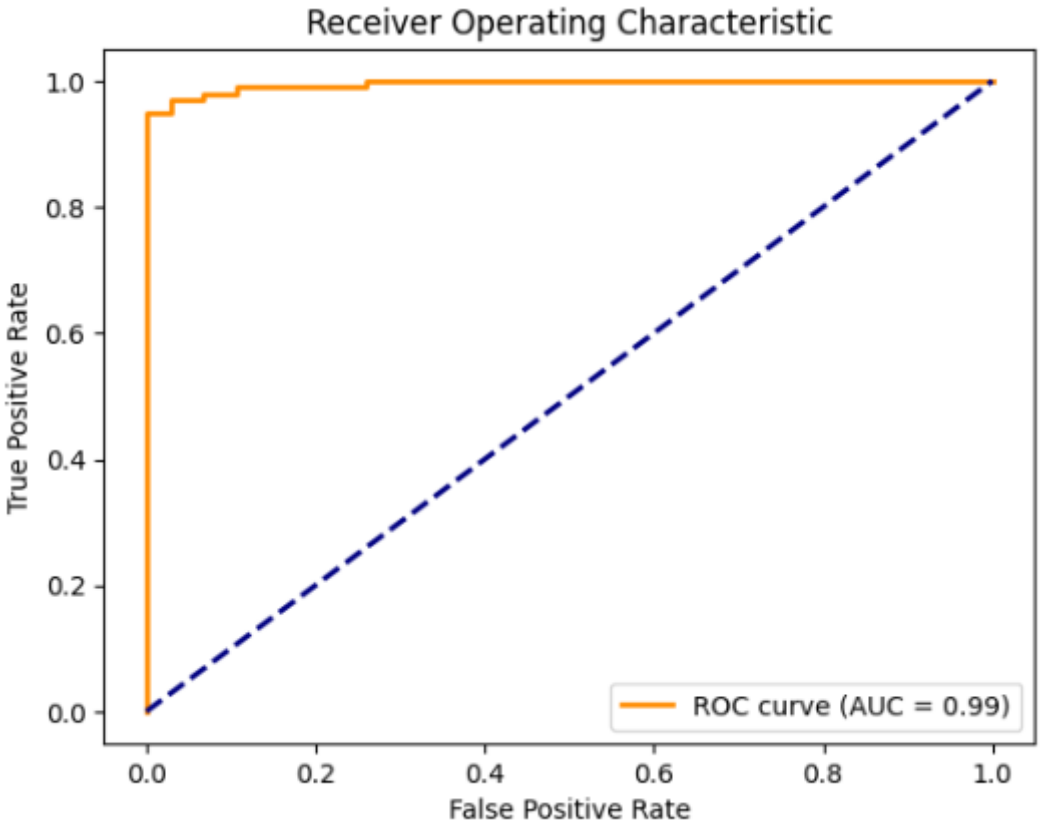
| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Non-snoring | 0.95 | 0.97 | 0.96 | 104 |
| Snoring | 0.97 | 0.95 | 0.96 | 96 |
| **Accuracy** | | | **0.96** | **200** |
| **Macro Avg** | 0.96 | 0.96 | 0.96 | 200 |
| **Weighted Avg** | 0.96 | 0.96 | 0.96 | 200 |

Confusion Matric:






The Receiver Operating Characteristic (ROC) curve illustrates the performance of the classification model, showing a high True Positive Rate against a low False Positive Rate. The curve is significantly above the diagonal baseline, indicating strong discriminative ability. The Area Under the Curve (AUC) is 0.99, which reflects excellent model performance, suggesting that the classifier can almost perfectly distinguish between the positive and negative classes.