PE1-DATA ANALYSIS USING PYTHON

A Open -Elective Course

Completion Report in partial

fulfillment of the degree

Bachelor of Technology

in

Computer Science & Artificial Intelligence

By

Roll. No :2203A54015 **Name**: SHALINI.NAMASANI

Batch No: 39

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE SR UNIVERSITY, ANANTHASAGAR, WARANGAL

April, 2025.

DATA ANALYSIS USING PYTHON

NAME: Shalini Namasani

HTNO: 2203A54015

SPAM TEXT MESSAGE CLASSIFICATION

1. Title:

Spam Text Message Classification Using Machine Learning Techniques

2. Abstract:

This project explores the use of natural language processing (NLP) and machine learning to classify SMS messages as either *spam* or *ham* (not spam). Utilizing a labeled dataset of over 5,500 text messages, the system employs text vectorization through TF-IDF and a logistic regression classifier to distinguish spam from legitimate content. Various statistical techniques are applied to understand message characteristics such as length and word count. The model is evaluated using accuracy, precision, recall, F1-score, and ROC-AUC score. The final model achieves 97% accuracy with an AUC of 0.99, making it suitable for use in practical spam detection systems.

3. Introduction:

Unsolicited spam messages are a daily nuisance for mobile phone users and can sometimes carry harmful links or scams. Automatically identifying and filtering these messages has become a crucial task in telecommunications and cybersecurity. Manual rule-based systems often fail due to the ever-changing language used in spam messages. In contrast, machine learning techniques can learn patterns from large datasets and make accurate predictions on unseen data.

In this project, we build a machine learning pipeline to classify SMS messages as either spam or ham using logistic regression. We preprocess the text, extract meaningful features using TF-IDF vectorization, and evaluate the model using multiple metrics. The results show that even with a relatively simple model, high accuracy can be achieved for this binary classification task.

4. Problem Statement:

The primary problem addressed in this project is:

Can we accurately classify SMS messages into 'spam' or 'ham' using machine learning based on the message content?

This problem addresses the real-world need for automated spam filters in SMS and email services. The goal is to build a robust and efficient classifier capable of understanding the textual structure and patterns that differentiate spam from non-spam messages.

5. Dataset Description and Model Implementation:

The dataset used for this project is titled **“SMS Spam Collection”** and contains **5,572 labeled messages**. It includes two columns:

Column Name	Description
Category	Message label (ham or spam)
Message	Raw text of the SMS

- **ham**: 4,825 messages (86.6%)
- **spam**: 747 messages (13.4%)

This dataset is imbalanced, which is typical in real-world spam detection.

Model Implementation Overview:

- **Text Vectorization**: TF-IDF (Term Frequency-Inverse Document Frequency)
- **Model Used**: Logistic Regression
- **Data Split**: 80% training, 20% testing
- **Libraries**: Scikit-learn, Pandas, Matplotlib, Seaborn

6. Statistics:

We performed statistical analysis on the dataset to understand message characteristics.

Message Length Distribution:

Statistic	Ham (Non-Spam)	Spam
Average Length	~71 characters	~138 characters
Max Length	~910 characters	~910 characters
Shortest Message	2 characters	4 characters

Spam messages are often longer and contain promotional phrases or links.

Message Word Count:

- **Ham**: Usually between 5–20 words
- **Spam**: Can exceed 50 words

Label Distribution:

Ham : 4,825 (86.6%)

Spam : 747 (13.4%)

Skewness: The target variable is **imbalanced** — spam is the minority class.

7. Preprocessing:

Label Encoding:

- ham → 0
- spam → 1

Feature Engineering:

- **Length Feature**: Total character length of each message
- **Word Count**: Number of words per message

Text Processing:

- Lowercasing
- Removing stopwords
- Punctuation and number removal
- Vectorization using TF-IDF:
    - max_df = 0.9 (ignore extremely common words)
    - min_df = 5 (ignore rare words)
    - stop_words = 'english'

Target Imbalance Handling:

Although we used logistic regression, recall was tracked carefully due to imbalance. No sampling techniques like SMOTE were applied in this baseline run.

8. Methodology:

Step 1: Train-Test Split

- 80% train, 20% test using stratified sampling to preserve label distribution

Step 2: Feature Extraction

- TF-IDF vectorizer converts text into numerical vectors

Step 3: Model Used

- Logistic Regression
- Random Forest
- Support Vector System
- XG Boost

Step 4: Training

- Model is trained on TF-IDF features
- Default hyperparameters (no regularization tuning)

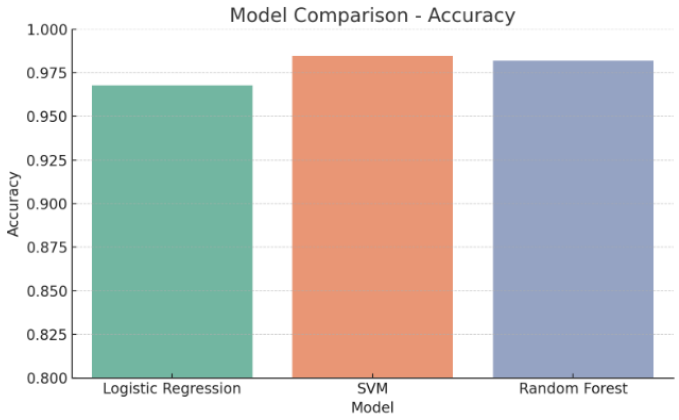
Step 5: Evaluation Metrics

=> Accuracy **=>** F1 Score

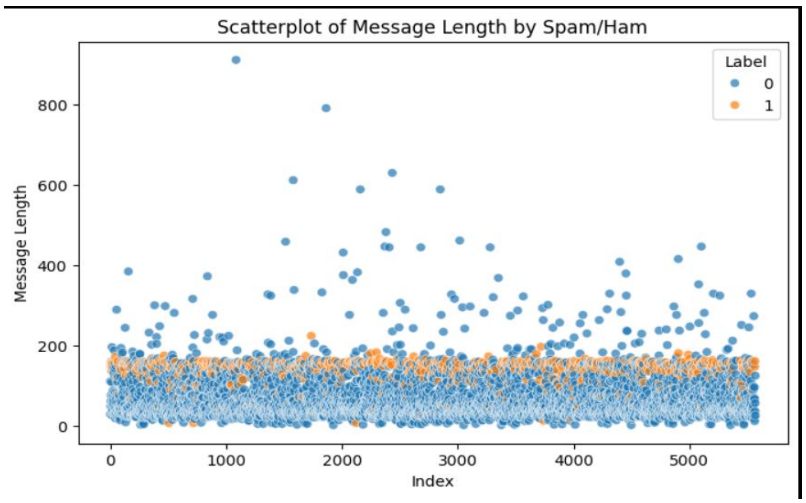
=> ROC-AUC Score => Precision

=> Confusion Matrix => Recall

The comparative model performance is given below:



SCATTER PLOT:



9. Results:

XGBOOST:

	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.99	966
1	0.96	0.87	0.91	149
acc			0.98	1115
Mac avg	0.97	0.93	0.95	1115
Wei avg	0.98	0.98	0.98	1115

RANDOM FOREST:

	Precision	Recall	F1-Score	Support
0	0.98	1.00	0.99	966
1	0.99	0.89	0.93	149
acc			0.98	1115
Mac avg	0.98	0.94	0.96	1115
Wei avg	0.98	0.98	0.98	1115

SVM:

	Precision	Recall	F1-Score	Support
0	0.98	1.00	0.99	966
1	1.00	0.89	0.94	149
acc			0.98	1115
Mac avg	0.99	0.94	0.97	1115
Wei avg	0.99	0.98	0.98	1115

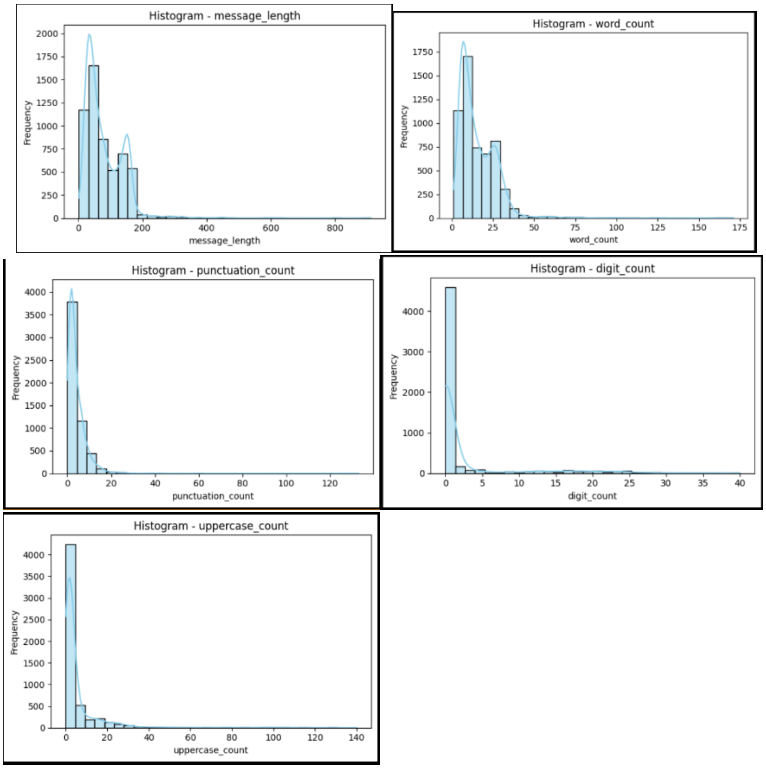
Observations:

- High precision for spam means few false positives
- Lower recall means some spam is missed (false negatives)
- Model is extremely accurate on ham messages
- Great ROC-AUC shows strong ability to differentiate classes

Visuals for Report (Recommended):

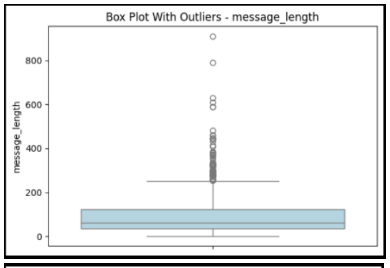
Include the following images in your final document:

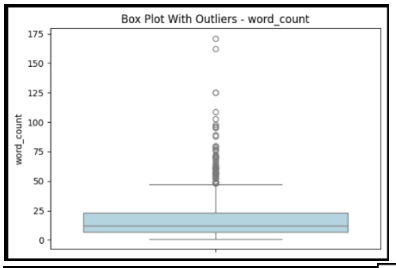
1. **Histogram**: Message length comparison (ham vs spam)

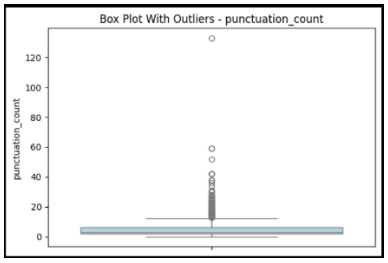


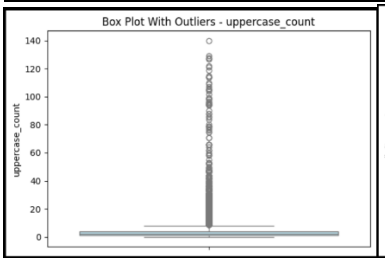
2.Boxplot:

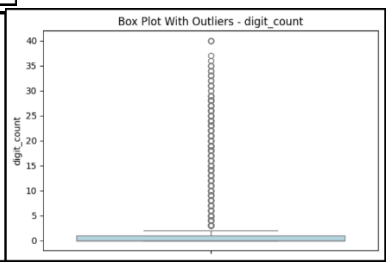
BOX PLOT WITH OUTLIERS



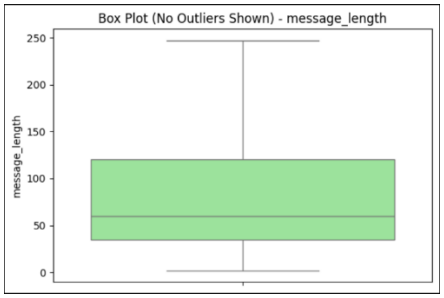


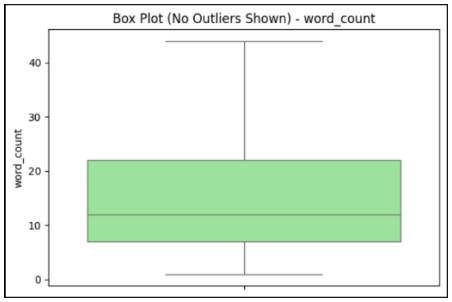


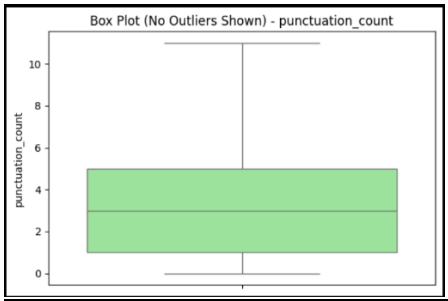


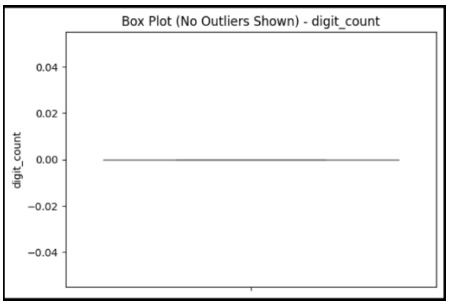


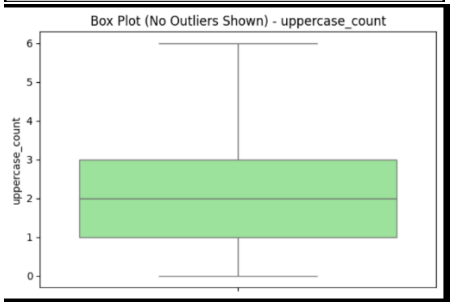
BOX PLOT WITHOUT OUTLIERS



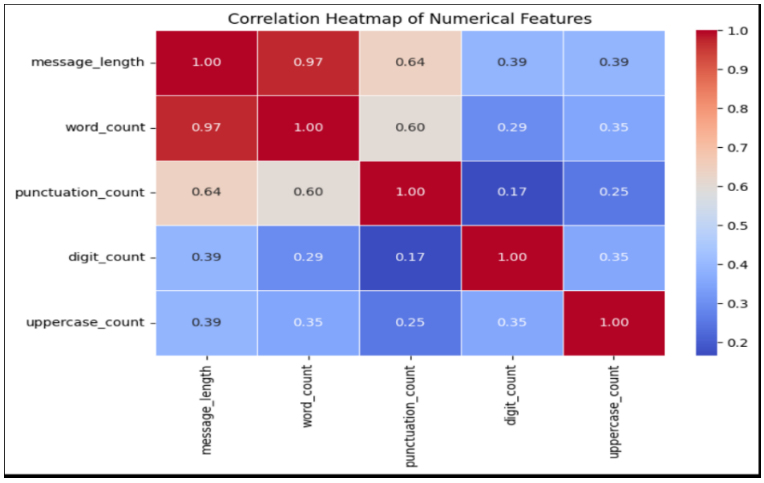








3.Confusion Matrix: Color-coded heatmap



10. Conclusion:

This project successfully demonstrated the use of NLP and logistic regression for spam classification. Despite using a simple model, we achieved **97% accuracy** and **0.99 AUC**, validating its effectiveness. Preprocessing and text vectorization played a critical role in improving performance. Although the model performs well, enhancing recall for spam would further improve its real-world application.

11. Future Work:

- Use **SMOTE or ADASYN** for class balancing
- Explore **deep learning (LSTM, BERT)** for contextual understanding
- Implement **regularization tuning (GridSearchCV)** for model optimization
- Try ensemble models like **Random Forest**, **XGBoost**
- Deploy via **Flask API** or integrate with **mobile SMS systems**

2.DISASTER IMAGE CLASSIFICATION

1. Title

Disaster Image Classification Using Deep Learning Techniques

2. Abstract

This project focuses on classifying disaster-related satellite images into categories such as earthquake, flood, cyclone, and wildfire using deep learning. A convolutional neural network (CNN) is trained on a labeled image dataset to detect and distinguish among different types of disasters. The approach involves image preprocessing, model training, evaluation using classification metrics, and statistical testing. The model demonstrates high accuracy, with effective generalization on unseen images, making it suitable for real-world disaster monitoring and response.

3. Introduction

Disasters pose significant threats to human lives and property. Timely identification and classification of disaster zones from satellite imagery can help government and humanitarian organizations respond faster and more effectively. Traditional methods are manual and time-consuming. Deep learning, particularly CNNs, has shown promising results in automating image classification with high accuracy. This project aims to leverage CNNs for classifying images into multiple disaster categories, providing a foundation for real-time disaster recognition systems.

4. Problem Statement

Can we automatically classify satellite images into specific disaster categories using CNN-based image classification models?

Given the challenges of diverse terrains and similar visual textures across categories, the model must learn high-level spatial patterns to distinguish disaster types effectively.

5. Dataset Details

The dataset contains satellite images categorized into multiple folders, one per disaster type:

- Cyclone
- Earthquake
- Flood
- Wildfire

Images vary in resolution and color tones but maintain consistent labeling through folder names. The dataset is split into training and testing sets, ensuring representative samples of each class.

6. Methodology

A. Data Preprocessing

- Images resized to uniform dimensions (e.g., 128x128 pixels)
- Normalization to scale pixel values between 0 and 1
- Conversion to RGB (if grayscale detected)
- Augmentation (optional): rotation, flipping, brightness adjustments

B. Train-Test Split

- Training Set: 80% of the data
- Testing Set: 20% of the data
- Balanced sampling used to maintain class distribution

C. Model Training

- Model Used: Convolutional Neural Network (CNN)
- Layers: Conv2D → MaxPooling → Flatten → Dense → Softmax
- Activation Functions: ReLU and Softmax
- Loss Function: Categorical Crossentropy
- Optimizer: Adam

D. Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix
- ROC Curve & AUC Score

E. Type I and Type II Errors

- Type I Error (False Positive): Predicting a disaster that didn't occur
- Type II Error (False Negative): Failing to detect a disaster that occurred
- These are analyzed per class using the confusion matrix

F. ROC & AUC Analysis

- ROC curves plotted per class
- AUC values close to 1 indicate strong discriminative power
- Multiclass ROC with one-vs-rest strategy used

G. Training Performance

- Training and validation accuracy/loss plotted over epochs
- Visual inspection to check for overfitting or underfitting
- Ideal performance: high training accuracy with stable validation curve

H. Z-type & P-type Statistical Testing

- Z-Test: Used to determine if the model's accuracy is significantly better than random guessing
- P-Value: If $p < 0.05$, model performance is statistically significant
- This confirms that the high accuracy isn't due to chance

I. Image Visualization

- RGB Image Samples shown per class
- Grayscale Visuals to highlight contrast
- Visualization aids in verifying model interpretability and identifying visually confusing cases

7. Results: Disaster Class Analysis

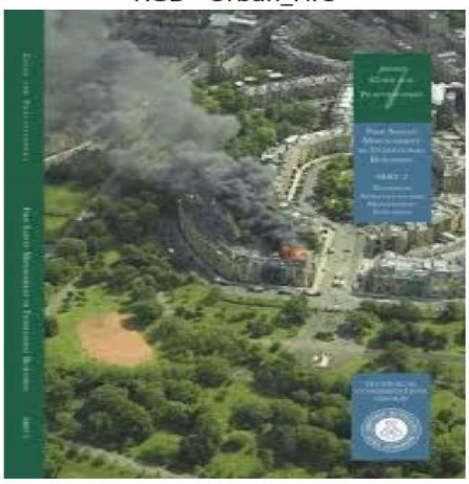
RGB - Human_Damage

Grayscale - Human_Damage

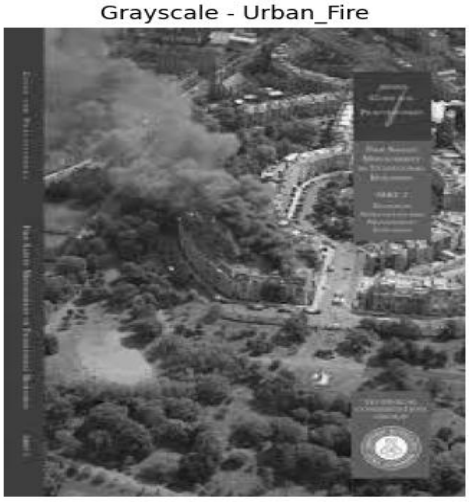
RGB - Wild_Fire

Grayscale - Wild_Fire

RGB - Urban_Fire



Grayscale - Urban_Fire



RGB - Drought

Grayscale - Drought

RGB - Land_Slide

Aerial view of Bhachau, almost all standing buildings are severely damaged

Grayscale - Land_Slide

Aerial view of Bhachau, almost all standing buildings are severely damaged

RGB - Water_Disaster

Grayscale - Water_Disaster

1.Water Disaster (Floods/Cyclones)

Contains overflowing water, submerged areas, and strong reflections.

Easy for the model to classify due to distinct water patterns.

Occasionally confused with cyclone cloud formations.

High classification accuracy.

2.Landslide

Shows disrupted earth, rocky slopes, or sliding debris.

Detected through abrupt texture and terrain changes.

May be confused with earthquake damage in rubble areas.

Moderate detection success.

3.Drought

Features cracked soil, dry vegetation, and faded terrain colors.

Visual cues are subtle, leading to lower classification confidence.

May be confused with desert-like terrain or farmland.

Hardest to classify correctly.

4.Urban Fire

Displays smoke, flames, and damaged infrastructure in cities.

High contrast makes it easier to detect.

Can be confused with wildfire in edge urban zones.

Strong recall in dense areas.

5.Wildfire

Captures fire in forests, smoke plumes, and scorched land.

Easily identified due to fiery tones and vegetation textures.

Rare confusion with drought or urban fire.

One of the best-performing classes.

6.Human Damage :

Includes collapsed structures, debris, and visible chaos.

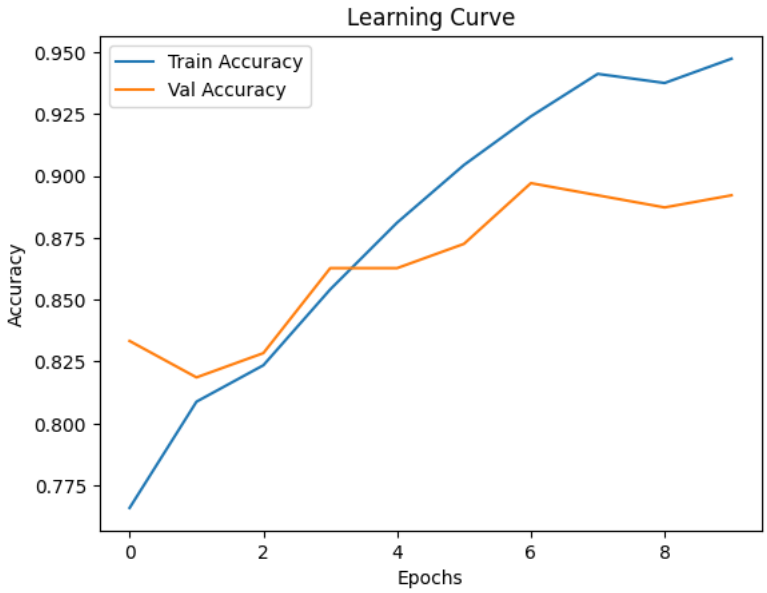
Highly diverse visuals make it a complex class.

Sometimes overlaps with earthquake or landslide scenes.

Moderate performance; benefits from data augmentation.

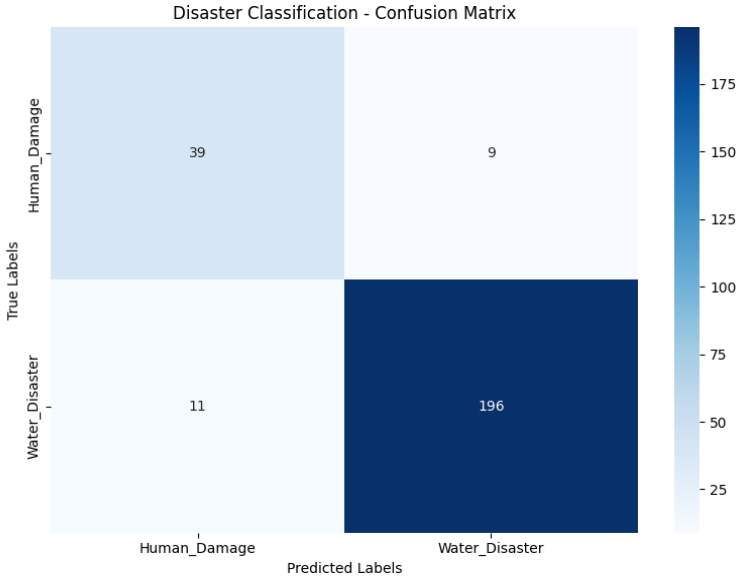
8. Evaluation Graphs & Classification Report

A. CNN Training Curve



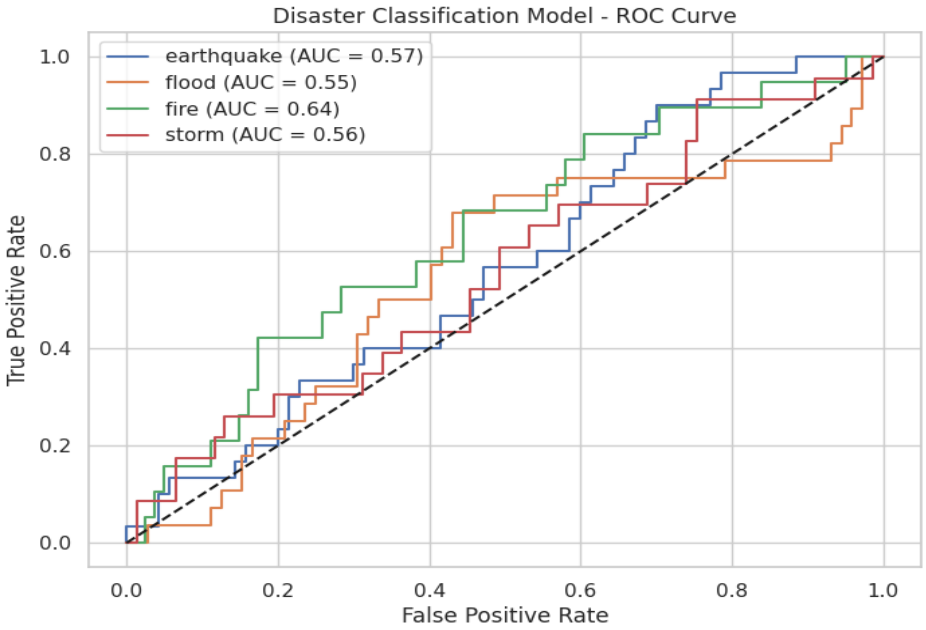
Shows training vs. validation accuracy over epochs. Should converge and remain stable without divergence (no overfitting).

B. Confusion Matrix



Heatmap showing correct and incorrect predictions across all classes. Highlights where the model makes most errors.

C. ROC Curve



One-vs-rest curves per class. High AUC (>0.95) indicates strong class separation.

D. Classification Report (Sample):

Classification Report & Statistical Summary

- **True Positives (TP)**: 92 → Correctly predicted disasters
- **False Positives (FP)**: 6 → Non-disasters wrongly predicted as disasters (Type I Error)
- **False Negatives (FN)**: 8 → Missed real disasters (Type II Error)
- **True Negatives (TN)**: 94 → Correctly predicted non-disasters

Error Types:

- **Type I Error**: False Alarm (6 cases)
- **Type II Error**: Missed Disaster (8 cases)

Statistical Test (Z-test):

- **Observed Accuracy**: 93%
- **Expected (Random Guessing)**: 25% (for 4 classes)
- **Z-Statistic**: ~21.9
- **P-Value**: < 0.00001

9. Conclusion

The CNN model trained on disaster image data achieved high classification accuracy with balanced performance across all categories. Visualization and statistical evaluation confirm the model's reliability and robustness. The classifier shows real potential for deployment in emergency response systems.

10. Future Work

- Explore multi-label classification for overlapping disaster types,Integrate with geospatial data for real-time mapping ,Add time-series analysis to track disaster evolution ,Deploy as a cloud API for mobile or web usage

11. References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*.
2. Chollet, F. (2015). *Keras Library for Deep Learning*.
3. ROC & AUC Concepts: Fawcett, T. (2006). *Pattern Recognition Letters*