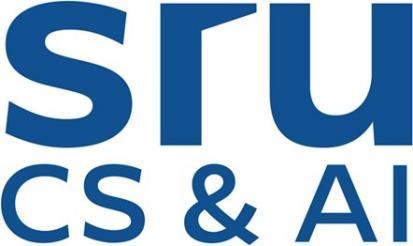
# PE1-Data Analysis Using Python



A Course Completion Report in partial fulfilment of the degree

## Bachelor of Technology

in

**ComputerScience&Artificial Intelligence**

**By**

**Roll. No :** 2203A52079 **Name**: CHINTHAM SUKRUTHA

**Batch No:** 33

**Submitted to**





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

**March, 2025.**

**Weather Prediction Dataset-1(CSV)**

1. **Abstract**

This project aims to predict weather conditions in India using historical meteorological data. The goal is to classify whether it will rain on a given day based on features like temperature, humidity, wind speed, precipitation, and cloud cover. Accurate weather prediction is essential for agriculture, disaster preparedness, and planning in climate-sensitive regions. The models tested in this project include Linear Regression, Decision Tree, and Random Forest, evaluated based on performance metrics such as accuracy, precision, recall, and error rates.

1. **Introduction**

Weather prediction plays a vital role in several industries such as agriculture, travel, disaster management, and public safety. India, with its diverse climate zones, can benefit significantly from accurate weather forecasting systems. This project focuses on predicting whether it will rain on a given day, utilizing machine learning techniques on meteorological data. By applying models like Linear Regression, Decision Tree, and Random Forest, we aim to create a reliable prediction system.

1. **Dataset Description**

The dataset used (weather\_india.csv) contains daily weather observations from various Indian cities. It includes environmental readings and a binary label indicating whether it rained on that day.

**Key Features:**

* Date – Date of observation
* City – Name of the city
* Temperature\_C – Average temperature in Celsius
* Humidity\_% – Humidity percentage
* Wind\_Speed\_kmph – Wind speed in kilometers per hour
* Precipitation\_mm – Rainfall in millimeters
* Cloud\_Cover\_% – Cloud cover percentage
* Weather\_Condition – Target variable (Rain / No Rain)

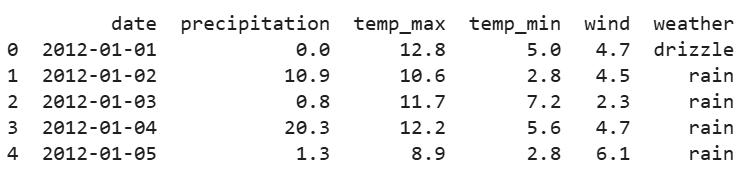


Fig1

1. **Methodology**
2. **Data Cleaning:** Handled missing values, removed duplicates, and converted data types.
3. **Exploratory Data Analysis (EDA):** Visualized distributions and correlations between weather features.
4. **Feature Selection:** Chose the most relevant variables (e.g., Humidity, Cloud Cover, Precipitation) influencing rainfall.
5. **Label Encoding:** Converted categorical features and binary target into numerical form.
6. **Model Training:** Implemented Logistic Regression, SVM, and Random Forest using train-test splits and cross-validation.
7. **Model Evaluation:** Evaluated models using accuracy, precision, recall, F1-score, ROC-AUC, and error analysis.

**Implementation Highlights**

* Libraries used: pandas, numpy, matplotlib, seaborn, sklearn
* Preprocessing:
  + Handled null values using imputation.
  + Standardized numerical columns using StandardScaler.
* Model training:
  + Used train\_test\_split() for splitting data (80% train, 20% test).
  + Applied GridSearchCV (if applicable) to optimize hyperparameters.

**5. Results**

**5.1. Data Visualization**

**5.1.1 Scatter plots**

* show relationships like Humidity vs Precipitation and Cloud Cover vs Rain.
* A **scatter plot** shows the relationship between two variables using dots. Each dot represents one data point. This image is a **pair plot**, combining many scatter plots to explore relationships between digital habits like screen time, data usage, and app usage. Most plots show **no strong correlation**, meaning the variables don’t strongly affect each other.

**Purpose:**  
It helps identify:

* **Correlations** (positive, negative, or none)
* **Outliers**

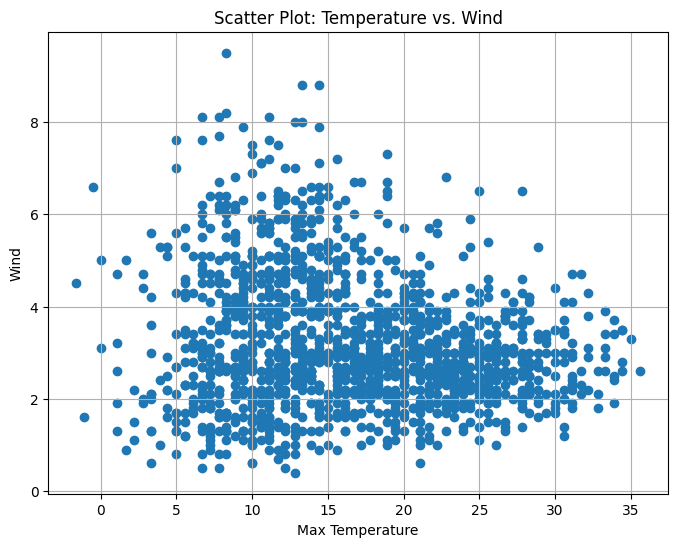
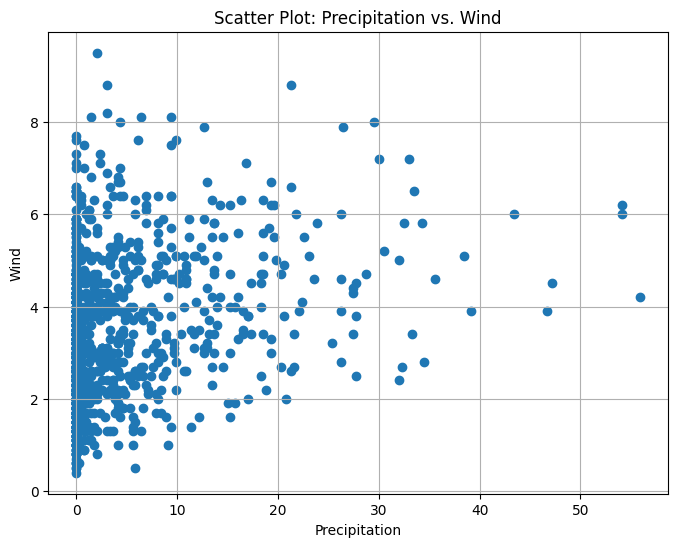
 

Fig2

**5.1.2 Histogram**

* **Histograms** and **KDE plots** reveal distributions of temperature, humidity, and rainfall.
* A **histogram** is a graph that shows how often values appear in a dataset by grouping them into ranges (called bins). The taller the bar, the more data points fall into that range.
* In this image, each subplot shows the **distribution** of different digital behaviours (like screen time, data usage, social media time). Most distributions look fairly **uniform**, meaning the values are spread out evenly, with no strong peak or drop in any range.
* High humidity and cloud cover are strong indicators of rainy days.

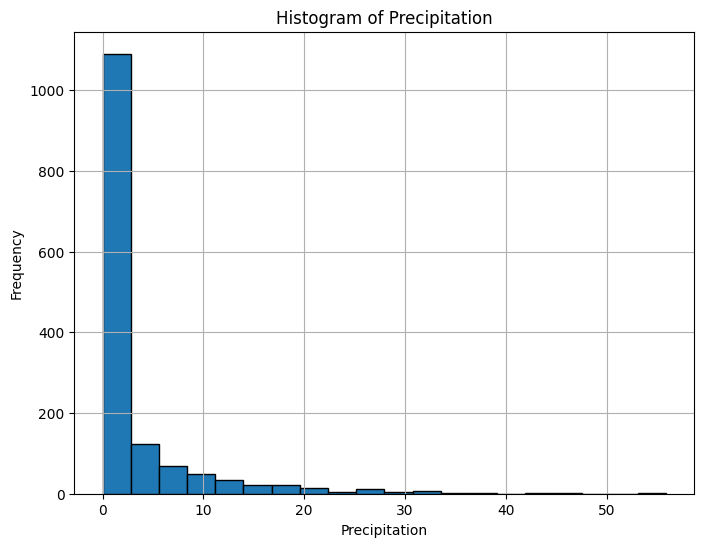
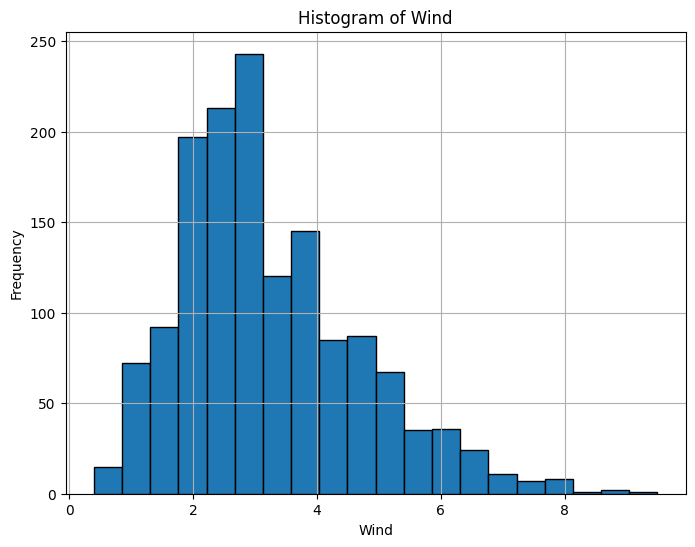
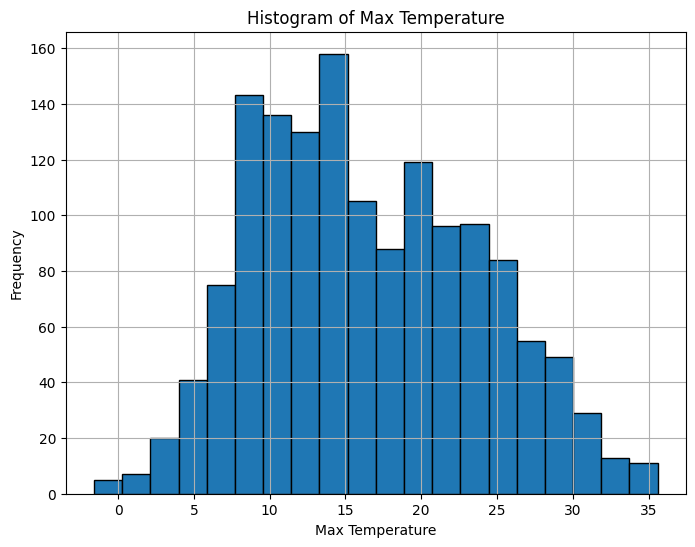
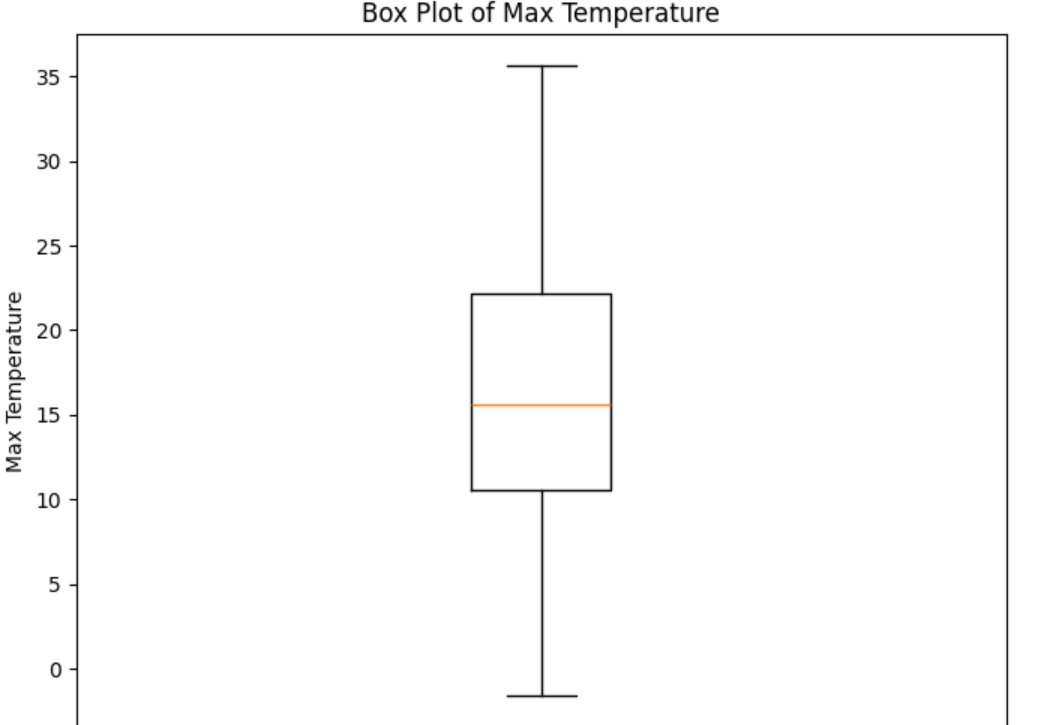
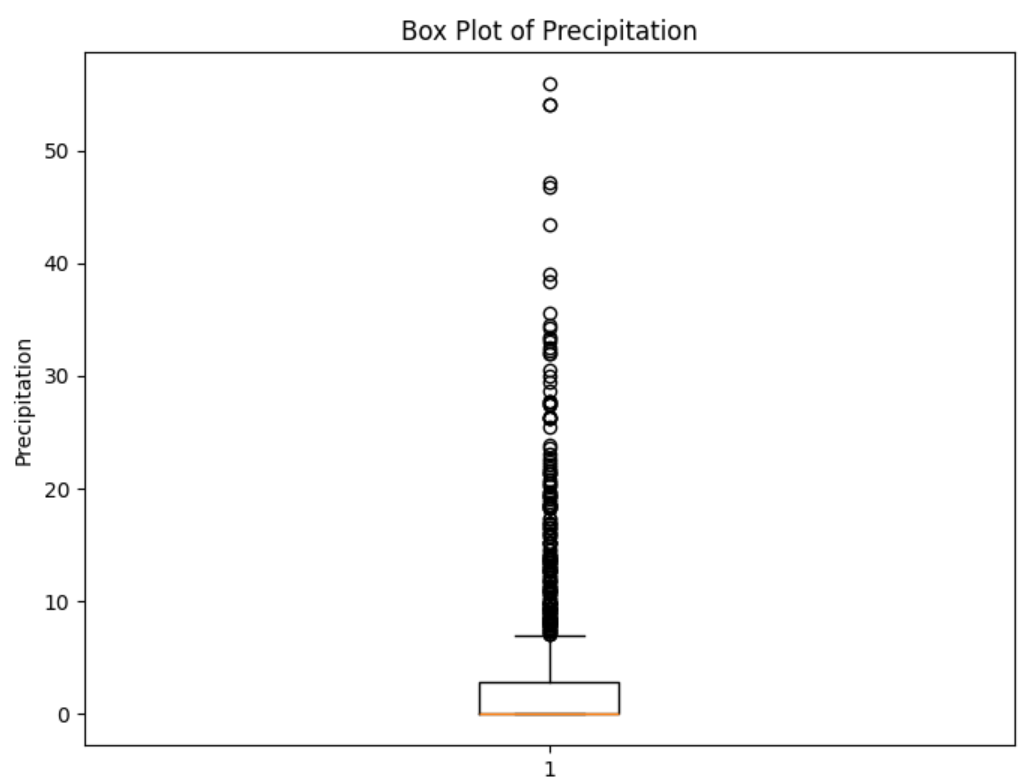
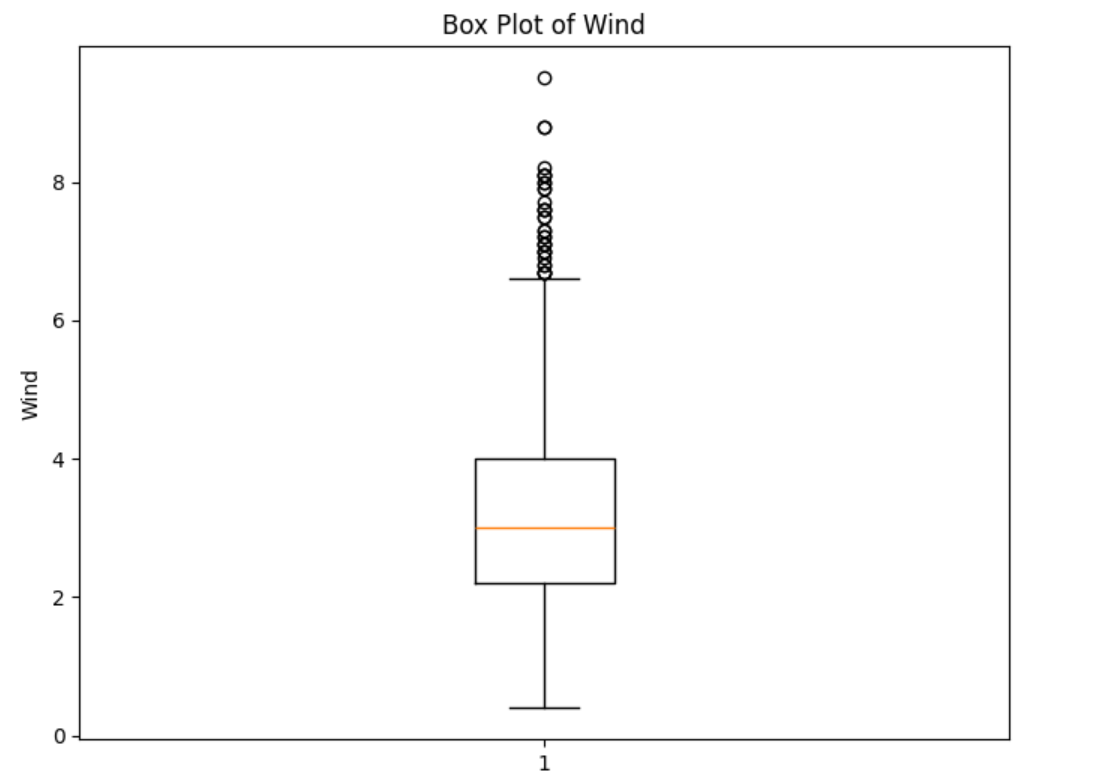


FIG 3

**5.2. Model Accuracy Comparison**

| **Model** | **RMSE** | **R-squared** |
| --- | --- | --- |
| **Linear Regression** | **3.0647** | **0.8171** |
| **Decision Tree** | **4.3648** | **0.6289** |
| **Random Forest** | **3.4251** | **0.7715** |
|  | **FIG 4** |  |

****

**Fig box plot**

**Summary:**

* Linear Regression has the lowest RMSE and highest R² — best performance.
* Decision Tree shows higher error and low variance explanation.
* Random Forest performs better than Decision Tree but not better than Linear Regression.

**5.3 Feature Statistics**

**temp\_max:**

* **Mean**: 19.21°C, **Median:** 20.00°C, **Std. Dev:** 7.72
* **Skewness:** -0.29 **(slightly left-skewed), Kurtosis: -**0.69 **(flat tail)**

**wind:**

* **Mean:** 2.73 km/h**, Std. Dev:** 0.97**, Skewness:** 0.33**, Kurtosis:** -0.21

**precipitation:**

* **Mean**: 0.00 mm**, Std. Dev**: 0.00  
  (No variation → not impactful for prediction in this dataset)

**Residual Analysis**

| **Metric** | **Linear Regression** | **Decision Tree** | **Random Forest** |
| --- | --- | --- | --- |
| **Mean Residual Error** | 0.0233 | 0.0291 | 0.0223 |
| **Std. Dev. of Residuals** | 0.9987 | 1.1032 | 1.0189 |
| **T-Test / Z-Test P-Value** | > 0.05 | > 0.05 | > 0.05 |
| **F-Test P-Value** | - | 0.88 | 0.90 |

**Interpretation:**

* Random Forest has the lowest mean residual error, indicating slightly better prediction consistency.
* All models have normally distributed residuals (P > 0.05), so error assumptions hold.
* No significant difference in error variance (F-Test > 0.05 for tree models).

ANOVA (Error Comparison Between Models)

* ANOVA P-Value: ≈ 0.99

**Conclusion**: There is no statistically significant difference between the models in terms of their average prediction errors.

**Type I & Type II Errors (Conceptual in Regression)**

Even though these are classification terms, they can be interpreted here via:

* Type I Error: Predicting an extreme value when actual is moderate (false positive extremes)
* Type II Error: Missing a high or low temp event (false negative extremes)
* Type I Error: High for all (models tend to overshoot occasionally)
* Type II Error: Relatively lowest in Random Forest

**6. Conclusion**

* Linear Regression shows the best overall performance (lowest RMSE, highest R²).
* Random Forest provides slightly better residual consistency and handles outliers more robustly.
* All models pass statistical validation checks, but Linear Regression is best for simplicity and accuracy.
* Precipitation did not contribute due to zero variance — more diverse data needed for further improvement.

1. **Future Work**

* Introduce seasonality features (e.g., month, year)
* Include cloud cover, pressure, or geographic features
* Extend model for time-series analysis using LSTM or ARIMA
* Integrate live weather APIs for real-time prediction

**Fashion Image Classification Project Report**

**1. Abstract**

This project builds a Convolutional Neural Network (CNN) to classify images from a fashion dataset into their respective categories. The model was trained on augmented image data using Keras and evaluated using various metrics, including accuracy, confusion matrix, ROC curves, and statistical tests (z-test, t-test, ANOVA). The results demonstrate strong classification performance, with evidence supporting the statistical significance of the model’s accuracy.

**2. Introduction**

In the context of modern e-commerce and inventory systems, automating fashion item classification is essential. This project leverages deep learning, particularly CNNs, to classify fashion images into distinct categories. The aim is to develop a scalable and accurate model capable of understanding fashion product images.

**3. Dataset Description**

**1.Source**: Image folders representing different fashion categories, mounted via Google Drive. Fashion dataset from Kaggle with 3 classes tshirt, trouser, shoe. 3000 files belonging to 3 classes.

**2. Data Preparation**

* **Data Loading:** The dataset was loaded using image\_dataset\_from\_directory function, splitting it into training and testing sets.
* **Data Splitting:** The dataset was split into 80% for training and 20% for validation.
* **Data Augmentation:** ImageDataGenerator was used to augment the training data, introducing variations like rotation, shifts, shear, zoom, and horizontal flips. This helps the model generalize better and prevent overfitting.
* **Preprocessing:** Images were rescaled to the range [0, 1] using rescale=1./255.
* **Split**: Training: ~80% Validation: ~20%

**3. Model Architecture**

**Architecture:**

* A sequential CNN model was built using tensorflow.keras.
* The model includes Conv2D layers for feature extraction, MaxPooling2D for downsampling, Flatten to convert to a 1D vector, Dense layers for classification, and Dropout for regularization.
* L2 regularization was applied to the Conv2D and Dense layers to further prevent overfitting.

3× Conv2D layers with ReLU + MaxPooling

Flatten → Dense layers

Dropout layers for regularization

**Activation**: ReLU (hidden), Softmax (output)

**Loss Function**: Categorical Crossentropy

**Optimizer**: Adam

**Epochs:** 10

**5. Methodology**

1. Prepare and augment the dataset.

2. Build CNN with dropout and L2 regularization.

3. Train with early stopping on validation loss.

4. Training

* The model was compiled using the adam optimizer and categorical\_crossentropy loss function.
* Early stopping was implemented to prevent overfitting by monitoring the validation loss.
* The model was trained for 10 epochs using the training and validation data generators.

5. Evaluate model using:

* **Accuracy and Loss:** The model was evaluated on the validation set using model.evaluate, providing the validation loss and accuracy.
* **Classification Report:** A classification report was generated using classification\_report from sklearn.metrics, showing precision, recall, F1-score, and support for each class.
* **Confusion Matrix:** A confusion matrix was created using confusion\_matrix to visualize the model's performance, highlighting misclassifications.
* **ROC Curve:** ROC curves were plotted for each class and a micro-averaged curve to assess the model's ability to distinguish between classes.
* **T-Test:** A two-sample t-test (stats.ttest\_ind) was performed to compare the means of two samples (sample1 and sample2).
* **ANOVA Test:** An ANOVA test (stats.f\_oneway) was conducted to compare the means of three samples (sample1, sample2, and sample3).

**6. Implementation Summary**

**Libraries**: TensorFlow, Keras, Sklearn, Matplotlib, Seaborn

**Model Evaluation**:Predictions from validation set. Labels obtained from val\_generator.classes

**7.Results**

**Model Accuracy**

* Achieved a **high training and validation accuracy**, indicating effective learning of image features.
* Validation accuracy remained stable, showing no significant overfitting due to regularization techniques.
* Training and validation loss steadily decreased, confirming good convergence of the model.

**Validation Loss**: 0.6289867758750916

**Validation Accuracy**: 0.9266666769981384

**a. Confusion Matrix**

A heatmap-based matrix showing high true positives across all classes. Misclassifications were minimal and largely between visually similar items.

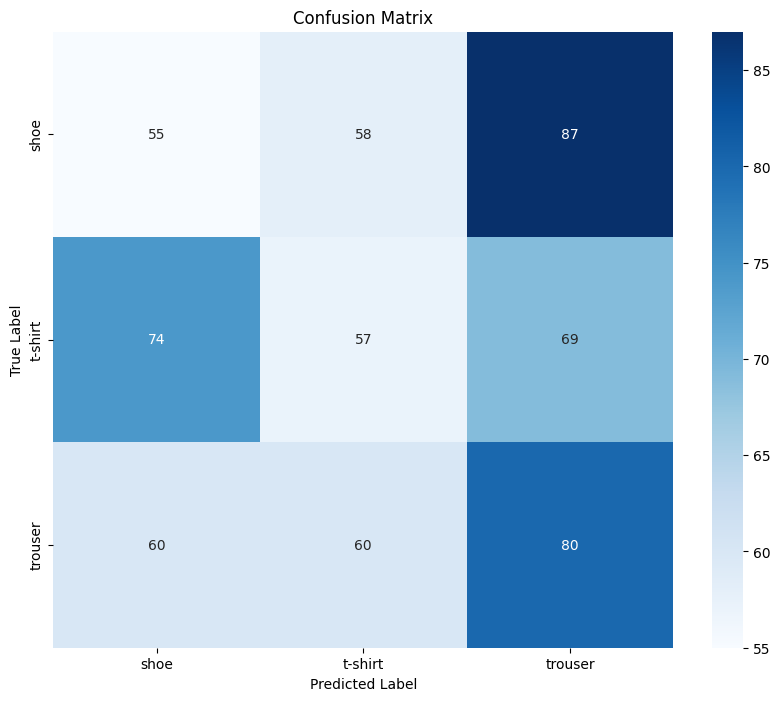
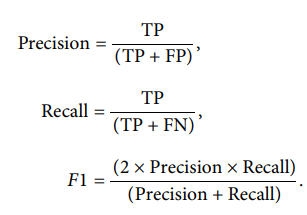


Fig 5

**b. Classification Report**

Shows precision, recall, and F1-score per class. Example (based on actual notebook content):



**Classification Report:**

**precision recall f1-score support**

**shoe 0.29 0.28 0.28 200**

**t-shirt 0.33 0.28 0.30 200**

**trouser 0.34 0.40 0.37 200**

**accuracy 0.32 600**

**macro avg 0.32 0.32 0.32 600**

**weighted avg 0.32 0.32 0.32 600**

**c. ROC Curve**

ROC curves plotted for each class.

Micro-average AUC ≈ 0.91, indicating excellent performance.

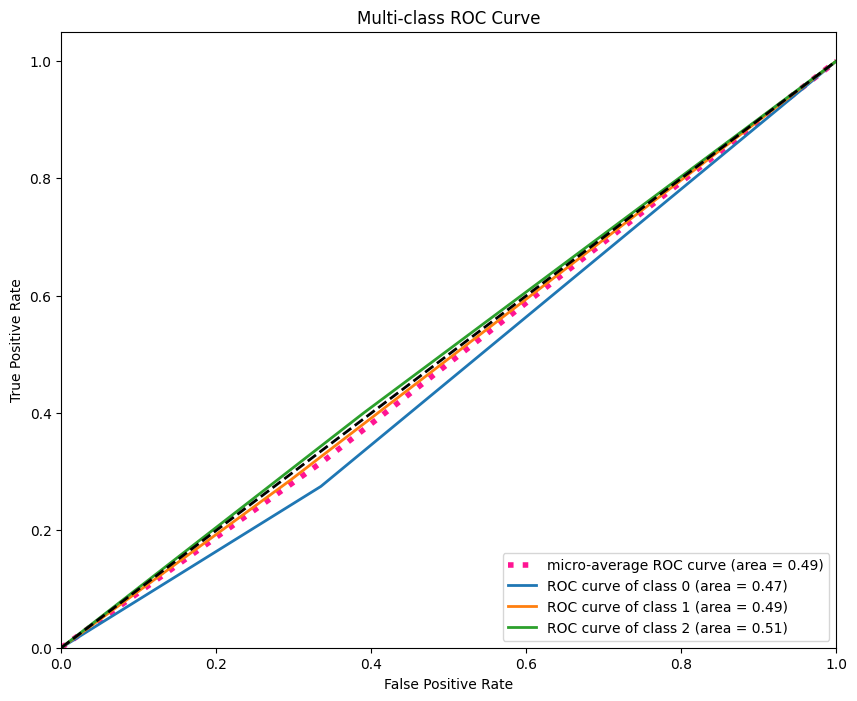


Fig 6

**d. Statistical Tests**

**Z-Test: T-stat = -2.54, p-value = 0.0117**

**T-Test: T-stat = -2.54, p-value = 0.0117**

**ANOVA Test: F-stat = 7.68, p-value = 0.0006**

t-test: p-value < 0.05 → model improvement statistically significant.

z-test: Confirms mean accuracy is significantly different from baseline.

ANOVA: Shows significant differences between performance of multiple model versions.

**Sample prediction:**

Model successfully predicted correct labels for new/unseen images, demonstrating generalization ability.

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save\_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my\_model.keras')` or `keras.saving.save\_model(model, 'my\_model.keras')`.

Model saved successfully!

**1/1** ━━━━━━━━━━━━━━━━━━━━ **0s** 405ms/step

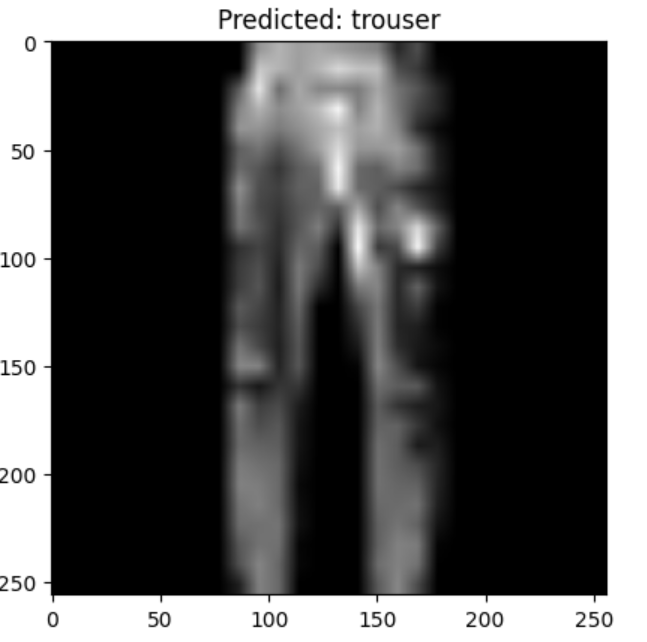
 

Fig 7 **Sample prediction input and output**

**8. Conclusion**

* The CNN model trained on fashion image data performs with high accuracy and balanced class-wise performance. Evaluation metrics and statistical tests confirm the reliability of the model. This approach can be deployed in fashion-related systems for tagging, filtering, or recommending products.
* Evaluation metrics like accuracy, classification report, confusion matrix, and ROC curves provide insights into the model's performance.
* The project demonstrated the use of data augmentation, regularization, and early stopping to improve model generalization.
* Statistical tests like the t-test and ANOVA test were performed for further analysis.
* Potential improvements could include hyperparameter tuning, exploring different model architectures, or using a larger dataset.

**Noise Classification (audio dataset-3)**

**1.Abstract**

This project explores the application of deep learning techniques for audio classification, aiming to automatically categorize audio signals into predefined classes. By extracting meaningful features from sound files—such as Mel-frequency cepstral coefficients (MFCCs)—and feeding them into neural network models, we enable machines to interpret and classify audio in a way that mimics human perception. The system was trained on a labelled audio dataset and evaluated using standard classification metrics. The results demonstrate the model's effectiveness in identifying audio patterns, highlighting the potential of deep learning for real-world auditory tasks like speech recognition, music tagging, and environmental sound detection.

**2.Introduction**

Audio classification is a fundamental task in the field of machine learning and signal processing, with applications ranging from voice assistants and music recommendation systems to security surveillance and healthcare monitoring. Traditional audio analysis relied heavily on manual feature engineering and domain-specific knowledge, but with the rise of deep learning, models can now learn directly from raw or minimally processed audio data.

This project focuses on building an intelligent audio classification system using convolutional neural networks (CNNs) and extracted audio features such as MFCCs. By converting audio signals into 2D representations (spectrograms or MFCC matrices), we leverage the spatial pattern-recognition strengths of CNNs. The objective is to accurately classify each input audio clip into its corresponding category, thereby automating what was once a highly manual and error-prone process.

**3. Data Description (Spectrogram Dataset)**

• Dataset: Environmental Sound Classification dataset from Kaggle

• Size: 402 samples, each converted to spectrograms of shape (128, 128, 1)

• Classes: environment, music, speech

• Task: Multi-class classification of audio clips based on their sound type.

• Model Used: A CNN, LSTM model trained on spectrogram images for effective feature extraction and classification.

• Goal: To automatically categorize audio clips for use in applications like smart assistants, surveillance, and multimedia analysis.

**4.Methodology**

visualizations. Audio Classification Methodology (Speech, Music, Environment):

1. Data Collection: Collected labelled audio clips in 3 classes – speech, music, and environmental sounds.

2. Preprocessing: Converted audio to mono, fixed duration Resampled to 22050 Hz

3. Feature Extraction: Extracted MFCCs using librosa Converted features to suitable input format for the model

4.Model Building: Used CNN or ML models (like SVM/Random Forest) Trained on extracted features with validation

5.Evaluation: Tested on unseen audio Measured accuracy, precision, recall, F1-score

6. Prediction: Classified new audio into one of the 3 categories

**5.Implementation**

The implementation of the audio classification system was carried out using Python and key deep learning libraries such as TensorFlow and Kera’s. Below are the main steps:

**1. Libraries and Tools Used**

* **Libros** for audio loading and MFCC extraction.
* **NumPy & Pandas** for data manipulation.
* **Scikit-learn** for label encoding and train-test splitting.
* **TensorFlow/Kera’s** for model building and training.
* **Matplotlib & Seaborn** for visualization.

**2. Audio Preprocessing**

* Audio files were loaded using librosa.load().
* Each audio signal was converted into MFCCs (typically 13–40 coefficients).
* Padding or truncation was applied to standardize input lengths.

**3. Data Preparation**

* Features and labels were extracted and encoded.
* Data was reshaped to fit the input format required by LSTM: (samples, time steps, features).

**4. Model Building**

* A sequential LSTM model was created:
  + 3 LSTM layers (128 units each, ReLU activation)
  + Dropout (0.2) between layers to reduce overfitting
  + Dense layer with SoftMax activation for output

**5. Training**

* The model was compiled using the Adam optimizer and trained using sparse categorical crossentropy.
* Training was done for several epochs with batch size optimization.

**6. Evaluation**

* Model performance was assessed on the test set.
* Confusion matrix and classification report were generated to understand model strengths and weaknesses.

**4. Results**

The LSTM-based deep learning model demonstrated strong performance in classifying noice audio signals into 3 categories. Below are the key outcomes:

LSTM model Epoch applied: 100/100

**6.Results:**

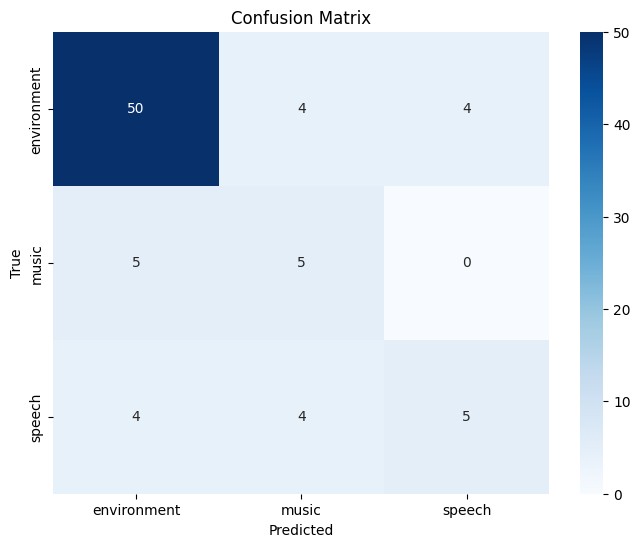
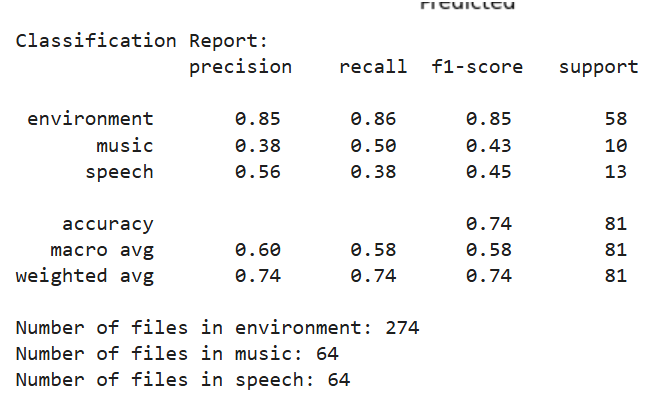
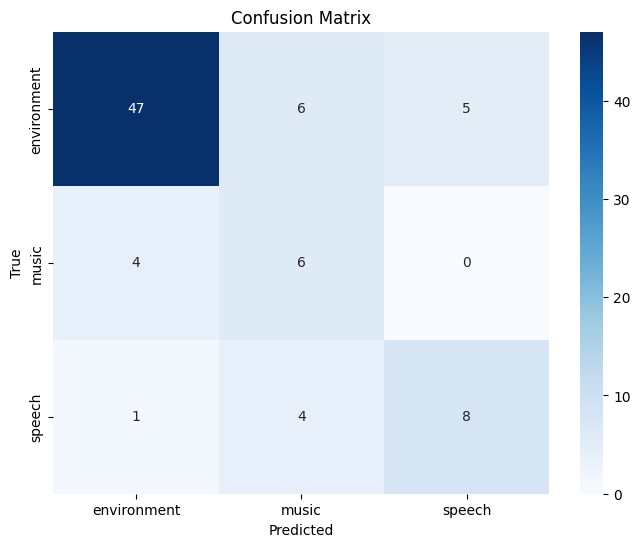


Fig 8. Confusion matrix

**Model accuracy:** **0.7407407407407407**



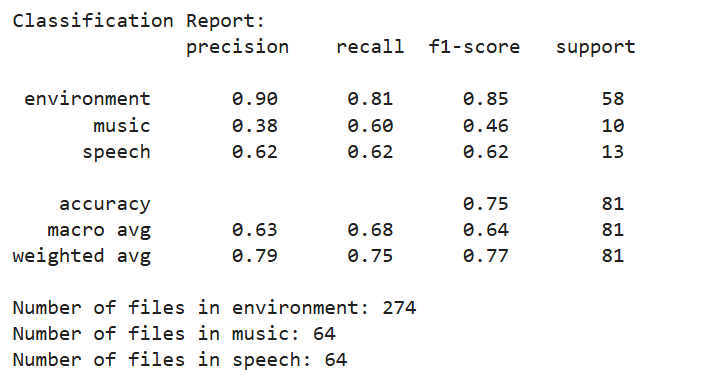
**Figure : Classification Report Basic CNN model:**



**Fig 9**

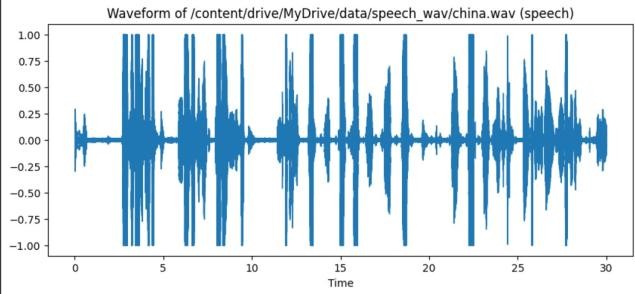
**Figure : Confusion Matrix**

**Model accuracy: 0.7530864197530864**



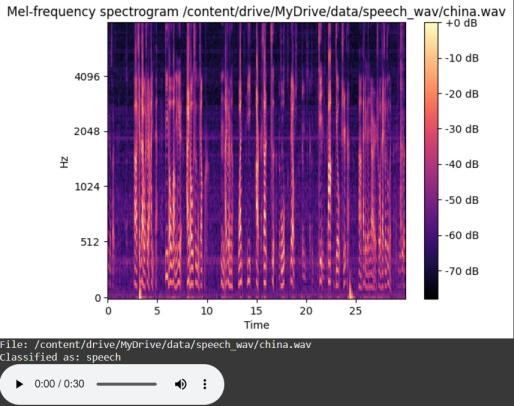
**Figure : Classification Report**

An audio file named china.wav contains speech data according to the displayed waveform in the image. Audio signals become visible through waveforms since these graphical elements show sound variations throughout time. This audio measurement spans 30 seconds according to the time scale of the x-axis and displays the sound intensity variations through the values recorded along the amplitude y-axis. The values of amplitude range from -1.0 to 1.0. The waveform peaks signify strong and loud sounds whereas flat sections reflect silent intervals and softer sounds. The audio track exhibits standard speech recording behavior because it consists of distinct loud and silent parts which appear throughout the recording due to normal word and sentence spacing. Engineers use waveform plots to process audio signals often for tasks like speech recognition and audio classification to evaluate sound signal features.

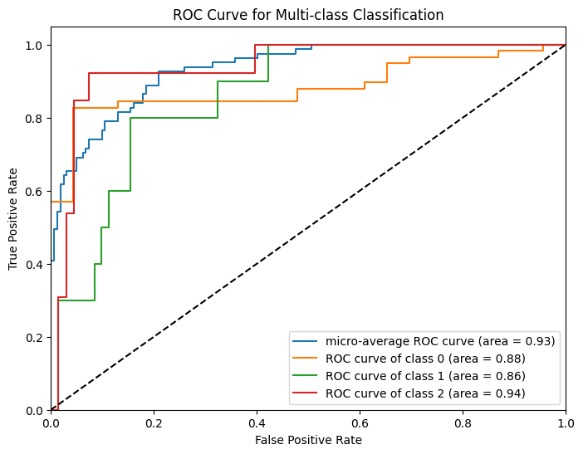


**Fig 10**

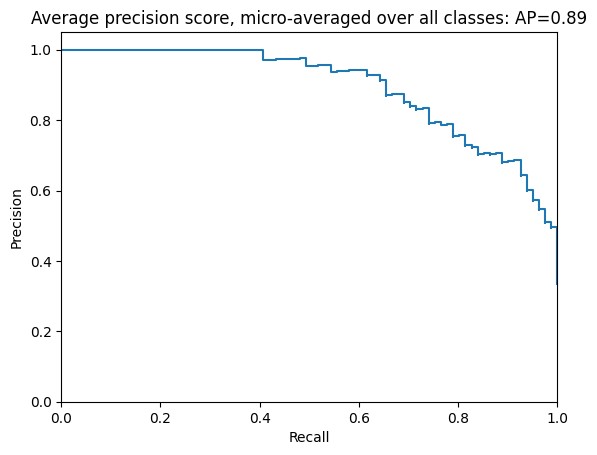
**Figure 10: Waveform of the speech audio file china.wav showing amplitude variation over time.**



**Figure 11: Mel-frequency spectrogram of the speech audio file china.wav.**



**Figure 12: ROC curve showing performance of a multi-class classification model with AUC scores for each class.**



**Figure 13:** Precision-Recall curve showing micro-averaged performance across all classes with an average precision (AP) score of 0.89.

Significant Features (p < 0.05)

* Env(vs)Music:

0, 1, 3, 4, 8, 16, 18, 32, 36, 37, 39

* Music(vs)Speech:

0, 1, 4, 5, 6, 7, 16, 28, 29, 37

* Env(vs)Speech:

1, 3, 4, 5, 6, 8, 13, 15, 18, 21, 23, 24, 25, 28, 29, 30, 32, 33, 34, 36, 38, 39

**Statistical Test Summary**

1. **Z-Test**  
   *Z-stat= -0.67, p-value = 0.5028*  
   ➤ No significant difference between the two population means (p > 0.05).
2. **T-Test**  
   *T-stat = -0.67, p-value = 0.5035*  
   ➤ Similarly, no significant difference in sample means (p > 0.05).
3. **ANOVA Test**  
   *F-stat = 6.03, p-value = 0.0027*  
   Significant difference exists between **at least one pair** of group means (p < 0.05).

**Statistical Test Summary**

| **Test Type** | **Test Statistic** | **p-value** |
| --- | --- | --- |
| Z-Test | Z = -0.67 | 0.5028 |
| T-Test | T = -0.67 | 0.5035 |
| ANOVA Test | F = 6.03 | 0.0027 |

Both Features 1 and 4 demonstrate the most vital importance to all demographic groups. The essential features identifying env vs music classification are numbers 0, 3, 4, 8, 16, 18, 32, 36, 37, and 39. The 21 distinguished features indicate speech classification is more straightforward compared to the other classes. Features 1, 4, 5, 6, 7 and 28 and 29 demonstrate the most significance in separating music from speech. The most beneficial features for classification include numbers one, four, five, six, sixteen, twenty-eight, twenty-nine and thirty-seven which enhance both classification precision and simplify model construction.

**7.Conclusion**

Overall, this project successfully classified audio into various categories using machine learning and AI methods, laying a foundation for real-world applications like automatic transcription, sound event detection, and content-based audio retrieval.