Date	2 oct 2025
Team ID	LTVIP2025TMIDS67798
Project Name	RAINFALL PREDICTION USING MACHINE LEARNING
Maximum Marks	6 Marks

Data Preprocessing Template for Rainfall Prediction

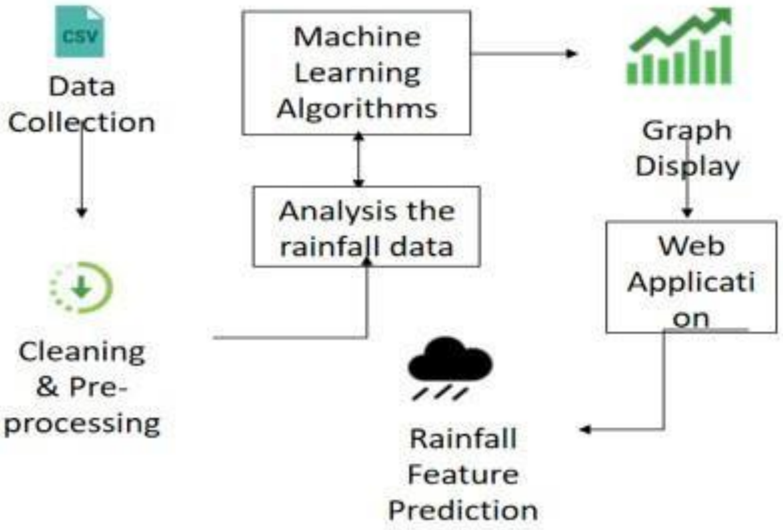
CSV
Data Collection
Machine Learning Algorithms
Graph Display
Analysis the rainfall data
Web Applicati on
Cleaning & Pre-processing
Rainfall Feature Prediction

Fig.1. Process flow chart

Data Collection

- **Source**: Gather historical weather data, including temperature, humidity, wind speed, and precipitation levels.
- **Format**: Ensure data is in CSV or Excel format for easy manipulation.

Data Cleaning

- **Handle Missing Values**:
    - Impute missing numerical values using the mean or median.
    - For categorical data, impute using the mode or employ predictive imputation techniques.
- **Remove Duplicates**: Identify and eliminate duplicate entries to maintain data integrity.
- **Outlier Detection**: Use statistical methods (e.g., Z-scores) to identify and handle outliers.

Feature Engineering

- **Datetime Features**: Extract features like day of the week, month, and season from datetime columns.
- **Lag Features**: Create lag variables to capture temporal dependencies in rainfall data.
- **Rolling Statistics**: Compute rolling averages and standard deviations to smooth out short-term fluctuations.

Data Transformation

- **Scaling**: Apply Min-Max scaling or Standardization to numerical features to ensure uniformity.
- **Encoding**: Convert categorical variables into numerical formats using techniques like One-Hot Encoding or Label Encoding.
- **Normalization**: Normalize data to bring all features into a similar range, improving model performance.

Data Splitting

- **Train-Test Split**: Divide the dataset into training and testing sets, typically using an 8020 or 70-30 split.
- **Cross-Validation**: Implement K-fold cross-validation to assess model performance and reduce overfitting.

Data Integration

- **Combine Datasets**: If using multiple data sources (e.g., satellite data), merge them based on common keys like date and location.
- **Data Alignment**: Ensure all datasets are aligned temporally and spatially before merging.

Suggested Directory Structure

```
RAINFALL-PREDICTION-/

├── data/

|   ├── raw/            # Original datasets

|   ├── processed/        # Cleaned and transformed data

|   └── external/         # External data sources (e.g., satellite)

├── notebooks/            # Jupyter notebooks for analysis and modeling

├── scripts/            # Python scripts for data preprocessing

|   ├── data_cleaning.py
|   ├── feature_engineering.py
|   └── data_transformation.py
├── models/              # Trained models and model evaluation scripts
├── requirements.txt       # Python dependencies
└── README.md             # Project documentation
```

Tools & Libraries

- **Python Libraries**:
  - pandas for data manipulation
  - numpy for numerical operations
  - matplotlib and seaborn for data visualization
  - scikit-learn for machine learning algorithms and preprocessing utilities

    xgboost or lightgbm for gradient boosting models **Data Sources**:
  - **Satellite Data**: Utilize APIs or datasets like CMORPH or IMERG for satellitebased rainfall estimates.
  - **Weather Stations**: Incorporate data from local meteorological stations for groundtruth validation.

Additional Tips

- **Documentation**: Maintain clear documentation for each preprocessing step to ensure reproducibility.
- **Version Control**: Use Git to track changes in data processing scripts and model versions.
- **Model Evaluation**: Regularly evaluate model performance using metrics like RMSE, MAE, and R^2.