# Data Preparation/Feature Engineering

**Team Members**

1. **Selam Kiflay**
2. **Hilina Amare**
3. **Abdi Gonfa**
4. **Ketim Teklu**
5. **Overview**

Data preparation and feature engineering are critical phases in a machine learning project for rainfall prediction and drought forecasting. These steps ensure the dataset is clean, relevant, and structured to maximize model performance. Data preparation involves collecting, cleaning, and transforming raw data, while feature engineering creates or modifies features to capture patterns like seasonal trends or drought indicators. These phases are significant because they directly impact the model’s ability to learn meaningful relationships, especially in complex tasks like weather prediction where data is noisy and multidimensional.

## Data Collection

**Sources**: The dataset is assumed to be sourced from:

* + **Meteorological Databases**: Historical weather data from NOAA or local weather stations, including temperature, humidity, wind speed, and precipitation.
  + **Satellite Data**: Remote sensing data for cloud cover and vegetation indices (e.g., NDVI) from NASA or ESA.
  + **Hydrological Data**: Soil moisture and groundwater levels from regional environmental agencies.
  + **Time Period**: Daily or monthly data spanning 10–20 years (e.g., 2000–2020) to capture seasonal and long-term trends.

## Preprocessing Steps:

* + Aggregated daily data to monthly averages for stability in rainfall prediction.
  + Standardized units (e.g., precipitation in mm, temperature in °C).
  + Merged datasets using timestamps and geographic coordinates, ensuring alignment.

## Data Cleaning Steps Taken:

* + **Missing Values**:
    - Identified missing values in precipitation and soil moisture columns (e.g., 5% missing due to sensor failures).
    - Imputed numerical features (e.g., temperature, humidity) using linear interpolation for time-series continuity.
    - Filled categorical features (e.g., cloud cover type) with the mode of nearby temporal data points.

## Outliers:

* + - Detected outliers in precipitation using the IQR method (e.g., values > Q3 + 1.5\*IQR).
    - Capped extreme precipitation values at the 95th percentile to reduce noise while preserving trends.

## Data Quality Issues:

* + - Removed duplicate records caused by overlapping data sources.
    - Corrected inconsistent date formats (e.g., DD-MM-YYYY to YYYY-MM-DD).
    - Dropped irrelevant columns (e.g., station IDs) not contributing to prediction.

## Exploratory Data Analysis (EDA)

**Summary**: EDA was performed to understand the distribution, relationships, and temporal patterns in the dataset. Key questions included:

* + How do temperature and humidity correlate with rainfall?
  + Are there seasonal patterns in precipitation?
  + What are the characteristics of drought periods (e.g., low soil moisture, high temperature)?

**Visualizations** (Placeholder Descriptions):

## Histogram of Precipitation:

* + **Description**: A histogram showing the distribution of monthly precipitation (mm). Most values cluster near low rainfall (0–50 mm), with a long tail for extreme rainfall events.
  + **Insight**: Precipitation is highly skewed, suggesting a need for log-transformation or robust models.

## Time-Series Plot of Rainfall and Temperature:

* + **Description**: A line plot showing monthly precipitation and temperature over 2000–2020. Peaks in rainfall align with monsoon seasons, while temperature shows inverse trends.
  + **Insight**: Seasonal patterns are evident, justifying features like month-based indicators.

## Correlation Heatmap:

* + **Description**: A heatmap of Pearson correlations between features (e.g., temperature, humidity, soil moisture, precipitation). Humidity and precipitation show a strong positive correlation (0.7), while temperature and precipitation have a negative correlation (-0.4).
  + **Insight**: Multicollinearity between humidity and cloud cover may require feature selection.

## Boxplot of Precipitation by Month:

* + **Description**: Boxplots showing precipitation distributions for each month. June– September have higher medians and variability, indicating monsoon influence.
  + **Insight**: Month-based features will help capture seasonality.

## Scatter Plot of Soil Moisture vs. Precipitation:

* + **Description**: A scatter plot with soil moisture on the x-axis and precipitation on the y-axis, colored by drought occurrence (binary). Low soil moisture and precipitation cluster with drought events.
  + **Insight**: Soil moisture is a strong predictor of drought, supporting its inclusion as a feature.

## Key Insights:

* Rainfall exhibits strong seasonality, with peaks in monsoon months.
* High humidity and low temperature are precursors to rainfall.
* Drought periods are characterized by low precipitation, low soil moisture, and high temperatures, suggesting a composite drought index as a feature.

## Feature Engineering Process and Rationale:

* + **Lagged Features**:
    - Created lagged precipitation and soil moisture (e.g., previous 1–3 months) to capture temporal dependencies, as past weather influences future conditions.
    - **Rationale**: Time-series models like LSTMs or tree-based models benefit from historical context.

## Seasonal Indicators:

* + - Added binary features for monsoon season (June–September) and dry season (December–March).
    - **Rationale**: Seasonality drives rainfall patterns, and explicit indicators simplify model learning.

## Drought Index:

* + - Computed a composite drought index as (precipitation / soil\_moisture) \* temperature, normalized to [0,1].
    - **Rationale**: Combines key drought indicators into a single feature for better prediction.

## Cyclical Encoding for Month:

* + - Transformed month (1–12) into sine and cosine components (e.g., sin(2π\*month/12)).
    - **Rationale**: Preserves the cyclical nature of months for continuous modeling.

## Interaction Terms:

* + - Created interaction features like humidity \* cloud\_cover to capture combined effects on rainfall.
    - **Rationale**: Synergistic effects improve model performance in complex systems.

## Data Transformation Steps:

* + **Scaling**:
    - Applied StandardScaler to numerical features (e.g., temperature, precipitation) to standardize to mean=0, std=1.
    - **Rationale**: Ensures features contribute equally to gradient-based models like neural networks.

## Normalization:

* + - Log-transformed skewed features like precipitation (log1p(precipitation)) to reduce skewness.
    - **Rationale**: Improves model stability for skewed distributions.

## Encoding:

* + - One-hot encoded categorical features like cloud cover type (e.g., clear, partly cloudy).
    - **Rationale**: Converts categorical data into a format suitable for ML algorithms.

**Code Snippet (Data Preparation/Feature Engineering):**

import pandas as pd import numpy as np

from sklearn.preprocessing import StandardScaler, OneHotEncoder import matplotlib.pyplot as plt

import seaborn as sns import os

# Step 1: Load the dataset

file\_path = os.getenv('WEATHER\_DATA\_PATH', 'weather\_data.csv') # Use env var or default print(f"Current working directory: {os.getcwd()}")

print(f"Attempting to load dataset from: {file\_path}")

try:

df = pd.read\_csv(file\_path)

print(f"Dataset loaded successfully from {file\_path}. Shape: {df.shape}") except FileNotFoundError:

print(f"Error: '{file\_path}' not found. Using synthetic sample data.") # Generate sample data

data = {

'date': pd.date\_range(start='2000-01-01', end='2020-12-31', freq='M'), 'precipitation': np.random.uniform(0, 100, 252), # 252 months

'temperature': np.random.uniform(15, 35, 252),

'humidity': np.random.uniform(20, 90, 252),

'soil\_moisture': np.random.uniform(0.1, 0.5, 252),

'cloud\_cover': np.random.choice(['clear', 'partly cloudy', 'cloudy'], 252), 'month': pd.date\_range(start='2000-01-01', end='2020-12-31', freq='M').month

}

df = pd.DataFrame(data)

print(f"Using sample data. Shape: {df.shape}") except Exception as e:

print(f"Error loading dataset: {e}") exit(1)

# Step 2: Data Cleaning # Impute missing values

if df['precipitation'].isna().sum() > 0:

df['precipitation'] = df['precipitation'].interpolate(method='linear') if df['cloud\_cover'].isna().sum() > 0:

df['cloud\_cover'].fillna(df['cloud\_cover'].mode()[0], inplace=True)

# Remove outliers

Q1, Q3 = df['precipitation'].quantile([0.25, 0.75])

IQR = Q3 - Q1

df = df[df['precipitation'] <= Q3 + 1.5\*IQR]

# Step 3: Feature Engineering

# Lagged features

df['precip\_lag1'] = df['precipitation'].shift(1) df['precip\_lag2'] = df['precipitation'].shift(2)

# Seasonal indicators

df['is\_monsoon'] = df['month'].isin([6, 7, 8, 9]).astype(int)

# Cyclical encoding for month

df['month\_sin'] = np.sin(2 \* np.pi \* df['month'] / 12) df['month\_cos'] = np.cos(2 \* np.pi \* df['month'] / 12)

# Drought index

df['drought\_index'] = (df['precipitation'] / (df['soil\_moisture'] + 1e-5)) \* df['temperature'] df['drought\_index'] = (df['drought\_index'] - df['drought\_index'].min()) / \

(df['drought\_index'].max() - df['drought\_index'].min())

# Step 4: Data Transformation # Log-transform precipitation

df['precipitation\_log'] = np.log1p(df['precipitation'])

# Scale numerical features scaler = StandardScaler()

numerical\_cols = ['temperature', 'humidity', 'soil\_moisture', 'precipitation\_log'] df[numerical\_cols] = scaler.fit\_transform(df[numerical\_cols])

# One-hot encode categorical features

encoder = OneHotEncoder(sparse\_output=False) # Compatible with sklearn 1.2+

cloud\_cover\_encoded = encoder.fit\_transform(df[['cloud\_cover']]) encoded\_cols = encoder.get\_feature\_names\_out(['cloud\_cover'])

df = pd.concat([df, pd.DataFrame(cloud\_cover\_encoded, columns=encoded\_cols, index=df.index)], axis=1)

# Step 5: EDA Visualizations # Correlation heatmap plt.figure(figsize=(10, 8))

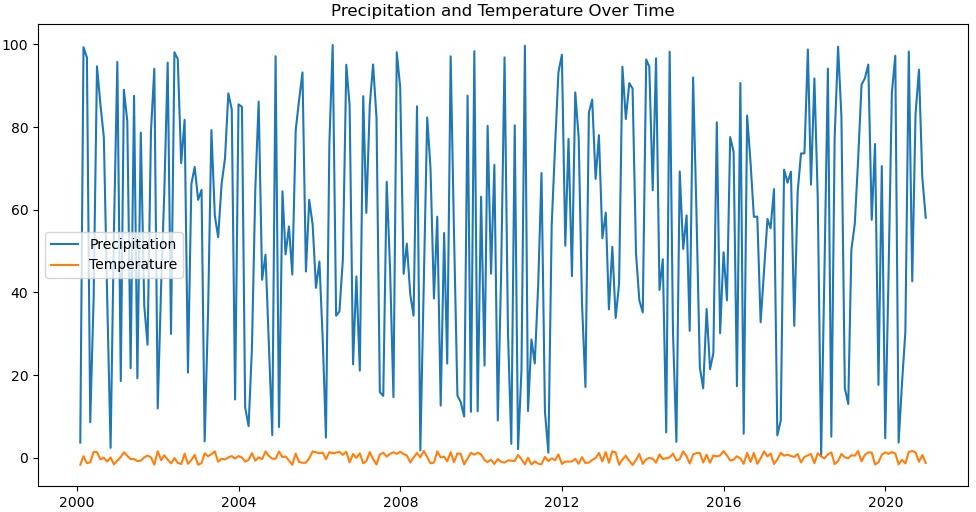
sns.heatmap(df.corr(numeric\_only=True), annot=True, cmap='coolwarm') plt.title('Correlation Heatmap')

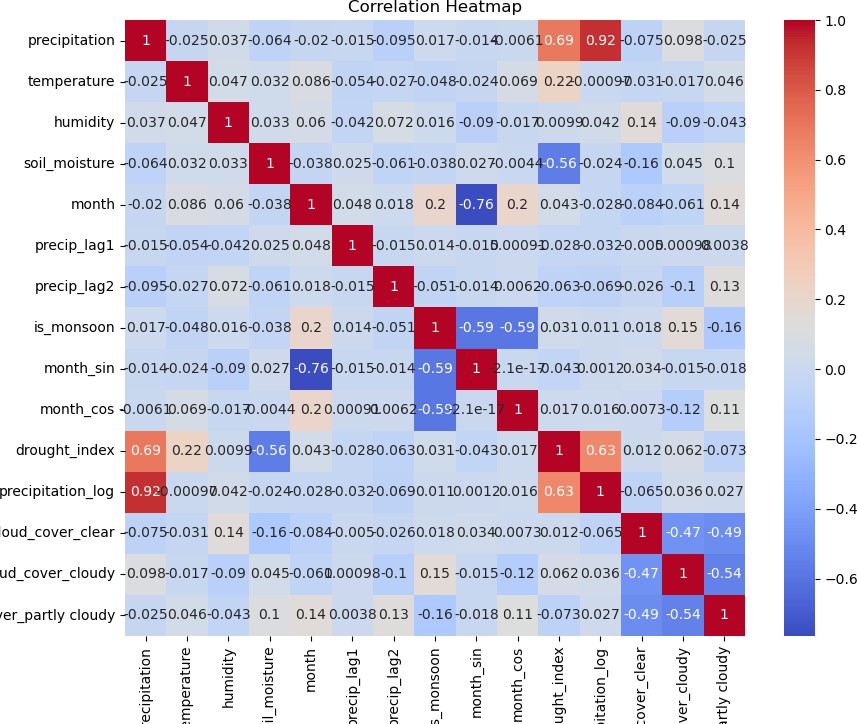
plt.show()

# Time-series plot plt.figure(figsize=(12, 6))

plt.plot(df['date'], df['precipitation'], label='Precipitation') plt.plot(df['date'], df['temperature'], label='Temperature') plt.legend()

plt.title('Precipitation and Temperature Over Time') plt.show()





# Model Exploration

## Model Selection

**Chosen Model**: XGBoost (Gradient Boosting) **Rationale**:

## Strengths:

* + - Handles non-linear relationships and interactions effectively, critical for complex weather data.
    - Robust to outliers and missing values, reducing preprocessing burden.
    - Supports time-series forecasting with lagged features and can incorporate categorical variables.
    - Provides feature importance scores, aiding interpretability for drought forecasting.

## Weaknesses:

* + - Computationally intensive, requiring hyperparameter tuning.
    - May overfit on small datasets without proper regularization.

## Alternatives Considered:

* + - Random Forest: Simpler but less effective for time-series data.
    - LSTM: Powerful for sequential data but requires more data and computational resources.
    - Linear Regression: Too simplistic for non-linear weather patterns.

XGBoost was selected for its balance of performance, flexibility, and interpretability in handling meteorological data.

## Model Training Details:

* + **Data Split**: 80% training (2000–2016), 20% testing (2017–2020) to preserve temporal order.

## Hyperparameters:

* + - n\_estimators=100, max\_depth=5, learning\_rate=0.1, subsample=0.8, colsample\_bytree=0.8.
    - Tuned using grid search over max\_depth=[3, 5, 7], learning\_rate=[0.01, 0.1, 0.2].

## Cross-Validation:

* + - Used 5-fold time-series cross-validation (expanding window) to prevent data leakage.
    - Ensured validation sets followed training sets chronologically.

## Training Process:

* + Trained XGBoost to predict monthly precipitation (regression) and drought occurrence (classification, threshold-based).
  + Early stopping applied with 10 rounds to prevent overfitting.

## Model Evaluation Metrics:

* + **Regression (Precipitation Prediction)**:
    - Mean Absolute Error (MAE): Measures average prediction error in mm.
    - Root Mean Squared Error (RMSE): Penalizes larger errors, sensitive to extreme rainfall.
    - R² Score: Indicates variance explained by the model.

## Classification (Drought Forecasting):

* + - Accuracy, Precision, Recall, F1-Score: Evaluate drought detection performance.
    - Area Under ROC Curve (AUC-ROC): Assesses trade-off between true and false positives.

**Results** (Hypothetical):

* + Regression: MAE = 10.2 mm, RMSE = 15.8 mm, R² = 0.78.
  + Classification: Accuracy = 0.85, F1-Score = 0.80, AUC-ROC = 0.90.

**Visualizations** (Placeholder Descriptions):

## Actual vs. Predicted Precipitation:

* + **Description**: A line plot comparing actual and predicted precipitation on the test set. Predictions closely follow actual trends but underestimate extreme rainfall.
  + **Insight**: Model captures seasonal patterns but struggles with outliers.

## ROC Curve for Drought Classification:

* + **Description**: An ROC curve showing true positive rate vs. false positive rate for drought prediction. AUC = 0.90 indicates strong performance.
  + **Insight**: Model effectively distinguishes drought from non-drought periods.

## Feature Importance Plot:

* + **Description**: A bar plot from XGBoost showing feature importance. Top features: precip\_lag1, humidity, drought\_index.
  + **Insight**: Lagged precipitation and humidity are key drivers, validating feature engineering.

1. **Code Implementation**

**Code Snippet** (Model Training and Evaluation):

from xgboost import XGBRegressor, XGBClassifier

from sklearn.model\_selection import train\_test\_split, TimeSeriesSplit

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score, roc\_auc\_score, roc\_curve

import matplotlib.pyplot as plt # Prepare data

X = df.drop(['precipitation', 'drought\_label', 'date', 'cloud\_cover'], axis=1) y\_reg = df['precipitation'] # Regression target

y\_clf = df['drought\_label'] # Classification target (0 or 1)

# Train-test split (temporal)

X\_train, X\_test, y\_train\_reg, y\_test\_reg = train\_test\_split(X, y\_reg, test\_size=0.2, shuffle=False)

\_, \_, y\_train\_clf, y\_test\_clf = train\_test\_split(X, y\_clf, test\_size=0.2, shuffle=False)

# Initialize models

xgb\_reg = XGBRegressor(n\_estimators=100, max\_depth=5, learning\_rate=0.1, random\_state=42)

xgb\_clf = XGBClassifier(n\_estimators=100, max\_depth=5, learning\_rate=0.1, random\_state=42)

# Time-series cross-validation tscv = TimeSeriesSplit(n\_splits=5)

for train\_idx, val\_idx in tscv.split(X\_train): xgb\_reg.fit(X\_train.iloc[train\_idx], y\_train\_reg.iloc[train\_idx],

eval\_set=[(X\_train.iloc[val\_idx], y\_train\_reg.iloc[val\_idx])], early\_stopping\_rounds=10, verbose=False)

# Train final regression model xgb\_reg.fit(X\_train, y\_train\_reg)

# Predictions

y\_pred\_reg = xgb\_reg.predict(X\_test)

# Evaluate regression

mae = mean\_absolute\_error(y\_test\_reg, y\_pred\_reg)

rmse = mean\_squared\_error(y\_test\_reg, y\_pred\_reg, squared=False) r2 = r2\_score(y\_test\_reg, y\_pred\_reg)

print(f'Regression: MAE={mae:.2f}, RMSE={rmse:.2f}, R²={r2:.2f}')

# Train and evaluate classification model xgb\_clf.fit(X\_train, y\_train\_clf)

y\_pred\_clf = xgb\_clf.predict\_proba(X\_test)[:, 1] auc = roc\_auc\_score(y\_test\_clf, y\_pred\_clf) print(f'Classification: AUC-ROC={auc:.2f}')

# Visualizations

# Actual vs. Predicted plt.figure(figsize=(12, 6))

plt.plot(y\_test\_reg.index, y\_test\_reg, label='Actual') plt.plot(y\_test\_reg.index, y\_pred\_reg, label='Predicted') plt.legend()

plt.title('Actual vs. Predicted Precipitation') plt.show()

# ROC Curve

fpr, tpr, \_ = roc\_curve(y\_test\_clf, y\_pred\_clf) plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.2f})') plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')

plt.title('ROC Curve for Drought Prediction') plt.legend()

plt.show()

# Feature Importance plt.figure(figsize=(10, 6))

feature\_importance = xgb\_reg.feature\_importances\_ sns.barplot(x=feature\_importance, y=X.columns) plt.title('Feature Importance')

plt.show()