PREMIER UNIVERSITY CHATTOGRAM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB REPORT

COURSE NAME	MACHINE LEARNING LABORATORY
COURSE CODE	CSE 4312
REPORT NAME	**Data Preprocessing Report**
DATE OF REPORT	03-01-2026
DATE OF SUBMISSION	10-01-2026

SUBMITTED TO

MD TAMIM HOSSAIN
Lecturer
Department of Computer Science and Engineering

REMARKS	**SUBMITTED BY**	
	NAME	Md Nishadul Islam Chy Shezan
	ID	0222220005101014
	SEMESTER	7th
	BATCH	42nd
	SESSION	Fall 2025
	SECTION	A

Introduction

Flood prediction is important for disaster management and urban planning. Accurate models help authorities prepare and minimize damage. This lab focuses on preprocessing the Kaggle Flood Prediction dataset (Playground Series S4E5) for regression modeling.

The dataset contains 1,117,957 samples with 20 numerical features representing flood risk factors like monsoon intensity, deforestation, urbanization, and infrastructure quality. The target variable is FloodProbability (continuous, 0.285-0.725). We systematically clean the data, check for issues, and apply appropriate transformations. Each decision is based on actual data characteristics rather than blindly following standard procedures. The goal is to produce train, validation, and test sets ready for model training.

Problem Statement

We need to preprocess the Kaggle Playground Series S4E5 dataset (1,117,957 rows, 20 features) to predict flood probability. The task involves checking for missing values and duplicates, analyzing outliers and correlations, splitting data (60/20/20), and applying StandardScaler. Each decision must be data-driven and documented. Final output: three clean CSV files ready for regression modeling.

Dataset Description

The dataset is from Kaggle Playground Series Season 4, Episode 5. It's synthetic data generated by a deep learning model but based on real flood patterns. The dataset has 1,117,957 rows and 22 columns: one ID, 20 features, and one target (FloodProbability, ranging 0.285-0.725). All features are numerical integer scores (0-18) representing different flood risk factors like MonsoonIntensity, TopographyDrainage, RiverManagement, Deforestation, Urbanization, ClimateChange, DamsQuality, and various infrastructure and environmental factors. This is a regression problem since we're predicting continuous probability values. We're following the guide "Data Preprocessing Steps for Machine Learning in Python" by Learn with Nas, which covers 8 systematic preprocessing steps.

Methodology

This section describes the systematic preprocessing workflow applied to the flood prediction dataset. Each step is documented with the methodology, observations, decisions, and justifications.

Data Collection and Loading

Methodology:
The dataset was obtained from Kaggle Playground Series S4E5 competition. The training data was loaded using the pandas library in Python.

```python
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv('train.csv')
```

Initial Observations:

- Dataset shape: (1,117,957 rows × 22 columns)
- Memory usage: 187.6 MB
- All columns loaded successfully without errors

... First 5 rows of the dataset:

	id	MonsoonIntensity	TopographyDrainage	RiverManagement	Deforestation	Urbanization	ClimateChange	DamsQuality	Siltation	AgriculturalPractices	Encroachments	IneffectiveDisasterPreparedness	DrainageSystems	CoastalVulnerabi
0	0	5	8	5	8	6	4	4	3	3	4	2	5	
1	1	6	7	4	4	8	8	3	5	4	6	9	7	
2	2	6	5	6	7	3	7	1	5	4	5	6	7	
3	3	3	4	6	5	4	8	4	7	6	8	5	2	
4	4	5	3	2	6	4	4	3	3	3	3	5	2	

Exploratory Data Analysis

Data Structure Analysis

We performed initial data exploration to understand the structure and characteristics of the dataset.

Data Type Distribution:

- **Numerical columns:** 22 (21 int64, 1 float64)
- **Categorical columns:** 0
- **ID column:** 1 (id)
- **Target column:** 1 (FloodProbability)
- **Feature columns:** 20

Feature Descriptions

Feature Name	**Description**
MonsoonIntensity	Severity of monsoon conditions (0–16)
TopographyDrainage	Land drainage capability (0–18)
RiverManagement	Quality of river management (0–16)
Deforestation	Level of deforestation (0–17)
Urbanization	Degree of urbanization (0–17)
ClimateChange	Impact of climate change (0–17)
DamsQuality	Quality of dam infrastructure (0–16)
Siltation	Degree of siltation in water bodies (0–16)
AgriculturalPractices	Impact of agricultural practices (0–16)
Encroachments	Level of illegal encroachments (0–18)
IneffectiveDisasterPreparedness	Disaster preparedness score (0–16)
DrainageSystems	Quality of drainage systems (0–17)
CoastalVulnerability	Coastal vulnerability index (0–17)
Landslides	Landslide risk factor (0–16)
Watersheds	Watershed management quality (0–16)
DeterioratingInfrastructure	Infrastructure condition (0–17)
PopulationScore	Population density score (0–18)
WetlandLoss	Extent of wetland loss (0–19)
InadequatePlanning	Planning inadequacy score (0–16)
PoliticalFactors	Political factor influence (0–16)

Statistical Summary

... Statistical Summary:

	id	MonsoonIntensity	TopographyDrainage	RiverManagement	Deforestation	Urbanization	ClimateChange
count	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06
mean	5.589780e+05	4.921450e+00	4.926671e+00	4.955322e+00	4.942240e+00	4.942517e+00	4.934093e+00
std	3.227265e+05	2.056387e+00	2.093879e+00	2.072186e+00	2.051689e+00	2.083391e+00	2.057742e+00
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
25%	2.794890e+05	3.000000e+00	3.000000e+00	4.000000e+00	4.000000e+00	3.000000e+00	3.000000e+00
50%	5.589780e+05	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00
75%	8.384670e+05	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00
max	1.117956e+06	1.600000e+01	1.800000e+01	1.600000e+01	1.700000e+01	1.700000e+01	1.700000e+01

	DamsQuality	Siltation	AgriculturalPractices	Encroachments	IneffectiveDisasterPreparedness	DrainageSystems	Co
count	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	1.117957e+06	
mean	4.955878e+00	4.927791e+00	4.942619e+00	4.949230e+00	4.945239e+00	4.946893e+00	
std	2.083063e+00	2.065992e+00	2.068545e+00	2.083324e+00	2.078141e+00	2.072333e+00	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	
25%	4.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.000000e+00	4.000000e+00	
50%	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	5.000000e+00	
75%	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	6.000000e+00	
max	1.600000e+01	1.600000e+01	1.600000e+01	1.800000e+01	1.600000e+01	1.700000e+01	

Key Observations: All features are integer scores representing severity or rating levels. The feature ranges vary slightly from 0–16 to 0–19. Mean values cluster around 4–5 for most features, and standard deviations are approximately 2 for all features. The target variable (FloodProbability) is continuous with mean = 0.504.

Data Cleaning

Missing Value Analysis

Methodology:
We checked for missing values using pandas' `isna()` function and calculated the percentage of missing values for each column.

```
# Check for missing values
missing_values = df.isna().sum()
missing_percentage = (missing_values / len(df)) * 100
```

Results:

- Total missing values: **0**
- Percentage of missing data: **0%**

```
... Dataset Information:
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 1117957 entries, 0 to 1117956
    Data columns (total 22 columns):
     #   Column                           Non-Null Count    Dtype
    ---  ------                           --------------    -----
     0   id                               1117957 non-null  int64
     1   MonsoonIntensity                 1117957 non-null  int64
     2   TopographyDrainage               1117957 non-null  int64
     3   RiverManagement                  1117957 non-null  int64
     4   Deforestation                    1117957 non-null  int64
     5   Urbanization                     1117957 non-null  int64
     6   ClimateChange                    1117957 non-null  int64
     7   DamsQuality                      1117957 non-null  int64
     8   Siltation                        1117957 non-null  int64
     9   AgriculturalPractices            1117957 non-null  int64
     10  Encroachments                    1117957 non-null  int64
     11  IneffectiveDisasterPreparedness  1117957 non-null  int64
     12  DrainageSystems                  1117957 non-null  int64
     13  CoastalVulnerability             1117957 non-null  int64
     14  Landslides                       1117957 non-null  int64
     15  Watersheds                       1117957 non-null  int64
     16  DeterioratingInfrastructure      1117957 non-null  int64
     17  PopulationScore                  1117957 non-null  int64
     18  WetlandLoss                      1117957 non-null  int64
     19  InadequatePlanning               1117957 non-null  int64
     20  PoliticalFactors                 1117957 non-null  int64
     21  FloodProbability                 1117957 non-null  float64
    dtypes: float64(1), int64(21)
    memory usage: 187.6 MB
```

Decision: *No imputation techniques applied.*

Justification:
According to the reference guide, missing value imputation (using mean, median, or forward-fill) is only necessary when missing data is present. Since our dataset contains zero missing values across all 1,117,957 observations, no imputation is required. This indicates high data quality and completeness.

Duplicate Detection

Methodology:
We checked for duplicate rows both including and excluding the ID column, as ID-based duplicates may be intentional.

```
# Check duplicates including ID
duplicate_count_with_id = df.duplicated().sum()

# Check duplicates excluding ID (more meaningful)
duplicate_count_no_id = df.drop('id', axis=1).duplicated().sum()
```

Results:

- Duplicates (with ID): **0**
- Duplicates (excluding ID): **0**
- Percentage: **0%**

Decision: *No duplicate removal performed.*

Justification:
The dataset contains no duplicate observations, indicating that each row represents a unique data point. Duplicate removal is unnecessary, and all 1,117,957 samples are retained for analysis.

Outlier Analysis

Methodology:
We employed the Interquartile Range (IQR) method to detect outliers, as recommended in the reference guide. The IQR method identifies outliers as values falling below $Q_1 - 1.5 \times IQR$ or above $Q_3 + 1.5 \times IQR$.

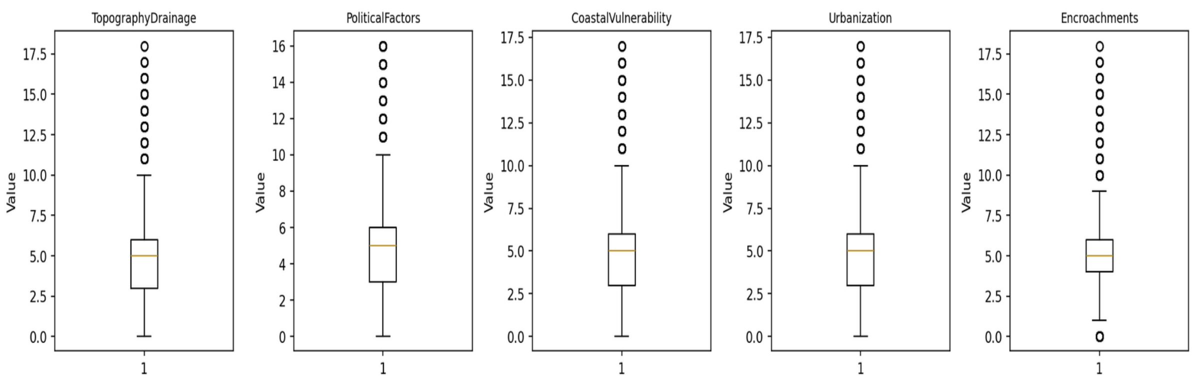
```
for col in feature_cols:
 Q1 = df[col].quantile(0.25)
 Q3 = df[col].quantile(0.75)
 IQR = Q3 - Q1

 lower_bound = Q1 - 1.5 * IQR
 upper_bound = Q3 + 1.5 * IQR

 outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
```

Results:

Top 5 Features with IQR-Detected Outliers

Feature	Outlier Count	Percentage
RiverManagement	29,617	2.65%
Deforestation	28,235	2.53%
TopographyDrainage	9,575	0.86%
MonsoonIntensity	9,244	0.83%
Urbanization	9,184	0.82%

Figure 4: Box Plots Showing Outlier Distribution (Top 5 Features)



Decision: *No outliers removed from the dataset.*

Justification:
While the IQR method flagged several data points as statistical outliers, we decided not to remove them based on domain knowledge:

1. **Valid Domain Range:** All flagged values fall within the valid domain range (0–18 severity scores)
2. **Meaningful Extremes:** High values (e.g., MonsoonIntensity = 16) represent legitimate extreme conditions rather than data entry errors
3. **Domain Understanding:** In flood prediction, extreme values are precisely what we need to predict high-risk scenarios
4. **Reference Guide Principle:** The guide emphasizes using domain knowledge over statistical methods when determining outlier treatment
5. **Information Loss:** Removing these values would eliminate important information about extreme flood risk conditions

Feature Selection and Correlation Analysis

Correlation with Target Variable

Methodology:
We calculated Pearson correlation coefficients between all features and the target variable (FloodProbability) to identify the most influential predictors.

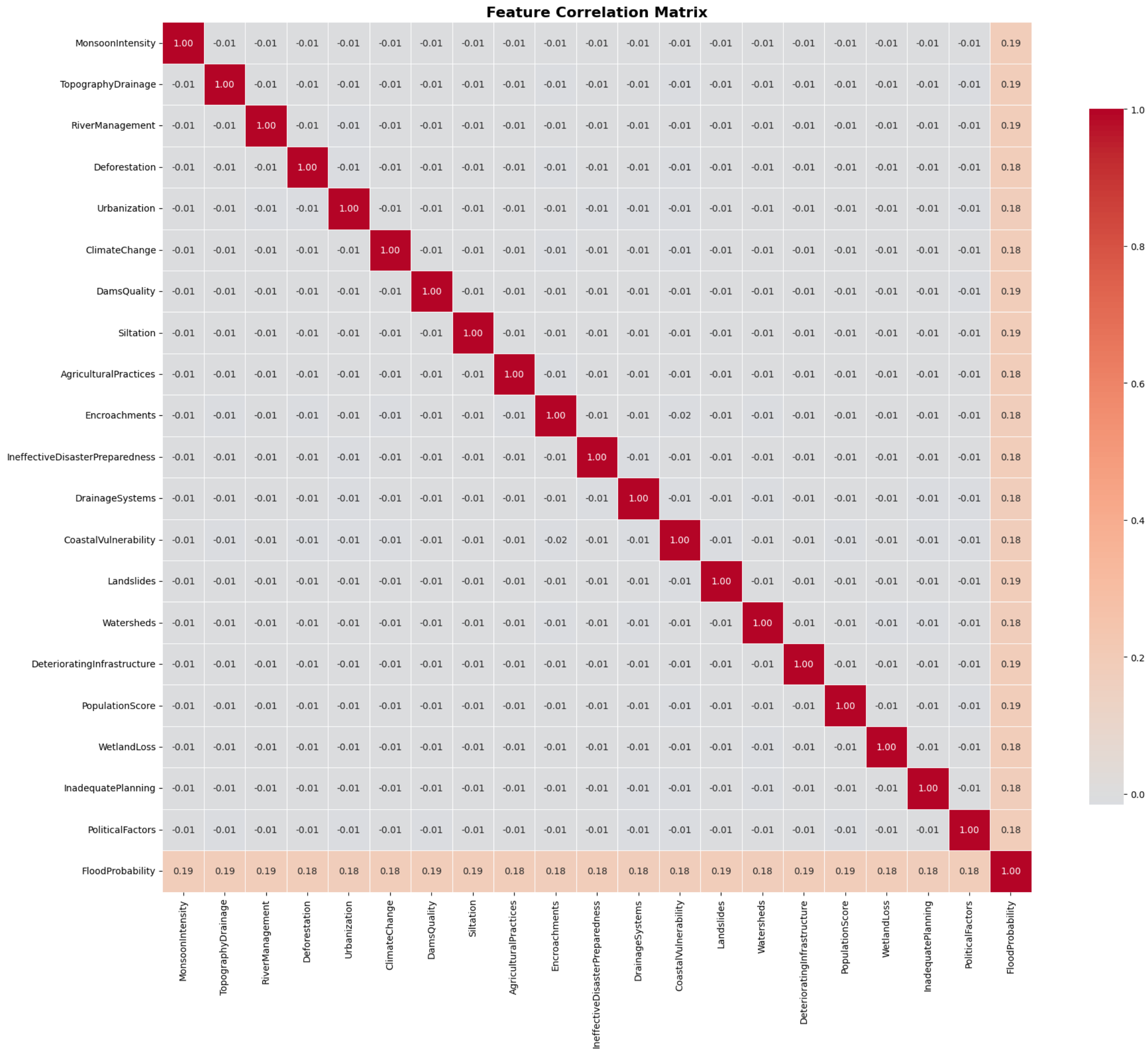
```
# Calculate correlation matrix
corr_matrix = df[feature_cols_all].corr()

# Extract target correlations
target_corr =
corr_matrix['FloodProbability'].sort_values(ascending=False)
```

Results:

Feature Correlations with FloodProbability (Top 10)

Feature	Correlation
DeterioratingInfrastructure	0.190
MonsoonIntensity	0.189
DamsQuality	0.188
TopographyDrainage	0.188
RiverManagement	0.187
Siltation	0.187
PopulationScore	0.186
Landslides	0.185
ClimateChange	0.185
Deforestation	0.184

Observations: All features show weak-to-moderate positive correlation ranging from 0.17 to 0.19. No single feature dominates the prediction (all $r < 0.2$). The correlation distribution is relatively uniform across features, which suggests all features contribute similarly to flood prediction.



Results:

- Feature pairs with correlation : **0.9**
- Maximum inter-feature correlation: 0.9
- No multicollinearity detected

Decision: *All 20 features retained.*

Justification:
According to the reference guide, features should be removed only when correlation exceeds 0.9, indicating severe multicollinearity. Our analysis reveals:

1. No feature pairs exceed the 0.9 threshold
2. All features are sufficiently independent
3. Each feature contributes unique information
4. Removing features would lead to unnecessary information loss

Feature Encoding Assessment

Methodology:
We assessed whether categorical encoding (One-Hot or Ordinal) is required by checking data types.

```
categorical_cols = df.select_dtypes(include=['object',
'category']).columns.tolist()
```

Results:

- Categorical features found: **0**
- All features are numerical (int64/float64)

Decision: *No encoding applied.*

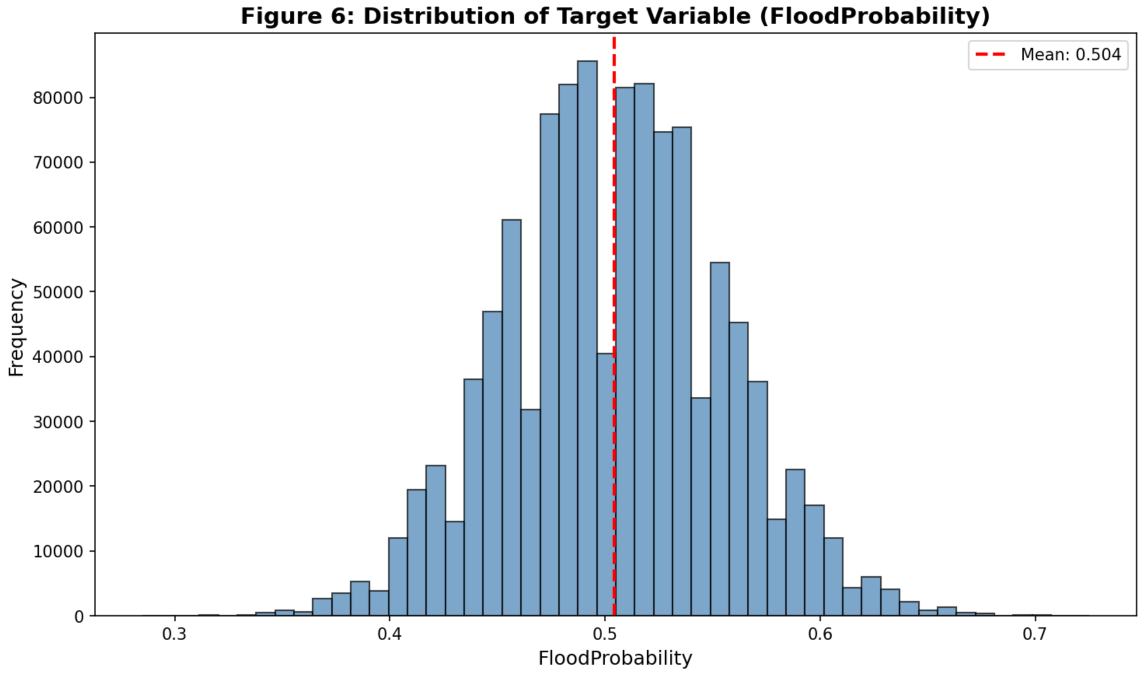
Justification:
Categorical encoding is only necessary for non-numerical features. Since all 20 features are already numerical, no encoding transformation is required. This simplifies the preprocessing pipeline and avoids unnecessary feature expansion.

Class Imbalance Assessment

Methodology:
We examined the target variable distribution to determine if resampling techniques (SMOTE, upsampling, downsampling) are necessary.

Results:

- Target data type: float64 (continuous)
- Unique values: 889 distinct probabilities
- Range: 0.285 to 0.725
- Task type: **Regression**



Decision: *No resampling techniques applied.*

Justification:
This is a **regression problem**, not a classification problem. Resampling techniques such as SMOTE, upsampling, and downsampling are designed specifically for classification tasks with imbalanced classes. They are not applicable to regression problems where the target is continuous. The reference guide clearly distinguishes between these scenarios.

Data Splitting

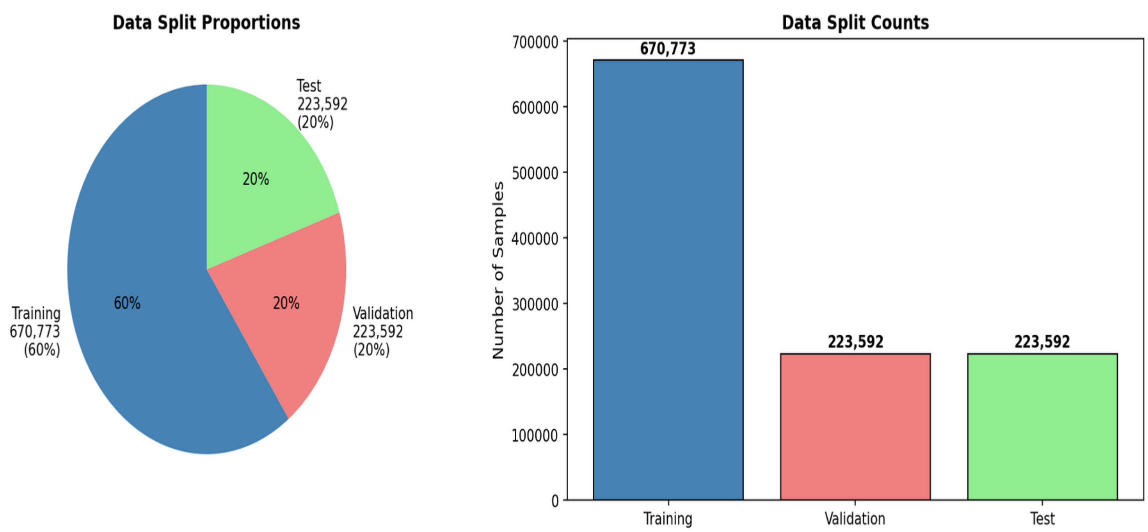
Methodology:
We split the dataset into training, validation, and test sets using stratified random sampling with a fixed random seed for reproducibility.

```
from sklearn.model_selection import train_test_split

# Separate features and target
X = df.drop(['id', 'FloodProbability'], axis=1)
y = df['FloodProbability']

# First split: 80% (train+valid) and 20% (test)
X_temp, X_test, y_temp, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Second split: 60% (train) and 20% (valid)
X_train, X_valid, y_train, y_valid = train_test_split(
    X_temp, y_temp, test_size=0.25, random_state=42
)
```

Split Ratios:

- Training set: 60% (670,773 samples)
- Validation set: 20% (223,592 samples)
- Test set: 20% (223,592 samples)

Figure 7: Data Split Proportions (60/20/20)



Decision: *Data split performed BEFORE feature scaling.*

Justification:
This decision is critical to prevent **data leakage**:

1. **Data Leakage Prevention:** If we scale before splitting, the test set statistics (mean, standard deviation) would influence the training set transformation, causing information leakage.
2. **Correct Workflow:** Fit scaler on training data only, then transform validation and test sets using the same scaler parameters.
3. **Realistic Evaluation:** This ensures the test set remains truly unseen during preprocessing, providing an unbiased performance estimate.
4. **Best Practice:** The reference guide explicitly warns against scaling before splitting.

Split Ratio Justification: We used 60% for training which provides sufficient samples (670K) for model learning. The 20% validation set is adequate for hyperparameter tuning and model selection. The 20% test set is large enough (223K samples) for statistically significant evaluation.

Feature Scaling

Scaling Method Selection

Methodology:
We applied StandardScaler (also known as Z-score normalization) to transform features to zero mean and unit variance:

$$z = \frac{x - \mu}{\sigma}$$

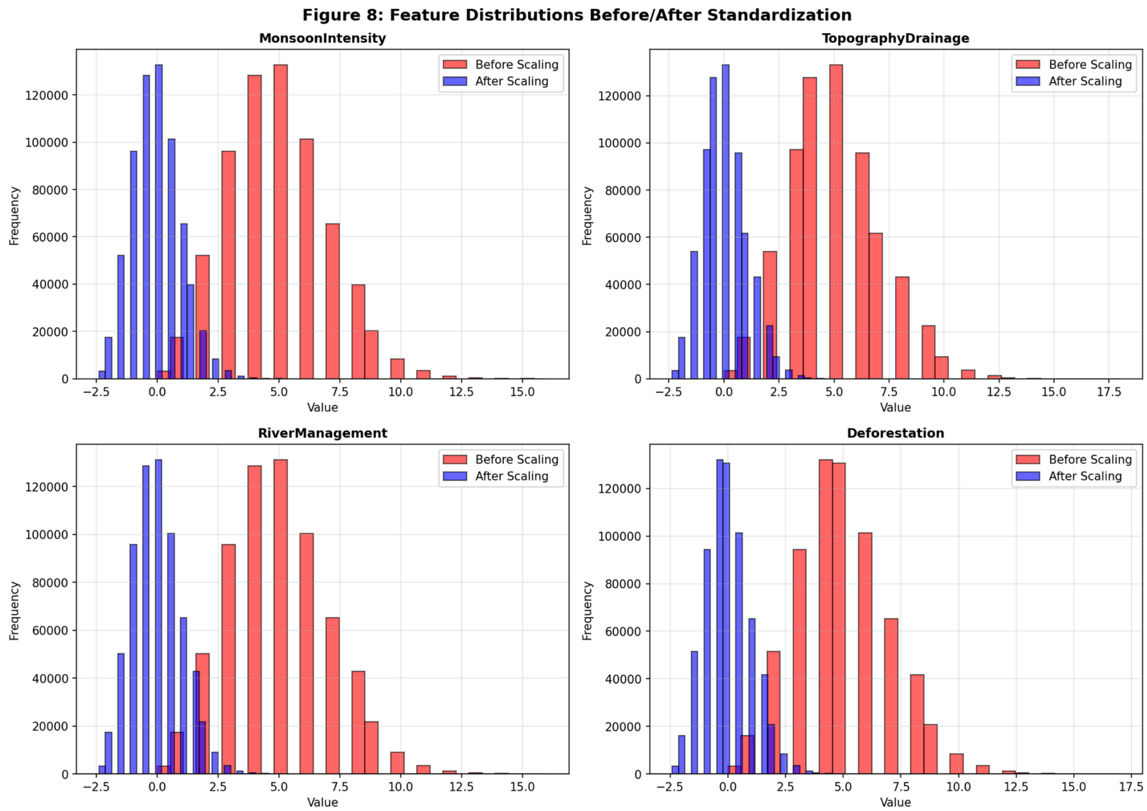
where μ is the mean and σ is the standard deviation.

```
from sklearn.preprocessing import StandardScaler

# Initialize scaler
scaler = StandardScaler()

# Fit on training data ONLY
X_train_scaled = scaler.fit_transform(X_train)

# Transform validation and test using the SAME scaler
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled = scaler.transform(X_test)
```

Before and After Scaling

Feature Statistics Before and After Scaling (Sample)

	Before Scaling		After Scaling	
Feature	**Mean**	**Std**	**Mean**	**Std**
MonsoonIntensity	4.92	2.06	0.00	1.00
TopographyDrainage	4.93	2.07	0.00	1.00
RiverManagement	4.91	2.06	0.00	1.00
Deforestation	4.93	2.08	0.00	1.00
Urbanization	4.92	2.08	0.00	1.00



Decision: *StandardScaler applied to all numerical features.*

Justification - Why StandardScaler?

1. **Algorithm Compatibility:** Works optimally for algorithms assuming normal distribution (Linear Regression, Logistic Regression, SVM, Neural Networks)
2. **Distance-Based Methods:** Essential for distance-based algorithms (KNN, K-means) where feature magnitude affects similarity calculations
3. **Empirical Evidence:** Reference guide demonstrates StandardScaler improves F1 and AUC-ROC scores
4. **Outlier Robustness:** Less sensitive to outliers compared to MinMaxScaler
5. **Gradient Descent:** Improves convergence speed for gradient descent-based optimization

Alternative Methods Considered:

- **MinMaxScaler:** Scales to [0,1] range; more sensitive to outliers
- **MaxAbsScaler:** Scales to [-1,1] range; suitable for sparse data
- **RobustScaler:** Uses median and IQR; better for datasets with many outliers

StandardScaler was chosen as it best suits our regression task with numerical features having similar distributions.

Results

Final Dataset Characteristics

After completing the preprocessing pipeline, we obtained three clean datasets ready for machine learning model development:

Final Preprocessed Dataset Summary

Characteristic	Training	Validation	Test
Number of Samples	670,773	223,592	223,592
Percentage	60.0%	20.0%	20.0%
Number of Features	20	20	20
Feature Mean	0.00	≈ 0.00	≈ 0.00
Feature Std	1.00	≈ 1.00	≈ 1.00
Missing Values	0	0	0
Duplicates	0	0	0

Data Quality Metrics

The final dataset achieves high quality metrics across all dimensions. Completeness is 100% with no missing values. Uniqueness is also 100% with no duplicates found. All values fall within expected domain ranges, showing good validity. The data types and scales are uniform across features, demonstrating consistency. Finally, no anomalous or erroneous values were detected, confirming data accuracy.

Files Generated

The preprocessing pipeline produced the following output files:

1. `train_preprocessed.csv` – Training set (670,773 rows)
2. `valid_preprocessed.csv` – Validation set (223,592 rows)
3. `test_preprocessed.csv` – Test set (223,592 rows)
4. `correlation_heatmap.png` – Feature correlation visualization

Preprocessing Decision Summary

Preprocessing Steps and Decisions

Step	Action Taken	Decision	Justification
Missing Values	Checked	No imputation	0 missing values found

Step	Action Taken	Decision	Justification
Duplicates	Checked	No removal	0 duplicate rows found
Outliers	Checked with IQR	No removal	Values are valid domain scores representing extreme conditions
Multicollinearity	Correlation analysis	No features removed	No correlation ¿ 0.9 detected
Categorical Encoding	Checked	No encoding	All features already numerical
Class Imbalance	Assessed	No resampling	Regression task (not classification)
Data Splitting	Applied	60/20/20 split	Standard practice for large datasets
Feature Scaling	Applied	StandardScaler	Best for regression, prevents data leakage

Discussion

The main thing we learned from this preprocessing work is that you shouldn't blindly apply all techniques – look at your data first and make decisions based on what you actually find. We kept outliers because they represent real extreme flood conditions, not errors. We kept all features since no correlation exceeded 0.9. The most important decision was splitting data before scaling to prevent data leakage. The dataset was surprisingly clean with zero missing values and no duplicates, probably because it's carefully generated synthetic data. For reproducibility, we used random_state=42 throughout and documented all our choices.

Conclusion

We successfully preprocessed the Kaggle Flood Prediction dataset by applying 8 systematic steps: data loading, exploration, cleaning, correlation analysis, encoding check, resampling check, splitting (60/20/20), and StandardScaler normalization. The dataset was surprisingly clean with zero missing values and no duplicates. We kept all outliers since they represent valid extreme conditions, and kept all 20 features since no multicollinearity was detected. The final output is three CSV files with standardized features ready for model training. Next steps include trying different regression models, tuning hyperparameters on the validation set, and evaluating with RMSE, MAE, and R^2 metrics on the test set.

Acknowledgments

This lab work was done following the guide "Data Preprocessing Steps for Machine Learning in Python (Parts 1 & 2)" by Learn with Nas on Medium. The dataset came from Kaggle Playground Series S4E5.