# UML Class Diagram - Hydrological Modeling System

This document describes the UML Class Diagram of a modular, extensible hydrological modeling system. It supports spatial-temporal inputs, physical-informed learning (PINN/XPINN), curriculum learning, ensemble rainfall forecasts, and future transfer learning to ungauged basins. The architecture enables staging experiments, modular data handling, and multiple model types.

## ConfigParser

* + load\_config(path: str) -> dict
* + validate\_schema(config: dict) -> bool

🔗 Loads XML/YAML configuration files with watershed info, grid type, dates, model setup.

## RainLoader

* + load\_netcdf(file: str) -> xarray.Dataset
* + interpolate(method='kriging') -> pd.DataFrame
* + subset\_by\_bbox(bbox) -> pd.DataFrame

🔗 Reads rainfall from NetCDF and interpolates it spatially (e.g., to UGRID). Supports future ensemble inputs.

## DischargeProcessor

* + load\_csv(file: str) -> pd.DataFrame
* + resample(freq='30min') -> pd.DataFrame
* + filter\_valid\_stations() -> list

🔗 Processes observed discharge CSV files. Supports frequency resampling and station filtering.

## SpatialMerger

* + merge\_rain\_discharge(rain\_df, q\_df) -> pd.DataFrame
* + attach\_static\_features(df, static\_map) -> pd.DataFrame

🔗 Merges rainfall, discharge and spatial features post-interpolation.

## FeatureEngineer

* + normalize(df: pd.DataFrame) -> pd.DataFrame
* + compute\_tau(df) -> pd.DataFrame
* + spatial\_embedding(df, method='grid\_id') -> pd.DataFrame
* + feature\_selection(df) -> pd.DataFrame

🔗 Handles feature scaling, time normalization, optional spatial embedding, and feature selection logic.

## DatasetBuilder

* + create\_sequences(df, seq\_len=6) -> TensorDataset
* + split\_by\_event(df, mode='strong') -> list[pd.DataFrame]
* + balance\_dataset() -> pd.DataFrame

🔗 Generates model-ready sequences, enables curriculum learning based on event strength or spatial criteria.

## ModelTrainer

* + train(model, dataloader, config) -> TrainedModel
* + compute\_loss(y\_pred, y\_true) -> torch.Tensor
* + compute\_physics\_residual(...) -> torch.Tensor

🔗 Trains a model (PINN/XPINN) with physics loss, using spatial-temporal features and rain/discharge inputs.

## Evaluator

* + evaluate(model, test\_loader) -> dict
* + plot\_hydrographs(...)
* + animate\_event\_rain\_and\_q(...) -> video

🔗 Evaluates model performance: generates plots, metrics, and rainfall-discharge animations.

## StagingRunner

* + run\_model\_comparison(models: list)
* + test\_physics\_variants()
* + retrain\_on\_new\_data()

🔗 Dedicated for experimentation: architecture comparison, physics variations, fine-tuning on new data.

## Class Dependencies

* ModelTrainer depends on: DatasetBuilder, FeatureEngineer, ConfigParser
* Evaluator depends on: ModelTrainer, RainLoader, DischargeProcessor
* StagingRunner depends on: ModelTrainer, Evaluator
* RainLoader / DischargeProcessor depends on: Independent

## הסבר על מסמך: ****UML Class Diagram - Hydrological Modeling System****

המסמך מתאר את מבנה המחלקות (classes) בקוד שלך — כלומר: את החלקים שבונים את המערכת, כל אחד עם אחריות ברורה, קלטים, ופלטים.

## 🧱 מחלקות עיקריות (Classes) והאחריות שלהן:

### 1. ConfigParser

* קורא את קובץ ה־XML או YAML שמגדיר את הריצה (איזה אגן, אילו תאריכים, סוג גריד, סוג מודל).
* מאפשר לבדוק אם הקובץ תקין.

### 2. RainLoader

* קורא קבצי גשם בפורמט NetCDF.
* מבצע אינטרפולציה (למשל קריגינג) כדי לשייך את הגשם לנקודות או לגריד.
* תומך בעתיד באנצמבל (תחזיות גשם מרובות).

### 3. DischargeProcessor

* טוען נתוני ספיקה (CSV).
* מאפשר שינוי תדירות (resample), סינון תחנות בעייתיות וכו'.

### 4. SpatialMerger

* מאחד את הגשם, הספיקה, והתכונות המרחביות לקובץ אחד סופי.
* מתבצע אחרי האינטרפולציה, כדי לא להעמיס עליה.

### 5. FeatureEngineer

* נרמול הפיצ'רים
* חישוב זמן מנורמל (tau)
* Spatial Embedding (למשל, שיוך של מיקום למטריצה)
* תשתית לבחירת פיצ'רים (feature selection)

### 6. DatasetBuilder

* הופך את הדאטה לרצפים (sequence) שמתאימים לאימון מודלים כמו LSTM.
* תומך בלמידה הדרגתית (לפי עוצמת האירוע או לפי אזור).
* יכול לאזן את הדאטה.

### 7. ModelTrainer

* מבצע את תהליך האימון בפועל.
* מחשב איבר איבוד (Loss) עם/בלי פיזיקה (PINN/XPINN).
* שומר את המודל המאומן.

### 8. Evaluator

* מחשב מדדים (MSE, R2, שגיאות וכו').
* מצייר גרפים של הידרוגרפים.
* מייצר סרטון של גשם שמגיע ותגובה בתחנות.

### 9. StagingRunner

* נועד לניסויים בלבד:
  + השוואת ארכיטקטורות (FCN מול LSTM)
  + שינוי loss
  + fine-tuning עם דאטה חדש
  + ניסוי במשוואות פיזיקליות שונות

## 🔗 קשרים בין המחלקות (תלות):

* ModelTrainer תלוי ב־ConfigParser, DatasetBuilder, FeatureEngineer
* Evaluator משתמש ב־ModelTrainer, RainLoader, DischargeProcessor
* StagingRunner מפעיל ניסויים על בסיס כל האחרים

## 🔄 Update: New Class – EquationAssigner

class EquationAssigner:  
 + assign\_equation(df: pd.DataFrame) -> pd.DataFrame  
 + compute\_manning(df: pd.DataFrame) -> pd.DataFrame  
  
🔗 Assigns governing equations to each spatial cell and estimates Manning's coefficient where needed.  
To be integrated before training, after feature engineering.