Pipeline: China Dataset for SEE Ensemble

This document describes the end-to-end pipeline for the **China** dataset in the ensemble Software Effort Estimation (SEE) experiment.

---

1. Dataset

Item	Value
Source	`datasets/china.arff`
Format	ARFF (Weka)
Target	`Effort` (dependent variable)
Features	All numeric attributes except `ID` and `Effort`
Dropped	`ID` (not used as a feature)

Feature columns used: AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDU_UFP, Resource, Dev.Type, Duration, N_effort.

---

2. Pipeline Overview

```
china.arff → Load ARFF → Drop ID, select numeric → Raw (X, y)
    → [Per iteration] Shuffle → Split 70/30 → Scale (fit on train) → Train
→ Predict → Metrics
    → Aggregate (mean ± std) over iterations
```

---

3. Step-by-Step

3.1 Load and prepare raw data

1. **Load ARFF**
   `scipy.io.arff.loadarff()` reads `datasets/china.arff`; rows are converted to a DataFrame with numeric types.

2. **Select columns**
   - Target: `Effort`
   - Drop: `ID`
   - Features: all remaining numeric columns (AFP, Input, Output, Enquiry, File, Interface, Added, Changed, Deleted, PDR_AFP, PDR_UFP, NPDR_AFP, NPDU_UFP, Resource, Dev.Type, Duration, N_effort).
3. **Drop missing**
   Rows with any NaN in features or target are removed.
4. **Output**
   Raw feature matrix `X` and target vector `y` (no scaling yet). Stored in `SEEFrame(X, y, feature_names)`.

3.2 Monte Carlo loop (per iteration)

For each of the `n_iterations` (e.g. 1000):

1. **Shuffle**
   Permute rows of `(X, y)` with a fixed seed for this iteration (reproducible).
2. **Split**
   - Train: first 70% of shuffled data
   - Test: remaining 30%
   - No overlap; test is never used for fitting.
3. **Scale (no leakage)**
   - Fit `MinMaxScaler` on **training** data only (X_train, y_train).
   - Transform both train and test with that scaler → [0, 1].
   - Result: `X_train_s`, `y_train_s`, `X_test_s`, `y_test_s`.
4. **Train**
   For each model (baselines + ensemble):
   - Clone the template, set `random_state=iter_seed`.
   - `model.fit(X_train_s, y_train_s)`.
5. **Predict**
   `y_pred = model.predict(X_test_s)` (on scaled test features only).

6. **Metrics (on scaled test)**
   On `(y_test_s, y_pred)` compute:
   MAR, MSE, RMSE, MAE, R^2.

7. **Store**
   Save this iteration's metrics per model.

3.3 After all iterations

- **Aggregate**
  For each model: mean and standard deviation of MAR, MSE, RMSE, MAE, R^2 across iterations.
- **Relative gain**
  For each model: RG vs Linear baseline (MAR).
- **Wilcoxon**
  Compare each model's MAR distribution vs Linear (across iterations); report p-value (H1: model better than Linear).

---

4. Models run on China

- **Baselines:** Linear Regression, ELM-2, ELM-5
- **Bagging:** B-LR, B-RR, B-RI, B-LA
- **Stacking:** ST-LR, ST-RR, ST-RI, ST-LA

Stacking uses `cv=5` for the meta-learner (out-of-fold base predictions; no leakage).

---

5. How to run

From the project root:

```
# China only (quick test: 5 iterations)
uv run python run_see.py --datasets china --max-iter 5

# China only (full: 1000 iterations)
uv run python run_see.py --datasets china --iterations 1000

# China with other datasets
uv run python run_see.py --datasets albrecht kemerer china --max-iter 5
```

Config for China is in `code/ensemble_see/config.py` under `DATASET_REGISTRY["china"]`.

---

6. Config summary (China)

Setting	Value
Path	`datasets/china.arff`
Target column	`Effort`
Drop columns	`ID`
Train ratio	0.70
Default iterations	1000
Normalization	Max-Min [0,1], fit on train only