Supplementary Material for

Applying Machine Learning Methods to improve Rainfall–Runoff Modeling in Subtropical River Basins

Haoyuan Yu[1], Qichun Yang[1,2*]

[1] Thrust of Earth, Ocean and Atmospheric Sciences, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511400, China

[2] Center for Ocean Research in Hong Kong and Macau, Hong Kong University of Science and Technology, Hong Kong, China

*	Correspondence: qichunyang@hkust-gz.edu.cn

Text S1. Details of the Wapaba model

Referring to Wang, *et al.* [1], the Wapaba model (Figure S1) calculations consist of five stages:

1.Total rainfall *P*(*t*) in time step *t* is partitioned to basin water consumption *X*(*t*) and yield *Y*(*t*). Catchment water consumption is the rainfall that replenishes the soil water stored and returns it to the atmosphere through evapotranspiration. The remaining rainfall is the catchment water yield.

2. Total water available for evapotranspiration *W*(*t*) is partitioned to actual evapotranspiration *ET*(*t*) and water remaining in the soil water store *S*(*t*).

3. The catchment water yield is partitioned into surface runoff $Q_s(t)$ and water that replenishes the groundwater store *R*(*t*).

4. The groundwater store is drained to give base flow $Q_b(t)$.

5. The surface runoff and base flow are summed to give the total monthly flow *Q*(*t*).

The following equations are:

$$\frac{Consumption}{Demand} = F\left(\frac{Supply}{Demand}, \alpha\right) = 1 + \frac{Supply}{Demand} - \left[1 + \left(\frac{Supply}{Demand}\right)^{\alpha}\right]^{\frac{1}{\alpha}} \quad \text{(S1)}$$

$$X(t) = X_0(t)F\left(\frac{P(t)}{X_0(t)}, \alpha_1\right) \quad \text{(S2)}$$

$$X_0(t) = ET_o(t) + \left(S_{max} - S(t-1)\right) \quad \text{(S3)}$$

$$Y(t) = P(t) - X(t) \quad (S4)$$

$$W(t) = S(t-1) + X(t) \quad (S5)$$

$$ET(t) = ET_o(t)F\left(\frac{W(t)}{ET_O(t)}, \alpha_2\right) \quad (S6)$$

$$S(t) = W(t) - ET(t) \quad (S7)$$

$$R(t) = \beta Y(t) \quad (S8)$$

$$Q_s(t) = Y(t) - R(t) \quad (S9)$$

$$Q_b(t) = G(t-1)\left(1 - e^{-\frac{T}{K}}\right) + R(t)\left(1 - \left(\frac{K}{T}\right)\left(1 - e^{-\frac{T}{K}}\right)\right) \qquad (S10)$$

$$G(t) = G(t-1) + R(t) - Q_b(t) \qquad (S11)$$

$$Q(t) = Q_s(t) + Q_b(t) \qquad (S12)$$

where $F()$ is as Equation (S1), referring to consumption curves, α_1 is the catchment consumption curve parameter, and $X_0(t)$ is the catchment water potential consumption. $ET_0(t)$ is the potential evapotranspiration, $S(t-1)$ is the amount of water held in the soil store at the end of time step $t-1$, and $S_{\max}$ is the maximum water-holding capacity of the soil store. α_2 is the evapotranspiration curve parameter. Parameter β is the proportion of the catchment water yield as groundwater. K is a time constant (in units of time) to produce base flow. T is the length of time step t.

Five parameters need to be calibrated in WAPABA, which are $S_{\max}$, α_1, $\alpha_{2,}\beta$, and K.

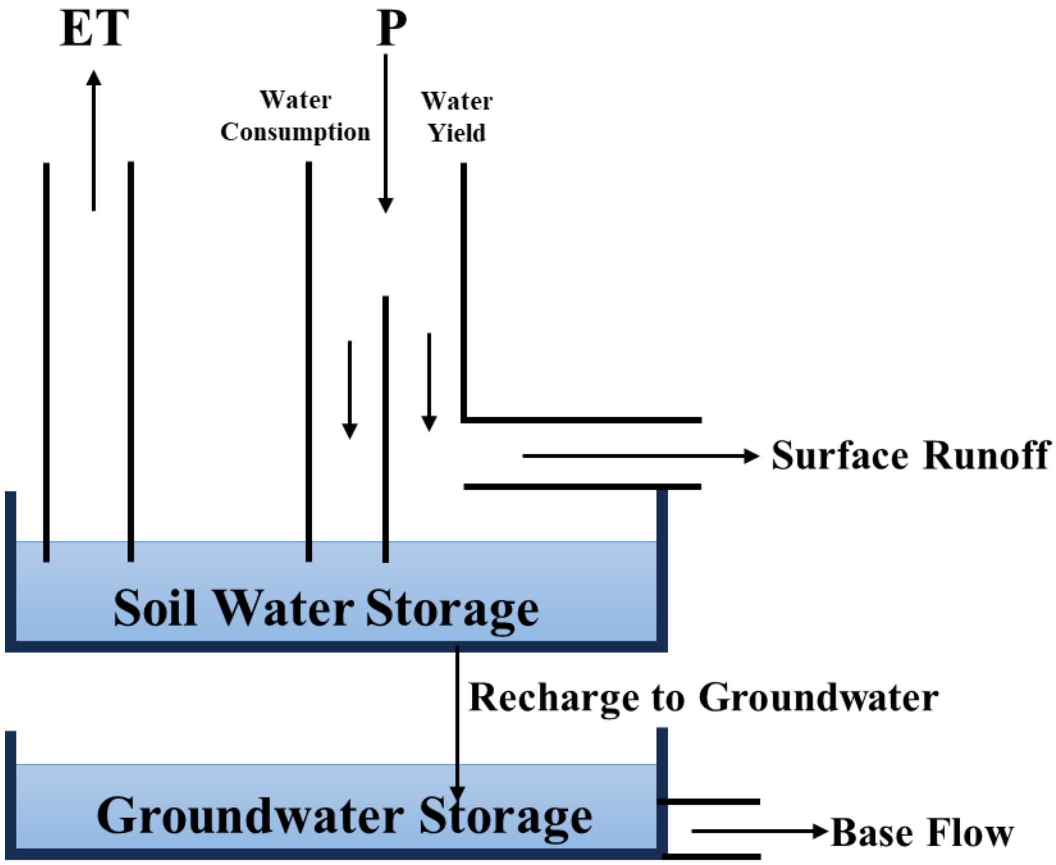

Figure S1. Schematic diagram of the WAPABA (water partition and balance) model. P means

precipitation and ET means reference crop evapotranspiration.

Text S2. Details of the LSTM model

The LSTM model architecture encompasses multiple components, each serving a specific purpose in the sequence learning process. An LSTM layer is designed with the following components:

Forget Gate: The forget gate f_t regulates the amount of information to discard from the cell state and is computed as:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \qquad (S13)$$

Input Gate: The input gate i_t determines which new information to store in the cell state and is calculated as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad (S14)$$

Candidate Cell State: The candidate cell state $\widetilde{c_t}$ computes the new candidate values that could be added to the cell state:

$$\widetilde{c_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \qquad (S15)$$

Cell State Update: The cell state c_t is updated based on the forget gate, input gate, and candidate cell state:

$$c_t = f_t * c_{t-1} + i_t * \widetilde{c_t} \qquad (S16)$$

Output Gate: The output gate o_t determines what information should be output as the hidden state and is computed as:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \qquad (S17)$$

Hidden State Update: The hidden state h_t is updated by applying the output gate to the cell state:

$$h_t = o_t * \tanh(c_t) \qquad (S18)$$

where $\mathrm{W_f}, \mathrm{W_i}, \mathrm{W_C}, \mathrm{W_o}$ are weight matrices for the forget gate, input gate, candidate cell state, and output gate, respectively; $\mathrm{b_f}, \mathrm{b_i}, \mathrm{b_C}, \mathrm{b_o}$ are bias terms corresponding to the forget gate, input gate, candidate cell state, and output gate, respectively; σ is the sigmoid function used to compute the outputs of the gates; tanh is the hyperbolic tangent function used to calculate the candidate cell state; h_{t-1} represents the previous time step's hidden state; x_t represents the input at the current time step; f_t is the output of the forget gate, controlling what information to forget from the cell state; i_t is the output of the input gate, controlling the flow of new information into the cell state; $\widetilde{C}_t$ represents the candidate cell state, calculating the new candidate values to be added to the cell state; C_t is the cell state at the current time step, representing the memory at that moment; o_t is the output of the output gate, controlling the flow of information from the cell state to the hidden state; and h_t is the hidden state at the current time step, representing the output at that moment.

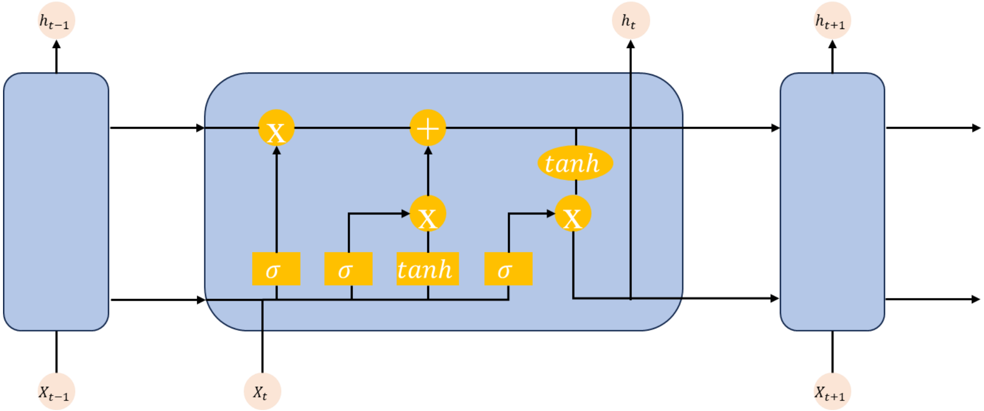

Figure S2. Schematic diagram of the Long Short-Term Memory Network.

Text S3. Details of the SVM model

Given a training set (x_i, y_i) with *m* samples, where $x_i \in R^n$ represents the input feature vector and $y_i \in R$ is the corresponding target variable, the SVM model aims to find a prediction function:

$$f(x) = \sum_{i=1}^{m} \alpha_i K(x_i, x) + b \qquad (S19)$$

This function needs to satisfy the following constraints for all *i*:

$$f(x_i) - y_i \le \epsilon + \xi_i \qquad (S20)$$

$$y_i - f(x_i) \le \epsilon + \xi_i \qquad (S21)$$

$$\xi_i \ge 0 \qquad (S22)$$

$$\sum_{i=1}^{m} \alpha_i \le C \qquad (S23)$$

where $f(x_i)$ is the predicted output, y_i is the actual target output, ϵ is the margin of tolerance, $\mathrm{\xi_i}$ are slack variables, and α is the Lagrange multiplier, which is constrained by the penalty parameter, C. The optimization problem of the SVR model with Radial Basis Function (RBF) kernel can be formulated as minimizing the loss function while meeting the constraints:

$$\min_{\alpha,b,\xi}\left(\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(x_i,x_j)+C\sum_{i=1}^{m}(\xi_i+\xi_i^*)\right) \qquad (S24)$$

where ξ_i^* typically represents the optimal value of the slack variable ξ_i after solving the optimization problem. This loss function is subjected to the following constraints:

$$y_i-\sum_{j=1}^{m}\alpha_j K(x_j,x_i)-b\le\epsilon+\xi_i \qquad (S25)$$

$$\sum_{i=1}^{m}\alpha_i K(x_i,x_j)+b-y_j\le\epsilon+\xi_j^* \qquad (S26)$$

$$\xi_i,\xi_i^*\ge 0 \qquad (S27)$$

where K represents the kernel function and b is the bias terms. By solving this optimization problem, the optimal SVR model parameters α and b can be obtained.

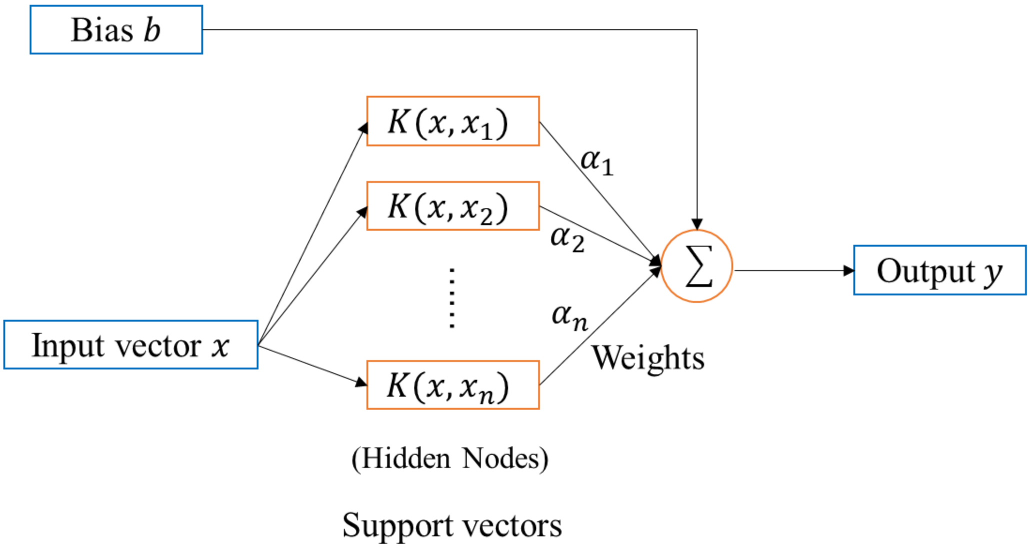

Figure S3. Schematic diagram of the Support Vector Machine.

Text S4. Details of the GPR model

Assuming the coordinates of the i-th point in space are $x^{(i)}$ and the data output at that point is $y^{(i)}$, the expression of the GPR model is given by:

$$\mathrm{y^{(i)}} = \mathrm{f}\left(\mathrm{x^{(i)}}\right) + \varepsilon \qquad (S28)$$

where ε represents the noise variable, which follows a normal distribution $\mathcal{N}(0, \sigma^2)$. In GPR, we assume that the function $f\left(x^{(i)}\right)$ follows a Gaussian process:

$$\mathrm{f}\left(\mathrm{x^{(i)}}\right) \sim \mathcal{GP}\left(\mathrm{m(x)}, \mathrm{K(x, x')}\right) \qquad (S29)$$

where $m(x)$ is the mean function, and $K(x, x')$ is the covariance function, also known as the kernel function. In this study, we use the RBF function:

$$K(x, x') = \sigma_f^2 \exp\left(-\frac{\left||x - x'|\right|^2}{2l^2}\right) \qquad (S30)$$

where x and x' are two points in the input space, $\left||x - x'|\right|^2$ denotes the squared Euclidean distance between them, σ_f^2 is the variance of the process, representing the variation of function outputs, and *l* is the length scale parameter, determining the smoothness or feature-length scale of the function values in the input space. By utilizing the optimization methods in Python, we can solve the optimal parameters of the model and make predictions.

Text S5. Details of the LR model

The Lasso Regression model is defined by the following objective function:

$$\min_{\beta_0, \beta}\left\{\frac{1}{2\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\left(\mathrm{y_i} - \beta_0 - \mathrm{x_i^T}\beta\right)^2 + \lambda|\beta|_1\right\} \qquad (S31)$$

where N is the number of observations, y_i represents the dependent variable for the $\mathrm{i^{th}}$ observation, $\mathrm{x_i}$is a feature vector for the i^{th} observation, β_0 is the intercept, β denotes the vector of coefficients associated with the features, λ is a non-negative regularization parameter controlling the strength of the L1 penalty, and $|\beta|_1$$ represents the L1 norm of the coefficient vector, promoting sparsity in the model parameters. A coordinate descent algorithm was used for optimizing the Lasso

Regression model, iteratively minimizing the objective function by updating one coefficient at a time until convergence.

Text S6. Details of the XGB model

The objective function of XGBoost integrates two components: the training loss and the regularization term, expressed as:

$$\mathrm{Obj} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{l}(\mathrm{y}_{\mathrm{i}}, \hat{\mathrm{y}}\mathrm{i}) + \sum \mathrm{k} = 1^{\mathrm{K}}\, \Omega(\mathrm{f}_{\mathrm{k}}) \qquad (S32)$$

where $l(y_i, \hat{y}i)$ denotes the loss function comparing the predicted value $\hat{y}i$ to the actual value y_i, and $\Omega(f_k)$ represents the regularization term for the k^{th} tree, aimed at controlling model complexity. The regularization term is defined as:

$$\Omega(\mathrm{f}) = \gamma\mathrm{T} + \frac{1}{2}\lambda \sum_{\mathrm{j}=1}^{\mathrm{T}} \mathrm{w}_{\mathrm{j}}^{2} \qquad (S33)$$

in which *T* is the number of leaves in the tree, w_{j} is the weight of the j^{th} leaf, and γ and λ are parameters that regulate the complexity of the tree and the square sum of the leaf weights, respectively. The model utilizes a gradient boosting framework, applying a coordinate descent algorithm to efficiently update the model by optimizing one coefficient at a time until convergence, which can be achieved by Python.

Text S7. Details of the LGBM model

The objective function of LightGBM integrates two components: the training loss and the regularization term, expressed as the equations (32) and (33), like XGB. The optimization of the objective function in LGBM is achieved through a gradient-based approach that iteratively updates the model by constructing new trees that predict the gradients of the loss with respect to the model's predictions, which is different from XGB. Such optimization processes can also be achieved by Python.

Text S8. Other figures and tables in the supporting information

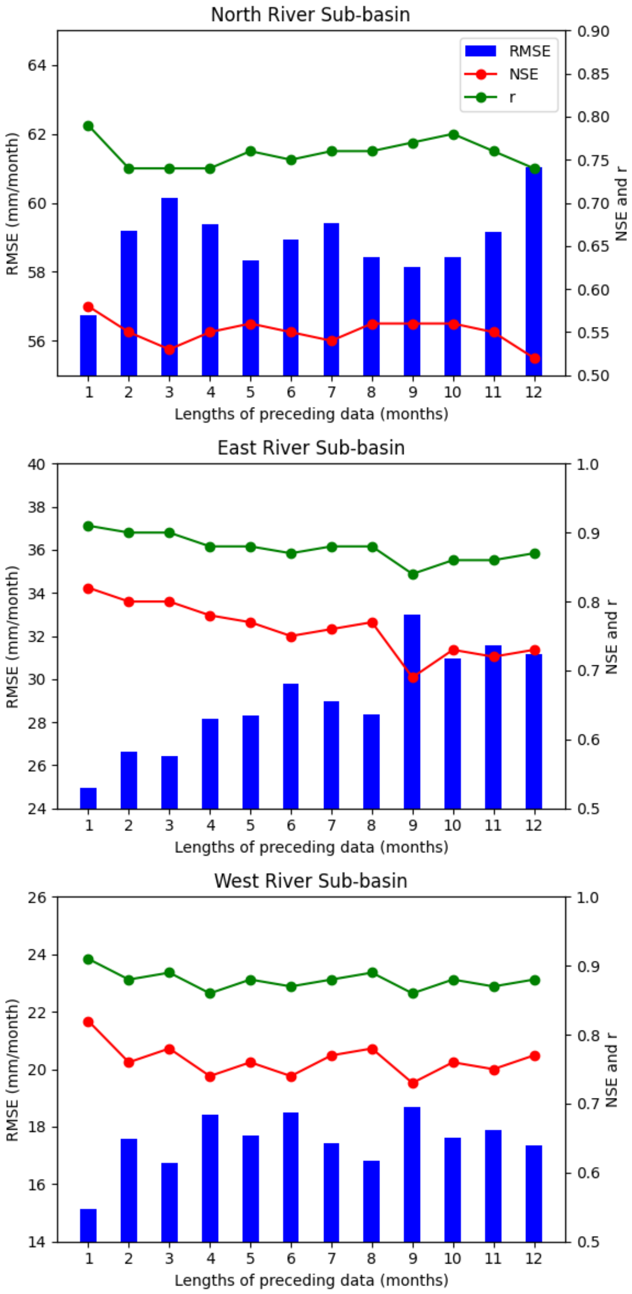

Figure S4. RMSE, NSE, and r for different input time lags (months ahead) for the LSTM model.

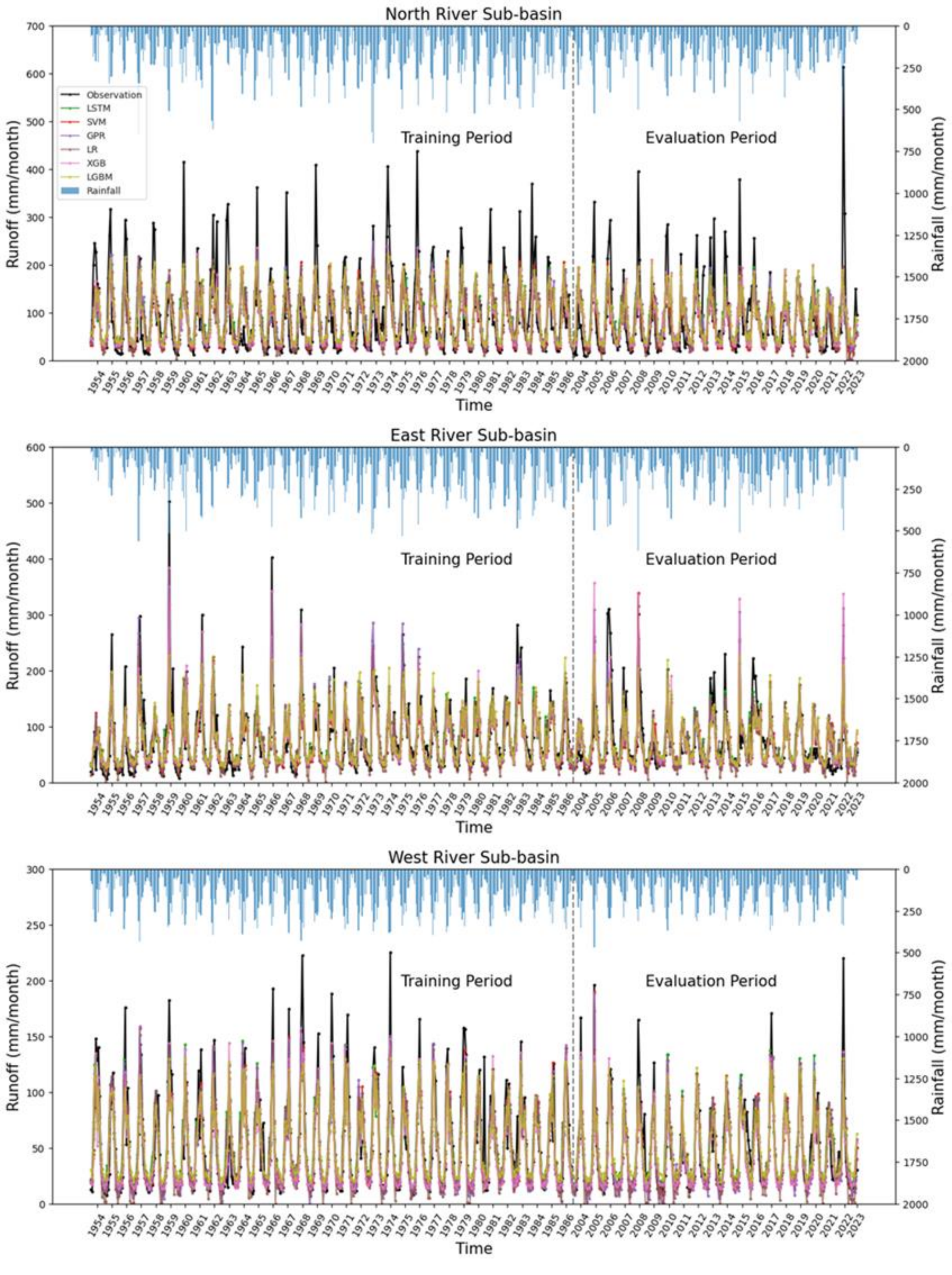

Figure S5. Runoff simulations using multiple machine learning models in Experiment 1 against observations among different sub-basins in the training (1954-1986) and evaluation (2004-2023) periods.

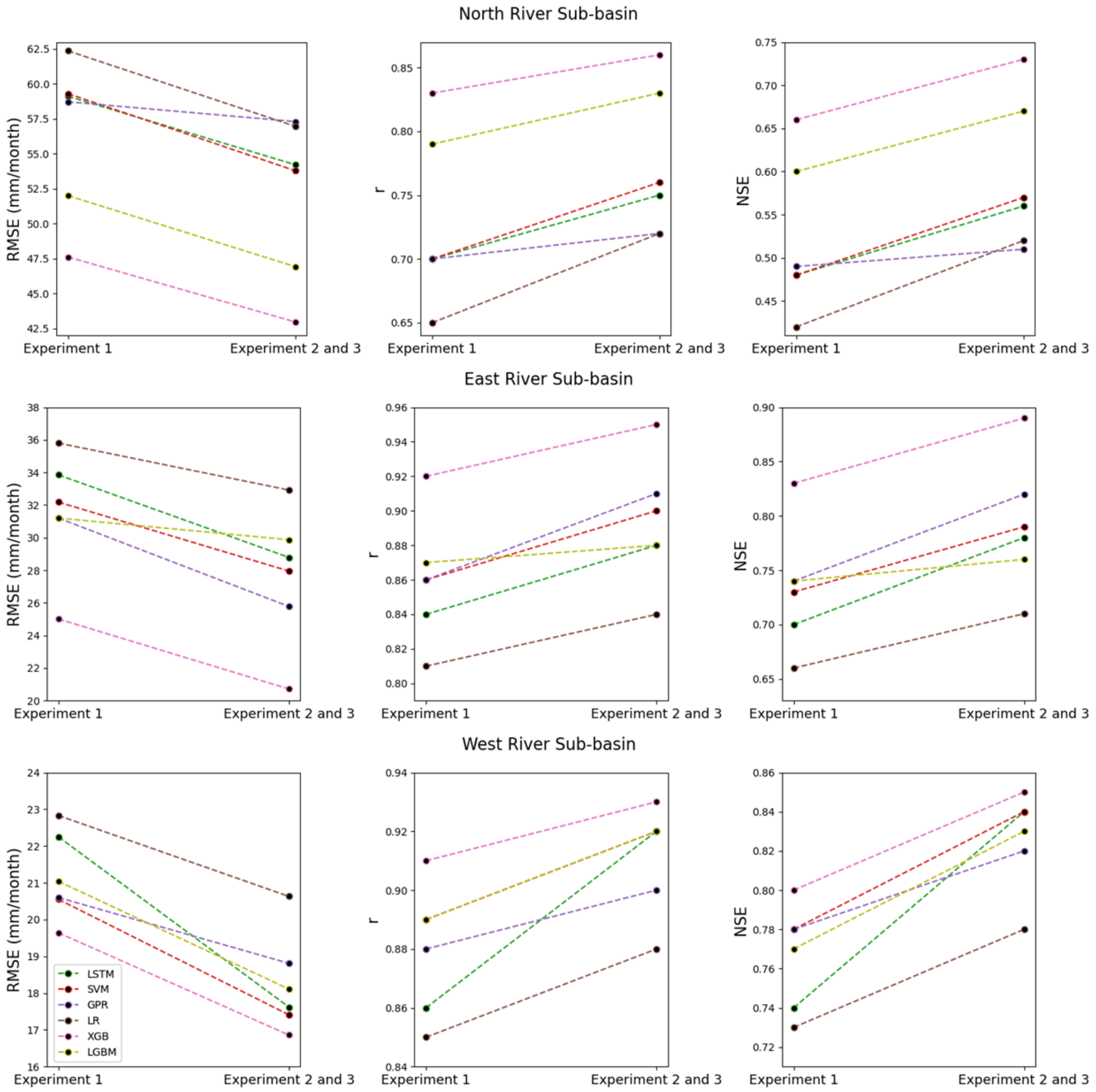

Figure S6. Root Mean Squared Error (RMSE), Correlation Coefficient (*r*), and Nash Sutcliffe efficiency coefficient (NSE) of different machine learning models in Experiments 1, 2, and 3 during the training period. Note that Experiments 2 and 3 have the same training results.

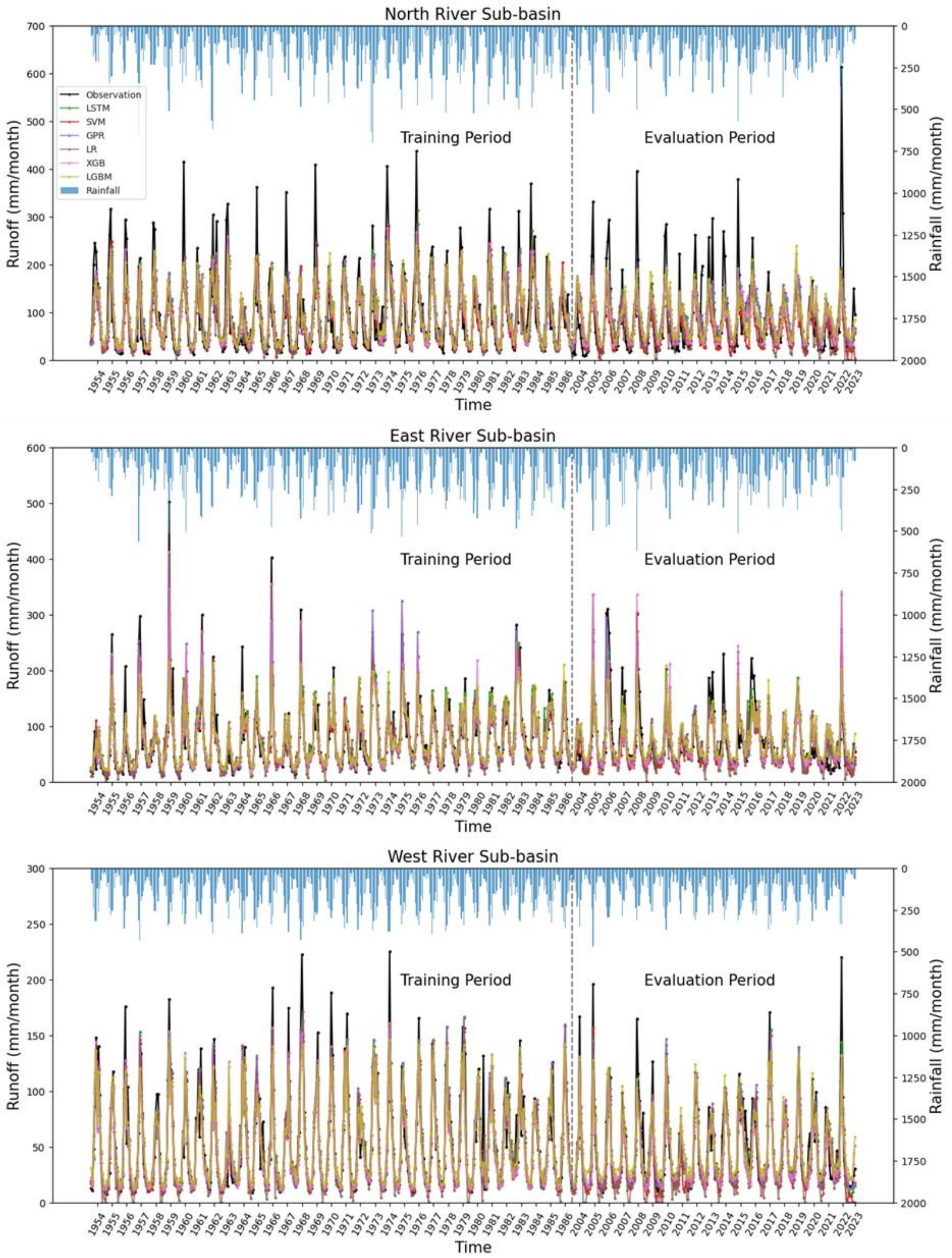

Figure S7. Runoff simulations using multiple machine learning models in Experiment 2 against observations among different sub-basins in the training (1954-1986) and evaluation (2004-2023) periods.

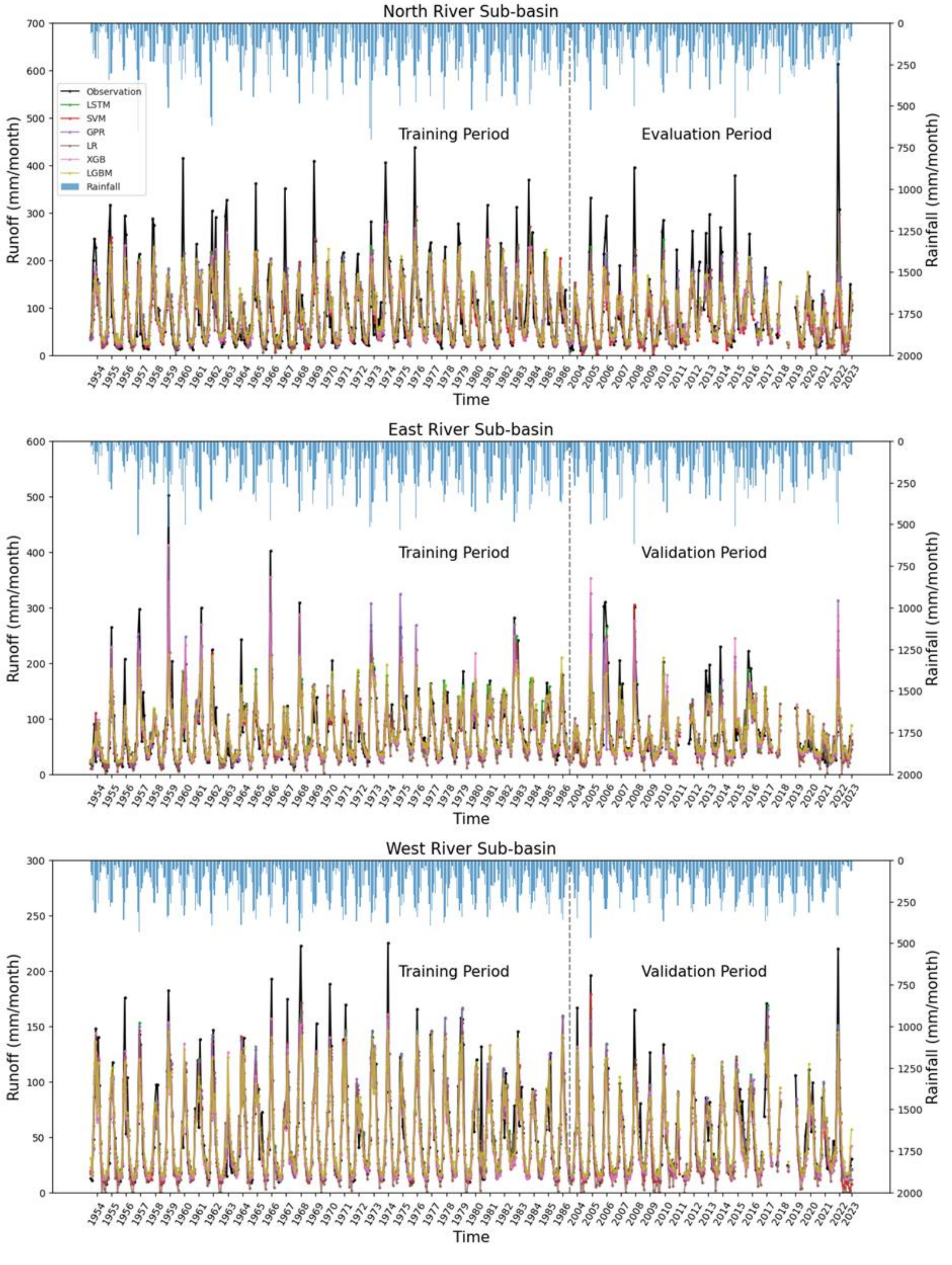

Figure S8. Runoff simulations using multiple machine learning models in Experiment 3 against observations among different sub-basins in the training (1954-1986) and evaluation (2004-2023) periods.

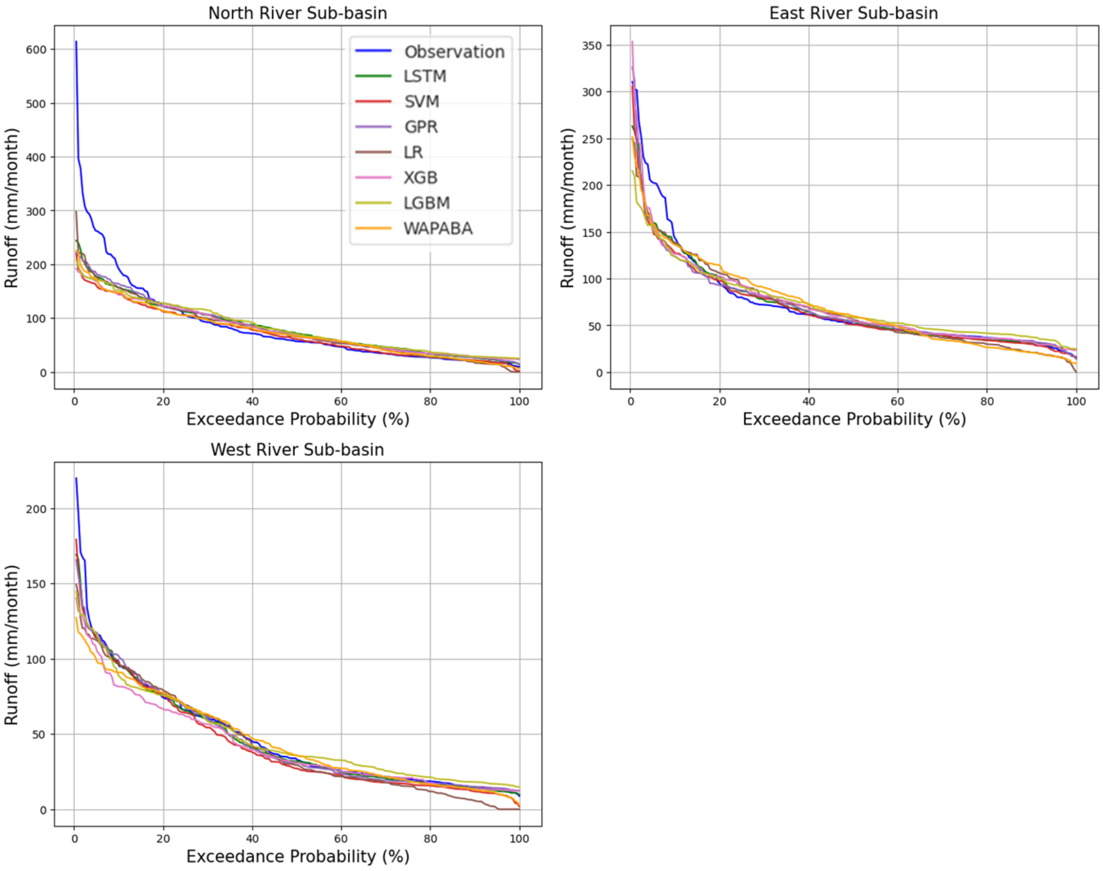

Figure S9. The Flow Duration Curves (FDC) of observations and simulations by all machine learning models and WAPABA in Experiment 3 in the North, East, and West River sub-basins. The x-axis represents the exceedance probability, indicating the probability that a specific runoff amount equals or exceeds a given runoff level shown on the y-axis.

Table S1. Statistical summary of variables used in this study. ‘Mean’ refers to the average value, ‘Std’ refers to the standard deviation. ‘Min’ refers to the minimum value in the data. ‘25%’, ‘50%’, and ‘75%’ refer to the 25th, 50th, and 75th percentiles, indicating that 25%, 50%, and 75% of the data values are lower than this value, respectively. ‘Max’ refers to the maximum value in the data.

	Runoff (mm/month)	**Precipitation (mm/month)**	**Vapor (kPa)**	**Wind (m/s)**	**Radiation (MJ/ (m2 day)**	**Maximum Temperature (℃)**	**Minimum Temperature (℃)**
				North River sub-basin			
Mean	89.05	161.04	1.81	0.66	11.62	22.40	12.22
Std	82.25	125.54	0.73	0.35	3.02	6.00	5.08
Min	8.91	0.67	0.50	0.03	5.22	7.99	0.63
25%	29.05	59.14	1.12	0.36	9.41	17.24	7.92
50%	60.81	133.49	1.76	0.65	11.57	23.54	12.47
75%	117.29	229.16	2.57	0.92	13.95	28.11	16.37
Max	613.85	699.79	2.96	1.77	18.88	30.11	21.85
				East River sub-basin			
Mean	75.46	145.90	1.91	0.81	12.39	23.74	12.92
Std	59.60	120.97	0.75	0.41	2.63	5.17	5.01
Min	6.11	0.21	0.44	0.01	5.43	10.57	-0.92
25%	36.38	46.12	1.21	0.48	10.44	19.46	8.82
50%	55.97	113.20	1.87	0.79	12.36	25.07	13.09
75%	96.26	221.10	2.69	1.12	14.30	28.56	16.93
Max	502.54	620.96	3.06	1.88	18.57	30.23	22.48
				West River sub-basin			
Mean	50.46	150.11	1.71	0.63	11.25	21.72	12.60
Std	41.58	97.62	0.62	0.27	2.70	5.32	4.08
Min	8.02	9.45	0.62	0.19	5.40	8.16	2.41
25%	18.16	67.23	1.13	0.44	8.97	17.17	9.34
50%	34.75	124.56	1.66	0.57	11.50	23.14	13.52

75%	72.49	222.63	2.33	0.75	13.49	26.65	15.85
Max	225.02	468.19	2.68	2.10	17.09	28.75	19.60

Table S2. Evaluation metric results among each machine learning model in Experiment 2 during the training (1954.01 to 1986.12) and evaluation periods (2004.01 to 2023.05) in different river sub-basins. The unit of Bias and RMSE is mm/month, and the other two evaluation metrics (*r* and NSE) are unitless. Note that the training results of Experiment 3 are the same as those in Experiment 2.

	LSTM	**SVM**	**GPR**	**LR**	**XGB**	**LGBM**
North River Sub-Basin						
Training Period						
Bias	-0.13	-7.63	-0.07	0.01	-2.53	0.95
RMSE	54.22	53.79	57.31	56.97	42.97	46.93
r	0.75	0.76	0.72	0.72	0.86	0.83
NSE	0.56	0.57	0.51	0.52	0.73	0.67
Evaluation Period						
Bias	4.78	-16.59	5.47	-10.78	-1.20	4.93
RMSE	56.83	64.88	56.48	58.12	63.80	63.39
r	0.79	0.68	0.79	0.74	0.66	0.67
NSE	0.53	0.39	0.54	0.51	0.41	0.42
East River Sub-Basin						
Training Period						
Bias	-0.01	-3.51	-0.02	0.01	-1.59	0.59
RMSE	28.79	27.95	25.78	32.92	20.72	29.87
r	0.88	0.9	0.91	0.84	0.95	0.88
NSE	0.78	0.79	0.82	0.71	0.89	0.76
Evaluation Period						
Bias	-2.06	-8.58	-0.72	-5.24	0.52	4.17
RMSE	27.88	32.98	35.77	32.49	37.09	34.87
r	0.87	0.82	0.78	0.82	0.76	0.79
NSE	0.75	0.65	0.59	0.66	0.56	0.61
West River Sub-Basin						
Training Period						
Bias	0.17	-1.06	0.00	0.07	-3.97	0.20
RMSE	17.62	17.41	18.81	20.63	16.86	18.11
r	0.92	0.92	0.90	0.88	0.93	0.92
NSE	0.84	0.84	0.82	0.78	0.85	0.83
Evaluation Period						
Bias	-3.68	-7.53	-3.04	-6.81	-4.88	1.93
RMSE	17.4	18.46	17.62	18.57	18.53	17.74
r	0.89	0.89	0.88	0.89	0.89	0.89
NSE	0.78	0.75	0.77	0.75	0.75	0.77

References

1. Wang, Q.J.; Pagano, T.C.; Zhou, S.L.; Hapuarachchi, H.A.P.; Zhang, L.; Robertson, D.E. Monthly versus daily water balance models in simulating monthly runoff. *Journal of Hydrology* **2011**, *404*, 166-175, doi:10.1016/j.jhydrol.2011.04.027.