Supplementary Information for

Extrapolation Validation (EV): A Universal Validation Method for Mitigating Machine Learning Extrapolation Risks

Mengxian Yu[1], Yin-Ning Zhou[2], Qiang Wang[1], Fangyou Yan[1*]

Text S1 Statistical parameters.

Statistical parameters are defined as follows:

Standard deviation (σ): $$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \frac{1}{n}\sum_{i=1}^{n} y)^2} \quad \text{(S1)}$$

Coefficient of determination (R^2): $$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{i,\text{exp}} - y_{i,\text{cal}})^2}{\sum_{i=1}^{n}(y_{i,\text{exp}} - \overline{y}_{\text{exp}})^2} \quad \text{(S2)}$$

Root mean squared error (RMSE): $$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{i,\text{exp}} - y_{i,cal})^2} \quad \text{(S3)}$$

Mean absolute error (MAE): $$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_{i,\text{exp}} - y_{i,cal}\right| \quad \text{(S4)}$$

where, n is the numbers of samples.

Text S2 Development of mathematical relationships models.

To establish ML models for multiple algorithms, the scikit-learn[1] version 1.3.0 package for Python[2] 3.11 was adopted to develop the MLR, LASSO, Ridge, GPR, MLP, AdaBoost, RF, GBDT, SVM, and KNN models, and the xgboost[3] version 1.7.6 package was applied to develop the XGBoost model. The independent variable data were standardized for modelling. The initial 33 ML models for the three datasets were established based on the default hyperparameters of a total of 11 ML algorithms within the scikit-learn[1] and xgboost[3] packages.

The phenomenon of extrapolation-failure was initially identified by analyzing the performance of the initial model training set, test (I) set, test (F) set, and test (B) set by R^2. Subsequently, the hyperparameters of 33 models were optimized (Figs. S1~S3) by exhaustive enumeration to observe the influence of hyperparameters on the performance of different algorithms. For selecting the model with the best extrapolation ability, the model with the smallest average value of $MAE_{training}$, $MAE_{test(F)}$ and $MAE_{test(B)}$ $((MAE_{training} + MAE_{test(F)} + MAE_{test(B)})/3)$ statistical parameters within the tuned hyperparameters was taken as the optimal model (Tables S4, S6 and S8). To be clear, During the development of machine learning models containing the parameter "random_state", this parameter is always set to 1. The R^2 of the optimal models for linear univariate, linear multivariate, and nonlinear multivariate data relationships of 11 ML algorithms further confirm the hypothesis that ML models involving tree algorithms may not have extrapolation ability.

Table S1 Statistical results of the initial hyperparametric ML models for data with linear univariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
Lasso	1.0000	2.9627	0.6703	2.8991	1.0000	1.0000	1.0000	1.0000
Ridge	0.5754	1.7048	0.3857	1.6682	1.0000	1.0000	1.0000	1.0000
SVM	98.4925	500.1807	40.1207	488.1564	0.8805	0.8202	0.9738	0.8397
GPR	0.0001	256.9459	0.0001	6.2930	1.0000	0.3048	1.0000	0.9983
MLP	691.4459	1146.0182	670.2223	200.2035	0.3231	1.0000	0.6214	1.0000
AdaBoost	6.0552	249.8868	4.5307	239.7142	0.9988	0.0000	0.9986	0.0000
XGBoost	0.3888	232.0081	0.4183	220.4561	1.0000	0.0000	1.0000	0.0000
RF	0.3940	232.7778	0.3881	220.9704	1.0000	0.0000	1.0000	0.0000
KNN	0.3651	235.2828	0.0000	223.7365	1.0000	0.0000	1.0000	0.0000
GBDT	1.0646	232.1061	1.1254	220.5793	1.0000	0.0000	0.9999	0.0000

Table S2 Statistical results of the initial hyperparametric ML models for data with linear multivariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
Lasso	1.0004	1.9373	0.6647	5.3968	1.0000	1.0000	1.0000	1.0000
Ridge	0.7173	7.8683	0.5561	158.4471	1.0000	1.0000	1.0000	0.0560
SVM	44.3835	297.7298	12.1532	294.7730	0.9212	0.9175	0.9899	0.5270
GPR	0.0001	465.6339	0.0000	102.1042	1.0000	0.9062	1.0000	0.6258
MLP	394.8063	576.1380	396.1819	336.3715	0.0783	0.9997	0.0910	0.6676
AdaBoost	3.8536	151.9961	2.9115	147.9956	0.9986	0.0000	0.9984	0.0000
XGBoost	0.2246	138.9467	0.2347	137.0528	1.0000	0.0000	1.0000	0.0000
RF	0.1703	139.4360	0.1737	137.3924	1.0000	0.0000	1.0000	0.0000
KNN	0.2211	140.9398	0.0001	139.0768	1.0000	0.0000	1.0000	0.0000
GBDT	0.6509	139.0176	0.6796	137.1404	1.0000	0.0000	0.9999	0.0000

Table S3 Statistical results of the initial hyperparametric ML models for data with nonlinear multivariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0038	0.0000	2.4726	1.0000	1.0000	1.0000	0.9996
Lasso	1.0100	0.7533	0.7541	57.6987	1.0000	1.0000	1.0000	0.9246
Ridge	0.7746	7.3083	0.5963	126.6090	1.0000	1.0000	1.0000	0.0378
SVM	63.2524	362.8745	20.7764	381.9859	0.8992	0.9132	0.9830	0.5085
GPR	0.0001	589.5278	0.0000	143.2726	1.0000	0.9078	1.0000	0.6095
MLP	535.0079	778.6396	534.5355	340.9841	0.0708	0.9998	0.0839	0.6845
AdaBoost	4.9844	181.0609	4.1720	197.7776	0.9985	0.0000	0.9979	0.0000
XGBoost	0.2602	166.3100	0.2871	182.8309	1.0000	0.0000	1.0000	0.0000
RF	0.2070	166.9202	0.2073	183.3232	1.0000	0.0000	1.0000	0.0000
KNN	0.2753	168.7369	0.0004	185.4687	1.0000	0.0000	1.0000	0.0000
GBDT	0.8094	166.4499	0.8436	182.9998	1.0000	0.0000	0.9999	0.0000

Table S4 Optimal hyperparameters of the optimal ML models for data with linear univariate functional relationships.

Model	Parameters
Lasso	max_iter = 100
Ridge	alpha = 0.1, max_iter = 100
SVM	C = 51, gamma = 0.001, kernel = 'linear'
GPR	kernel=2**2 * DotProduct(sigma_0=4)
MLP	activation = 'identity', hidden_layer_sizes = (210, 10), solver = 'lbfgs'
AdaBoost	learning_rate = 0.11, n_estimators = 200
XGBoost	learning_rate = 0.19, max_depth = 7, n_estimators = 600
RF	max_depth = 9, n_estimators = 10
KNN	n_neighbors = 1
GBDT	learning_rate = 0.05, max_depth =5, n_estimator = 600

Table S5 Statistical results of the optimal ML models for data with linear univariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
Lasso	0.1000	0.2963	0.0670	0.2899	1.0000	1.0000	1.0000	1.0000
Ridge	0.0577	0.1710	0.0387	0.1673	1.0000	1.0000	1.0000	1.0000
SVM	0.0579	0.1716	0.0388	0.1679	1.0000	1.0000	1.0000	1.0000
GPR	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
MLP	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
AdaBoost	10.1147	259.6758	7.4628	249.4922	0.9966	0.0000	0.9959	0.0000
XGBoost	0.0035	231.8071	0.0039	220.2600	1.0000	0.0000	1.0000	0.0000
RF	0.5990	232.6736	0.6088	220.4324	1.0000	0.0000	1.0000	0.0000
KNN	0.0000	231.8060	0.0000	220.2589	1.0000	0.0000	1.0000	0.0000
GBDT	0.0000	231.8060	0.0000	220.2589	1.0000	0.0000	1.0000	0.0000

Table S6 Optimal hyperparameters of the optimal ML models for data with linear multivariate functional relationships.

Model	Parameters
Lasso	alpha = 0.3, max_iter = 100, tol = 0.0002
Ridge	alpha = 1.9, max_iter = 100
SVM	C = 191, gamma = 0.001, kernel = 'linear'
GPR	kernel=2**2 * DotProduct(sigma_0=3)
MLP	activation = 'identity', alpha = 0.0005, hidden_layer_sizes = (110, 10), solver = 'lbfgs'
AdaBoost	learning_rate = 0.13, n_estimators = 200
XGBoost	learning_rate = 0.13, max_depth = 7, n_estimators = 900
RF	max_depth = 9, max_features = 4, n_estimators = 10
KNN	n_neighbors = 1
GBDT	learning_rate = 0.05, max_depth =5, n_estimator = 500

Table S7 Statistical results of the optimal ML models for data with linear multivariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	1.0000	1.0000
Lasso	0.3048	0.2207	0.2088	3.9420	1.0000	1.0000	1.0000	0.9978
Ridge	1.2107	10.8737	0.9283	113.7655	0.9999	1.0000	0.9999	0.0008
SVM	0.0700	0.9778	0.0758	37.3265	1.0000	1.0000	1.0000	0.7957
GPR	0.0000	0.0000	0.0000	0.0001	1.0000	1.0000	1.0000	1.0000
MLP	0.0006	0.0244	0.0005	5.2772	1.0000	1.0000	1.0000	0.9967
AdaBoost	5.7320	156.6871	4.0441	153.0914	0.9970	0.0000	0.9968	0.0000
XGBoost	0.0036	138.8537	0.0040	136.9603	1.0000	0.0000	1.0000	0.0000
RF	0.3001	139.3735	0.2986	137.0648	1.0000	0.0000	1.0000	0.0000
KNN	0.0000	138.8526	0.0000	136.9591	1.0000	0.0000	1.0000	0.0000
GBDT	0.0000	138.8526	0.0000	136.9591	1.0000	0.0000	1.0000	0.0000

Table S8 Optimal hyperparameters of the optimal ML models for data with nonlinear multivariate functional relationships.

Model	Parameters
Lasso	alpha = 1.7, max_iter = 600
Ridge	alpha = 1.9, max_iter = 100
SVM	C = 101, gamma = 0.011, kernel = 'sigmoid'
GPR	kernel=1**2 * DotProduct(sigma_0=2)
MLP	activation = 'identity', alpha = 0.0007, hidden_layer_sizes = (210, 310), solver = 'lbfgs'
AdaBoost	learning_rate = 0.15, n_estimators = 200
XGBoost	learning_rate = 0.17, max_depth = 7, n_estimators = 700
RF	max_depth = 9, max_features = 5, n_estimators = 10
KNN	n_neighbors = 1
GBDT	learning_rate = 0.19, max_depth =7, n_estimator = 200

Table S9 Statistical results of the optimal ML models for data with nonlinear multivariate functional relationships.

Model	$RMSE_{training}$	$RMSE_{test(F)}$	$RMSE_{test(I)}$	$RMSE_{test(B)}$	$R^2_{training}$	$R^2_{test(F)}$	$R^2_{test(I)}$	$R^2_{test(B)}$
MLR	0.0000	0.0038	0.0000	2.4726	1.0000	1.0000	1.0000	0.9996
Lasso	1.7006	3.0380	1.1622	4.7048	1.0000	1.0000	1.0000	0.9990
Ridge	1.3402	10.6818	1.0273	76.6308	0.9999	1.0000	1.0000	0.4168
SVM	0.5614	8.3121	0.2512	24.8271	1.0000	1.0000	1.0000	0.9576
GPR	0.0000	0.0038	0.0000	2.4724	1.0000	1.0000	1.0000	0.9996
MLP	0.0038	0.0901	0.0034	0.3379	1.0000	1.0000	1.0000	1.0000
AdaBoost	7.1512	187.5066	5.1517	204.2875	0.9970	0.0000	0.9966	0.0000
XGBoost	0.0033	166.2165	0.0035	182.7721	1.0000	0.0000	1.0000	0.0000
RF	0.3807	166.8446	0.3883	182.9056	1.0000	0.0000	1.0000	0.0000
KNN	0.0000	166.2155	0.0000	182.7710	1.0000	0.0000	1.0000	0.0000
GBDT	0.0000	166.2155	0.0000	182.7710	1.0000	0.0000	1.0000	0.0000

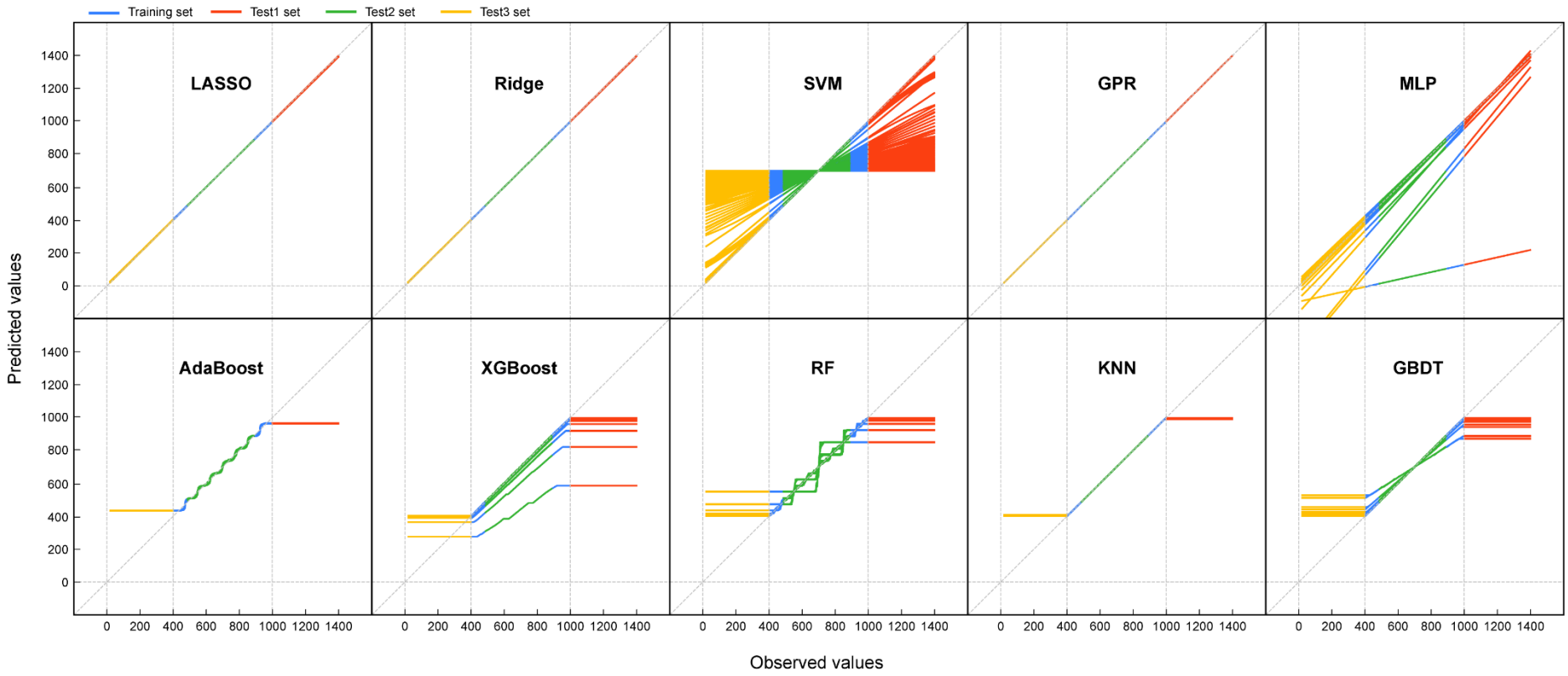


Fig. S1 Hyperparametric adjustment process for linear univariate ML models.

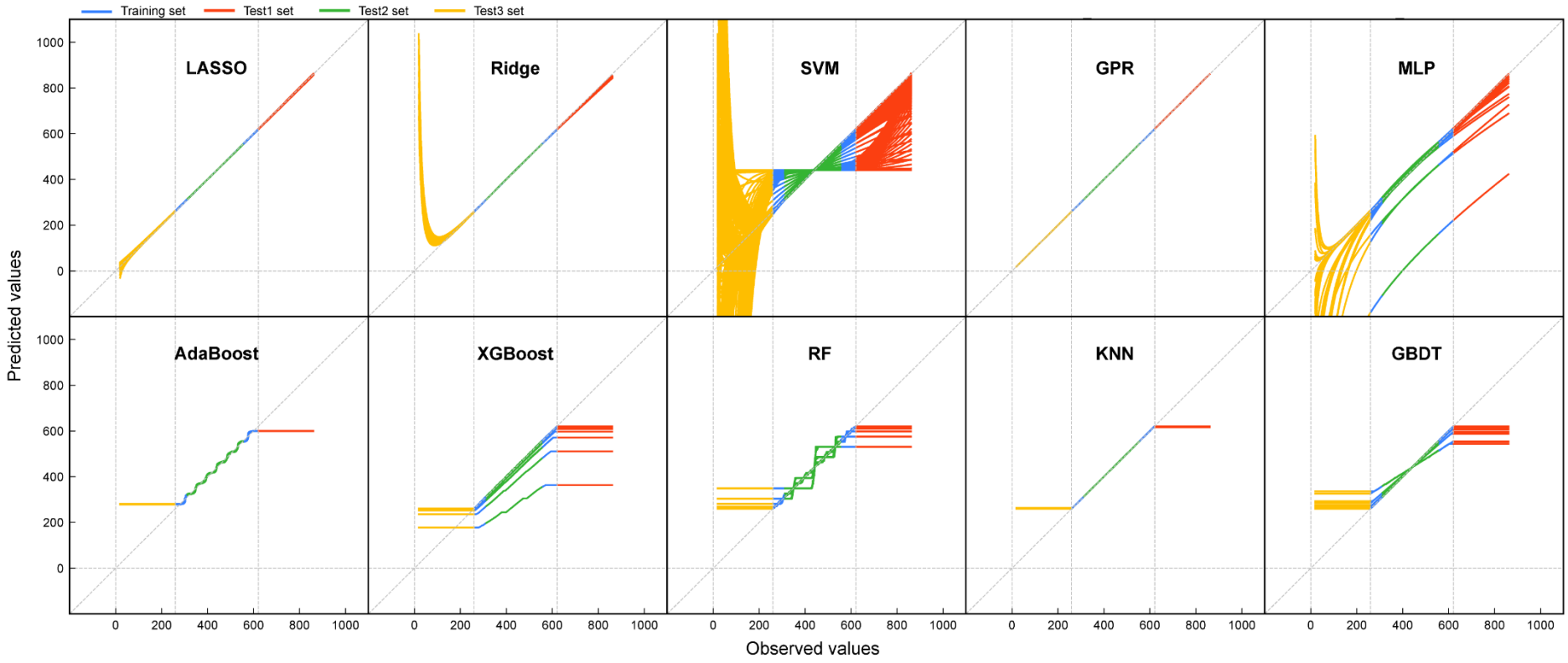


Fig. S2 Hyperparametric adjustment process for linear multivariate ML models.

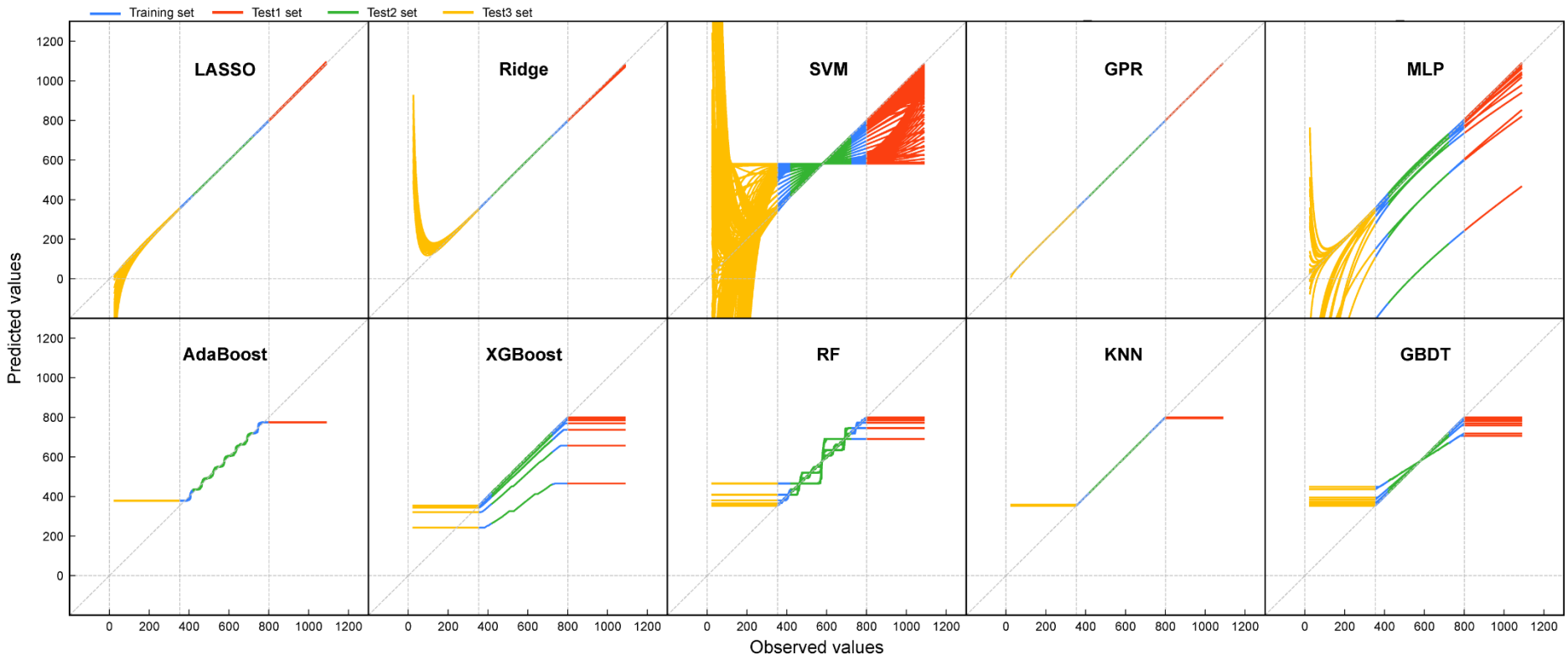
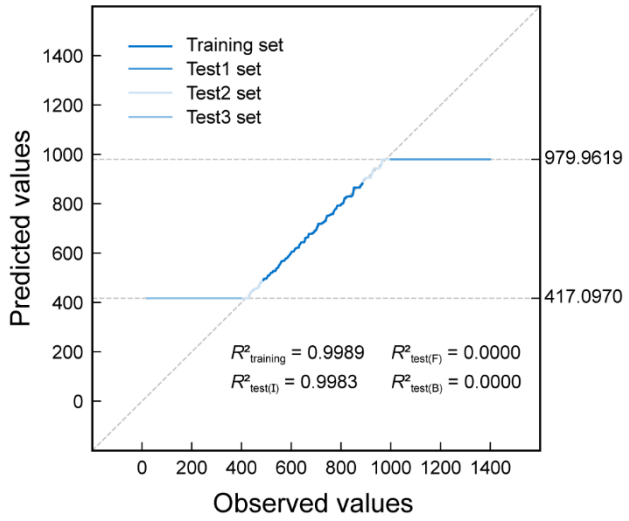


Fig. S3 Hyperparametric adjustment process for nonlinear multivariate ML models.

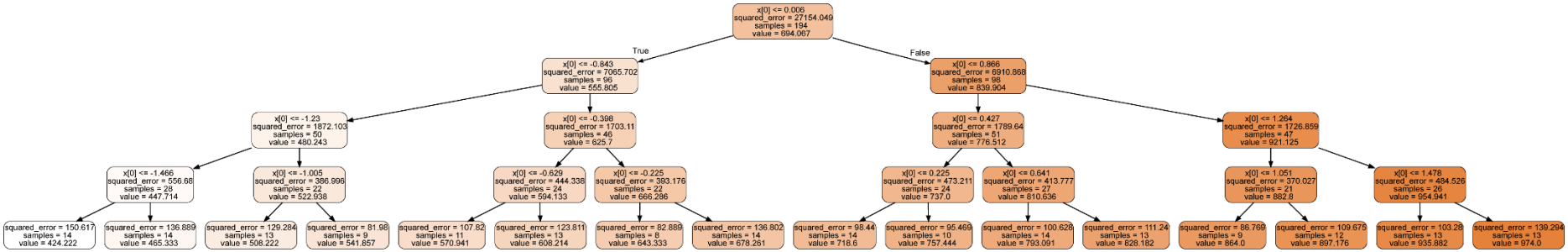


82

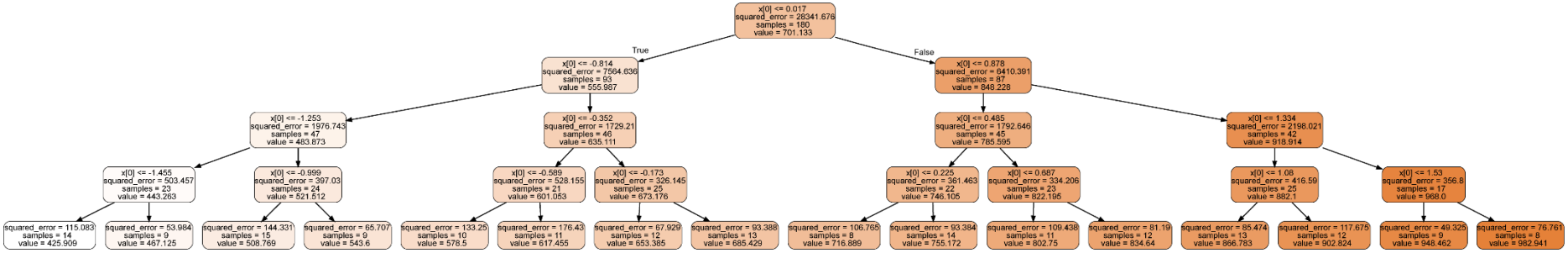
83 **Fig. S4** Example Random Forest (RF) model predicted values *vs*. observed values. Note: The RF model with linear univariate relationships dataset which has 10

84 DTs with a depth of 4 each (n_estimators = 10, max_depth = 4).

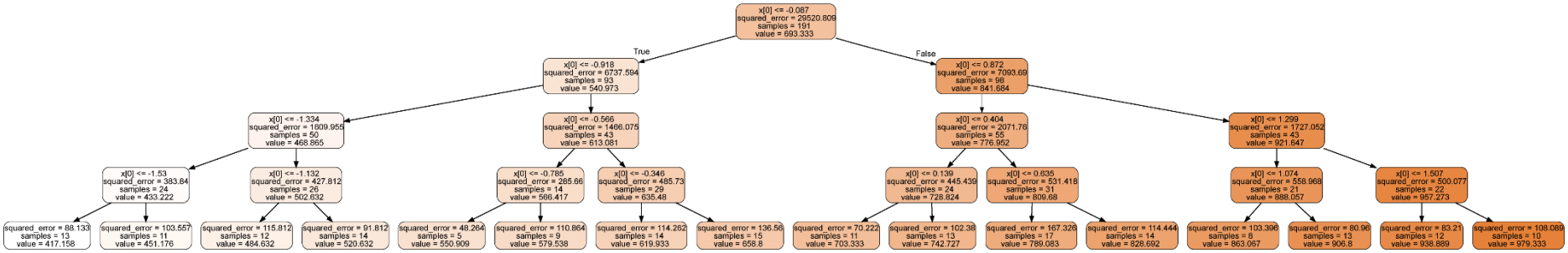
85


x[0] <= 0.006
squared_error = 27154.049
samples = 194
value = 694.067
True
False
x[0] <= -0.843
squared_error = 7065.702
samples = 96
value = 555.805
x[0] <= 0.866
squared_error = 6910.868
samples = 98
value = 839.904
x[0] <= -1.23
squared_error = 1872.103
samples = 50
value = 480.243
x[0] <= -0.398
squared_error = 1703.11
samples = 46
value = 625.7
x[0] <= 0.427
squared_error = 1789.64
samples = 51
value = 776.512
x[0] <= 1.264
squared_error = 1726.859
samples = 47
value = 921.125
x[0] <= -1.466
squared_error = 556.68
samples = 28
value = 447.714
x[0] <= -1.005
squared_error = 386.996
samples = 22
value = 522.938
x[0] <= -0.629
squared_error = 444.336
samples = 24
value = 594.133
x[0] <= -0.225
squared_error = 393.176
samples = 22
value = 666.286
x[0] <= 0.225
squared_error = 473.211
samples = 24
value = 737.0
x[0] <= 0.641
squared_error = 413.777
samples = 27
value = 810.636
x[0] <= 1.051
squared_error = 370.027
samples = 21
value = 882.8
x[0] <= 1.478
squared_error = 484.526
samples = 26
value = 954.941
squared_error = 150.617
samples = 14
value = 424.222
squared_error = 136.889
samples = 14
value = 465.333
squared_error = 129.284
samples = 13
value = 508.222
squared_error = 81.98
samples = 9
value = 541.857
squared_error = 107.82
samples = 11
value = 570.941
squared_error = 123.811
samples = 13
value = 608.214
squared_error = 82.889
samples = 8
value = 643.333
squared_error = 136.802
samples = 14
value = 678.261
squared_error = 98.44
samples = 14
value = 718.6
squared_error = 95.469
samples = 10
value = 757.444
squared_error = 100.628
samples = 14
value = 793.091
squared_error = 111.24
samples = 13
value = 828.182
squared_error = 86.769
samples = 9
value = 864.0
squared_error = 109.679
samples = 12
value = 897.176
squared_error = 103.28
samples = 13
value = 935.882
squared_error = 139.294
samples = 13
value = 974.0

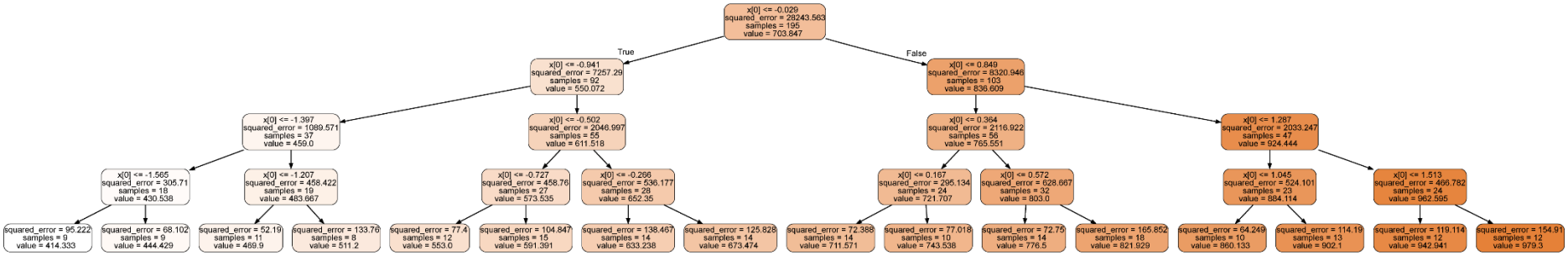
86


x[0] <= 0.017
squared_error = 28341.676
samples = 180
value = 701.133
True
False
x[0] <= -0.814
squared_error = 7564.636
samples = 93
value = 555.987
x[0] <= 0.878
squared_error = 6410.391
samples = 87
value = 848.228
x[0] <= -1.253
squared_error = 1976.743
samples = 47
value = 483.873
x[0] <= -0.352
squared_error = 1729.21
samples = 46
value = 635.111
x[0] <= 0.485
squared_error = 1792.646
samples = 45
value = 785.595
x[0] <= 1.334
squared_error = 2198.021
samples = 42
value = 918.914
x[0] <= -1.455
squared_error = 503.457
samples = 23
value = 443.263
x[0] <= -0.999
squared_error = 397.03
samples = 24
value = 521.512
x[0] <= -0.589
squared_error = 528.155
samples = 21
value = 601.053
x[0] <= -0.173
squared_error = 326.145
samples = 25
value = 673.176
x[0] <= 0.225
squared_error = 361.463
samples = 22
value = 746.105
x[0] <= 0.687
squared_error = 334.206
samples = 23
value = 822.195
x[0] <= 1.08
squared_error = 416.59
samples = 25
value = 882.1
x[0] <= 1.53
squared_error = 356.8
samples = 17
value = 968.0
squared_error = 115.083
samples = 14
value = 425.909
squared_error = 53.984
samples = 9
value = 467.125
squared_error = 144.331
samples = 15
value = 508.769
squared_error = 65.707
samples = 9
value = 543.6
squared_error = 133.25
samples = 10
value = 578.5
squared_error = 176.43
samples = 11
value = 617.455
squared_error = 67.929
samples = 12
value = 653.385
squared_error = 93.388
samples = 13
value = 685.429
squared_error = 106.765
samples = 8
value = 716.889
squared_error = 93.384
samples = 14
value = 755.172
squared_error = 109.438
samples = 11
value = 802.75
squared_error = 81.19
samples = 12
value = 834.64
squared_error = 85.474
samples = 13
value = 866.783
squared_error = 117.675
samples = 12
value = 902.824
squared_error = 49.325
samples = 9
value = 948.462
squared_error = 76.761
samples = 8
value = 982.941

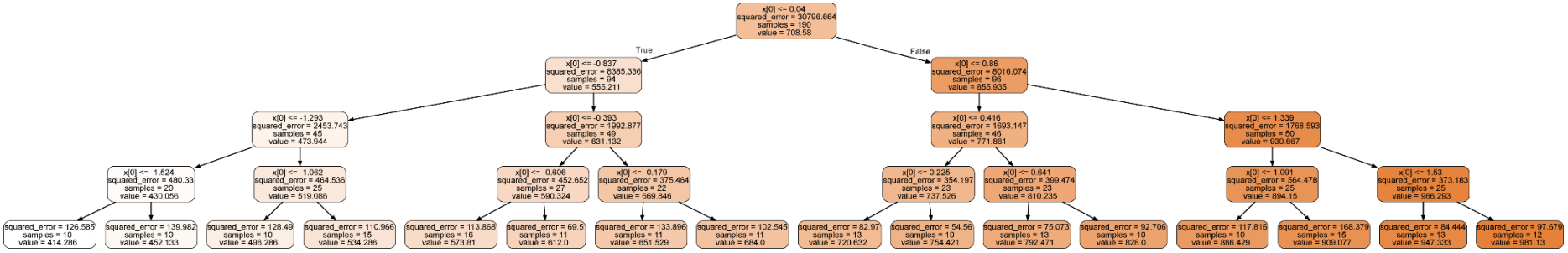
87


x[0] <= -0.087
squared_error = 29520.806
samples = 191
value = 693.333
True
False
x[0] <= -0.918
squared_error = 6737.594
samples = 93
value = 540.973
x[0] <= 0.872
squared_error = 7093.69
samples = 98
value = 841.684
x[0] <= -1.334
squared_error = 1609.955
samples = 50
value = 468.865
x[0] <= -0.566
squared_error = 1466.075
samples = 43
value = 613.081
x[0] <= 0.404
squared_error = 2071.76
samples = 55
value = 776.952
x[0] <= 1.299
squared_error = 1727.052
samples = 43
value = 921.647
x[0] <= -1.53
squared_error = 383.84
samples = 24
value = 433.222
x[0] <= -1.132
squared_error = 427.812
samples = 26
value = 502.632
x[0] <= -0.785
squared_error = 285.66
samples = 14
value = 566.417
x[0] <= -0.346
squared_error = 485.73
samples = 29
value = 635.48
x[0] <= 0.139
squared_error = 445.439
samples = 24
value = 728.824
x[0] <= 0.635
squared_error = 531.418
samples = 31
value = 809.68
x[0] <= 1.074
squared_error = 558.968
samples = 21
value = 888.057
x[0] <= 1.507
squared_error = 500.077
samples = 22
value = 957.273
squared_error = 88.133
samples = 13
value = 417.158
squared_error = 103.557
samples = 11
value = 451.176
squared_error = 115.812
samples = 12
value = 484.632
squared_error = 91.812
samples = 14
value = 520.632
squared_error = 48.264
samples = 5
value = 550.909
squared_error = 110.864
samples = 9
value = 579.538
squared_error = 114.262
samples = 14
value = 619.933
squared_error = 136.56
samples = 15
value = 658.8
squared_error = 70.222
samples = 11
value = 703.333
squared_error = 102.36
samples = 13
value = 742.727
squared_error = 167.326
samples = 17
value = 789.083
squared_error = 114.444
samples = 14
value = 828.692
squared_error = 103.396
samples = 8
value = 863.067
squared_error = 80.96
samples = 13
value = 906.8
squared_error = 83.21
samples = 12
value = 938.889
squared_error = 108.089
samples = 10
value = 979.333

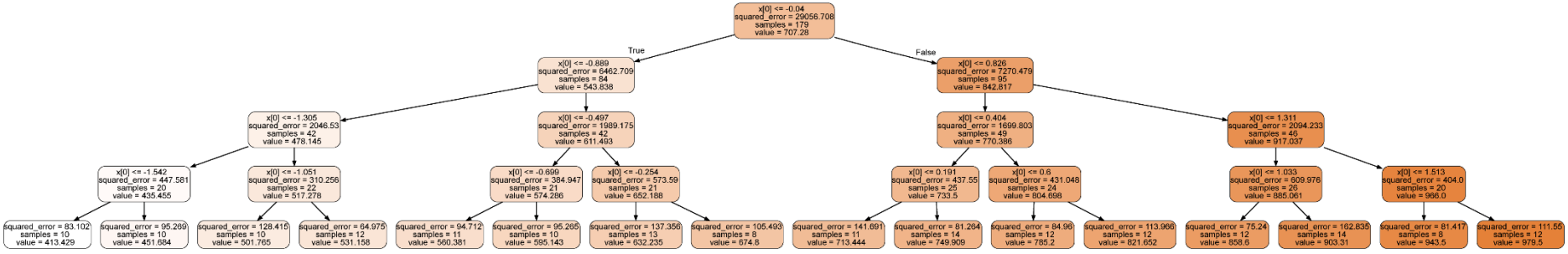
88


x[0] <= -0.029
squared_error = 28243.563
samples = 195
value = 703.847
True
False
x[0] <= -0.941
squared_error = 7257.29
samples = 92
value = 550.072
x[0] <= 0.849
squared_error = 8320.946
samples = 103
value = 836.609
x[0] <= -1.397
squared_error = 1089.571
samples = 37
value = 459.0
x[0] <= -0.502
squared_error = 2046.997
samples = 55
value = 611.518
x[0] <= 0.364
squared_error = 2116.922
samples = 56
value = 765.551
x[0] <= 1.287
squared_error = 2033.247
samples = 47
value = 924.444
x[0] <= -1.565
squared_error = 305.71
samples = 18
value = 430.538
x[0] <= -1.207
squared_error = 458.422
samples = 19
value = 483.667
x[0] <= -0.727
squared_error = 458.76
samples = 27
value = 573.535
x[0] <= -0.266
squared_error = 536.177
samples = 28
value = 652.35
x[0] <= 0.167
squared_error = 295.134
samples = 24
value = 721.707
x[0] <= 0.572
squared_error = 628.667
samples = 32
value = 803.0
x[0] <= 1.045
squared_error = 524.101
samples = 23
value = 884.114
x[0] <= 1.513
squared_error = 466.782
samples = 24
value = 962.595
squared_error = 95.222
samples = 9
value = 414.333
squared_error = 68.102
samples = 9
value = 444.429
squared_error = 52.19
samples = 11
value = 469.9
squared_error = 133.76
samples = 8
value = 511.2
squared_error = 77.4
samples = 12
value = 553.0
squared_error = 104.847
samples = 15
value = 591.391
squared_error = 138.467
samples = 14
value = 633.238
squared_error = 125.828
samples = 14
value = 673.474
squared_error = 72.388
samples = 14
value = 711.571
squared_error = 77.018
samples = 10
value = 743.538
squared_error = 72.75
samples = 14
value = 776.5
squared_error = 165.852
samples = 18
value = 821.929
squared_error = 64.249
samples = 10
value = 860.133
squared_error = 114.19
samples = 13
value = 902.1
squared_error = 119.114
samples = 12
value = 942.941
squared_error = 154.91
samples = 12
value = 979.3

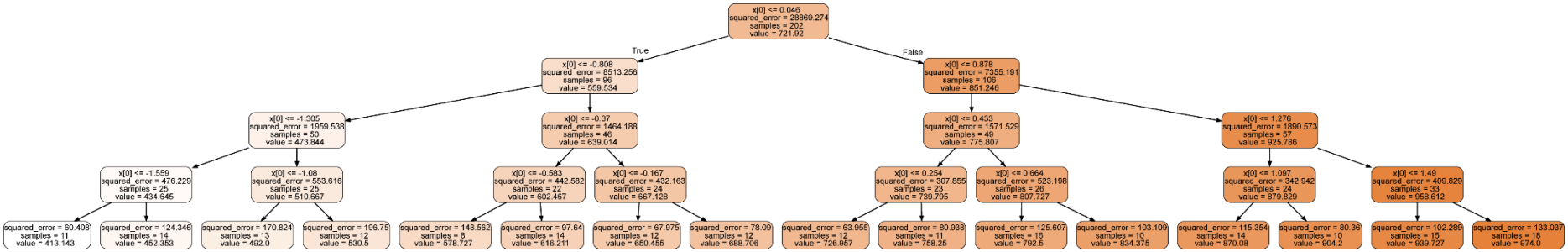
89


x[0] <= 0.04
squared_error = 30796.664
samples = 190
value = 708.58
True
False
x[0] <= -0.837
squared_error = 8385.336
samples = 94
value = 555.211
x[0] <= 0.86
squared_error = 8016.074
samples = 96
value = 855.935
x[0] <= -1.293
squared_error = 2453.743
samples = 45
value = 473.944
x[0] <= -0.393
squared_error = 1992.877
samples = 49
value = 631.132
x[0] <= 0.416
squared_error = 1693.147
samples = 46
value = 771.861
x[0] <= 1.339
squared_error = 1768.593
samples = 50
value = 930.667
x[0] <= -1.524
squared_error = 480.33
samples = 20
value = 430.056
x[0] <= -1.062
squared_error = 464.536
samples = 25
value = 519.086
x[0] <= -0.606
squared_error = 452.652
samples = 27
value = 590.324
x[0] <= -0.179
squared_error = 375.464
samples = 22
value = 669.846
x[0] <= 0.225
squared_error = 354.197
samples = 23
value = 737.526
x[0] <= 0.641
squared_error = 399.474
samples = 23
value = 810.235
x[0] <= 1.091
squared_error = 564.478
samples = 25
value = 894.15
x[0] <= 1.53
squared_error = 373.183
samples = 25
value = 966.293
squared_error = 126.585
samples = 10
value = 414.286
squared_error = 139.982
samples = 10
value = 452.133
squared_error = 128.49
samples = 10
value = 496.286
squared_error = 110.966
samples = 15
value = 534.286
squared_error = 113.868
samples = 16
value = 573.81
squared_error = 69.5
samples = 11
value = 612.0
squared_error = 133.896
samples = 11
value = 651.529
squared_error = 102.545
samples = 11
value = 684.0
squared_error = 82.97
samples = 13
value = 720.632
squared_error = 54.56
samples = 10
value = 754.421
squared_error = 75.073
samples = 13
value = 792.471
squared_error = 92.706
samples = 10
value = 828.0
squared_error = 117.816
samples = 10
value = 866.429
squared_error = 168.379
samples = 15
value = 909.077
squared_error = 84.444
samples = 13
value = 947.333
squared_error = 97.679
samples = 12
value = 981.13

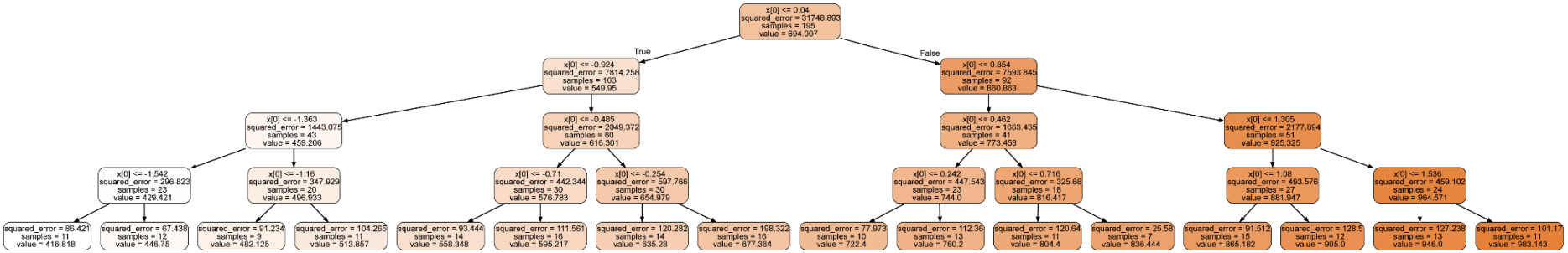
90


x[0] <= -0.04
squared_error = 29056.708
samples = 179
value = 707.28
True
False
x[0] <= -0.889
squared_error = 6462.709
samples = 84
value = 543.838
x[0] <= 0.826
squared_error = 7270.479
samples = 95
value = 842.817
x[0] <= -1.305
squared_error = 2046.53
samples = 42
value = 478.145
x[0] <= -0.497
squared_error = 1989.175
samples = 42
value = 611.493
x[0] <= 0.404
squared_error = 1699.803
samples = 49
value = 770.386
x[0] <= 1.311
squared_error = 2094.233
samples = 46
value = 917.037
x[0] <= -1.542
squared_error = 447.581
samples = 20
value = 435.455
x[0] <= -1.051
squared_error = 310.256
samples = 22
value = 517.278
x[0] <= -0.699
squared_error = 384.947
samples = 21
value = 574.286
x[0] <= -0.254
squared_error = 573.59
samples = 21
value = 652.188
x[0] <= 0.191
squared_error = 437.55
samples = 25
value = 733.5
x[0] <= 0.6
squared_error = 431.048
samples = 24
value = 804.698
x[0] <= 1.033
squared_error = 609.976
samples = 26
value = 885.061
x[0] <= 1.513
squared_error = 404.0
samples = 20
value = 966.0
squared_error = 83.102
samples = 10
value = 413.429
squared_error = 95.289
samples = 10
value = 451.684
squared_error = 128.415
samples = 10
value = 501.765
squared_error = 64.975
samples = 12
value = 531.158
squared_error = 94.712
samples = 11
value = 560.381
squared_error = 95.265
samples = 10
value = 595.143
squared_error = 137.356
samples = 13
value = 632.235
squared_error = 105.493
samples = 8
value = 674.8
squared_error = 141.691
samples = 11
value = 713.444
squared_error = 81.264
samples = 14
value = 749.909
squared_error = 84.96
samples = 12
value = 785.2
squared_error = 113.966
samples = 12
value = 821.652
squared_error = 75.24
samples = 12
value = 858.6
squared_error = 162.835
samples = 14
value = 903.31
squared_error = 81.417
samples = 8
value = 943.5
squared_error = 111.55
samples = 12
value = 979.5

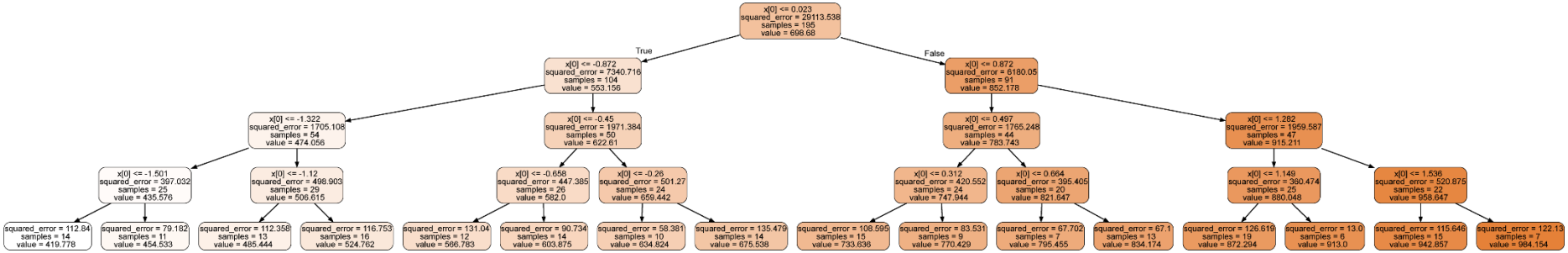
91


x[0] <= 0.046
squared_error = 28869.274
samples = 202
value = 721.92
True
False
x[0] <= -0.808
squared_error = 8513.256
samples = 96
value = 559.534
x[0] <= 0.878
squared_error = 7355.191
samples = 106
value = 851.246
x[0] <= -1.305
squared_error = 1959.538
samples = 50
value = 473.844
x[0] <= -0.37
squared_error = 1464.188
samples = 46
value = 639.014
x[0] <= 0.433
squared_error = 1571.529
samples = 49
value = 775.807
x[0] <= 1.276
squared_error = 1890.573
samples = 57
value = 925.786
x[0] <= -1.559
squared_error = 476.229
samples = 25
value = 434.645
x[0] <= -1.08
squared_error = 553.616
samples = 25
value = 510.667
x[0] <= -0.583
squared_error = 442.582
samples = 22
value = 602.467
x[0] <= -0.167
squared_error = 432.163
samples = 24
value = 667.128
x[0] <= 0.254
squared_error = 307.855
samples = 23
value = 739.795
x[0] <= 0.664
squared_error = 523.198
samples = 26
value = 807.727
x[0] <= 1.097
squared_error = 342.942
samples = 24
value = 879.829
x[0] <= 1.49
squared_error = 409.829
samples = 33
value = 958.612
squared_error = 60.408
samples = 11
value = 413.143
squared_error = 124.346
samples = 14
value = 452.353
squared_error = 170.824
samples = 13
value = 492.0
squared_error = 196.75
samples = 12
value = 530.5
squared_error = 148.562
samples = 8
value = 578.727
squared_error = 97.64
samples = 14
value = 616.211
squared_error = 67.975
samples = 12
value = 650.455
squared_error = 78.09
samples = 12
value = 688.706
squared_error = 63.955
samples = 12
value = 726.957
squared_error = 80.938
samples = 11
value = 758.25
squared_error = 125.607
samples = 16
value = 792.5
squared_error = 103.109
samples = 10
value = 834.375
squared_error = 115.354
samples = 14
value = 870.08
squared_error = 80.36
samples = 10
value = 904.2
squared_error = 102.289
samples = 15
value = 939.727
squared_error = 133.037
samples = 18
value = 974.0

92


x[0] <= 0.04
squared_error = 31748.893
samples = 195
value = 694.007
True
False
x[0] <= -0.924
squared_error = 7814.258
samples = 103
value = 549.95
x[0] <= 0.854
squared_error = 7593.845
samples = 92
value = 860.863
x[0] <= -1.363
squared_error = 1443.075
samples = 43
value = 459.206
x[0] <= -0.485
squared_error = 2049.372
samples = 60
value = 616.301
x[0] <= 0.462
squared_error = 1663.435
samples = 41
value = 773.458
x[0] <= 1.305
squared_error = 2177.894
samples = 51
value = 925.325
x[0] <= -1.542
squared_error = 296.823
samples = 23
value = 429.421
x[0] <= -1.16
squared_error = 347.929
samples = 20
value = 496.933
x[0] <= -0.71
squared_error = 442.344
samples = 30
value = 576.783
x[0] <= -0.254
squared_error = 597.766
samples = 30
value = 654.979
x[0] <= 0.242
squared_error = 447.543
samples = 23
value = 744.0
x[0] <= 0.716
squared_error = 325.66
samples = 18
value = 816.417
x[0] <= 1.08
squared_error = 493.576
samples = 27
value = 881.947
x[0] <= 1.536
squared_error = 459.102
samples = 24
value = 964.571
squared_error = 86.421
samples = 11
value = 416.818
squared_error = 67.438
samples = 12
value = 446.75
squared_error = 91.234
samples = 9
value = 482.125
squared_error = 104.265
samples = 11
value = 513.857
squared_error = 93.444
samples = 14
value = 558.348
squared_error = 111.561
samples = 16
value = 595.217
squared_error = 120.282
samples = 14
value = 635.28
squared_error = 198.322
samples = 16
value = 677.364
squared_error = 77.973
samples = 10
value = 722.4
squared_error = 112.36
samples = 13
value = 760.2
squared_error = 120.64
samples = 11
value = 804.4
squared_error = 25.58
samples = 7
value = 836.444
squared_error = 91.512
samples = 15
value = 865.182
squared_error = 128.5
samples = 12
value = 905.0
squared_error = 127.238
samples = 13
value = 946.0
squared_error = 101.17
samples = 11
value = 983.143

93


x[0] <= 0.023
squared_error = 29113.538
samples = 195
value = 698.68
True
False
x[0] <= -0.872
squared_error = 7340.716
samples = 104
value = 553.156
x[0] <= 0.872
squared_error = 6180.05
samples = 91
value = 852.178
x[0] <= -1.322
squared_error = 1705.108
samples = 54
value = 474.056
x[0] <= -0.45
squared_error = 1971.384
samples = 50
value = 622.61
x[0] <= 0.497
squared_error = 1765.248
samples = 44
value = 783.743
x[0] <= 1.282
squared_error = 1959.587
samples = 47
value = 915.211
x[0] <= -1.501
squared_error = 397.032
samples = 25
value = 435.576
x[0] <= -1.12
squared_error = 498.903
samples = 29
value = 506.615
x[0] <= -0.658
squared_error = 447.385
samples = 26
value = 582.0
x[0] <= -0.26
squared_error = 501.27
samples = 24
value = 659.442
x[0] <= 0.312
squared_error = 420.552
samples = 24
value = 747.944
x[0] <= 0.664
squared_error = 395.405
samples = 20
value = 821.647
x[0] <= 1.149
squared_error = 360.474
samples = 25
value = 880.048
x[0] <= 1.536
squared_error = 520.875
samples = 22
value = 958.647
squared_error = 112.84
samples = 14
value = 419.778
squared_error = 79.182
samples = 11
value = 454.533
squared_error = 112.358
samples = 13
value = 485.444
squared_error = 116.753
samples = 16
value = 524.762
squared_error = 131.04
samples = 12
value = 566.783
squared_error = 90.734
samples = 14
value = 603.875
squared_error = 58.381
samples = 10
value = 634.824
squared_error = 135.479
samples = 14
value = 675.538
squared_error = 108.595
samples = 15
value = 733.636
squared_error = 83.531
samples = 9
value = 770.429
squared_error = 67.702
samples = 7
value = 795.455
squared_error = 67.1
samples = 13
value = 834.174
squared_error = 126.619
samples = 19
value = 872.294
squared_error = 13.0
samples = 6
value = 913.0
squared_error = 115.646
samples = 15
value = 942.857
squared_error = 122.13
samples = 7
value = 984.154

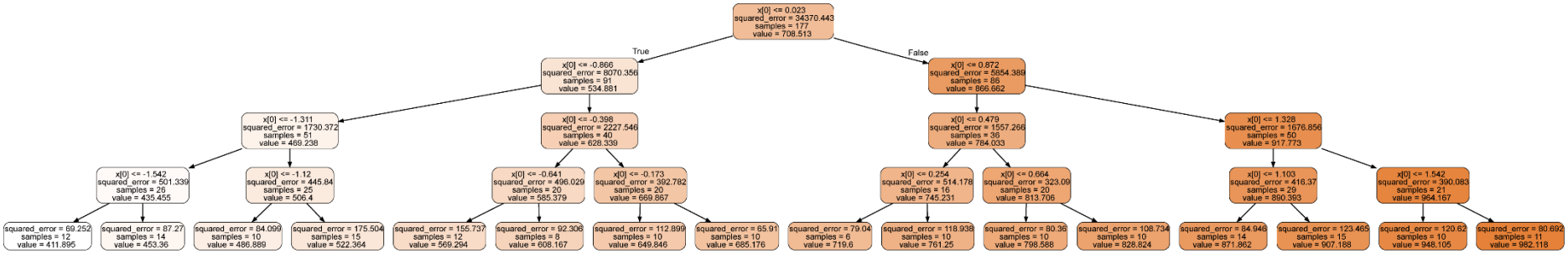


Fig. S5 10 decision trees (DTs) of example random forest (RF) model visualization.

Text S3 Extrapolation validation of mathematical relationships models

Applying the EV method to evaluate the extrapolation ability of the three functional relations optimal model established by 11 ML algorithms. Serialized extrapolation was performed on the *h* and independent variables of the training set, and the training (EV) and test (EV) sets were re-divided in the ratio of 8:2, the ED was calculated and the extrapolation ability of the model was evaluated. (Figs. 4a, b and c, Fig. S6 and S7).

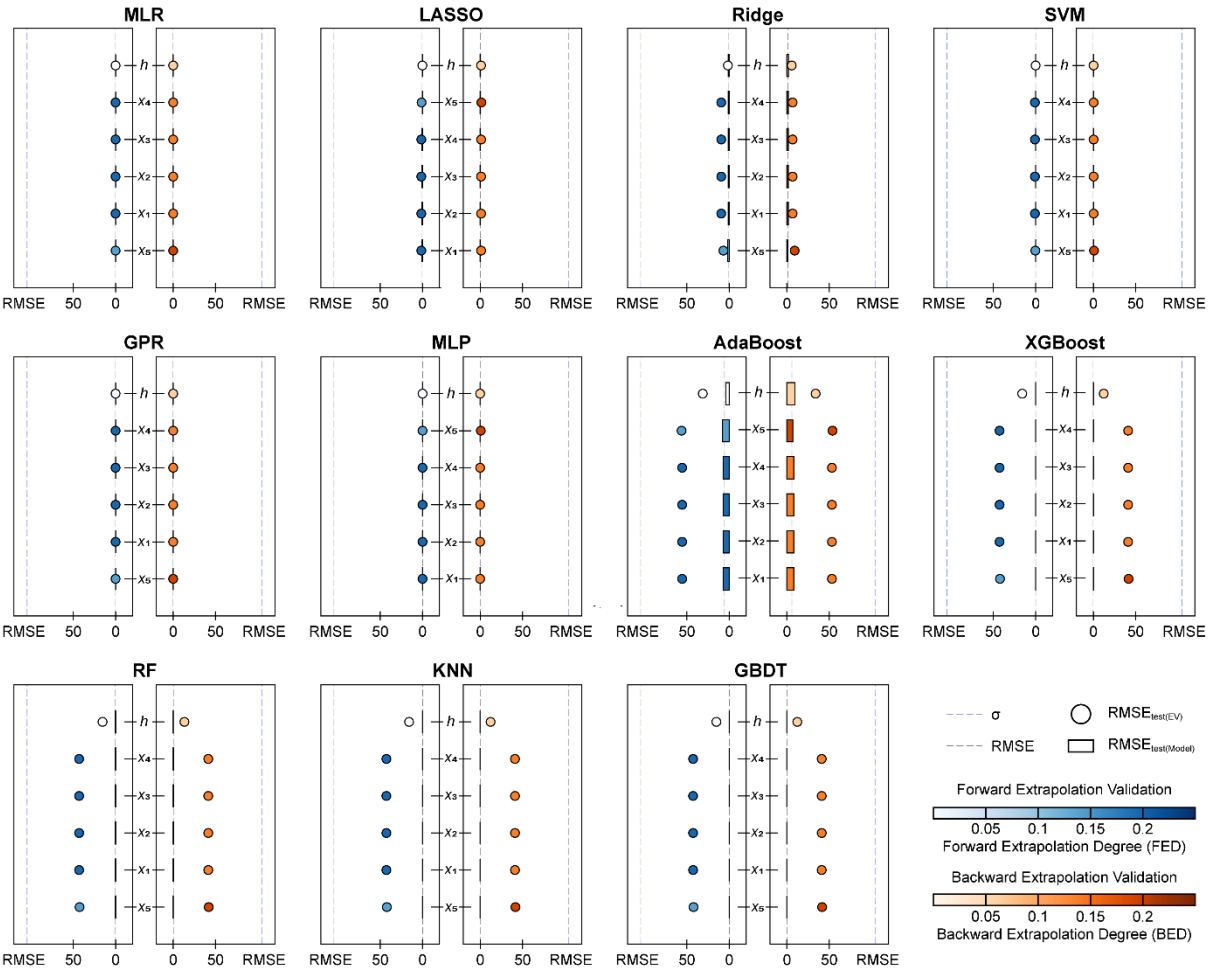


Fig. S6 Extrapolation validation (EV) results of MLR, LASSO, Ridge, SVM, MLP, AdaBoost, RF, KNN, GBDT and GPR Model for ML models developed with data from linear multivariate.

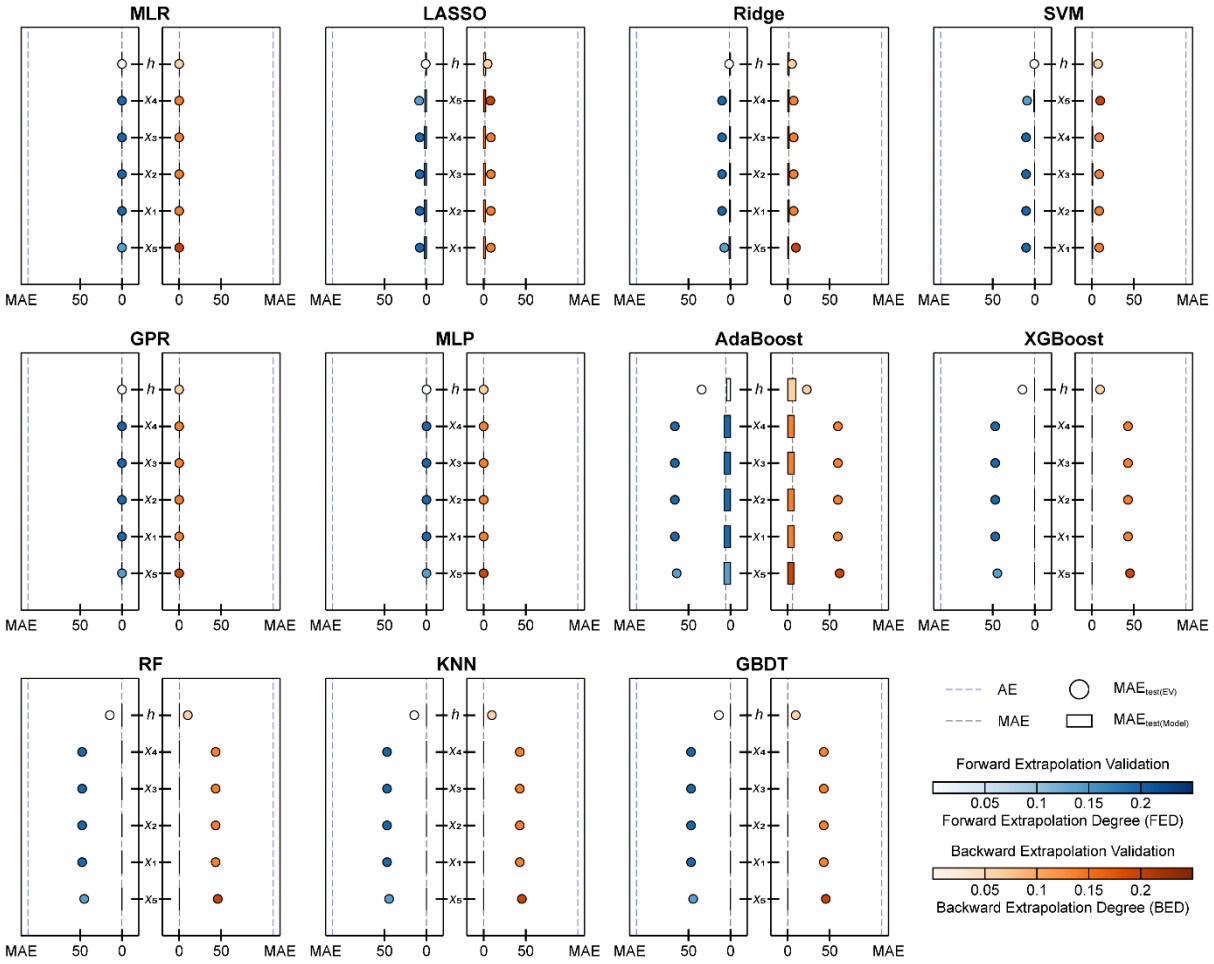


Fig. S7 Extrapolation validation (EV) results of MLR, LASSO, Ridge, SVM, MLP, AdaBoost, RF, KNN, GBDT and GPR Model for ML models developed with data from nonlinear multivariate.

Text S4 Development of PI-T_g models

The 29 norm indexes (I) required for MLR models developed in the literature were calculated. With these 29 Is as the independent variables and the PI glass transition temperature (T_g) data provided in the literature as the dependent variables, 10 ML algorithmic models for LASSO, Ridge, GPR, MLP, AdaBoost, RF, GBDT, SVM, KNN and XGBoost were developed using scikit-learn[1] version 1.3.0 and xgboost[3] version 1.7.6. The I data were standardized for modelling. The ratio of the training set to the test set in the modeling is completely consistent with the settings in the literature. Using R^2 and MAE as statistical parameters to evaluate the established model, the optimal model hyperparameters were determined with the best average performance between the training and test sets. (Table S10 and Fig. S8). To be clear, During the development of machine learning models containing the parameter "random_state", this parameter is always set to 1.

Table S10 Hyperparameters of the PI-T_g ML model developed in this study.

Model	Parameters
Lasso	alpha = 0.042, tol = 0.0009
Ridge	alpha = 0.99, max_iter = 100
SVM	C = 750, epsilon = 1.3, gamma = 0.009, kernel = 'rbf'
GPR	kernel=kernel=2**2 * DotProduct(sigma_0=2) + WhiteKernel(noise_level=0.9)
MLP	activation = 'identity', alpha = 0.0001, hidden_layer_sizes = (150, 50), solver = 'adam'
AdaBoost	learning_rate = 0.9, n_estimators = 750
XGBoost	learning_rate = 0.09, max_depth = 4, n_estimators = 280
RF	max_depth = 9, max_features = 9, n_estimators = 150
KNN	n_neighbors = 2
GBDT	learning_rate = 0.08, max_depth =5, n_estimator = 250

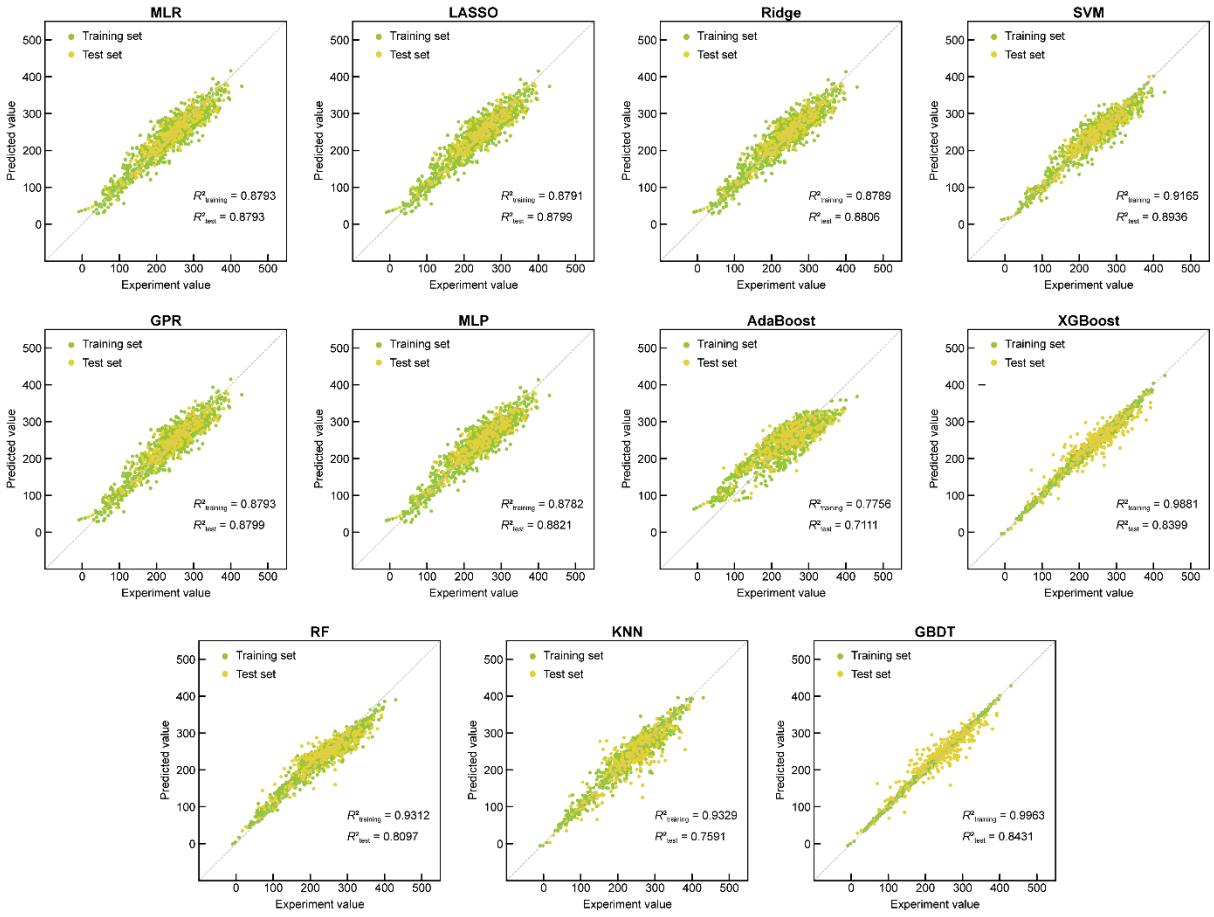


Fig. S8 PI-T_g ML models predicted *vs.* experimental values.

Text S5 Extrapolation validation of PI-T_g models.

Evaluation of the extrapolation ability of 11 PI-T_g models by using the EV method. Serialized extrapolation was performed on the *h* and *I*s of the training set, and the training (EV) and test (EV) sets were re-divided in the ratio of 8:2, the ED was calculated and the extrapolation ability of the model was evaluated.

References

1. Scikit-learn Scikit-learn. https://scikit-learn.org/stable/.
2. Python Python. www.python.org/.
3. xgboost xgboost. https://xgboost.readthedocs.io/en/stable/.
4. joblib joblib. https://joblib.readthedocs.io/en/stable/.