Domain-Based Fuzzing for Supervised Learning of Anomaly Detection in Cyber-Physical Systems (Appendix)

HERMAN WIJAYA, Singapore University of Technology and Design, Singapore
MAURÍCIO ANICHE, Delft University of Technology, Netherlands
ADITYA MATHUR, Singapore University of Technology and Design, Singapore

A APPROACH

A.1 Equivalence Class Partitioning

The specification of the raw water process marks the water level sensor reading into certain predefined constants of *LL* (Very low), *L* (Low), *H* (High) and *HH* (Very high) as 250 mm, 500 mm, 800 mm and 1000 mm, respectively. In normal operation, the motorized valve *MV*101 should open when the water level reaches L and close when it reaches H. When the water level in the ultra-filtration process is low, the PLC will turn on the pump P101 to transfer water to the ultra-filtration tank in the subsequent process. Regardless of whether the ultra-filtration process needs water or not, the pump P101 should stop when the water level reaches *LL*. Likewise, the reading of FIT101 should indicate a normal operating flow between 0.0 to 4.40 m^3/h.

Tables 1, 2 and 3 show the details of equivalence classes partitions of *LIT*101, *FIT*101 and *MV*101 respectively.

Table 1. Equivalence Class Partitioning LIT101

Partition	LIT101 reading	Expected MV101 state	Expected P101 state
Underflow	<= *LL*	OPEN	OFF
Low	>= *LL* and <= *L*	OPEN	ON or OFF
Normal	> *L* and < *H*	OPEN or CLOSED	ON or OFF
High/Overflow	>= *H* and < *HH*	CLOSED	ON or OFF

Table 2. Equivalence Class Partitioning of FIT101

Partition	FIT101 reading	Expected MV101 state
Zero	0.0	CLOSED
Flowing	>= 0.0 and <= 4.40	OPEN
Abnormal	< 0.0 or >= 4.40	OPEN or CLOSED

Table 3. Equivalence Class Partitioning of MV101

Partition	*MV*101 command	*MV*101 state
Closed	1	CLOSED
Open	2	OPEN

A.2 Learning the Classifiers – Evaluation

Confusion matrix is used to represent the performance of each classifier.

Table 4. Confusion matrix of anomaly detection model

		Predicted Class	
		Abnormal	***Normal***
Actual Class	***Abnormal***	True Positive	False Negative
	Normal	False Positive	True Negative

As shown in Table 4, the normal and abnormal columns represent the predicted class, whereas the normal and abnormal rows represent the actual class. The intersection between columns and rows show the corresponding counts which provide a basic information of the performance of the classifier:

- True Positive (TP) represents the number of correctly detected anomalies
- False Positive (FP) represents the number of false alarms in the model
- False Negative (FN) represents the number of undetected anomalies
- True Negative (TN) represents the number of correctly classified normal traces

The following performance evaluation metrics were used to evaluate the classifiers: accuracy, error rate, precision, recall, and F1 scores.

Accuracy is the percentage of the correctly classified instances while the error rate is the percentage of the misclassified instances from the dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$ErrorRate = 1 - Accuracy$$

Precision score is calculated as the number of correctly detected anomalies over the number of samples that are classified as anomalies, while recall score is calculated as the number of correctly detected anomalies over the number of samples that should have been classified as anomalies.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Formally, the precision score measures positive predictive value and the recall score measures sensitivity of the classifier.

F1 score computes both the precision and recall as a harmonic average to balance between the positive predictive value and sensitivity, which is desirable for anomaly detection models.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

B RESULTS DETAILS

B.1 Supervised Classifiers Performance

B.1.1 ***Multilayer Perceptron (MLP).*** Our MLP implementation uses the following model parameters.

- Fully connected networks (Dense)
- 1 input layer with 450 neurons, 'tanh' activation function
- 3 hidden layers with 100 neurons, 50 neurons and 10 neurons respectively, 'tanh' activation function
- 1 output layer with 1 neuron, 'sigmoid' activation function
- 'Adam' optimizer, learning rate of 0.001, 'binary cross-entropy' loss function
- 40 epochs with batch size of 10

Tables 5 and 6 show the confusion matrix and performance metrics evaluation of the MLP model on the testing dataset.

Table 5. MLP confusion matrix

MLP Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	47	1
	Normal	2	70

Table 6. MLP performance metrics

Metrics	Score
Time to build model	5.085 sec
Accuracy	97.50
Precision	95.92
Recall	97.92
F1	96.91

Table 7 shows the performance metrics evaluation of the MLP model by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

Table 7. MLP k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	95.00	98.11	98.33	100.00	100.00
Precision	88.89	100.00	100.00	100.00	100.00
Recall	87.50	95.83	95.83	100.00	100.00
F1	94.12	97.87	97.87	100.00	100.00

B.1.2 ***Support Vector Machine (SVM).*** Our SVM classifiers are implemented in two different kernels, the RBF kernel and linear kernel, as the following model parameters.

- RBF kernel
  C=1.0, kernel='rbf', gamma=0.02, cache_size=1000
- Linear kernel
  C=0.01, kernel='linear', gamma='auto', cache_size=1000

Tables 8 and 9 show the confusion matrix and tables 10 and 11 show the performance metrics evaluation of the SVM models on the testing dataset.

Table 8. SVM-RBF confusion matrix

SVM-RBF Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	48	0
	Normal	4	68

Table 9. SVM-LNR confusion matrix

SVM-LNR Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	35	13
	Normal	2	70

Table 10. SVM-RBF performance metrics

Metrics	Score
Time to build model	0.010 sec
Accuracy	96.67
Precision	92.31
Recall	100.00
F1	96.00

Table 11. SVM-LNR performance metrics

Metrics	Score
Time to build model	0.008 sec
Accuracy	87.50
Precision	94.60
Recall	72.92
F1	82.35

Tables 12 and 13 shows the performance metrics evaluation of the SVM models by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

Table 12. SVM-RBF k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	93.33	95.00	98.33	95.00	95.00
Precision	85.71	95.65	96.00	95.65	88.89
Recall	100.00	91.67	100.00	91.67	100.00
F1	92.31	93.61	97.96	93.62	94.12

Table 13. SVM-LNR k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	81.67	93.33	85.00	86.67	85.00
Precision	100.00	100.00	85.71	94.44	89.47
Recall	54.17	83.33	75.00	70.83	70.83
F1	70.27	90.91	80.00	80.95	79.07

B.1.3 ***Logistic Regression (LR).*** Our LR implementation uses the following model parameters.

- C=0.05, solver='liblinear', max_iter=1000

Tables 14 and 15 show the confusion matrix and performance metrics evaluation of the LR model on the testing dataset.

Table 14. LR confusion matrix

LR Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	42	6
	Normal	5	67

Table 15. LR performance metrics

Metrics	Score
Time to build model	0.120 sec
Accuracy	90.83
Precision	89.36
Recall	87.50
F1	88.42

Table 16 shows the performance metrics evaluation of the LR model by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

B.1.4 ***Decision Tree (DT).*** Our DT implementation uses the default model parameters of Scikit-learn's DecisionTreeClassifier as the following.

- criterion='gini', splitter='best', max_features=None

Table 16. LR k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	88.67	85.00	85.00	88.83	91.67
Precision	100.00	85.71	89.47	94.74	100.00
Recall	66.67	75.00	70.83	75.00	79.17
F1	80.00	80.87	79.07	83.71	88.37

Tables 17 and 18 show the confusion matrix and performance metrics evaluation of the DT model on the testing dataset.

Table 17. DT confusion matrix

DT Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	46	2
	Normal	5	67

Table 18. DT performance metrics

Metrics	Score
Time to build model	0.015 sec
Accuracy	94.17
Precision	90.20
Recall	95.83
F1	92.93

Table 19 shows the performance metrics evaluation of the DT model by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

Table 19. DT k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	98.33	96.67	93.33	95.00	93.33
Precision	100.00	100.00	88.46	92.00	100.00
Recall	95.83	91.67	95.83	95.83	83.33
F1	97.87	95.65	92.00	93.88	90.91

B.1.5 ***Random Forest (RF)***. Our RF implementation uses the default model parameters of Scikit-learn's RandomForestClassifier as the following.

- n_estimators=100, criterion='gini', bootstrap=True, max_features='auto'

Table 20. RF confusion matrix

RF Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	46	2
	Normal	0	72

Table 21. RF performance metrics

Metrics	Score
Time to build model	0.080 sec
Accuracy	98.33
Precision	100.00
Recall	95.83
F1	97.87

Tables 20 and 21 show the confusion matrix and performance metrics evaluation of the RF model on the testing dataset.

Table 22 shows the performance metrics evaluation of the RF model by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

Table 22. RF k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	98.33	96.67	96.67	100.00	93.33
Precision	100.00	100.00	100.00	100.00	100.00
Recall	95.83	91.67	91.67	100.00	83.33
F1	97.87	95.65	95.65	100.00	90.91

B.1.6 ***Extremely Randomized Trees (ExraTrees)***. Our ExtraTrees implementation uses the default model parameters of Scikit-learn's ExtraTreesClassifier as the following.

- n_estimators=100, criterion='gini', bootstrap=False, max_features='auto'

Tables 23 and 24 show the confusion matrix and performance metrics evaluation of the ExtraTrees model on the testing dataset.

Table 23. ExtraTrees confusion matrix

ExtraTrees Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	48	0
	Normal	1	71

Table 25 shows the performance metrics evaluation of the ExtraTrees model by cross-validation with the entire dataset using k-fold cross-validation, with k=5.

Table 24. ExtraTrees performance metrics

Metrics	Score
Time to build model	0.066 sec
Accuracy	99.17
Precision	97.96
Recall	100.00
F1	98.97

Table 25. ExtraTrees k-fold cross-validation

Metrics	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Accuracy	100.00	96.67	98.33	100.00	98.33
Precision	100.00	100.00	100.00	100.00	100.00
Recall	100.00	91.67	95.83	100.00	95.83
F1	100.00	95.65	97.87	100.00	97.87

B.2 Unsupervised Models Performance

B.2.1 ***One-Class Support Vector Machine (OC-SVM)***. Our OC-SVM implementation uses following model parameters.

- nu=0.01, kernel='rbf', gamma=0.0256, cache_size=1000

Tables 26 and 27 show the confusion matrix and performance metrics evaluation of the OC-SVM model on the testing dataset.

Table 26. OC-SVM confusion matrix

OC-SVM Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	120	0
	Normal	1	55

Table 27. OC-SVM performance metrics

Metrics	Score
Time to build model	0.006 sec
Accuracy	91.15
Precision	87.59
Recall	100.00
F1	93.39

B.2.2 ***Isolation Forest (IF).*** Our IF implementation uses following model parameters.

- n_estimators=100, behaviour='new', contamination=0.27, bootstrap=False, verbose=0

Tables 28 and 29 show the confusion matrix and performance metrics evaluation of the IF model on the testing dataset.

Table 28. IF confusion matrix

IF Confusion Matrix		**Predicted Class**	
		Abnormal	***Normal***
Actual Class	***Abnormal***	95	25
	Normal	24	48

Table 29. IF performance metrics

Metrics	Score
Time to build model	0.150 sec
Accuracy	74.48
Precision	79.83
Recall	79.17
F1	79.50