Medical Application Prototype

PDA Prototype

As a proof of concept, we have developed a distributed component. We developed an android mobile application to manage the data and faults on the patient side. Prediction Models are assumed to be trained offline and can be automatically deployed on the cloud or data center. For this purpose, we have developed a GUI based gadget to train, test, and deploy prediction models to Firebase cloud data based on which the mobile application can download and use in fault detection.

Training and Anlysis

In **Figure 1**, The training process can be conducted using SVM, ANN, or Decision tree. This application java based and it is developed in a Netbeans project. The data engineer configures the training process and launches the training process. Validation is conducted either by applying 10-Cross-validation or by providing a validation dataset. After that, the model can be deployed to the cloud, FireBase DB, see Figure 2. The backend of the this application is WEKA.

In **Figure 1**, the following options are available:

1. Browse data: browse and upload the data set in a csv format. The following is a sample of the data set snapshots:

Sensor Dataset snapshot and summary

Pkt	forwarded	dropped	Retries	battery	Quality tx	Quality rx	Parent rssi	Label
1	0	3	930	25	15	15	234	battery alarm
2	0	3	932	25	15	15	234	battery alarm
3	0	3	936	25	15	15	233	battery alarm
4	7994	7	853	28	15	13	237	normal data
5	7994	7	856	28	15	15	237	normal data
…	7978	7	851	28	7	15	237	hardware alarm

The Sensor dataset can be found in the github link of the project and it has the following distribution:

Sensor Dataset summary

Fault/Label type	Percentage/count
Battery faults	61%
Hardware faults	8%
Normal	31%
Total Data	29603

Snapshot of Synthesis data sets

Synthesis data set – A			Synthesis data set with conditions –B			
Sensor ID	Reading	Label	Sensor ID	Reading	Condition	Label
159	1.632	Fault	54	4.08	After Surgery	Fault
5	-1.34	Fault	80	4.17	Septic shock	Fault

480	7.502	Normal	76	13.6	Anesthesia	Normal
150	2.80	Fault	20	-4.4	labor-Mother	Fault
310	4.142	Fault	47	11.3	Anesthesia	Normal
450	0.088	Fault	66	8.15	Normal	Normal
383	11.59	Normal	30	4.78	Labor-Baby	Fault
215	4.896	Fault	95	-0.4	Normal	Fault
….	….	….	82	3.79	After Surgery	Fault
			…	….	…	

2. Browse Configuration: upload a configuration file that holds parameters for SVM/DT, see sample configuration in the below Tables for SVM and Decision tree.

#SVM configuration C= 2^11, gamma = 2^-2
svm_type=0
kernel=2
C=1
gamma=0.15
coef0=0
degree=2
	##Decision Tree parameter configuration
	U=1
	#Use unpruned tree.
	O=1
	#Do not collapse tree.
	C=0.55
	#Set confidence threshold for pruning.(default 0.25)
	M=3
	#Set minimum number of instances per leaf. (default 2)
	R=1
	#Use reduced error pruning.
	N=3
	#Set number of folds for reduced error pruning. One fold is used as pruning set. (default 3)
	B=1
	#Use binary splits only.
	S=1
	#Don't perform subtree raising.
	L=1
	#Do not clean up after the tree has been built.
	A=1
	#Laplace smoothing for predicted probabilities.
	J=1
	#Do not use MDL correction for info gain on numeric attributes.
	Q=1
	#Seed for random data shuffling (default 1).
	doNotMakeSplitPointActualValue=1
	#Do not make split point actual value.

3. Browse Test Dataset: browse and select a testing data set.
4. Train ANN/SVM/DT Training: conduct machine training of the selected data set and apply the user picked training method.
5. Save Model: save the prediction model for upload on the cloud to be used for the Mobile application.

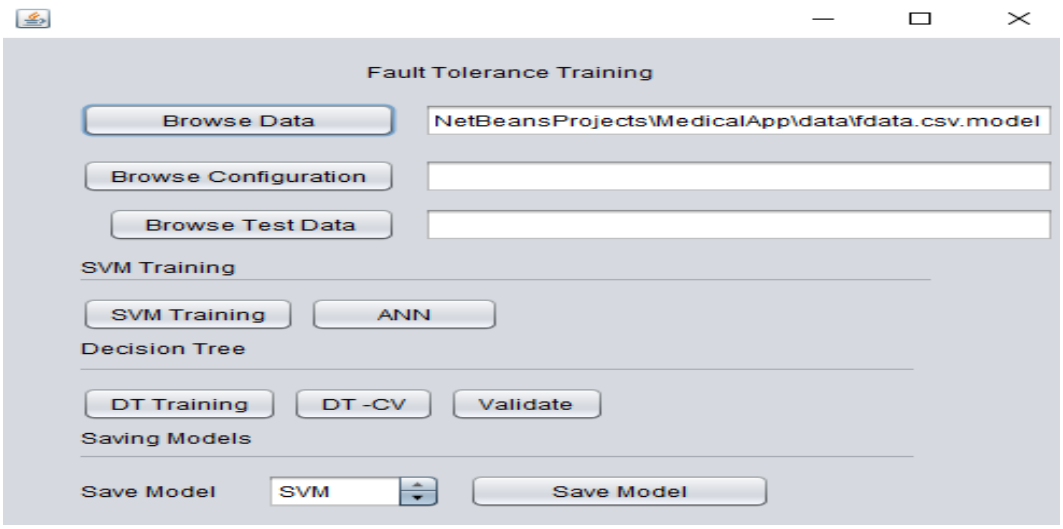


Figure 1. Training Model Gadget

Uploading and Managing Prediction Models

In **Figure 2**, the upload and managing of the prediction models is presented. Observe that we are focusing in this report on fault tolerance prediction models. After training is over, tested prediction models are uploaded to the Cloud specifically Firebase DB (Google).
In application is python based and it has simple operations as follows:

1. Retrieve a model by model id.
2. Upload a model to the cloud
3. Upload a sensor profile

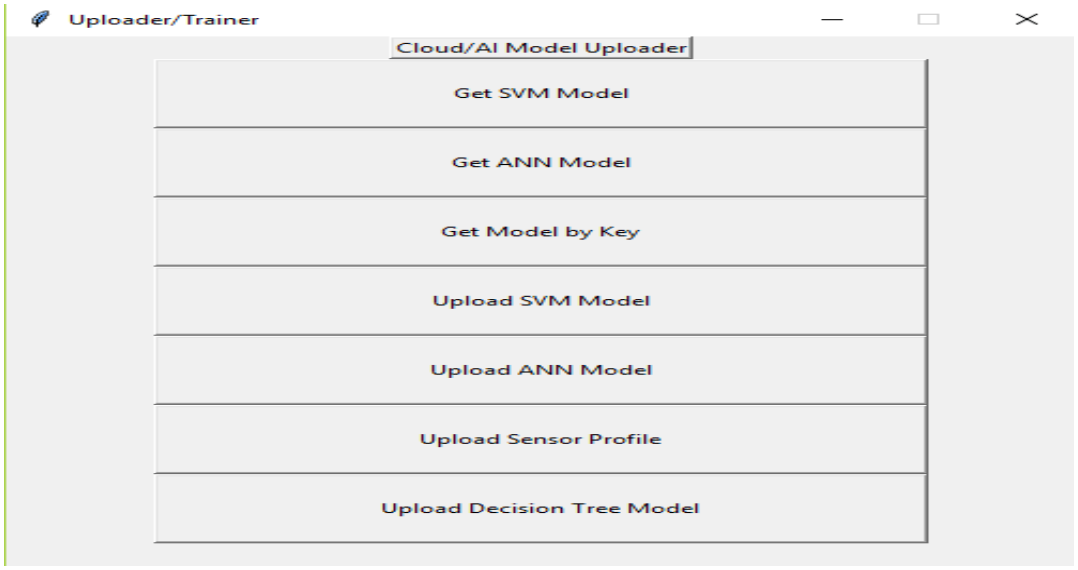


Figure 2. Model Uploader Gadget

The result of uploading can be found on FB console, see figures below.

Figure 3 DT model persisted on the Firebase DB on the cloud

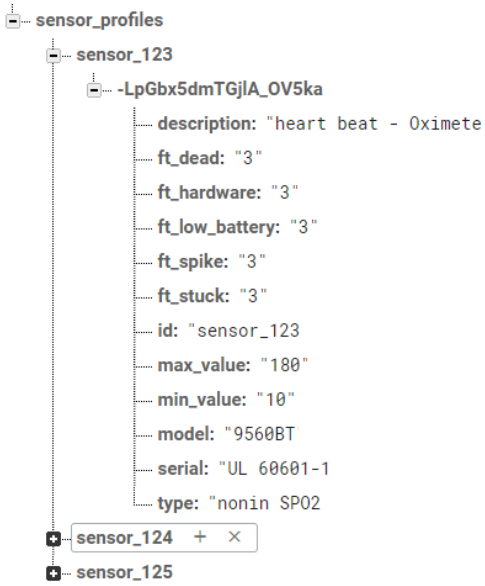


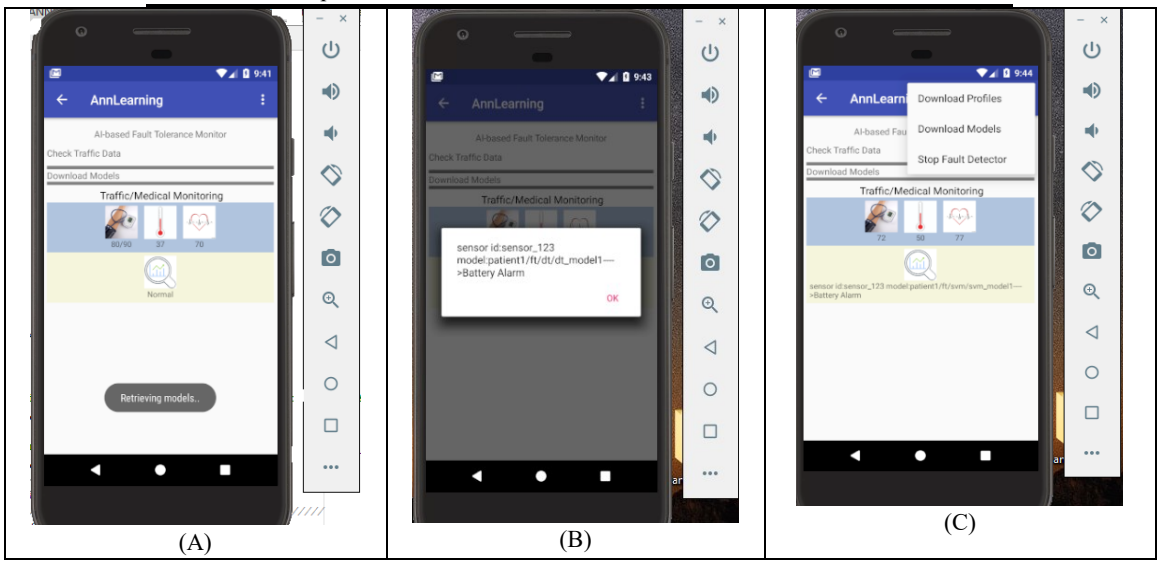
Figure 4 Sensor Profile model persisted on the Firebase on the cloud.

Mobile Application and Fault Detection

The mobile application is designed to accommodate WEKA models, LIBSVM models, and TensorFlow models. Each model might be trained to predict certain kinds of faults. In our prototype, all models were trained to predict four kinds of sensor faults along with 2 types of software faults, see **Table 1**. The design is adaptable to include more faults and predictors with small modifications. The application is also capable of automatically update its current prediction models. We have deployed the prediction models presented in the results section on the mobile prototype. **Figure 5** depicts three functional screenshots of the prototype, namely, downloading/retrieving prediction models from the cloud, detecting battery fault, and capability of data management (sensors profiles and prediction models). All the gadgets projects including the mobile prototype, are available online.

Table 1. List of detected faults in our prototype

Sensor Faults	Software Faults
Hardware Fault	Connection Lost
Battery Fault	Malformed Prediction model
Stuck at zero	Dis-functioning Prediction model
Spiked value	



(A) (B) (C)

Figure 5. Prototype Screenshots

6. Conclusions

The design of a fault tolerance system requires a great deal of care when it involves human lives. In this paper, we have explored the different faults that can occur in WBAN medical monitoring application. We proposed a framework for WBAN sensor and traffic management. The framework incorporates data management and machine learning-based fault detection. The framework is flexible to integrate different machine learning techniques that can be updated and deployed to detect new faults. The framework can accommodate different sensors through profiling.

We presented the results of three different learning techniques to detect several different faults. We proposed a threshold-based detection approach using machine learning and presented high accuracy results using Decision Trees and SVM.

As proof of concept, we developed and presented a prototype for the training process, model deployment, and model prediction. We presented a mobile application that can detect several sensor and software faults.

Future work will include exploring the software side of the WBAN applications to examine additional software faults. Next, we would like to conduct larger-scale management of prediction models on the data center to uncover additional faults. Finally, we would like to deploy our prototype on a real patient site to explore networking faults and the possibilities of using other technologies in case of communications loss.