[Group 11] Assignment #4: Analytics tools and cloud platform using real sensors

1. Overview of System Flow

1. We connect the Raspberry Pi and Laptop to the MQTT broker provided by IBM cloud
2. After establishing connection to MQTT broker, when we open/close the door, we detect an open/close sequence using the logic mentioned in the section 4
3. We then create features required for the SVM, and call the REST API for the model deployed on IBM cloud with feature vector as payload
4. Upon receiving the prediction as a response from the REST API, we publish the result to the MQTT broker on the topic “DoorStatus”
5. The Laptop is subscribed to the topic “DoorStatus” so whenever a prediction is published on the topic is received by Laptop and we print the door status with timestamp on the Laptop

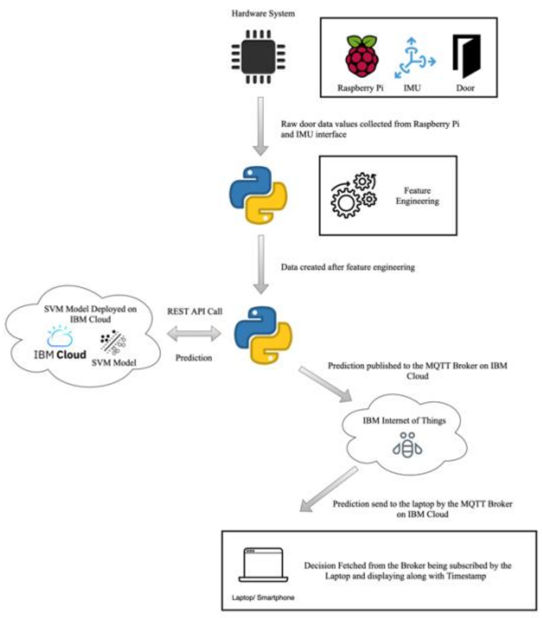
Hardware System
Raspberry Pi
IMU
Door
Raw door data values collected from Raspberry Pi and IMU interface
Feature Engineering
Data created after feature engineering
SVM Model Deployed on IBM Cloud
IBM Cloud
SVM Model
REST API Call
Prediction
Prediction published to the MQTT Broker on IBM Cloud
IBM Internet of Things
Prediction send to the laptop by the MQTT Broker on IBM Cloud
Decision Fetched from the Broker being subscribed by the Laptop and displaying along with Timestamp
Laptop/ Smartphone

Figure 1. High Level System Overview

2. Hardware

The IMU 6050 sensor used to determine the linear and angular acceleration of the door was communicated with the raspberry pi through Inter-Integrated Circuit (I2C) communication. This is an ideal communication method for this sensor as it allows multiple different readings to be sent using only a 2 wire connection using the Serial Data Line (SDA) and the Serial Clock Line (SCL) pins. The remaining two connections seen in figure one are just to supply power to the device. The raspberry pi was mounted on the edge of the door as seen in figure 3.

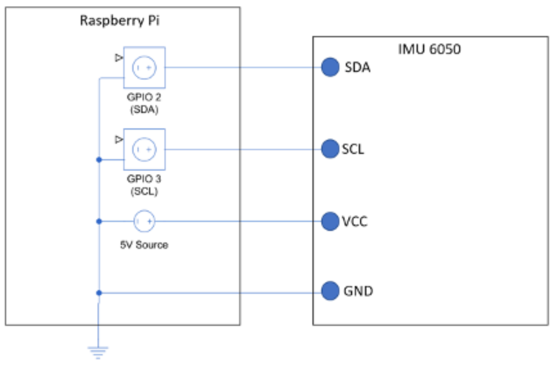

Figure 2. Hardware diagram for Raspberry Pi and IMU

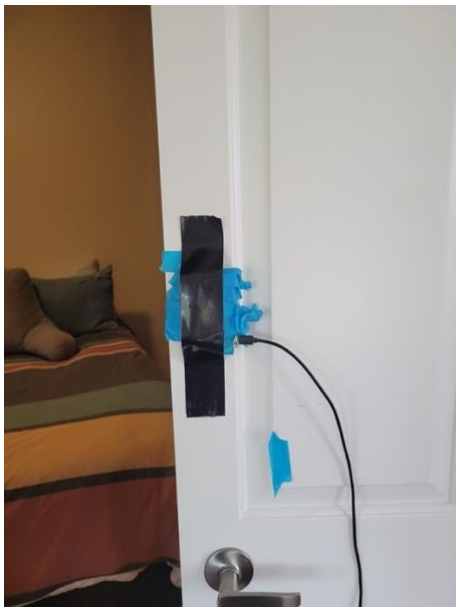

Figure 3. Raspberry Pi mounted onto door with lots of tape.

3. Data Collection

We collected approximately 350 samples by sticking the RaspberryPi with IMU on the door. After collecting and labeling the samples we created different files for training and test data based on 70:30 split.

4. Detecting Sequence of Door Open/Close

To detect a sequence of IMU values for Door Open/Close we employed a sliding window approach. We took the moving average of 3 values for Gyroscope x values. Depending on a threshold value (50), we decide if the sequence of door opening/closing has started or ended. When the moving average is above the threshold we determine it as the start of the sequence and when the moving average goes below the threshold again we determine it as the end of the sequence.

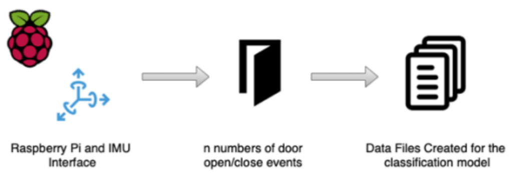
Raspberry Pi and IMU Interface
n numbers of door open/close events
Data Files Created for the classification model

Figure 4. Data Collection

To make it more precise from human error, we only end a sequence if the moving average is below the threshold for 3 consecutive values given by IMU.

5. Feature Engineering

Upon receiving the values from IMU (ax, ay, az, gz, gy, gz) for a sequence of Door Open/Close, we divided the sequence into 3 windows. For each window, we experimented with two different feature selection approaches.

1. First, we selected mean values for ax, gx, and gz from each window. This resulted in a feature vector of size 9 values (3 features from each window)
2. Second we experimented with selecting 4 features from each window, i.e mean and standard deviation of each window along with mean and standard deviation of gyroscope values. This resulted in a feature vector of size 12 values (4 features from each window).

In our case, the first approach of selecting ax, gx and gz worked out best when calculating accuracy on a test set.

6. Model Selection and Hyper Parameter Tuning

We have taken an SVM classifier with RBF kernel to detect door opening and closing using IMU values. We performed a grid search on C, gamma, PCA components, and Feature selection window size to select the best set of hyperparameters.

Hyperparameter	**Value1**	**Value2**	**Value3**	**Value4**
SVM C	0.05	0.1	0.5	0.5
SVM Gamma	0.1	0.5	-	-
PCA components	2	4		
Feature selection window size	3	4	-	-

The following were the best set of hyperparameters:-
SVM C - 0.05
SVM Gamma - 0.1
PCA Components - 2
Feature selection window size – 3

7. Model Training and Accuracy

We had split our dataset into a train and test split of 70:30 with 244 samples in the train set and 100 samples in the hold out test set. We had an equal distribution of Open and Close samples in both the train and test set.

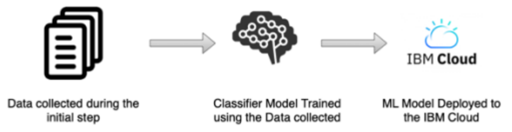
IBM Cloud
Data collected during the initial step
Classifier Model Trained using the Data collected
ML Model Deployed to the IBM Cloud

Figure 5. Model Training and Deployment

After training the model using the above set of hyperparameters using 3 fold cross validation, we achieved an accuracy of 100% on the hold out test set. We went with the cross validation approach and not a hold out validation set for training because of a lack of training examples. We also tried out different fold values for cross validation and 3 fold cross validation gave the best test set accuracy.
We have also tested our model heavily in a real world environment, giving 100% accuracy.

8. Links

Github:
https://github.ncsu.edu/jwboerge/IoT_ASN4_Group11