

Introduction

The purpose of this application note is to help users understand the design of the ML training tool and to provide guidance on how to configure the parameters for variant applications. The training tool we designed takes advantage of acceleration data from Kionix tri-axial accelerometers. For each application, the goal is to generate a classifier that can recognize different states with the ML training tool. For example, the classifier generated by the tool for the vacuum cleaner application is able to recognize surfaces as carpet, tile, and rubber. When using the ML training tool for different applications, it is very likely that some customization will be required to optimize performance. The information provided here will help the users get the most out of the ML training tool.

Application Schematic

This section shows the layout of the ML training tool GUI, which has 5 parts, Preprocessing, Axis Selection, Feature Extraction, Training and Prediction. Raw data will be preprocessed to eliminate noise, and feature extraction will be performed after data preprocessing. Finally, we can generate an ML model with well formatted feature data. Parameters in preprocessing and feature extraction phase are highly configurable and can be modified based on user's needs and users are free to use or skip any data preprocessing step by checking or unchecking each procedure.

Offline prediction phase can test the robustness of the model trained in previous phases. Online prediction phase can predict labels in real-time using the model selected.

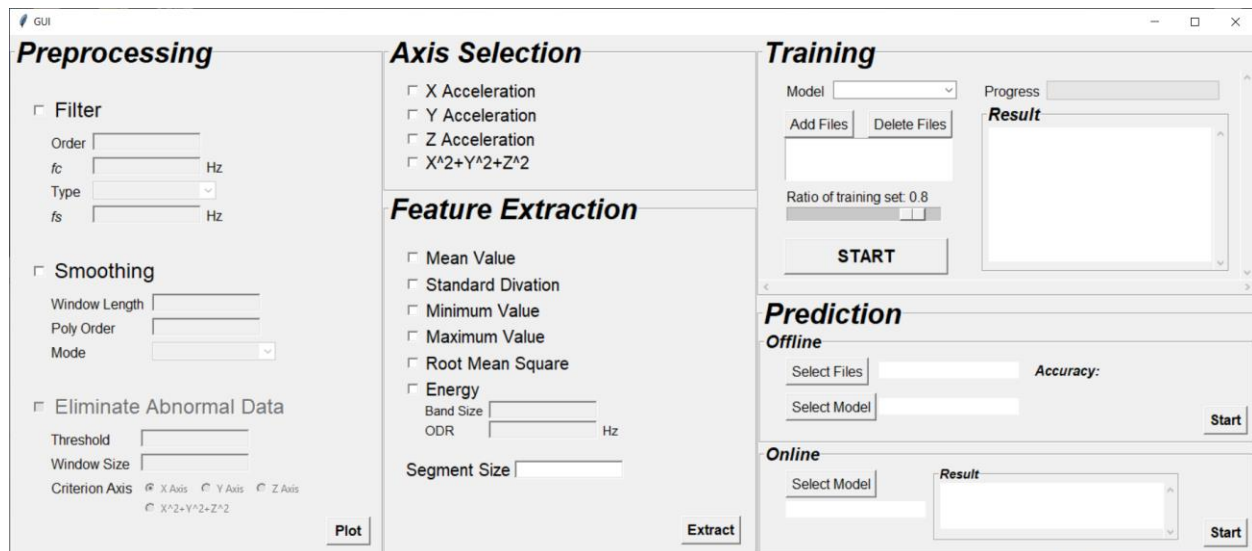


Fig 1. GUI Schematic

Design Overview

Preprocessing

For preprocessing section, filtering, smoothing and data elimination are not mandatory. It totally depends on if these techniques are going to make the classification performance better. In most cases, they are helpful. To use the filter or smoothing module, you can simply check the checkbox and fill in all required parameters. One thing worth noticing is once you choose to use any module, you'll have to set all parameters. If you missed any of them, a warning window will pop out to remind you.

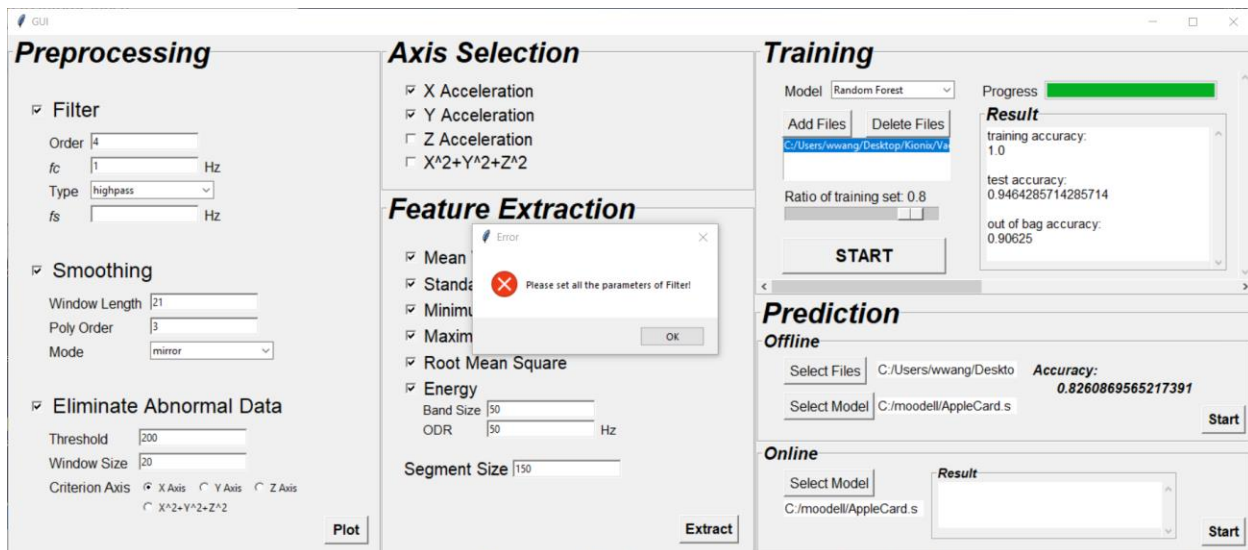


Fig 2. Warning window

We use python built-in signal.butter and signal.savgol_filter packages to do the filtering and smoothing. For more information about these packages and definitions of all parameters, please refer to the following links

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html>

https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.signal.savgol_filter.html

Data elimination module is used to get rid of data that are considered abnormal. In the application of surface recognition, data are abnormal when the cleaner confronts obstacles or redirects causing huge spikes in the plot. We designed the threshold to be the variance of the data in a group of data. The size of the group can be customized by users via segment size entry.

There is also a plot button for users to visualize differences before and after data preprocessing. After setting all parameters in data preprocessing section and

selecting input data, we are good to start plotting. Notice that before plotting, users will have to add a compilation of input data in the training section.

Figure 1 / 17

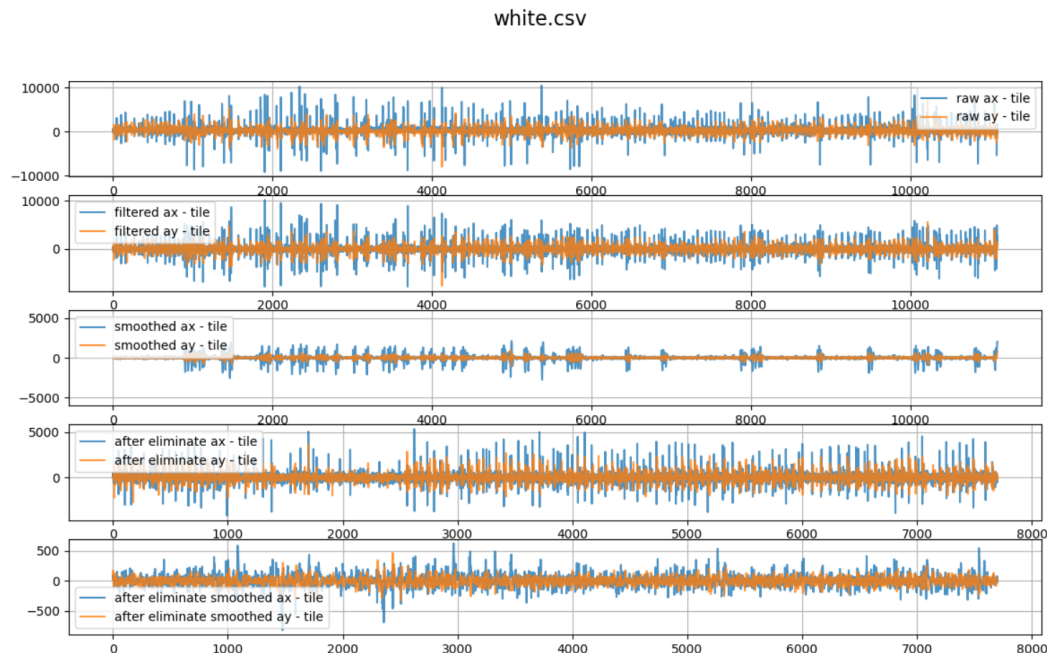


Fig 3. Visualization of Data

Axis Selection

Acceleration data from Kionix sensor is always tri-axial, but when it comes to classification and prediction, more data is not always good and sometimes can bring up even more noise. Axis Selection allows user to choose acceleration data from any axis. User can train the machine learning model with data on only one axis, two axes, or all of them.

Feature Extraction

In this section, users are able to select features that distinguish different classes the most. In general, the more unique the features are that you are using to characterize a class, the better accuracy you will get. We provide mean, standard deviation, maximum, minimum, root mean square and energy for users to choose. In most applications, these features are enough, in later version, we are going to add more features.

Training

Fig 4. Model Selection

After setting all parameters in previous sections, we can start training the machine learning model. In the training section, there two machine learning models to choose from they are Random Forest and Support Vector Machine (SVM). Click Add Files button to start adding a compilation of raw data, the compilation csv file looks like the following:

	A	B	C	D	E
1	file	dir	label	header	
2	white.csv	raw_data\\log_files	tile	6	
3	red.csv	raw_data\\log_files	rubber	6	
4	carpet.csv	raw_data\\log_files	carpet	6	
5	tile_white.csv	raw_data\\log_files	tile	6	
6	meeting_carpet_boost.csv	raw_data\\log_files	carpet	4	
7	meeting_carpet_standard.csv	raw_data\\log_files	carpet	4	
8	office_carpet_boost.csv	raw_data\\log_files	carpet	4	
9	office_carpet_standard.csv	raw_data\\log_files	carpet	4	
10	office_wmarvel_boost.csv	raw_data\\log_files	tile	4	
11	office_wmarvel_max.csv	raw_data\\log_files	tile	4	
12	office_wmarvel_standard.csv	raw_data\\log_files	tile	4	
13	red_marvel_boost.csv	raw_data\\log_files	rubber	4	
14	red_marvel_max.csv	raw_data\\log_files	rubber	4	
15	meeting_carpet_max.csv	raw_data\\log_files	carpet	4	
16	office_carpet_max.csv	raw_data\\log_files	carpet	4	
17	red_marvel_standard.csv	raw_data\\log_files	rubber	4	
18					

Fig 5. Raw Data Compilation

So, whenever you want to train the model with more data, simply add the new data files and labels to the compilation. Ratio of training set entry is to split raw data as training set and testing set to do cross validation. Users can set the ratio of training

set from 0.1 to 0.9. In general, training set should be larger than testing set. To start training the model, simply click 'START' button and a window will pop out allowing users to choose directory where they want to save the model. When the training process is done, users can see training accuracy, testing accuracy and out of bag accuracy on the GUI. Given these results, users can adjust parameters in previous sections to get better performance accordingly.

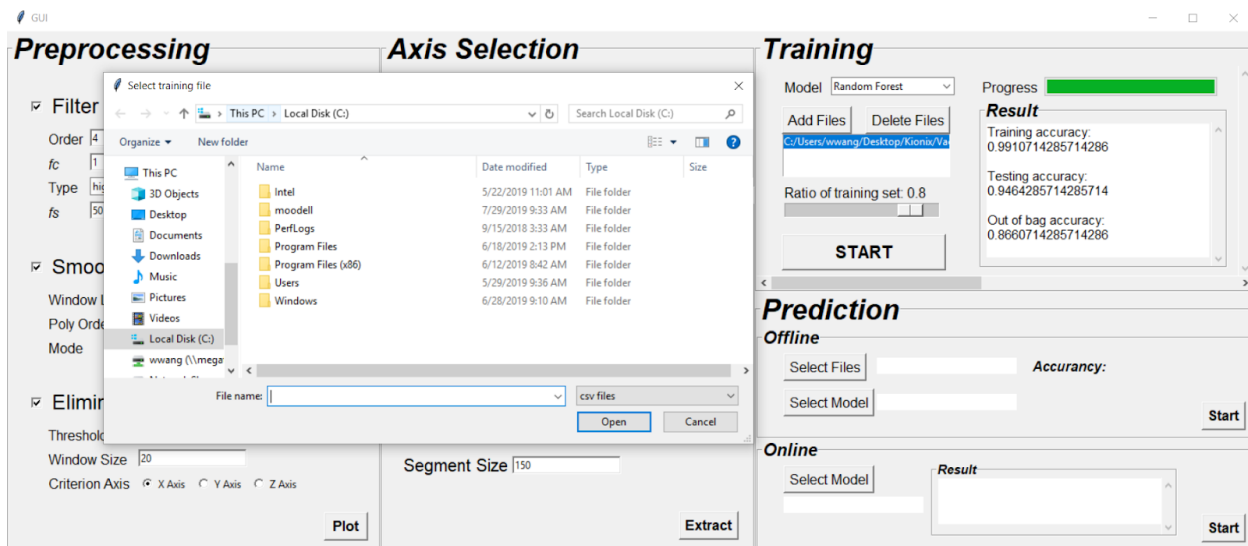


Fig 6. Start and save training model

Prediction

After a model has been successfully trained and saved, users are able to predict new input data either offline or online. To predict offline data, select saved model and a compilation of test data, the testing accuracy will display on the console.

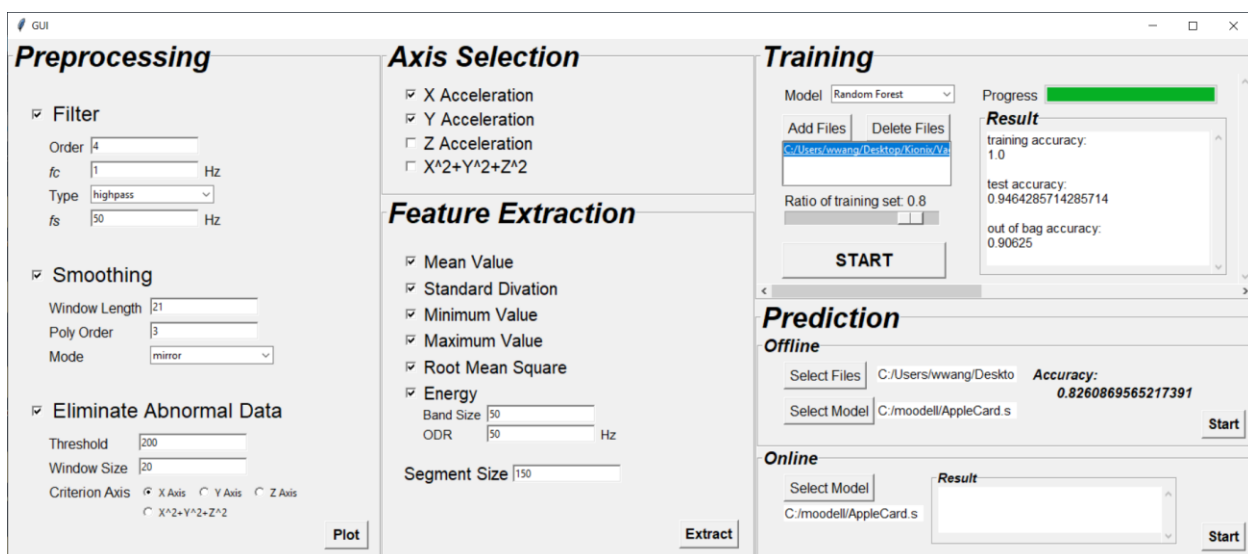


Fig 7. Predict offline with saved model

To predict online data via Kionix sensor node, connect sensor node to the laptop and select saved model in the prediction online select, online prediction results will display on the console like following.

```
(base) C:\Users\wwang\Desktop\Kionix\Vacuum\Cleaner>python newMLAPP.py  
rubber  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
tile  
rubber  
tile  
tile  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet  
carpet
```

Fig 8. Predict online with saved model

To only quit the online prediction processing, disconnect the sensor node.

Quick Start Implementations

Here we present an example on how to set all the parameters to achieve the best model for a certain application.

Surface Recognition Example

In this example, we are going to establish a model that can recognize different surfaces such as carpet, tile, and rubber using accelerometer data.

Data Acquisition

Accelerometer data are collected by Kionix KMX62 sensor node, which is mounted on the top of the vacuum cleaning robot and parallel to the ground. We ran the robot and collect data on three different types of surfaces carpet, tile, and rubber for at least 3 minutes.

Data Preprocessing

Check 'Filter', select 'highpass', and set other parameters to do DC offset.

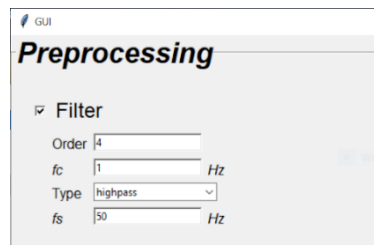


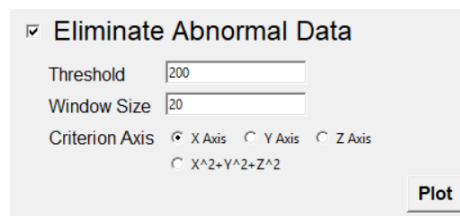
Fig 9. Parameters of Filter

Check 'Smoothing', set window length, poly order, and mode to smooth the data. Smoothed data can clearly show intervals when the robot confronts bumps.



Fig 10. Parameters of Smoothing

Check 'Eliminate Abnormal Data'. Based on smoothed data, we can easily find those data indicate that robot confronted bumps or did a right or left turn, since they have larger vibration. Threshold in this part means the threshold of variance among smoothed data in window size. Since there are three axis, if we eliminate data separately, the dimension may not match, so we are supposed to select one axis as criterion axis.



☒ **Eliminate Abnormal Data**

Threshold

Window Size

Criterion Axis ☒ X Axis ☐ Y Axis ☐ Z Axis
☐ $X^2+Y^2+Z^2$

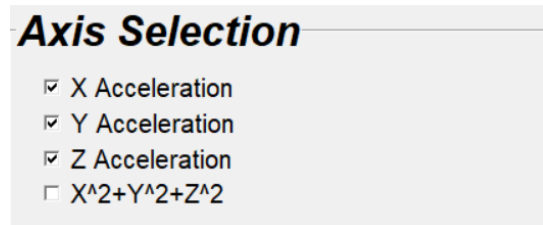
Plot

Fig 11. Parameters of Elimination

Click 'Plot' button will show a bunch of figures after each preprocessing step.

Axis Selection

X acceleration data have more obvious bump data, so it's much easier to eliminate them using previous steps. Also, it contains enough information of different surfaces to be classified.



Axis Selection

☒ X Acceleration

☒ Y Acceleration

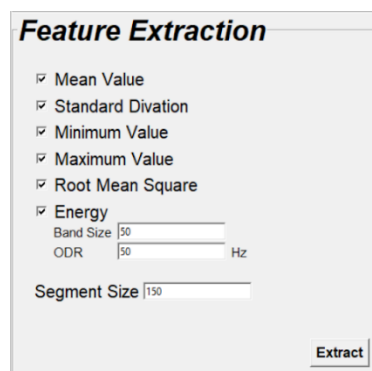
☒ Z Acceleration

☐ $X^2+Y^2+Z^2$

Fig 12. Axis Selection

Feature Extraction

In this example, we select all the features to achieve the best accuracy. Set segment size as 150, which means for every 150 preprocessed data, we extract the following features and it becomes one data with those features, waiting for training.



Feature Extraction

☒ Mean Value

☒ Standard Deviation

☒ Minimum Value

☒ Maximum Value

☒ Root Mean Square

☒ Energy

Band Size

ODR Hz

Segment Size

Extract

Fig 13. Different features

Click 'Extract' button can save those extracted features and their labels as 'npv' files.

Training

Select training model, here we select Random Forest as the model. Model information is stored in 'train.py' file. Initial parameters can be modified in this file, and more other models can be added.

Next to 'Model', training file can be added or deleted. Here we select the file which contains all the raw data and their labels.

Ratio of training set is usually 0.7 or 0.8, which means 70% or 80% of data becomes training data to train the model, and the other becomes test data to verify the model. Then click 'START', training starts, and the progress bar will show the progress of training.

Result will show in the 'Result' console.

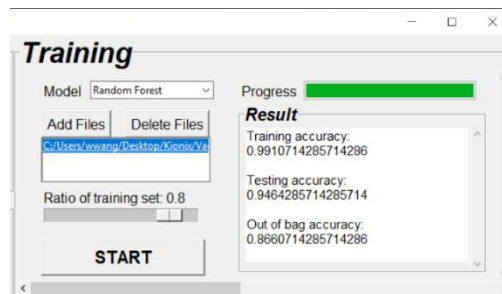


Fig 14. Training results

Prediction

Offline Prediction

In this section, model trained in training section can be used to test new offline data. Select files used to test, and select any model trained in previous section. Click 'Start', accuracy will show in the right part.

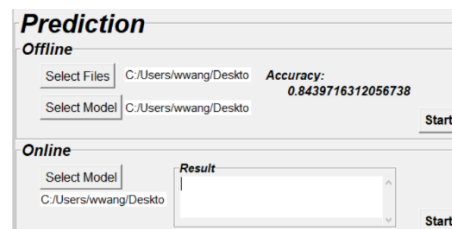


Fig 15. Offline prediction results

Online Prediction

In this section, we can use any model trained in training section to do real-time prediction.

Select model, and click 'Start', the results will show in console.

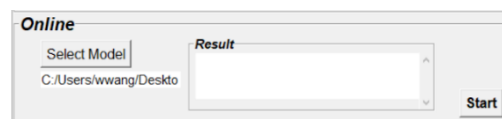


Fig 16. Online prediction

```
(base) C:\Users\wwang\Desktop\Kionix\Vacuum\Cleaner>python MLApp.py
carpet
carpet
tile
carpet
carpet
tile
rubber
rubber
rubber
carpet
```

Fig 17. Results in console

Machine Health Example

Data Acquisition

We start by making a compilation of machine data under different situations, we label the data as 'Normal' (without load) and 7Hz_3load_motor etc.

1	file	dir	label	header
2	7Hz_0load_motor.csv	raw_data\\log_files	Normal	4
3	7Hz_3load_motor.csv	raw_data\\log_files	7Hz_3load_motor	4
4	7Hz_0load_block.csv	raw_data\\log_files	Normal	4
5	7Hz_3load_block.csv	raw_data\\log_files	7Hz_3load_block	4
6	9Hz_0load_motor.csv	raw_data\\log_files	Normal	4
7	9Hz_3load_motor.csv	raw_data\\log_files	9Hz_3load_motor	4
8	9Hz_0load_block.csv	raw_data\\log_files	Normal	4
9	9Hz_3load_block.csv	raw_data\\log_files	9Hz_3load_block	4
10	5Hz_0load_block.csv	raw_data\\log_files	Normal	4
11	5Hz_3load_block.csv	raw_data\\log_files	5Hz_3load_block	4
12	5Hz_0load_motor.csv	raw_data\\log_files	Normal	4
13	5Hz_3load_motor.csv	raw_data\\log_files	5Hz_3load_motor	4
14				

Fig 18. Machine Raw Data Compilation

By plotting the data in data preprocessing section, we can see that the raw data is already very clean and easy to differentiate between different classes. So, it's not necessary to use any data preprocessing technique.

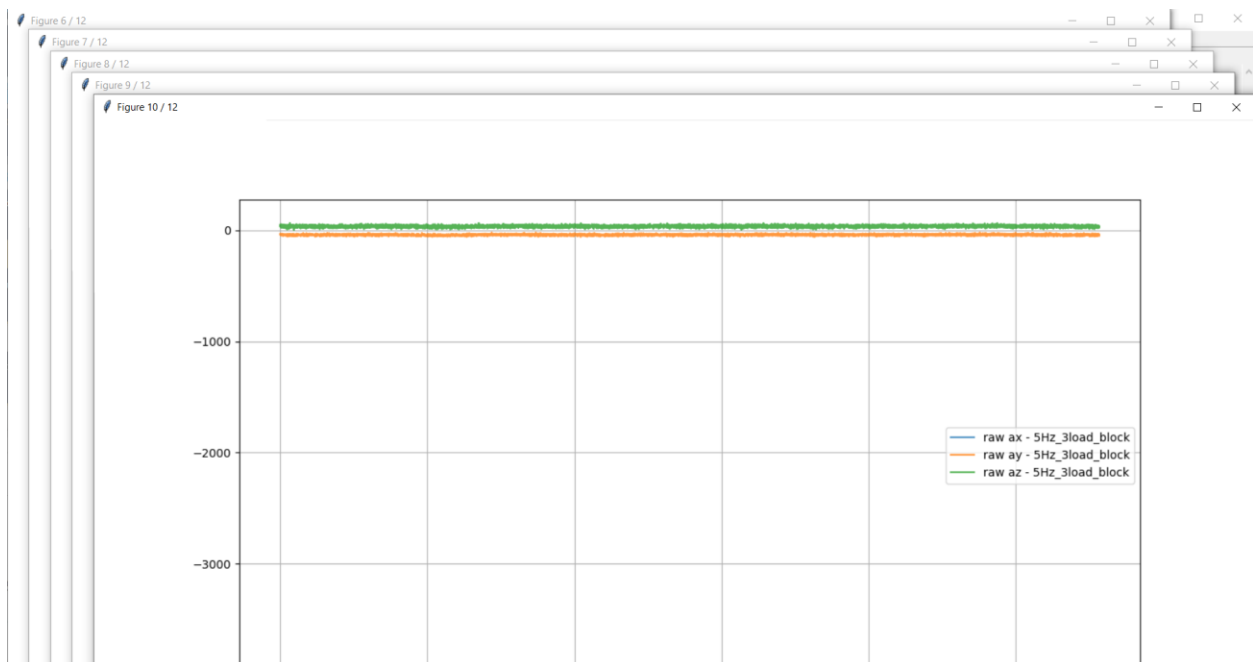


Fig 19. Visualization of Machine Data

Feature Extraction and Training

We extract all six features from the raw data and train the model with random forest classifier, both testing accuracy and training accuracy are good. So we are good to start predicting online.

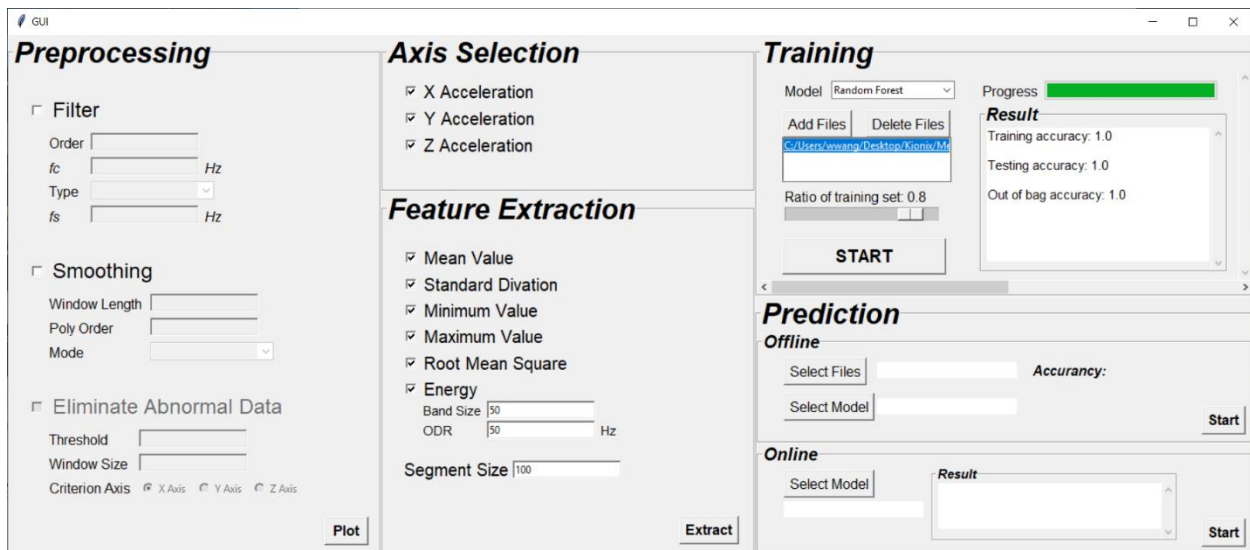
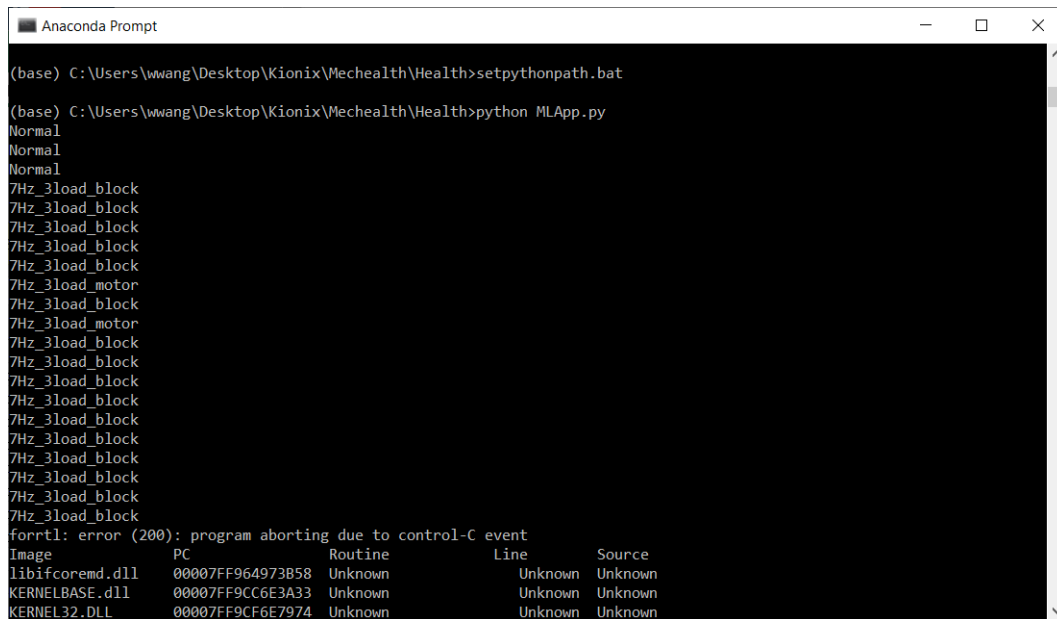


Fig 20. Feature Extraction and Training

Predict Online



```
(base) C:\Users\wwang\Desktop\Kionix\Mechhealth\Health>setpythonpath.bat

(base) C:\Users\wwang\Desktop\Kionix\Mechhealth\Health>python MLApp.py
Normal
Normal
Normal
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_motor
7Hz_3load_block
7Hz_3load_motor
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
7Hz_3load_block
forrtl: error (200): program aborting due to control-C event
Image                PC                Routine              Line        Source
libifcoremd.dll      00007FF964973B58  Unknown              Unknown     Unknown
KERNELBASE.dll       00007FF9CC6E3A33  Unknown              Unknown     Unknown
KERNEL32.DLL         00007FF9CF6E7974  Unknown              Unknown     Unknown
```

Fig 21. Results in console

References

Random Forest

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>

SVM

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>

More supervised models

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

The Kionix Advantage

Kionix technology provides for X, Y, and Z-axis sensing on a single, silicon chip. One accelerometer can be used to enable a variety of simultaneous features including, but not limited to:

- Hard Disk Drive protection
- Vibration analysis
- Tilt screen navigation
- Sports modeling
- Theft, man-down, accident alarm
- Image stability, screen orientation & scrolling
- Computer pointer
- Navigation, mapping
- Game playing
- Automatic sleep mode

Theory of Operation

Kionix MEMS linear tri-axis accelerometers function on the principle of differential capacitance. Acceleration causes displacement of a silicon structure resulting in a change in capacitance. A signal-conditioning CMOS technology ASIC detects and transforms changes in capacitance into an analog output voltage, which is proportional to acceleration. These outputs can then be sent to a micro-controller for integration into various applications.

For product summaries, specifications, and schematics, please refer to the Kionix MEMS accelerometer product catalog at:

<http://www.kionix.com/parametric/Accelerometers>