Physical Therapy Modeling

Ethan Straub, Liam Hayes

Data

Session 1	Session 2	Session 3	Session 4	Session 5
Exercise 1	Exercise 1	Exercise 1	Exercise 1	Exercise 1
Exercise 2	Exercise 2	Exercise 2	Exercise 2	Exercise 2
Exercise 3	Exercise 3	Exercise 3	Exercise 3	Exercise 3
Exercise 4	Exercise 4	Exercise 4	Exercise 4	Exercise 4
Exercise 5	Exercise 5	Exercise 5	Exercise 5	Exercise 5
Exercise 6	Exercise 6	Exercise 6	Exercise 6	Exercise 6
Exercise 7	Exercise 7	Exercise 7	Exercise 7	Exercise 7
Exercise 8	Exercise 8	Exercise 8	Exercise 8	Exercise 8

Data

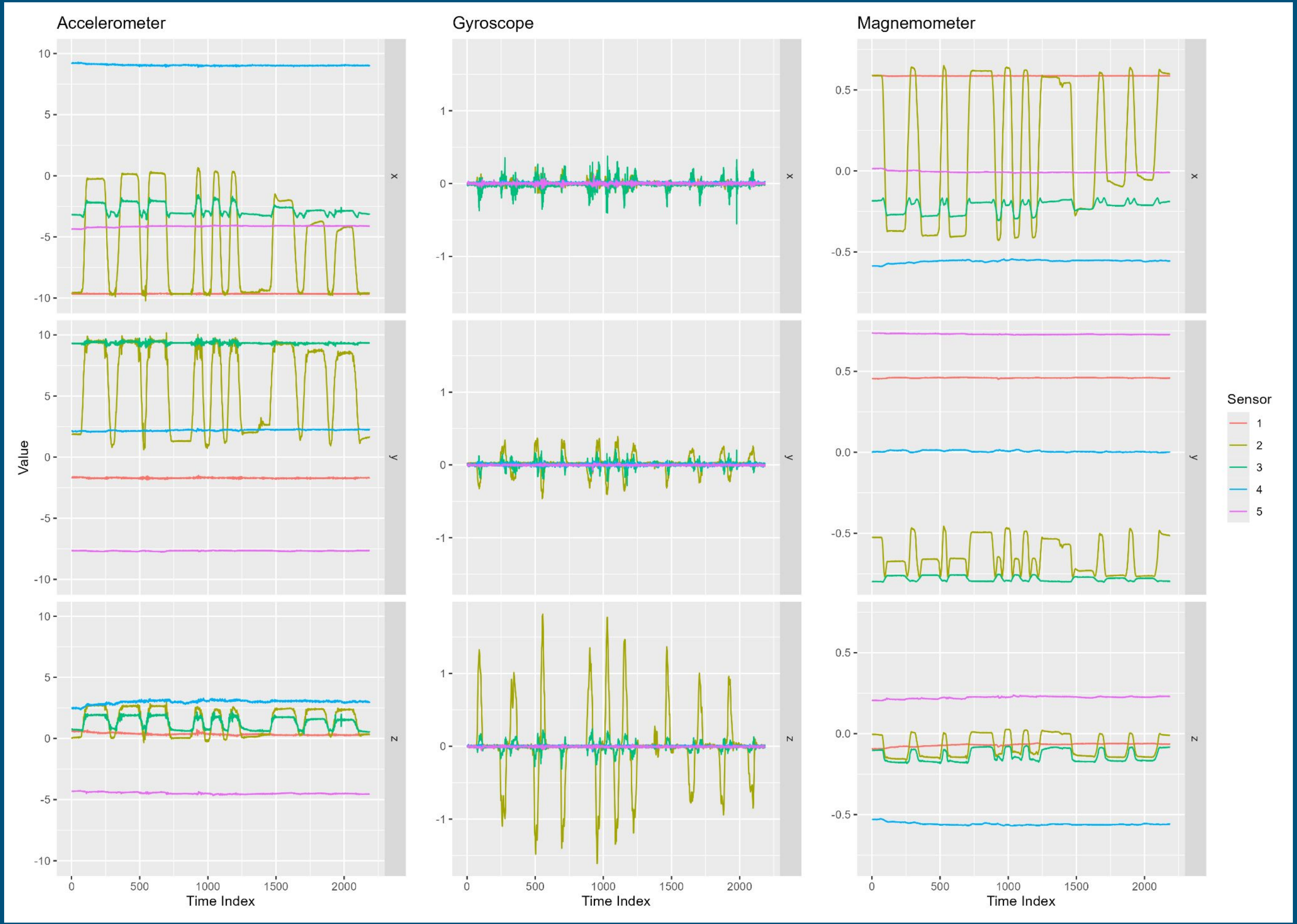
Session 1

Exercise 1

5 sensors

3 types of measurement per sensor

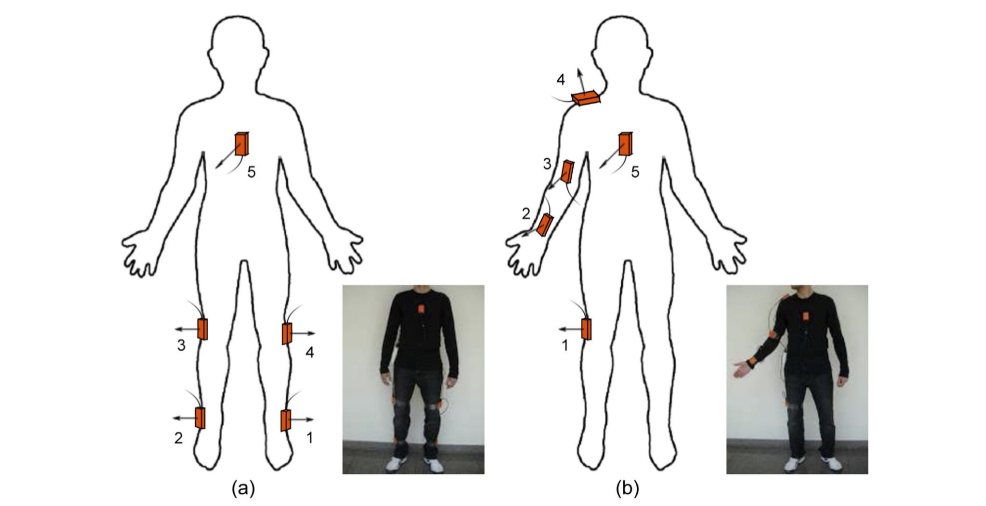
X, Y, and Z directions

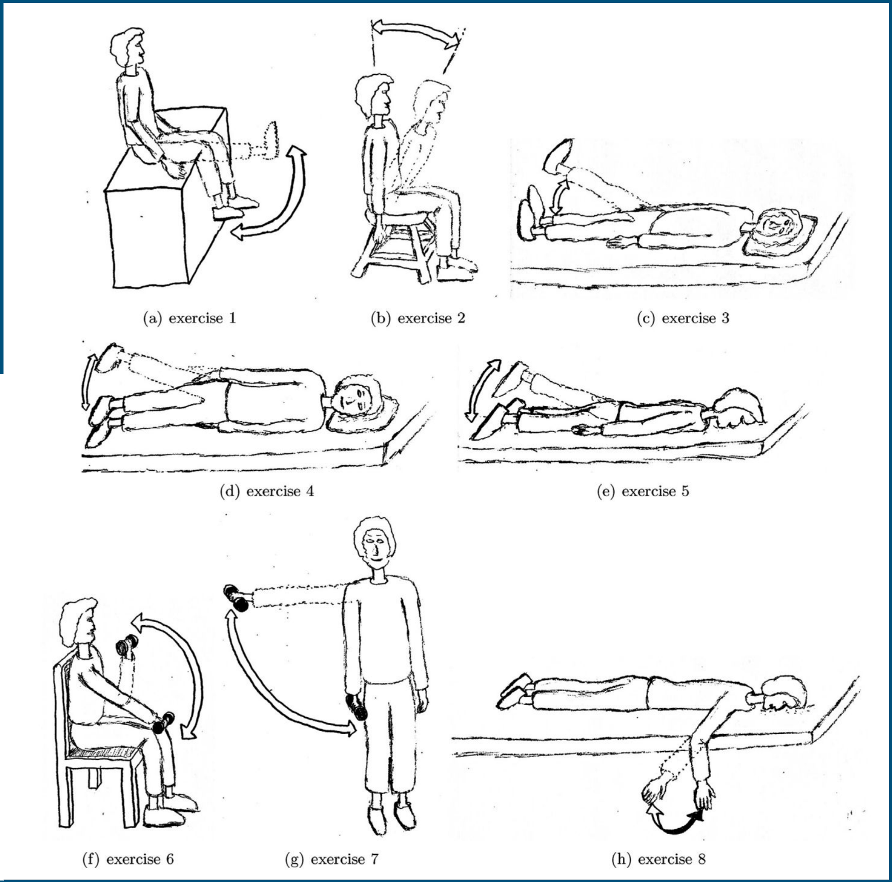


Sensors and Exercises

Right: Each of the 8 exercises in our dataset

Below: Two different sensor configurations depending on upper or lower body exercise





Research Questions

Can we correctly determine what exercise someone is doing?

Can we eliminate some sensor(s) or time spent performing the exercise and still accurately classify the exercise?

Modeling Methods

1. Machine learning methods (Random Forests, SVM, KNN) using fitted regression model parameters)
2. Use Dynamic Time Warping (DTW) to identify and classify exercises.

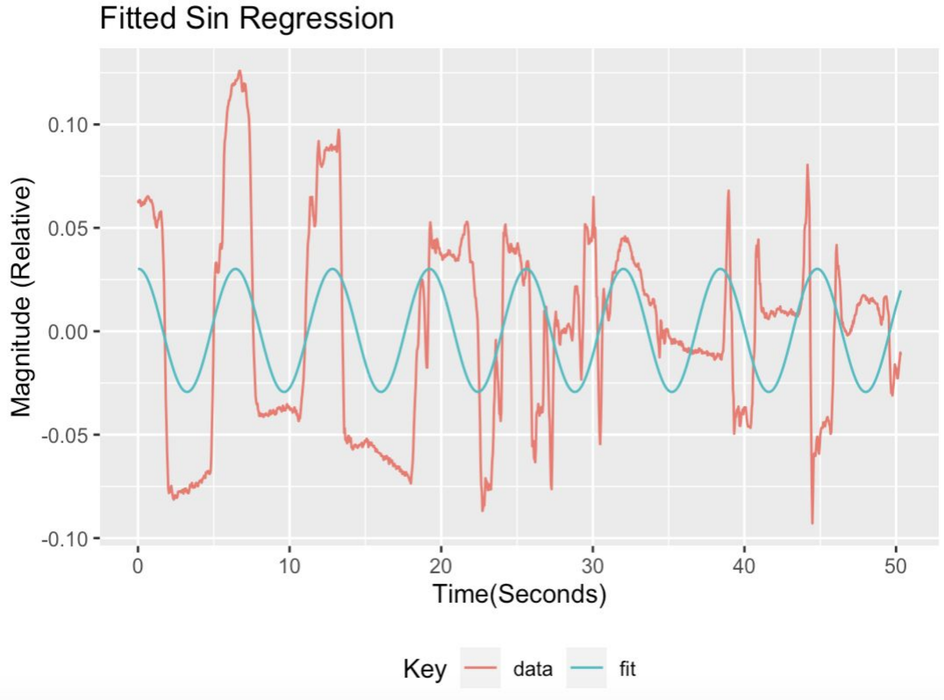
Method 1

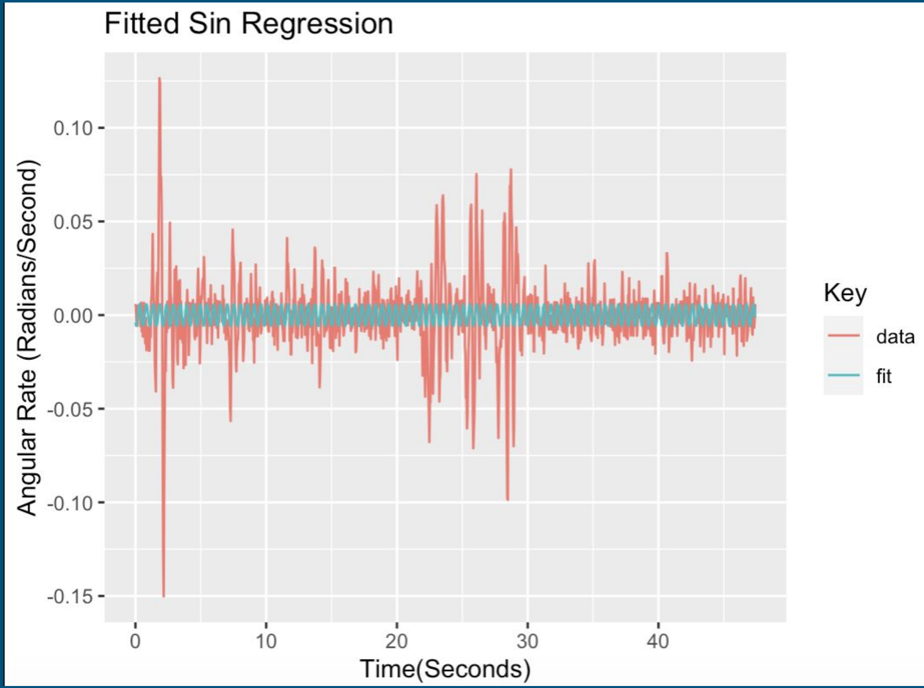
Goal: Create a feature matrix (matrix of predictor variables) with some kind of information from each series that a classifier can use to differentiate between the exercises

Idea 1: Use the amplitudes of sin regression models fitted to a detrended version of each series

Idea 2: Use the variance of each series

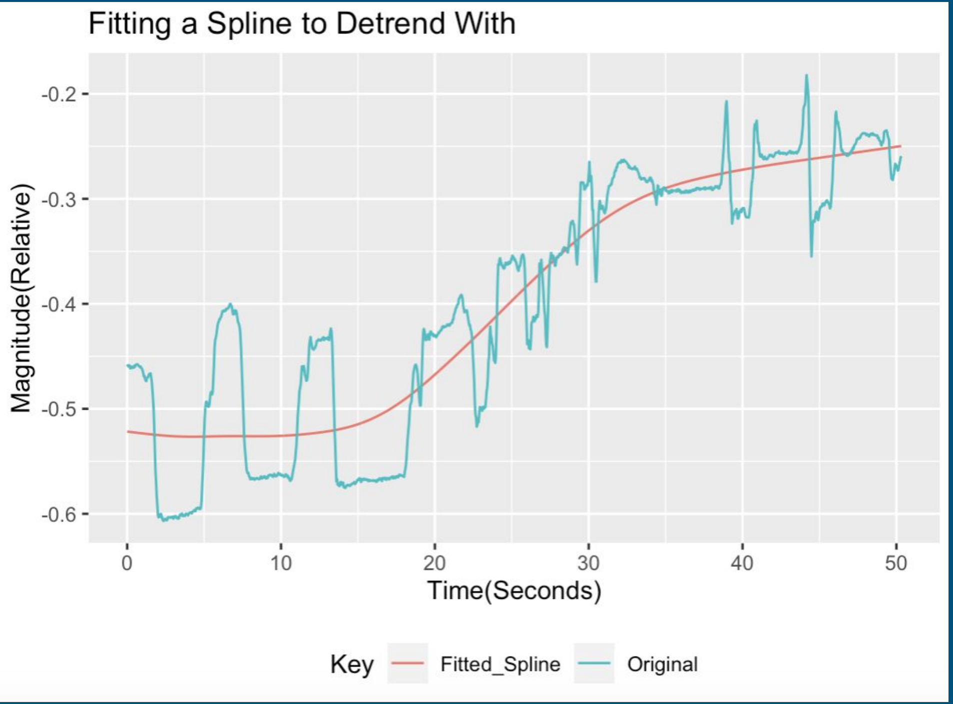
Fitting Sin Regression to the Detrended Series

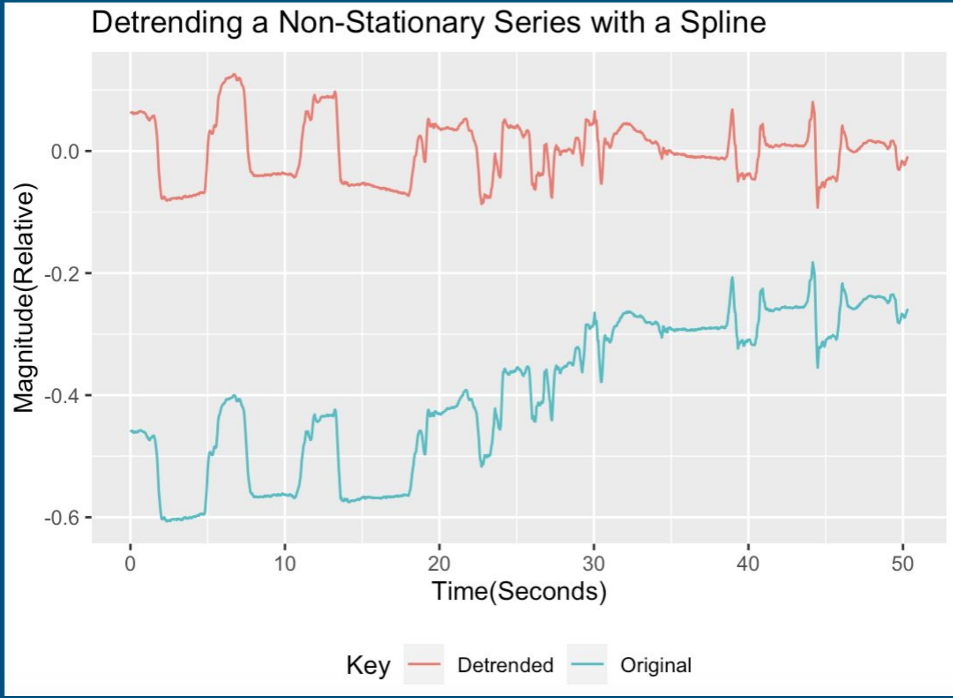




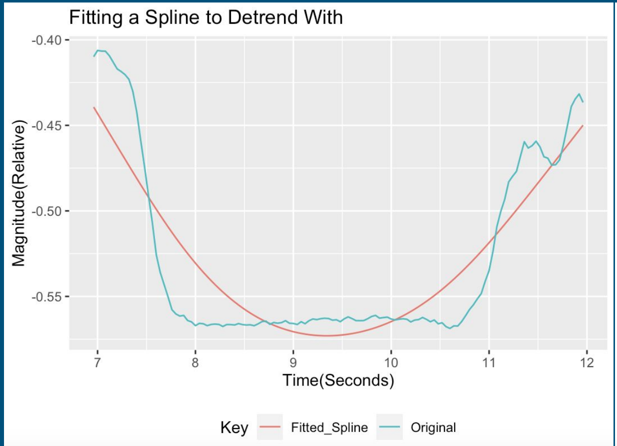
Achieving a Constant Mean (Detrending)

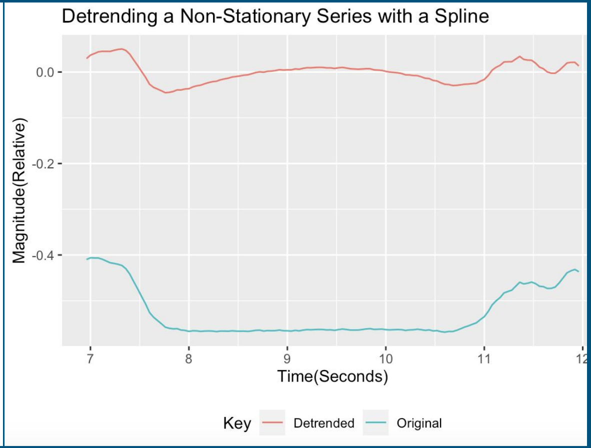
Good Fit (lambda = 0.05)

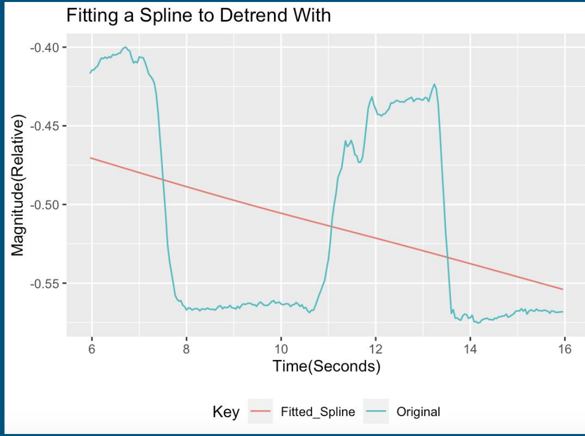


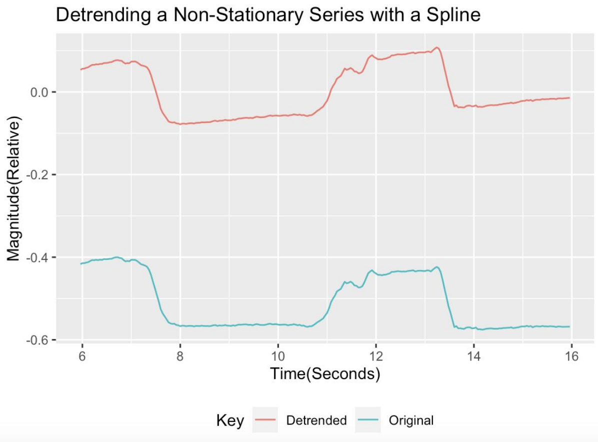


Issues with shorter series









Solution: If a series is less than 15 seconds, do not detrend it.

A Glimpse of the Inputs to the Classifiers

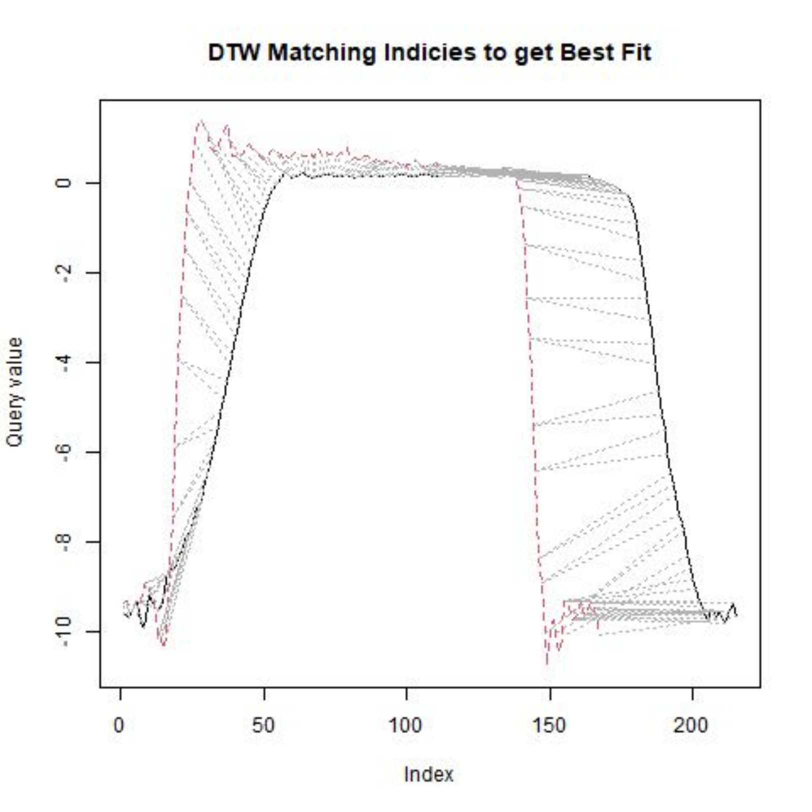
Not a variable. 5 sessions = 5 fold cross validation. Since we are trying to evaluate the classifier's performance on new people. The classifier should not be evaluated with a person that it was trained on.

Ex: Amplitude of the fitted sin regression model of a detrended series from session 1, exercise 3, sensor 1, angular rate in the X direction.

		Sensor 1			
		Acceleration			Angular Rate
session	exercise	x	y	z	x
1	1	0.003	0.009	0.022	0.001
1	2	0.007	0.015	0.028	0.004
1	3	0.028	0.012	0.069	0.005
1	4	0.009	0.052	0.021	0.005
1	5	0.096	0.103	0.035	0.013
1	6	0.003	0.001	0.009	0.001

Dynamic Time Warping

- Algorithm that matches each point on a signal to a point on a different signal by “warping time”.
- Useful for aligning signals.
- Computes a distance measure.



Classify With Dynamic Time Warping

1. Find the average signal for each 8 exercise types.
2. Compare an exercise to each average exercise using DTW.
3. Classify based on the lowest distance measure.

Computing the Average Exercise

Steps:

1. Align the signals.
2. Take the average at each index.

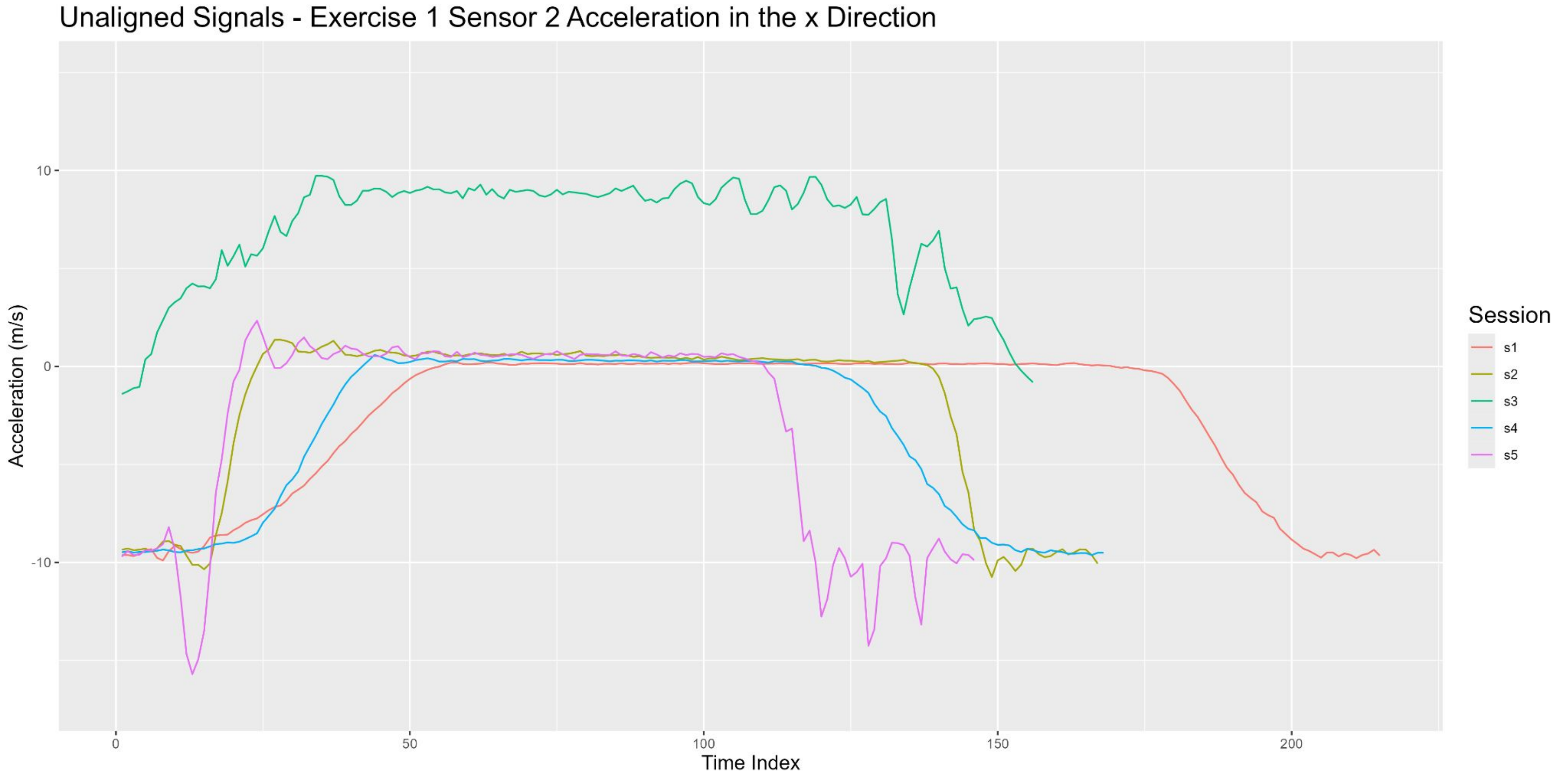
For each exercise:

5 sensors
3 measurements per sensor
3 directions per measurement

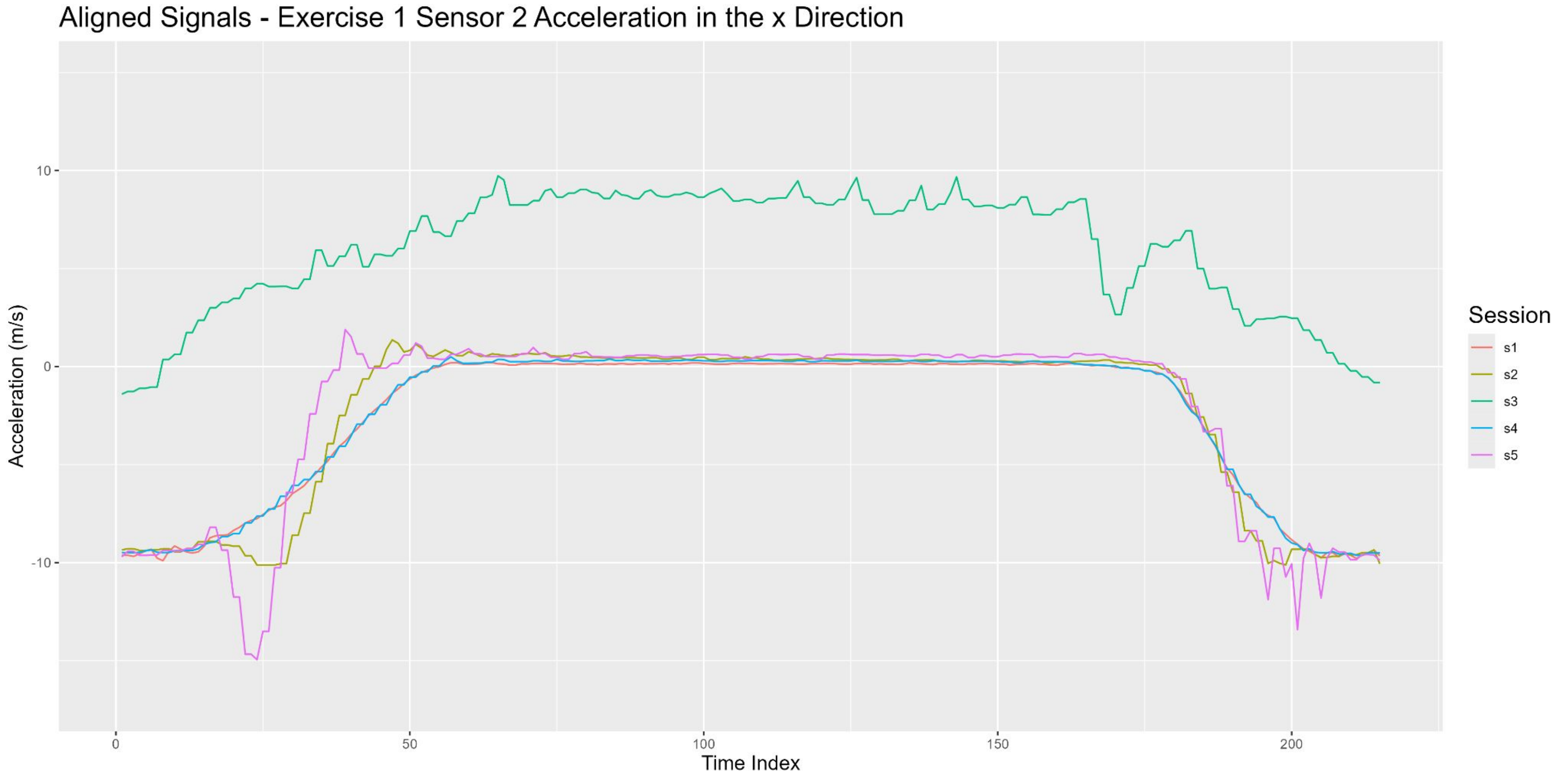
This means we have 5 x 3 x 3 = 45 individual signals for each exercise.

We compute the average of each of these.

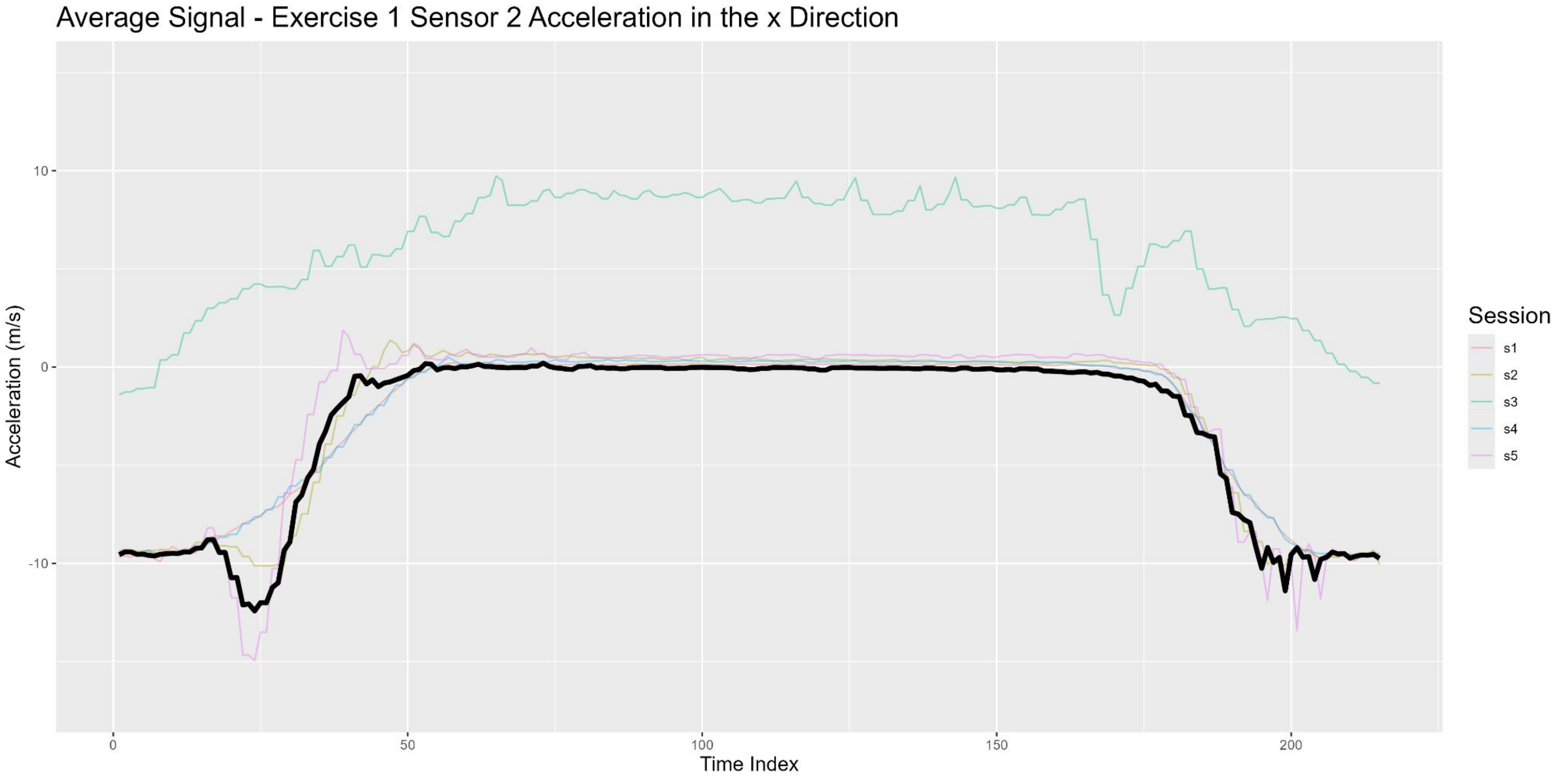
Computing the Average Exercise



Computing the Average Exercise



Computing the Average Exercise



Classifying an Exercise

We want to classify an exercise *e*.

1. Select the *n* highest variance signals to compare.
2. Compare the *n* highest variance signals from *e* to the corresponding signals from each 8 average exercises using DTW.
3. Classify as the exercise with the lowest total DTW distance measure

Finding When an Exercise is Happening

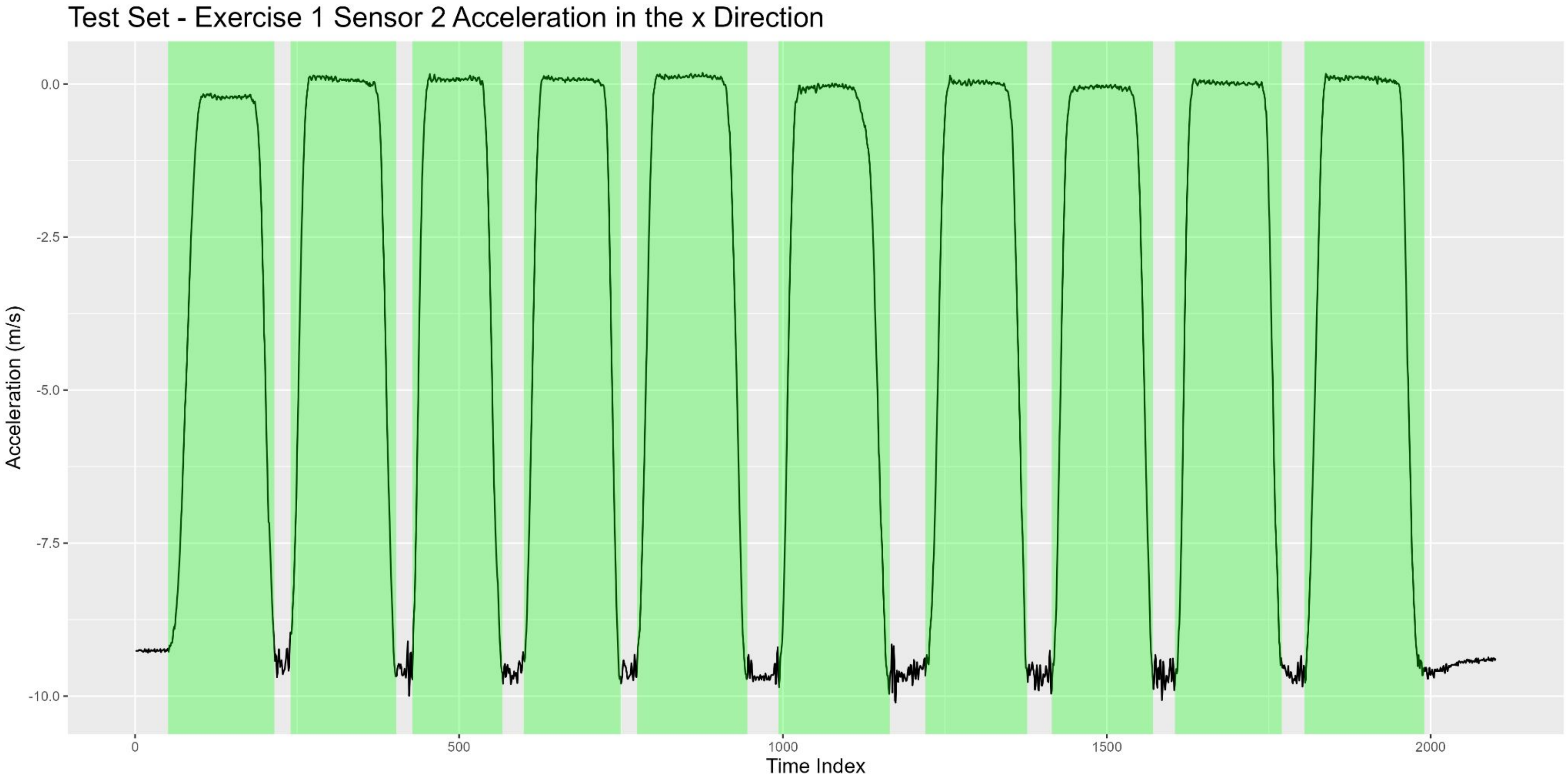
For each average exercise:

1. Compare against a sliding window of the same size across the session.
2. Store the distance measure for each window in a vector.

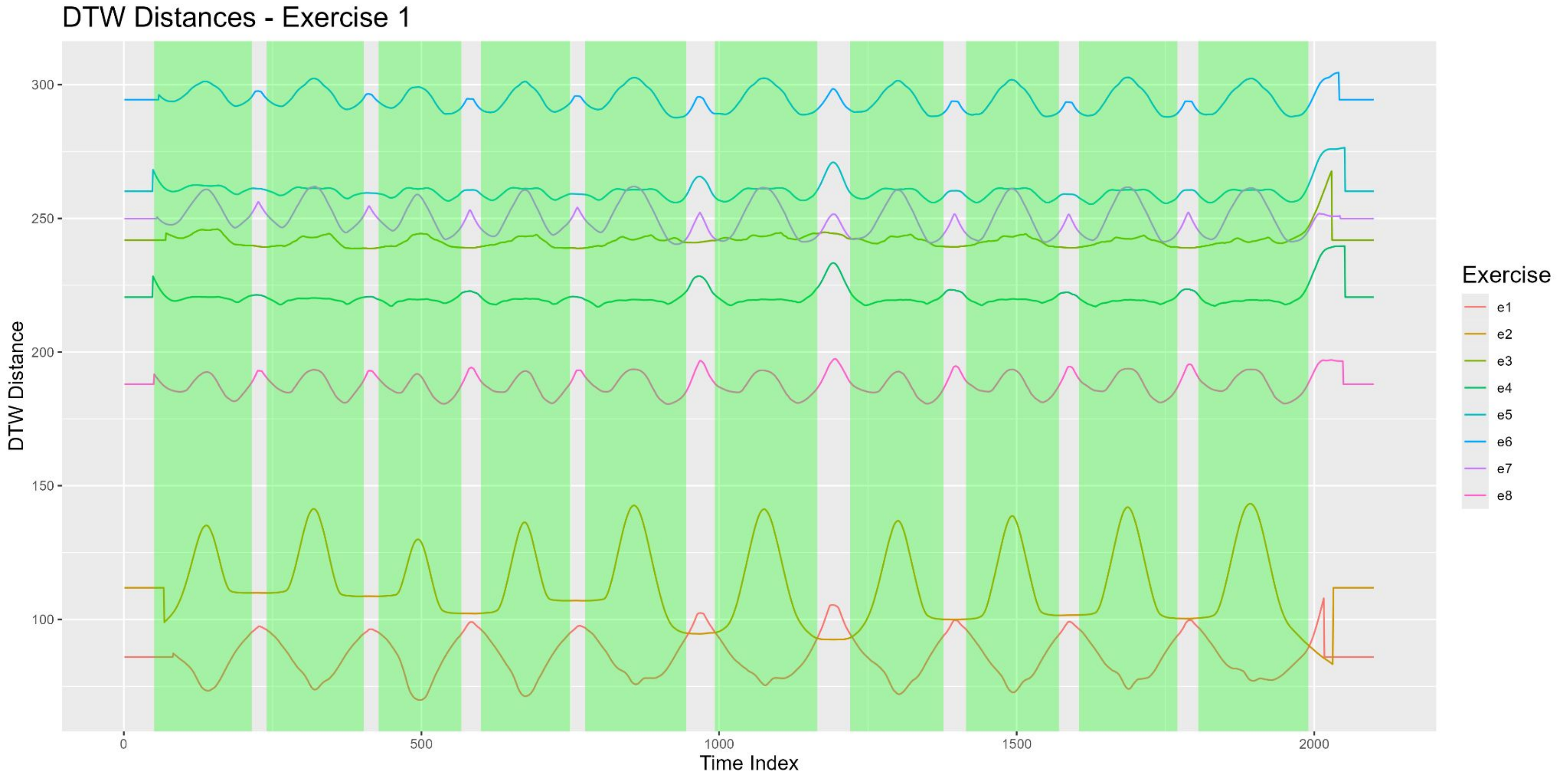
Once we have the 8 distance measure vectors:

1. Select the vector with the lowest mean.
2. Find the local minima of that vector.

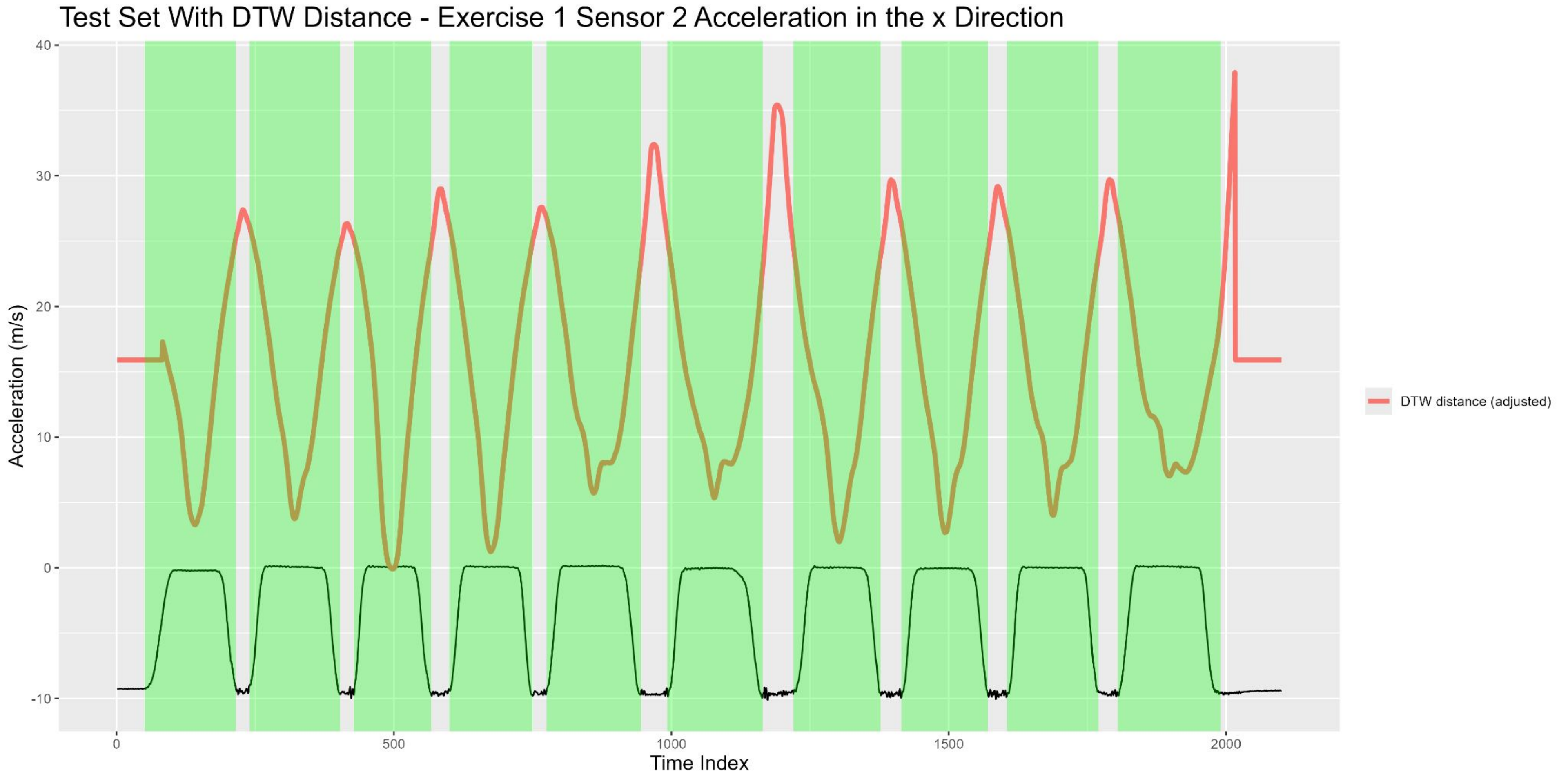
Finding When an Exercise is Happening



Finding When an Exercise is Happening



Finding When an Exercise is Happening



Testing and Results

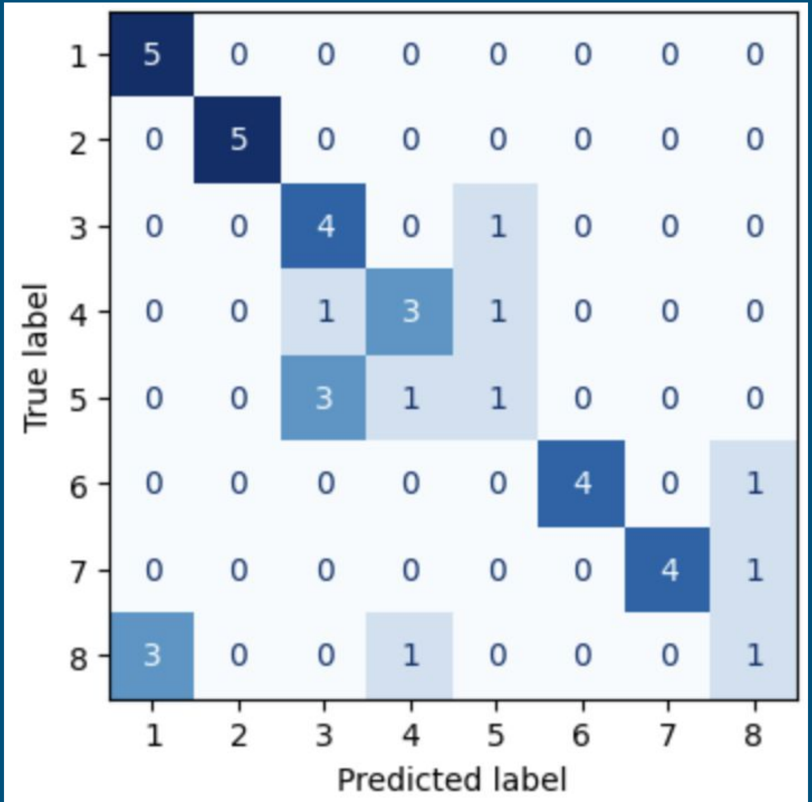
K Nearest Neighbors

Estimated Accuracy = 0.675

Parameter Grid

weights	k	p
uniform	1	1
uniform	1	2
uniform	3	1
uniform	3	2
uniform	7	1
uniform	7	2
distance	1	1
distance	1	2
distance	3	1
distance	3	2
distance	7	1
distance	7	2

Confusion Matrix



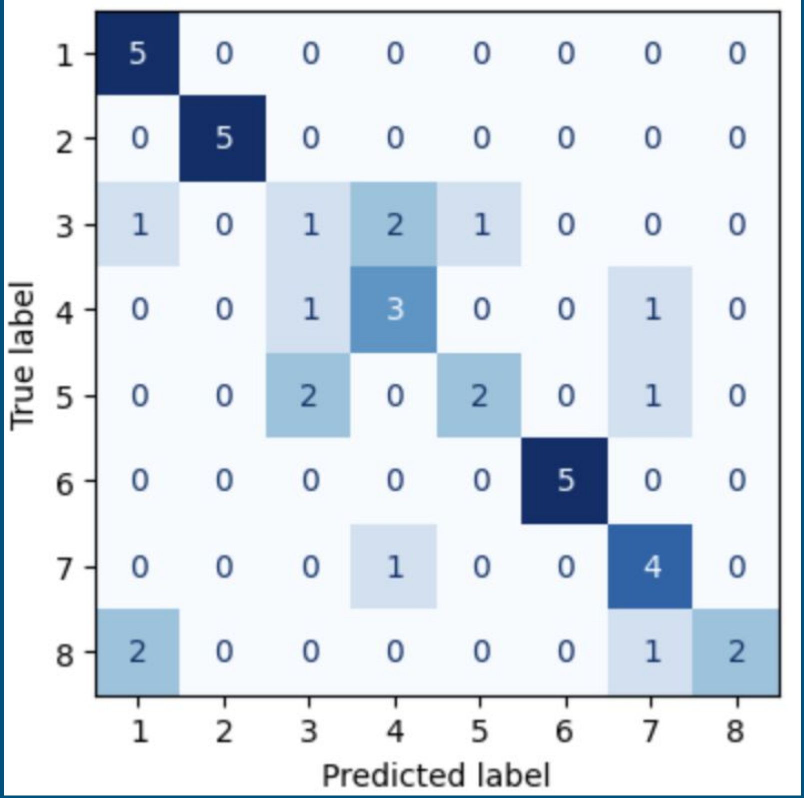
Support Vector Machines

Estimated Accuracy = 0.675

Parameter Grid

C	kernel	gamma
0.1	linear	NA
1	linear	NA
10	linear	NA
100	linear	NA
1000	linear	NA
1	rbf	0.001
1	rbf	0.1
10	rbf	0.001
10	rbf	0.1
100	rbf	0.001
100	rbf	0.1

Confusion Matrix



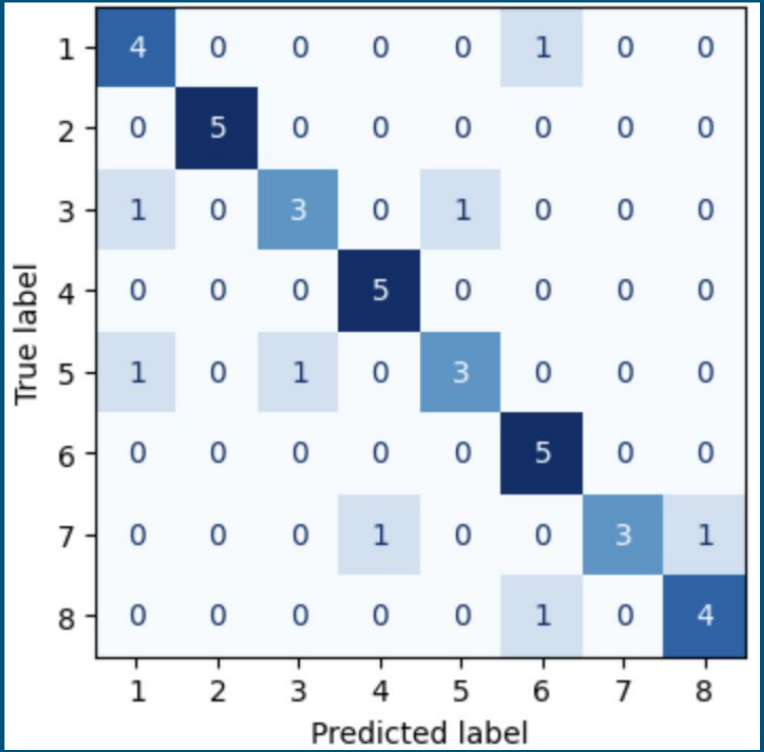
Random Forests

Estimated Accuracy = 0.8

Parameter Grid

```
{'n_estimators': [10, 20, 50, 100]}
```

Confusion Matrix



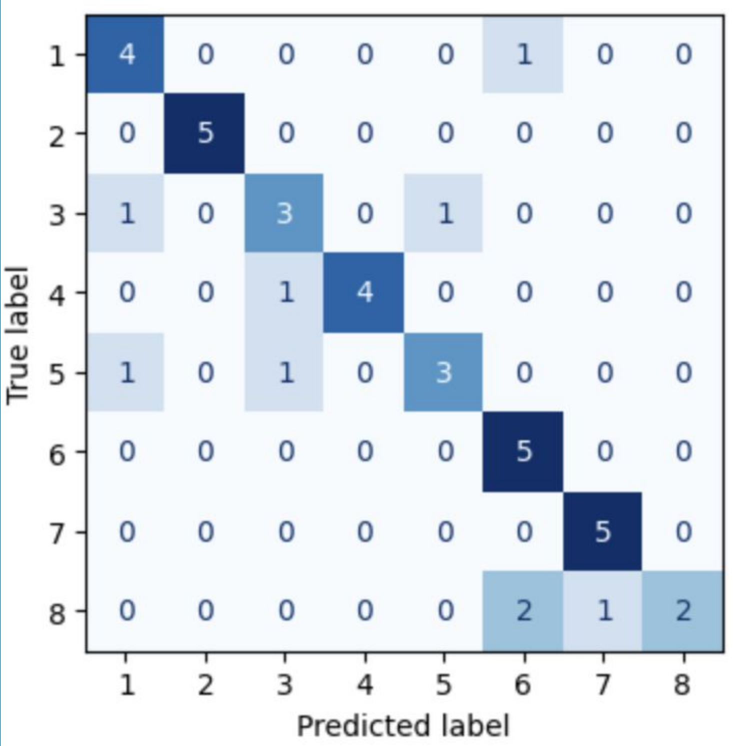
Random Forests using Variances

Variance of Session 1, exercise 2, sensor 1, acceleration in the y direction

		Sensor 1			
		Acceleration			Angular Rate
session	exercise	x	y	z	x
1	1	0.000	0.002	0.009	0.000
1	2	0.001	0.002	0.006	0.000
1	3	0.005	0.001	0.017	0.001
1	4	0.001	0.039	0.024	0.001
1	5	0.037	0.074	0.010	0.003
1	6	0.000	0.000	0.001	0.000

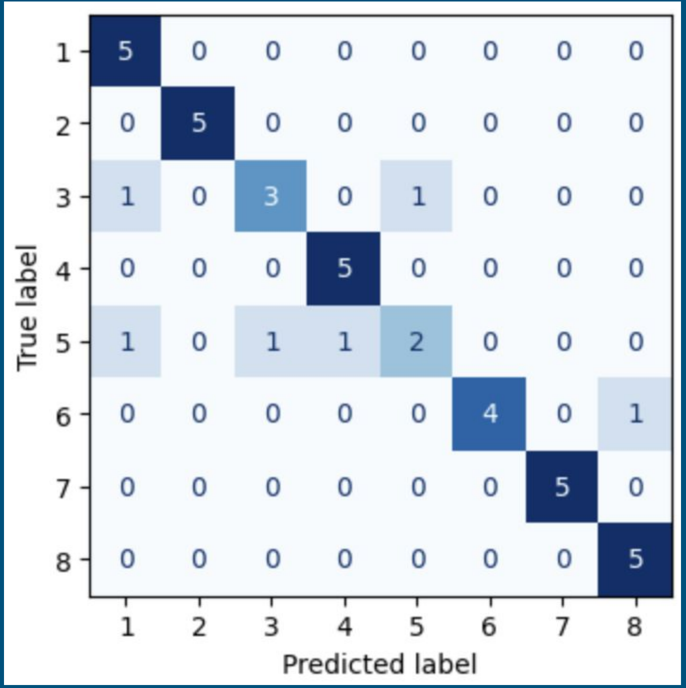
Estimated Accuracy = 0.775

Confusion Matrix



Random Forests Best Classifier

Confusion Matrix



			Magnitude Sensor 4		Acceleration Sensor 1	
Session	Exercise		y	z	x	y
1	1	. . .	0.002	0.002	0.000	0.000
1	2		0.000	0.000	0.007	0.015
1	3		0.004	0.006	0.000	0.000
1	4		0.003	0.010	0.000	0.000
1	5		0.003	0.002	0.000	0.000
1	6		0.000	0.000	0.003	0.001

Estimated Accuracy = 0.875

Trying other methods

Principal Component Analysis

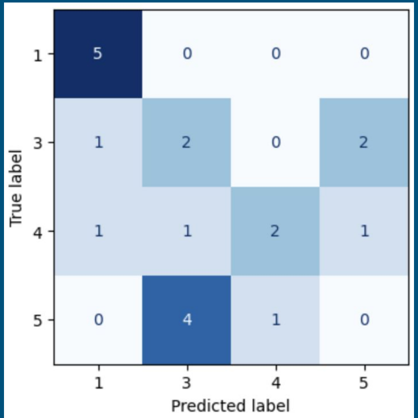
Random Forests

```
Number of principal components:  1
Train accuracy:  1.0
Test accuracy:  0.375
Number of principal components:  2
Train accuracy:  1.0
Test accuracy:  0.625
Number of principal components:  4
Train accuracy:  1.0
Test accuracy:  0.75
Number of principal components:  6
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  8
Train accuracy:  1.0
Test accuracy:  0.625
Number of principal components:  10
Train accuracy:  1.0
Test accuracy:  0.75
Number of principal components:  12
Train accuracy:  1.0
Test accuracy:  0.75
Number of principal components:  14
Train accuracy:  1.0
Test accuracy:  0.75
Number of principal components:  16
Train accuracy:  1.0
Test accuracy:  0.625
Number of principal components:  18
Train accuracy:  1.0
Test accuracy:  0.625
```

SVM

```
Number of principal components:  1
Train accuracy:  0.5
Test accuracy:  0.375
Number of principal components:  2
Train accuracy:  0.59375
Test accuracy:  0.5
Number of principal components:  4
Train accuracy:  0.96875
Test accuracy:  0.375
Number of principal components:  6
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  8
Train accuracy:  1.0
Test accuracy:  0.625
Number of principal components:  10
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  12
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  14
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  16
Train accuracy:  1.0
Test accuracy:  0.5
Number of principal components:  18
Train accuracy:  1.0
Test accuracy:  0.5
```

Lower Body Exercises Only



Time or Sensor Restrictions

Description	Test_Accuracy
10 Seconds	0.725
20 Seconds	0.725
Sensor 2 Only	0.600
Sensors 2,3, and 5	0.675
Acceleration Only	0.725
Angular Rate Only	0.700
Magnitude Only	0.675

DTW Classification

Tuning Parameters

1. Smoothing window (w)
2. Number of signals (n)

Testing Method

Train the model on sessions 2-5.

Test on session 1.

Try to minimize n while maintaining accuracy.

DTW Classification Results

Using LOOCV, we found that w=10 and n=28 minimizes n while maximizing accuracy.

On the test set, this gave us 100% accuracy for classifying individual exercises of any type (correctly performed, quickly performed, and low amplitude).

DTW Finding When

Testing Method

Use the classification model with w=10 and n=28 in the algorithm for finding when. Train on sessions 2-5 and test on session 1

If the local minima is:

1. Within the range of when the exercise is happening, classify as correct.
2. Outside of the range of when the exercise is happening, classify as incorrect.
3. Within the range of an exercise that already has a local minima assigned to it, classify as incorrect.

DTW Finding When Results

On the test set:

1. 75.63% accuracy overall.
2. 98.75% accuracy on correctly performed exercises.
3. 81.25% accuracy on exercises performed too quickly.
4. 45% accuracy on low amplitude exercises.

Conclusions

- We are able to classify exercises based on motions sensors with very good accuracy.
- Finding when/how many reps of an exercise someone is doing is slightly trickier but still reasonably accurate.
- In order for this to be used in practice for real physical therapy patients, we would need a way to score *how well* they did the exercise. Then we could provide helpful feedback to the patient and compute an effectiveness score of the session.

References

Yurtman, A. Billur, B (2014). “Automated Evaluation of Physical Therapy Exercises Using Multi-Template Dynamic Time Warping on Wearable Sensor Signals” *Elsevier Health*

Van Boxtel, G. (2021). “Signal Processing in R”

Giorgino, T. (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package” *Journal of Statistical Software*