Data Analysis of 3 EMG sensors using MATLAB and R

Hector Estrella
Embedded Electrical and Computer Systems

San Francisco State University,
December 2019

SF State

Agenda

- Project's motivations
- Hardware: 3 EMG sensors and Arduino Microcontroller Unit (MCU)
- Software: Using Data Analysis Techniques
- Software: Using Feature Extraction
- Software: Classification Learner
- Software: Using R to find fastest compilation time by using least amount of training data
- Results
- Summary
- References
- Questions

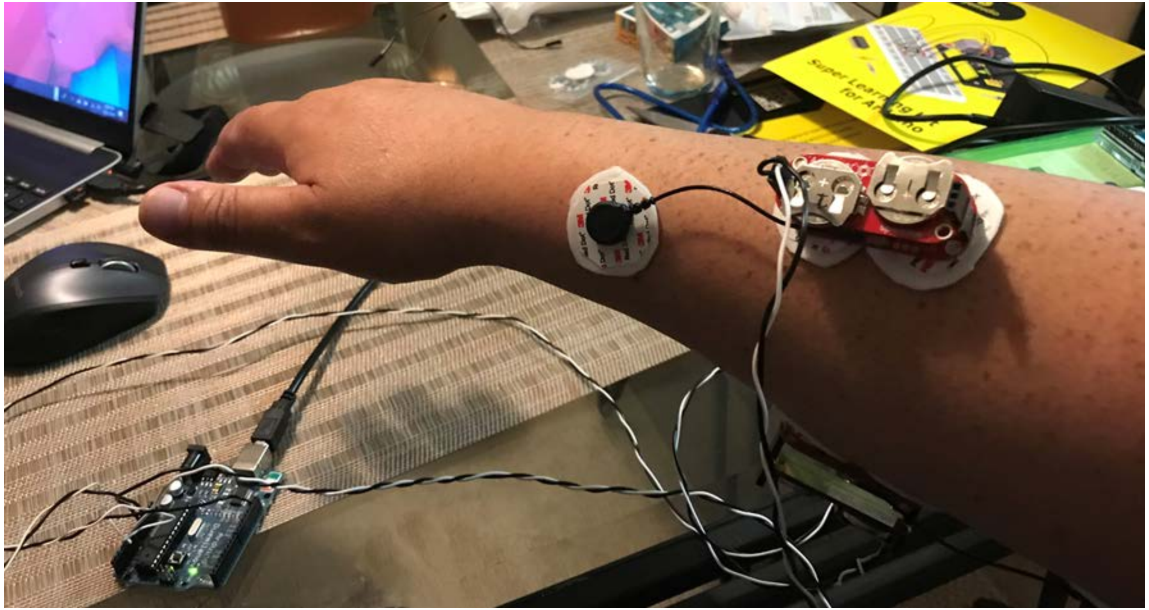
Project’s Motivations

- EMG sensors have been used to track hand movements (Chen).
- When analyzing EMG sensor data, feature extraction has been previously used (Phinyomark).
- MATLAB can be programmed to use feature extraction.
- Classification Learner in MATLAB uses Data Analysis Techniques.
- Classification Learner is used to train and test data using various Data Analytical Techniques.
- Run a feature extraction program in MATLAB with Classification Learner and get 100% recognition of the hand movements in the fastest time.
- R will be used to find the lowest amount of training data needed to get close to 100% testing results.

Hardware: 3 EMG sensors, 3 batteries, Arduino MCU, laptop

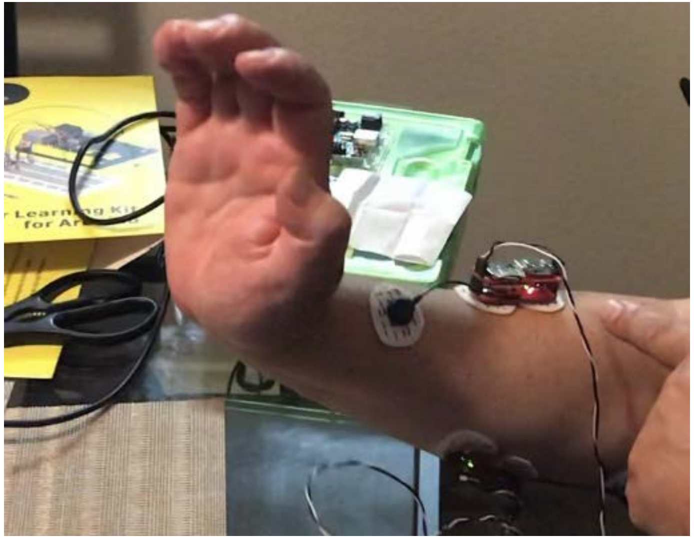
- Front EMG sensor: located in the Extensor Digitorum region.
- Back EMG sensor: located in Brachioradialis region.
- Biceps EMG sensor: located in Biceps Brachii region.
- Sensor pads to connect to muscle and ground (bone).
- Battery for each sensor.
- Sensor connected to inputs and ground of Arduino.
- Arduino connected to laptop.

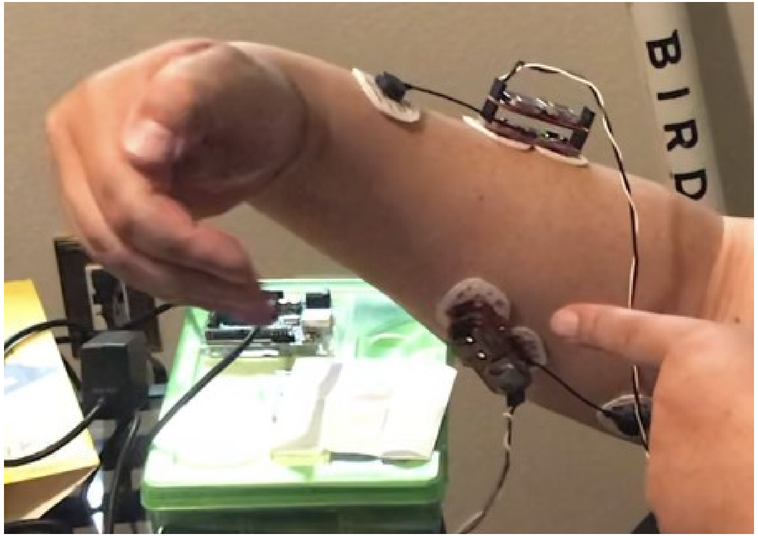
Rest Motion: Keep Hand at Rest. No movement.
The “Front” EMG sensor is in the Extensor Digitorim muscle region.



Wrist Extension: Moving the wrist from down to upward.

Supination: Moving the right wrist clockwise.
The “Back” EMG sensor is in the Brachioradialis muscle region.

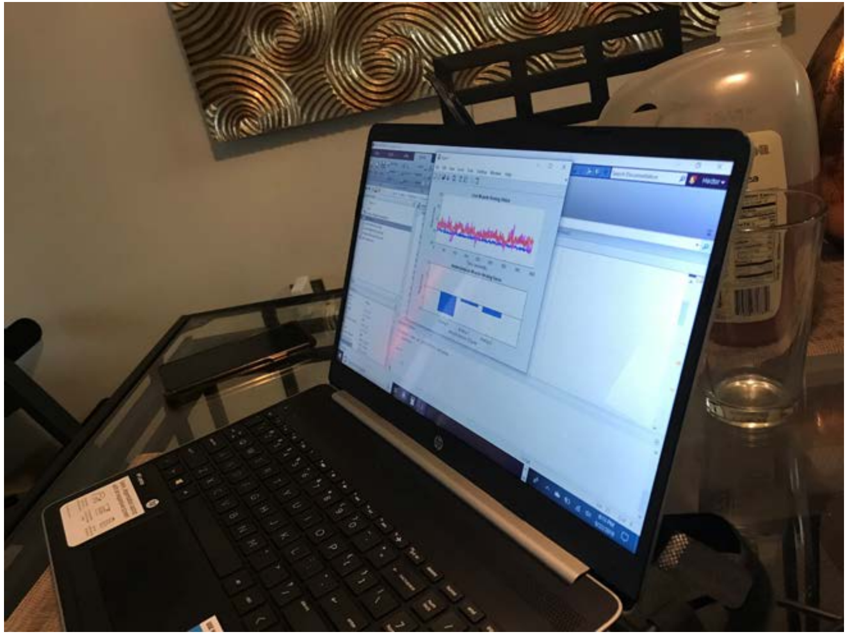


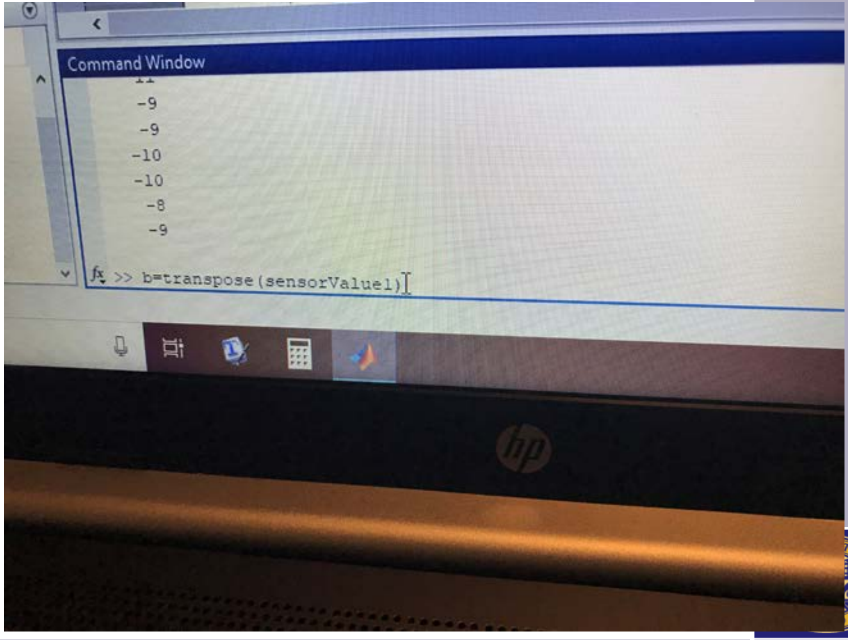


Bicep Flex: Moving the right forearm upward.
The “Biceps” EMG sensor is in the Biceps Brachii muscle region.

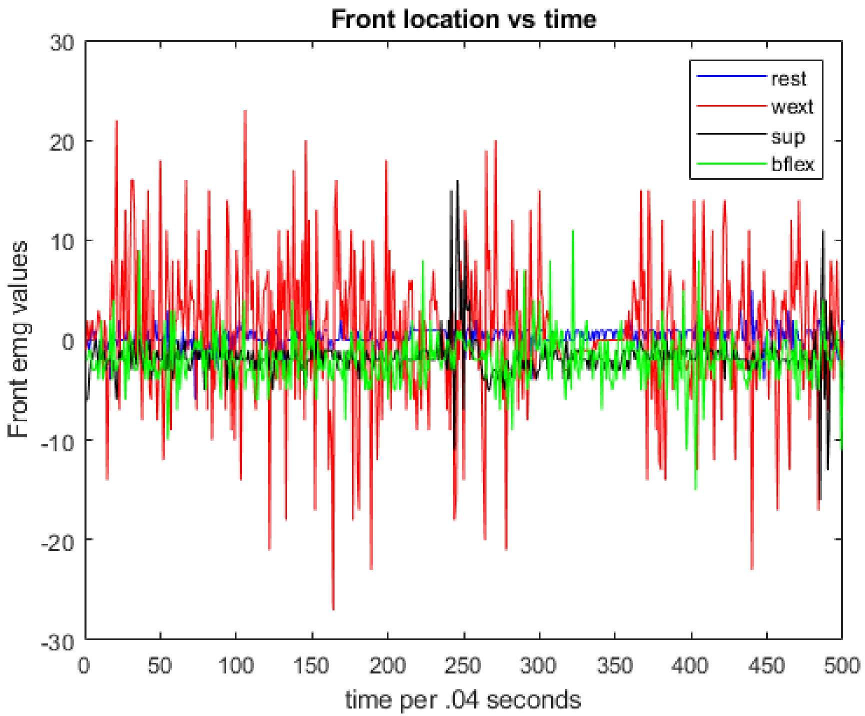
SF STATE

Example of Data Collection on the “Front” EMG.

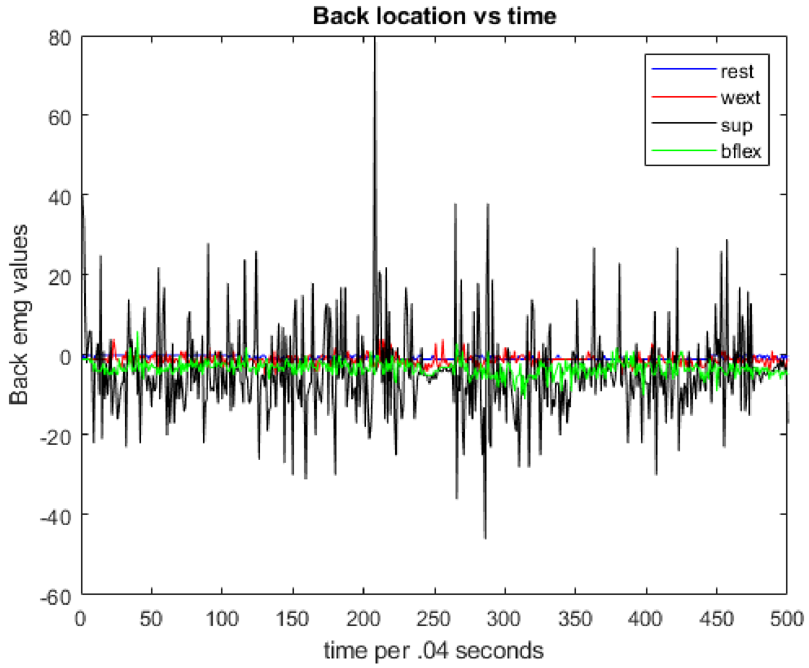




Front EMG Sensor: Located in Exterior Digitorum. Wrist Extension has highest EMG values.

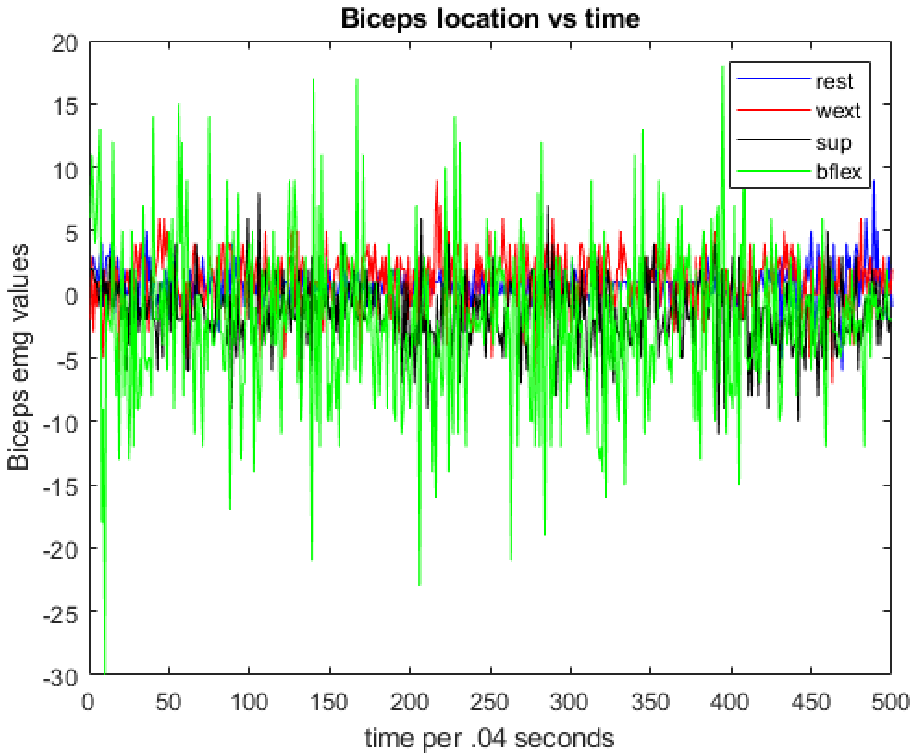


Back EMG Sensor: Located in Brachioradialis. Supination has highest EMG values.



SF STATE

Biceps EMG Sensor: Located in Biceps Brachii. Biceps Flex has highest EMG values.



Feature Extraction

- There are 2 training and 1 testing sets of data with 500 points each for 1500 total data points for each EMG sensor (Biceps, Front, Back).
- Raw data is segmented by Analysis Windows into Window Length (WL) and Window Increment (Winc).
- Examples will be shown.
- The observations will have 12 features which will be our predictors.

SF State

Method

- There were 3 trials for Rest, Wrist Extension (Wext), Supination (Sup), and Biceps Flex (Bflex).
- 500 values of each EMG sensor data were collected for each hand movement at a frequency of 25 Hz.
- Once the data was collected, 2 trials were initially set as training data and the last trial as testing data.
- Matlab was used to do Feature Extraction for the data. The Window Length (WL) and Window Increment (Winc) could be changed to lower the number of observations to make the program run faster.
- Initially, WL = 100 and Winc = 2.
- Winc was increased to 5, 10, and finally 20.
- R could be used to find the lowest % of training data that will give 100% testing results.
- By finding the limit, we could find the fastest time to train and test our data.

3 EMG sensors (Biceps, Front, Back) and 4 features (MAV, Waveform Length, Zero crossings and number of slope sign changes) for each sensor gives 12 total features.

Mean absolute value (MAV) $\mathrm{MAV} = \frac{1}{N}\sum_{n=1}^{N}\left|x_n\right|$

Waveform length (WL) $\mathrm{WL} = \sum_{n=1}^{N-1}\left|x_{n+1} - x_n\right|$

Zero crossing (ZC)

$$\mathrm{ZC} = \sum_{n=1}^{N-1}\left[\operatorname{sgn}\left(x_n \times x_{n+1}\right) \cap \left|x_n - x_{n+1}\right| \geq threshold\right];$$

$$\operatorname{sgn}(x) = \begin{cases} 1, & if\ x \geq threshold \\ 0, & \text{otherwise} \end{cases}$$

Slope Sign Change (SSC)

$$\mathrm{SSC} = \sum_{n=2}^{N-1}\left[f\left[\left(x_n - x_{n-1}\right)\times\left(x_n - x_{n+1}\right)\right]\right];$$

$$f(x) = \begin{cases} 1, & if\ x \geq threshold \\ 0, & \text{otherwise} \end{cases}$$

SF State

3 EMG sensors (Biceps, Front, Back) and 4 features (MAV, Waveform Length, Zero crossings and number of slope sign changes) for each sensor gives 12 total features. This is in R with the first 27 observations.

biceps.mav	biceps.len	biceps.zc	biceps.turns	front.mav	front.len	front.zc	front.turns	back.mav	back.len	back.zc	back.turns	obs	class
1.286400	5.951707	45.000000	33.000000	0.896800	4.922931	45.000000	27.000000	0.112800	1.296099	9.000000	3.000000	1.000000	1
1.122400	5.309440	47.000000	33.000000	0.783600	4.823210	38.000000	24.000000	0.112800	1.260976	8.000000	3.000000	2.000000	1
0.958200	4.493849	52.000000	33.000000	0.766400	4.548106	37.000000	21.000000	0.295200	1.379944	17.000000	4.000000	3.000000	1
0.862800	3.673528	54.000000	29.000000	0.713600	4.765658	39.000000	24.000000	0.455000	1.132527	19.000000	3.000000	4.000000	1
0.8504000	3.3190813	56.0000000	29.0000000	0.6372000	3.5888091	42.0000000	23.0000000	0.4998000	0.9813466	27.0000000	4.0000000	5.0000000	1
0.729200	2.756337	50.000000	28.000000	0.562000	2.980846	36.000000	24.000000	0.490200	1.059436	36.000000	10.000000	6.000000	1
0.735800	2.875202	49.000000	28.000000	0.565000	2.887283	36.000000	23.000000	0.466200	1.255252	43.000000	13.000000	7.000000	1
0.777000	3.172186	52.000000	29.000000	0.592800	1.889649	30.000000	18.000000	0.442200	1.339754	37.000000	10.000000	8.000000	1
0.747600	2.994547	56.000000	29.000000	0.628800	1.976283	30.000000	17.000000	0.435200	1.361285	36.000000	9.000000	9.000000	1
0.720000	2.945139	60.000000	29.000000	0.599000	1.738348	36.000000	13.000000	0.411800	1.377782	30.000000	6.000000	10.000000	1
0.700000	2.613342	61.000000	17.000000	0.597600	2.027459	42.000000	15.000000	0.364800	1.479990	28.000000	7.000000	11.000000	1
0.721400	2.752507	59.000000	25.000000	0.640000	2.033380	40.000000	6.000000	0.282200	1.469638	20.000000	6.000000	12.000000	1
0.668000	2.350665	53.000000	21.000000	0.640000	2.177857	49.000000	16.000000	0.268800	1.488802	20.000000	6.000000	13.000000	1
0.686000	2.670240	45.000000	22.000000	0.581400	1.705513	49.000000	12.000000	0.268800	1.488802	20.000000	6.000000	14.000000	1
0.650600	2.649684	39.000000	19.000000	0.603200	1.795706	47.000000	13.000000	0.255000	1.534412	21.000000	7.000000	15.000000	1
0.648400	3.000496	35.000000	20.000000	0.542000	1.369699	48.000000	20.000000	0.163800	1.495230	16.000000	7.000000	16.000000	1
0.638800	2.482896	34.000000	17.000000	0.495200	1.737359	57.000000	27.000000	0.112800	1.331221	10.000000	4.000000	17.000000	1
0.763200	3.289526	32.000000	20.000000	0.604000	2.488105	59.000000	28.000000	0.095000	1.340977	10.000000	5.000000	18.000000	1
0.945800	4.679170	37.000000	23.000000	0.729200	3.051001	60.000000	27.000000	0.255000	1.349608	14.000000	6.000000	19.000000	1

3 EMG sensors (Biceps, Front, Back) and 4 features (MAV, Waveform Length, Zero crossings and number of slope sign changes) for each sensor gives 12 total features. This is in R with the last 27 observations.

biceps.mav	biceps.len	biceps.zc	biceps.turns	front.mav	front.len	front.zc	front.turns	back.mav	back.len	back.zc	back.turns	obs	class
4.260000	15.263895	54.000000	29.000000	1.943000	9.969346	51.000000	33.000000	1.697600	8.702854	49.000000	27.000000	232.000000	4
4.240000	15.877855	50.000000	34.000000	1.846000	10.300472	50.000000	34.000000	1.743000	9.011027	49.000000	26.000000	233.000000	4
3.784000	15.736103	47.000000	33.000000	1.584000	8.858665	49.000000	33.000000	1.749800	9.354596	45.000000	27.000000	234.000000	4
4.069800	18.710839	50.000000	34.000000	1.516000	8.457500	50.000000	32.000000	1.604000	8.866755	42.000000	29.000000	235.000000	4
4.551000	20.918615	44.000000	36.000000	1.570800	7.892266	48.000000	28.000000	1.392800	8.003930	42.000000	27.000000	236.000000	4
4.718800	23.606602	39.000000	34.000000	1.539600	6.809195	46.000000	24.000000	1.280400	6.126561	38.000000	26.000000	237.000000	4
4.260000	23.235498	35.000000	31.000000	1.740000	8.376482	26.000000	22.000000	1.168000	6.159995	35.000000	24.000000	238.000000	4
4.020000	20.119178	51.000000	30.000000	1.777600	8.505434	50.000000	27.000000	1.285200	5.820634	47.000000	24.000000	239.000000	4
4.219400	17.769035	46.000000	28.000000	1.812000	8.986536	50.000000	30.000000	1.466000	5.599550	50.000000	21.000000	240.000000	4
4.16000	15.43301	45.00000	29.00000	1.68320	8.06915	55.00000	31.00000	1.44560	5.99724	50.00000	26.00000	241.00000	4
4.156000	15.340213	46.000000	27.000000	1.474200	7.961315	53.000000	30.000000	1.516400	6.864991	44.000000	26.000000	242.000000	4
3.984000	14.791245	42.000000	27.000000	1.253000	6.265779	43.000000	28.000000	1.530000	7.335208	41.000000	30.000000	243.000000	4
4.121200	15.099311	39.000000	29.000000	1.491400	6.646889	47.000000	30.000000	1.455200	6.843570	35.000000	28.000000	244.000000	4
4.030000	15.638966	45.000000	31.000000	1.884000	6.892399	46.000000	27.000000	1.514000	7.240658	30.000000	26.000000	245.000000	4
3.990400	14.533458	44.000000	30.000000	2.139000	8.018980	49.000000	29.000000	1.701600	7.802743	32.000000	25.000000	246.000000	4
4.46420	16.49822	41.00000	31.00000	2.49680	10.44932	51.00000	33.00000	1.74000	6.88522	36.00000	23.00000	247.00000	4
5.340000	21.532748	42.000000	32.000000	2.735200	12.724251	53.000000	33.000000	1.994000	7.646001	46.000000	23.000000	248.000000	4
5.914200	24.066072	43.000000	32.000000	2.630400	12.333797	53.000000	32.000000	2.126000	8.194127	47.000000	25.000000	249.000000	4
5.766000	23.696493	45.000000	34.000000	2.593800	12.152266	54.000000	31.000000	2.363400	8.844857	57.000000	25.000000	250.000000	4
5.664200	23.582306	49.000000	32.000000	2.603200	12.333740	54.000000	31.000000	2.188000	7.878246	60.000000	27.000000	251.000000	4
5.466000	23.214734	51.000000	35.000000	2.656000	11.714239	57.000000	31.000000	2.069800	7.609125	57.000000	27.000000	252.000000	4

Data Analysis: Linear Discriminant Analysis (LDA)

- We are trying to separate the classes (they are Rest, Wext, Sup, and Bflex).
- To separate the classes, we assume that observations are drawn from a Gaussian distribution and it makes estimates of the following mean and variance distributions:
- Mean: $\hat{\boldsymbol{u}}_{\boldsymbol{k}} = \frac{\mathbf{1}}{\boldsymbol{n}_{\boldsymbol{k}}} \sum_{\boldsymbol{i}:\boldsymbol{y}_{\boldsymbol{i}}=\boldsymbol{k}} \boldsymbol{x}_{\boldsymbol{i}}$
- Variance: $\hat{\boldsymbol{\sigma}}^{\mathbf{2}} = \frac{\mathbf{1}}{\boldsymbol{n}-\boldsymbol{K}} \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{K}} \sum_{\boldsymbol{i}:\boldsymbol{y}_{\boldsymbol{i}}=\boldsymbol{k}} (\boldsymbol{x}_{\boldsymbol{i}} - \hat{\boldsymbol{u}}_{\boldsymbol{k}})^{\mathbf{2}}$
- where n is the number of training observations, and n_k is the number of observations in the *k*th class.

Data Analysis: Linear Discriminant Analysis (LDA)

- Estimating probability class: $\hat{\boldsymbol{\pi}}_{\boldsymbol{k}} = \frac{\boldsymbol{n_k}}{\boldsymbol{n}}$

where *n* is the number of training observations, n_k is the number of observations in the *k*th class.

- An observation is assigned *X* = *x* to the class for which the:

- Discriminant Function $\hat{\boldsymbol{\delta}}_{\boldsymbol{k}}(\boldsymbol{x}) = \boldsymbol{x}\frac{\hat{\boldsymbol{\mu}}_{\boldsymbol{k}}}{\hat{\boldsymbol{\sigma}}^{2}} - \frac{\hat{\boldsymbol{u}}_{\boldsymbol{k}}^{2}}{2\hat{\boldsymbol{\sigma}}^{2}} + \boldsymbol{log}(\hat{\boldsymbol{\pi}}_{\boldsymbol{k}})$

  is Largest

- LDA is trying to cluster the data points of the classes together by using their mean and variance. This will separate the four classes.

Data Analysis: Quadratic Discriminant Analysis (QDA)

- We are trying to separate the classes (the classes are Rest, Wext, Sup, and Bflex) from a graph.
- To separate the classes, we assume that observations are drawn from a Gaussian distribution. but we do not make estimates of the mean and variance.
- Bayes classifier assigns observation *X* = *x* to the class for which:

- Discriminant Function is largest

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\boldsymbol{\Sigma}_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\boldsymbol{\Sigma}_k^{-1}x + x^T\boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\boldsymbol{\Sigma}_k^{-1}\mu_k - \frac{1}{2}\log|\boldsymbol{\Sigma}_k| + \log\pi_k\end{aligned}$$

- QDA is trying to cluster the data points of the classes together by using their mean and variance.
- This will also separate the classes from one another.

Data Analysis: K nearest neighbors (KNN) with N = 10.

• KNN is trying to separate the four classes.
• For positive integer *K* and a test observation x_0, it finds the *K* points in the training data nearest to x_0 = N_0.

• KNN will use the Bayes Classifier: $\boldsymbol{Pr(Y = j|X = x_0)}$
• KNN applies Bayes rule and classifies the test observation x_0 to the class with the largest probability. $\boldsymbol{Pr(Y = j|X = x_0) = \frac{1}{K}\sum_{i\in N_0} I(y_i = j)}$

• KNN will classify observation x_0 by considering the plurality of its 10 nearest neighbors.

Data Analysis: Tree.

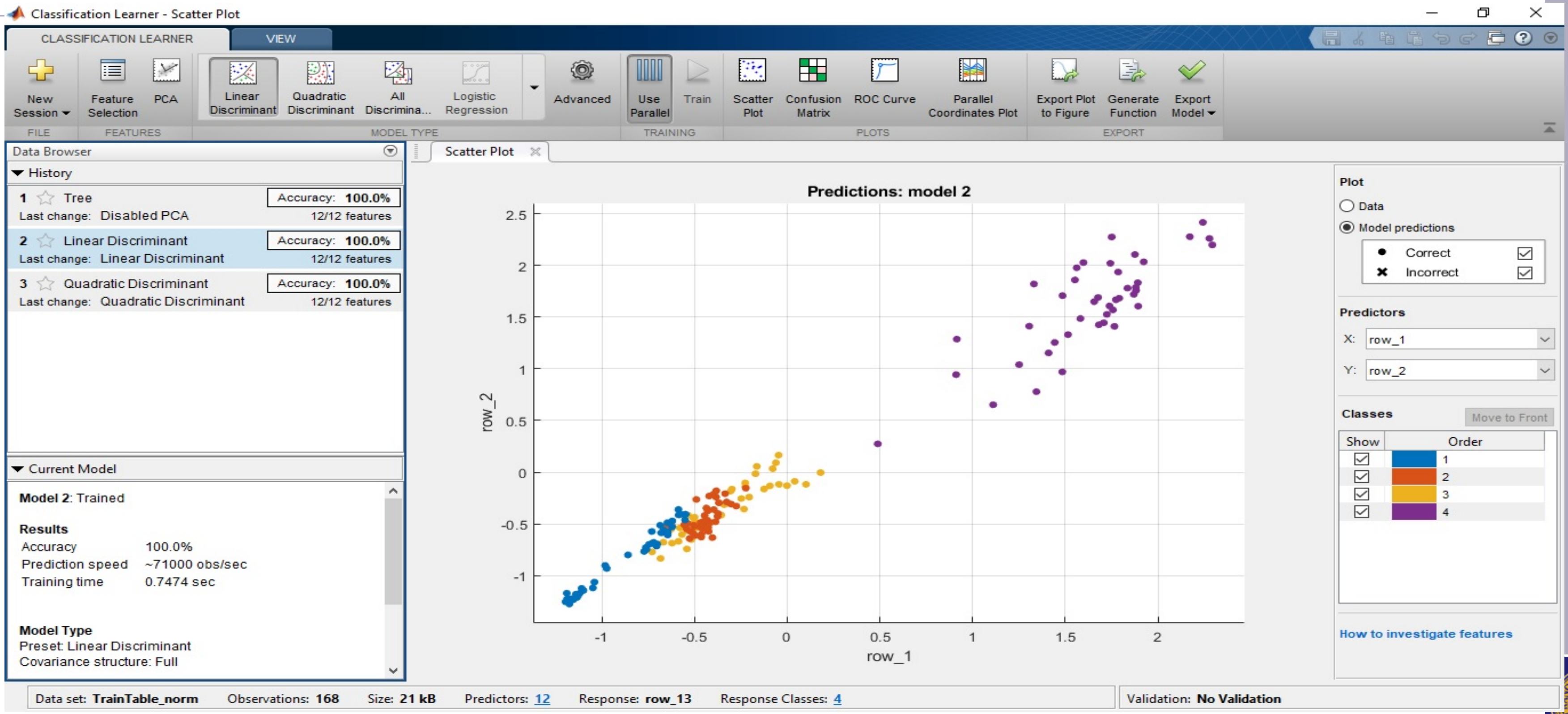
- We are trying to separate the classes in a Tree Structure.
- We have a total of 12 predictors or features.
- We divide the data by a certain value of the predictors, using as little predictors as possible.
- Hopefully, not all 12 predictors will be needed.
- A Tree example will be shown later.

Theory on LDA, QDA, KNN, and Tree

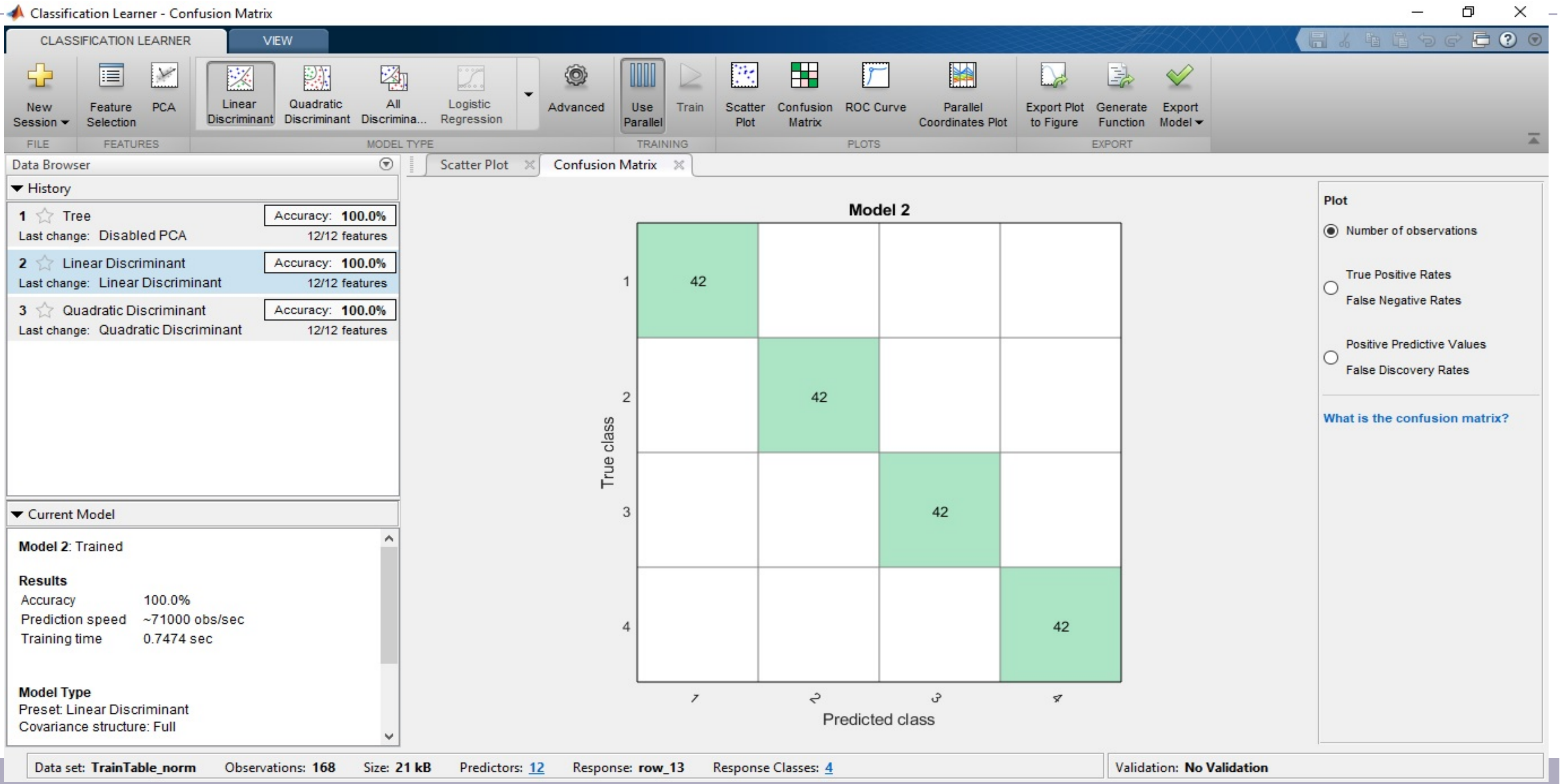
- If the data is linear, then LDA will be more accurate than QDA and QDA will overfit the training data and give worse results in testing than training.
- If the data is quadratic, QDA will be more accurate than LDA.
- As the number of observations decrease, than LDA should give better results.
- KNN gives better results when N is larger. We will use N = 10.
- Decision Tree is easy to visualize, but the calculations are longer than LDA, QDA, and KNN.
- It will take a lot longer to compute Tree when the predictors are very large (features > 1000). It could take several hours.

SF STATE

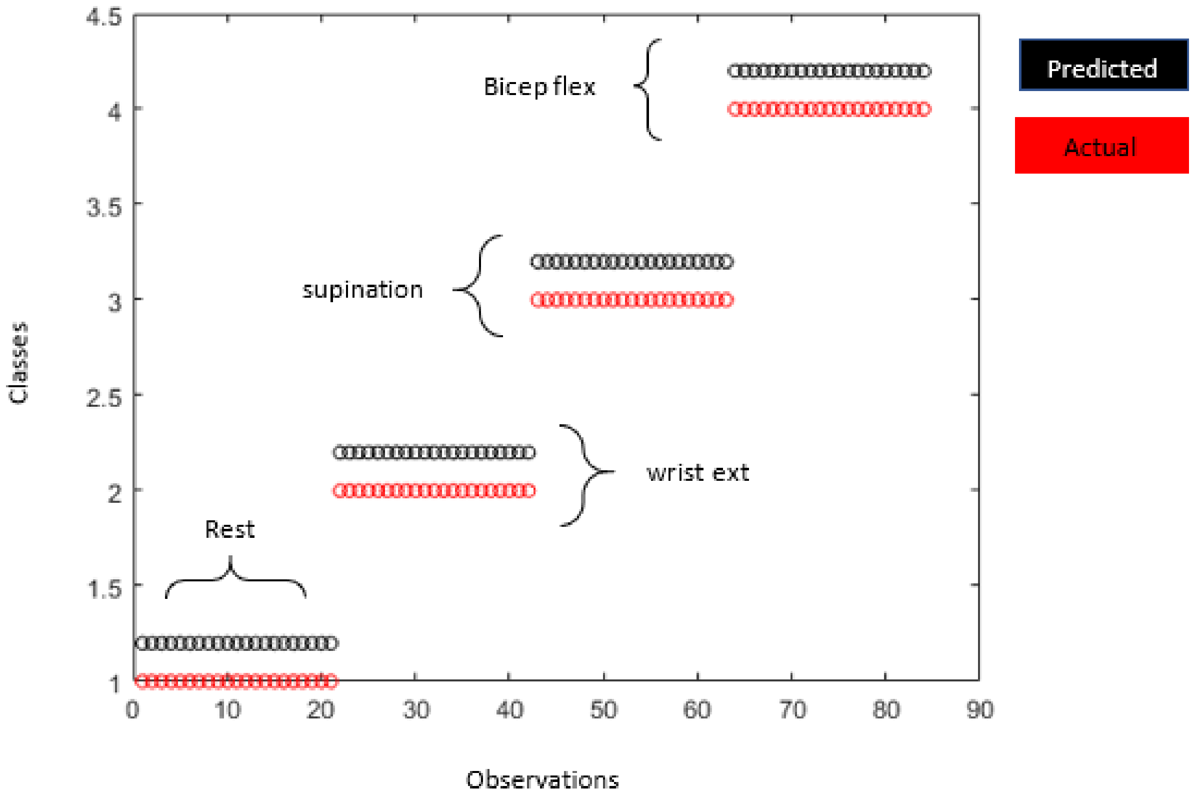
LDA for 2 train (168 observations), 1 test, 100 = WL, Winc = 20, 21 kB, time is 0.747 s. For classes: 1 is rest, 2 is Wext, 3 is Sup, 4 is Bflex.



Confusion Matrix for LDA for 2 training, 1 testing, WL = 100, Winc = 20, 100% training results.



Test result for LDA, 2 training (67%), 1 testing (33%), 84 testing observations, WL = 100, WInc = 20, normalized. The test results achieve 100% accuracy.



Using R to find the lowest possible training set to get 100% accuracy on either LDA, QDA, KNN, or Tree

- There are a total of 252 total observations on the 3 sets (84 for each set).
- We have split them up in Matlab using 2 training and 1 test set.
- R can randomly split the 252 observations into % for training and testing.
- Classification Learner cannot randomly split the data.
- To run faster Matlab compilation time, we can lower the training set until we get less than 100% accuracy.
- R was run with different percentage training data.

SF STATE

Using R to find the lowest possible training set to get 100% accuracy on either LDA, QDA, KNN, or Tree

% of Total Data	LDA		QDA		KNN	N = 10	Tree	
	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate	Testing Accuracy	Error Rate
67	100	0	99.76	0.24	99.76	0.24	99.52	0.48
33	100	0	96.69	3.31	99.65	0.35	99.29	0.71
30	100	0	97.86	2.14	99.55	0.45	99.44	0.56
28	100	0	97.36	2.64	99.67	0.33	98.35	1.65
27	99.89	0.11	97.67	2.33	99.13	0.87	99.35	0.65

- LDA was giving 100% testing accuracy until it reached 27% of the data.
- KNN with N = 10 averaged 99.13% accuracy and Tree averaged 99.35% accuracy with 27% training data.

Confusion Matrix for the test results for 2 random data sets for LDA for WL = 100, Winc = 20 at 27% of training data using R. LDA gives 100% testing results.

68 training (top) and
184 testing (bottom) observations

lda.train_class	bflex	rest	sup	wext
bflex	20	0	0	0
rest	0	9	0	0
sup	0	0	17	0
wext	0	0	0	22

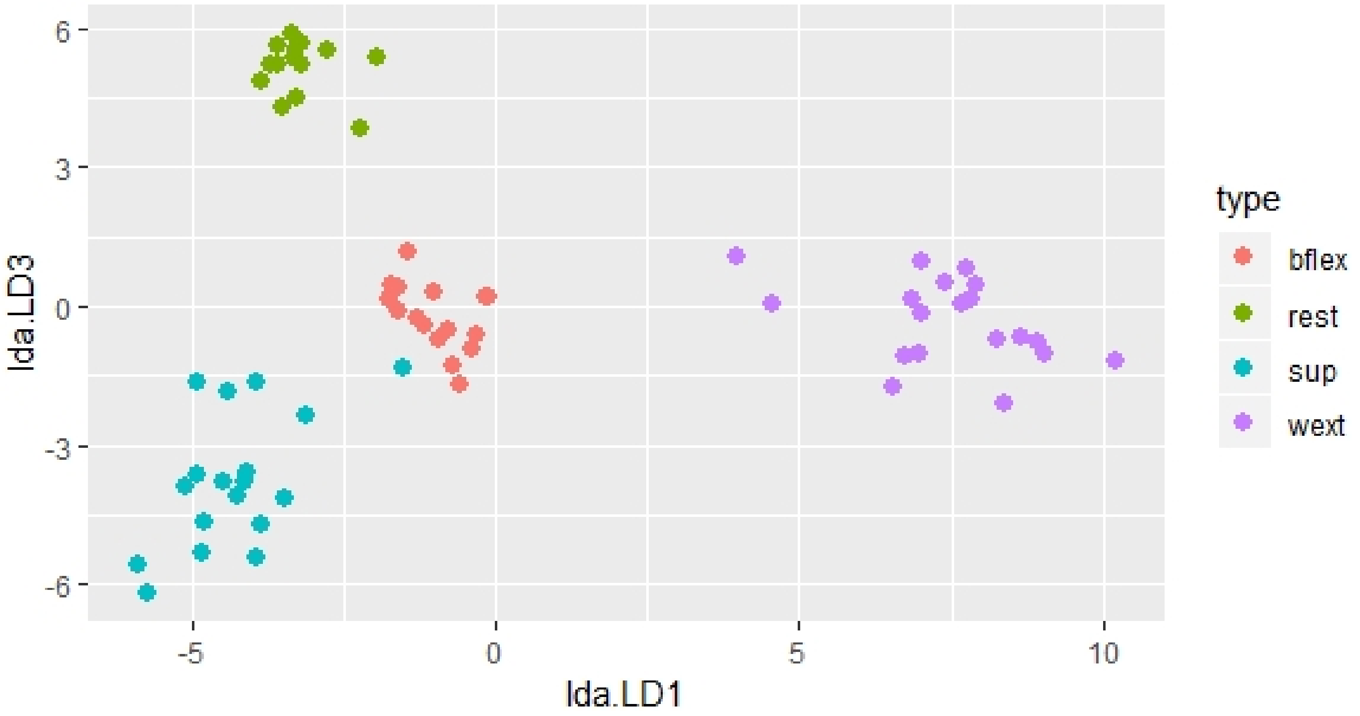
lda.class	bflex	rest	sup	wext
bflex	43	0	0	0
rest	0	54	0	0
sup	0	0	46	0
wext	0	0	0	41

68 training (top) and
184 testing (bottom) observations

lda.train_class	bflex	rest	sup	wext
bflex	19	0	0	0
rest	0	14	0	0
sup	0	0	18	0
wext	0	0	0	17

lda.class	bflex	rest	sup	wext
bflex	44	0	0	0
rest	0	49	0	0
sup	0	0	45	0
wext	0	0	0	46

LDA in R: Separation of classes for WL = 100, Winc = 20 at 27% (68 observations) of training data using R.



Confusion Matrix for the test results for 2 random data sets for KNN for N = 10, WL = 100, Winc = 20 at 27% of total data using R. KNN for N = 10 gives 99.5% testing results for both.

68 training observations
184 testing observations
Mean = .99456
Error rate = .005435

```
         test.Y
knn.pred bflex rest sup wext
   bflex    40    0   0    0
   rest      0   51   0    0
   sup       0    0  46    0
   wext      0    0   1   46
```

68 training observations
184 testing observations
Mean = .99456
Error rate = .005435

```
         test.Y
knn.pred bflex rest sup wext
   bflex    43    0   0    0
   rest      0   54   0    0
   sup       0    0  45    0
   wext      0    0   1   41
```

SF STATE

Confusion Matrix for the test results for 2 random data sets for Tree WL = 100, Winc = 20 at 27% of total data using R. Tree gives 97.8% and 99.5% testing accuracy.

68 training observations
184 testing observations
Mean = .97826
Error rate = .0217

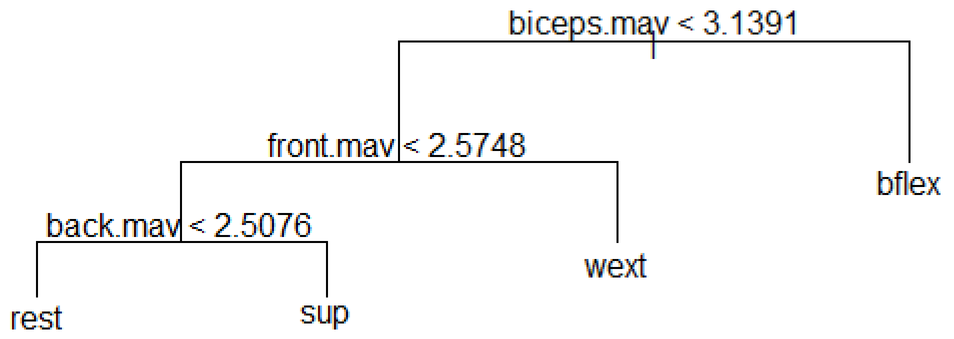
tree.pred	bflex	rest	sup	wext
bflex	42	0	0	0
rest	1	54	0	3
sup	0	0	46	0
wext	0	0	0	38

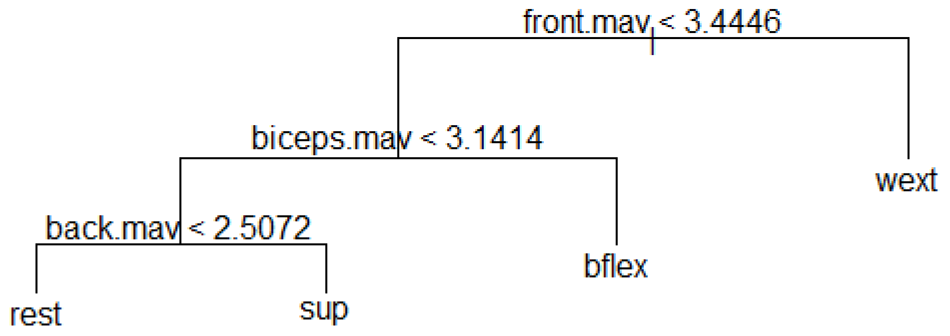
68 training observations
184 testing observations
Mean = .99456
Error rate = .0054

tree.pred	bflex	rest	sup	wext
bflex	39	0	0	0
rest	1	51	0	0
sup	0	0	47	0
wext	0	0	0	46

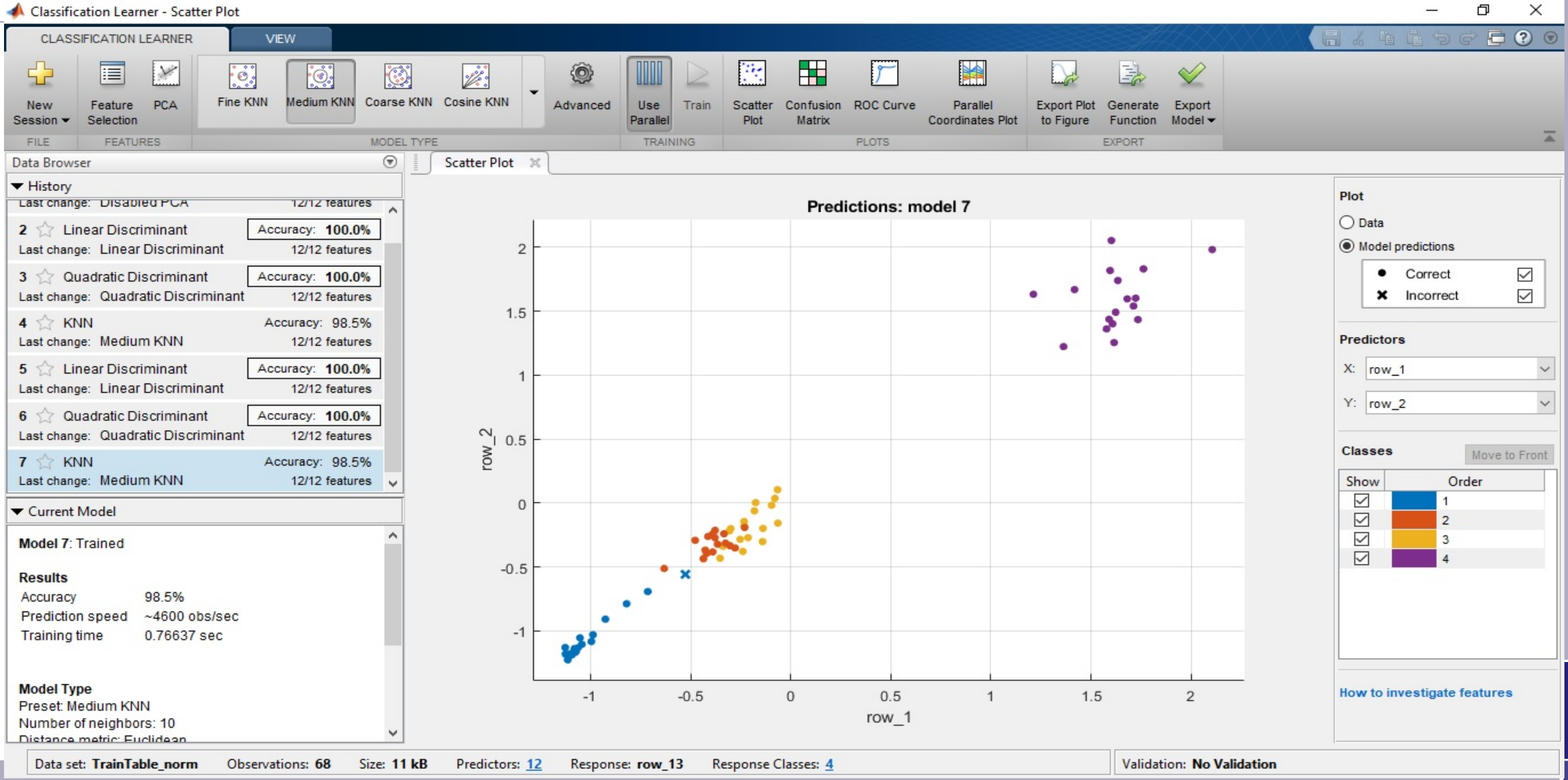
SF State

Tree structure for 2 random data sets for Tree using WL = 100, Winc = 20 at 27% of total data using R. Tree gives 97.8% and 99.5% testing results.

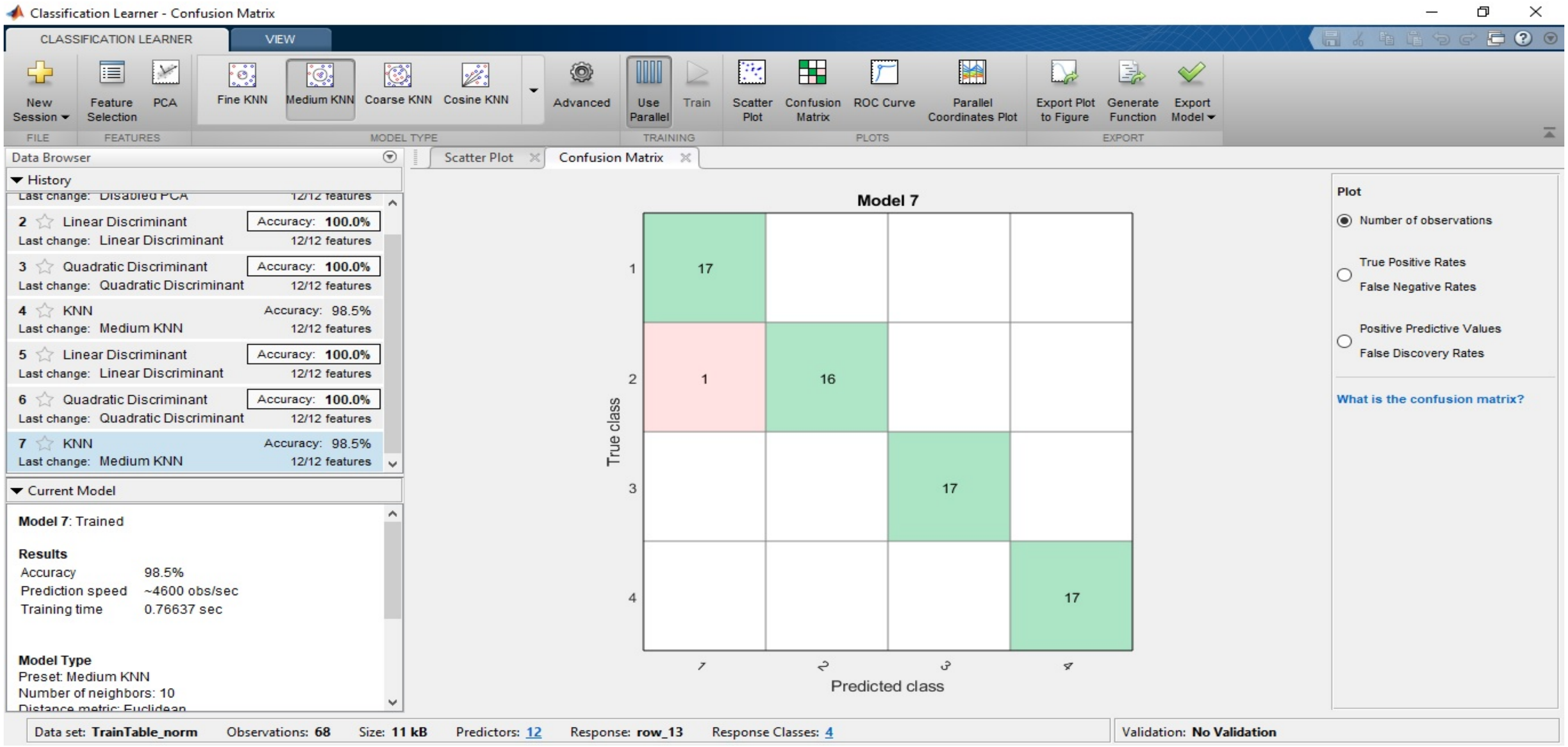




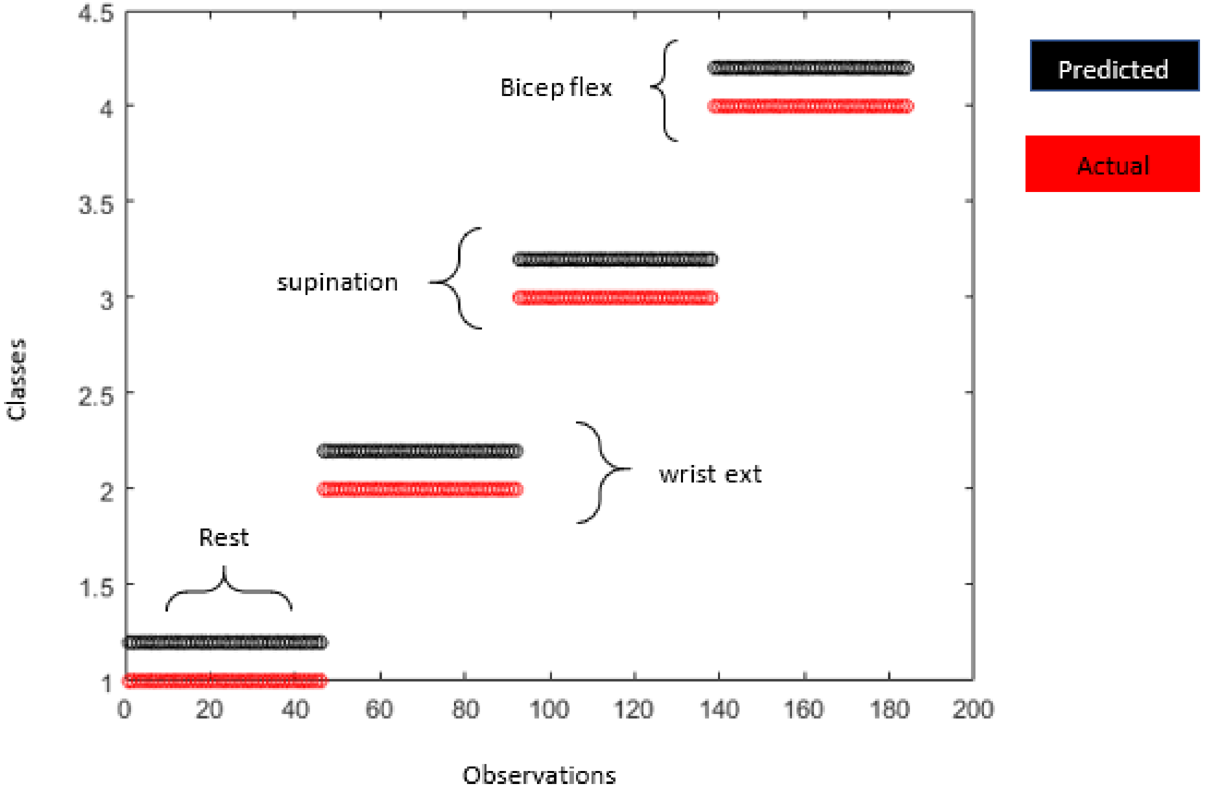
KNN with N = 10, 27% training, 68 observations, 100 = WL, Winc = 20, 11kB, time is 0.766 s. Classes: 1 is rest, 2 is Wext, 3 is Sup, 4 is Bflex.



Confusion Matrix for KNN with N = 10 for 27% training (68 observations) 11 kB data, WL = 100, Winc = 20. 98.5% training results.



KNN with N = 10 for 27% training, 73% testing (184 testing observations), WL = 100, Winc = 20. 100% testing accuracy.



Summary of Classification Learner results

		Training Accuracy %	Testing Accuracy %	Program Time seconds	Compile Time seconds	Total Time seconds
1	LDA; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.845	12.009
2	QDA; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.777	11.941
3	KNN; WL = 100, Winc = 2; N = 10 1608 obs; 167 kB; 67% training	99.9	100	11.164	0.759	11.923
4	Tree; WL = 100, Winc = 2; 1608 obs; 167 kB; 67% training	100	100	11.164	0.767	11.931
5	LDA; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.831	6.245
6	QDA; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.743	6.157
7	KNN; WL = 100, Winc = 5; N = 10 648 obs; 70 kB; 67% training	100	100	5.414	0.776	6.19
8	Tree; WL = 100, Winc = 5; 648 obs; 70 kB; 67% training	100	100	5.414	0.772	6.186
9	LDA; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	100	3.2	0.825	4.025
10	QDA; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	99.39	3.2	0.771	3.971
11	KNN; WL = 100, Winc = 10; N =10 328 obs; 37 kB; 67% training	100	100	3.2	0.763	3.963
12	Tree; WL = 100, Winc = 10; 328 obs; 37 kB; 67% training	100	100	3.2	0.76	3.96

Summary of Classification Learner results

		Training Accuracy %	Testing Accuracy %	Program Time seconds	Compile Time seconds	Total Time seconds
13	LDA; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.712	2.872
14	QDA; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.771	2.931
15	KNN; WL = 100, Winc = 20; N = 10 168 obs; 21 kB; ; 67% training	100	100	2.16	0.763	2.923
16	Tree; WL = 100, Winc = 20; 168 obs; 21 kB; 67% training	100	100	2.16	0.76	2.92
17	LDA; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	100	100	2.038	0.76	2.798
18	QDA; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	99.3	85.12	2.038	0.745	2.783
19	KNN; WL = 100, Winc = 20; N = 10 84 obs; 12 kB; 33% training	100	100	2.038	0.765	2.803
20	Tree; WL = 100, Winc = 20; 84 obs; 12 kB; 33% training	100	80.95	2.038	0.771	2.809
21	LDA; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	98.37	1.902	0.749	2.651
22	QDA; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	76.09	1.902	0.777	2.679
23	KNN; WL = 100, Winc = 20; N =10 68 obs; 11kB; 27% training	98.5	100	1.902	0.765	2.667
24	Tree; WL = 100, Winc = 20; 68 obs; 11 kB; 27% training	100	82.07	1.902	0.758	2.66

Timing when lowering observations

- KNN runtime was 10% times faster when going from 168 to 68 observations.
- KNN runtime was 49% times faster when going from 328 to 68 observations.

	1608 obs WL = 100 67% train	167 KB Winc = 2	648 obs WL = 100 67% train	70 KB Winc = 5	328 obs WL = 100 67% train	37 KB Winc = 10	168 obs WL = 100 67% train	21 KB Winc = 20	84 obs WL = 100 33% train	12 KB Winc = 20	68 obs WL = 100 27% train	11 KB Winc = 20		
	Total Time seconds	Faster	Total Time seconds	Faster than 1608 obs	Total Time seconds	Faster than 648 obs	Total Time seconds	Faster than 328 obs	Total Time seconds	Faster than 168 obs	Total Time seconds	Faster than 84 obs	Faster than 168 obs	Faster than 328 obs
LDA	12.009	N/A	6.245	1.92	4.025	1.55	2.872	1.40	2.798	1.03	2.651	1.06	1.08	1.52
QDA	11.941	N/A	6.157	1.94	3.971	1.55	2.931	1.35	2.783	1.05	2.679	1.04	1.09	1.48
KNN with N = 10	11.923	N/A	6.19	1.93	3.963	1.56	2.923	1.36	2.803	1.04	2.667	1.05	1.10	1.49
Tree	11.931	N/A	6.186	1.93	3.96	1.562	2.92	1.36	2.81	1.04	2.66	1.06	1.10	1.49

Experiments and Results

- MATLAB with Classification Learner was used for Data Analysis of 4 hand movements (Rest, Wext, Sup, and Bflex).
- R lowered the observations to get 100% testing results.
- The lower limit was 27% training and 78% testing for LDA in R.
- We achieved 100% testing results for LDA and KNN for 33% training and 67% testing.
- Lowering to 68 observations gave results of 8% faster for LDA and 10% faster for KNN with N = 10.

Summary

- MATLAB was programmed for Feature Extraction of our 4 hand gestures.
- Classification Learner was used to compile our 12 predictors.
- R can be used to find the lowest limit of training data needed to get 100% results.
- By utilizing the testing results of R, the lower limit was reduced from 168 to 68 observations.
- At our lowest limit, we found that the total time was 2.67 s for KNN with N = 10. The runtime was reduced 10% from 168 to 68 obs.
- MATLAB, Classification Learner, and R can be utilized for future EMG sensor data analysis.

References

- Chen, Xiang, et al. "Hand gesture recognition research based on surface EMG sensors and 2D-accelerometers." *2007 11th IEEE International Symposium on Wearable Computers*. IEEE, 2007.
- James, Gareth, et al. *An introduction to statistical learning*. Vol. 112. New York: springer, 2013.
- Lotte, Fabien. "A new feature and associated optimal spatial filter for EEG signal classification: Waveform Length." *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. IEEE, 2012.
- Phinyomark, A., et al. "Evaluation of EMG feature extraction for movement control of upper limb prostheses based on class separation index." *5th Kuala Lumpur International Conference on Biomedical Engineering 2011*. Springer, Berlin, Heidelberg, 2011.
- Teetor, Paul. "R Cookbook: Proven Recipes for Data Analysis." *Statistics, and Graphics. Cambridge: O'Reilly* (2011).
- Wickham, Hadley, and Garrett Grolemund. *R for data science: import, tidy, transform, visualize, and model data*. O'Reilly Media, Inc., 2016.
- Zhang, Xiaorong, and He Huang. "A real-time, practical sensor fault-tolerant module for robust EMG pattern recognition." *Journal of neuroengineering and rehabilitation* 12.1 (2015): 18.
- Zhang, Xiaorong, He Huang, and Qing Yang. "Real-time implementation of a self-recovery EMG pattern recognition interface for artificial arms." *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2013.

SF STATE