Using Features in Embedded Machine Learning

Michael Stanley

http://sensip.asu.edu

Mike.Stanley@ieee.org

Arizona State University

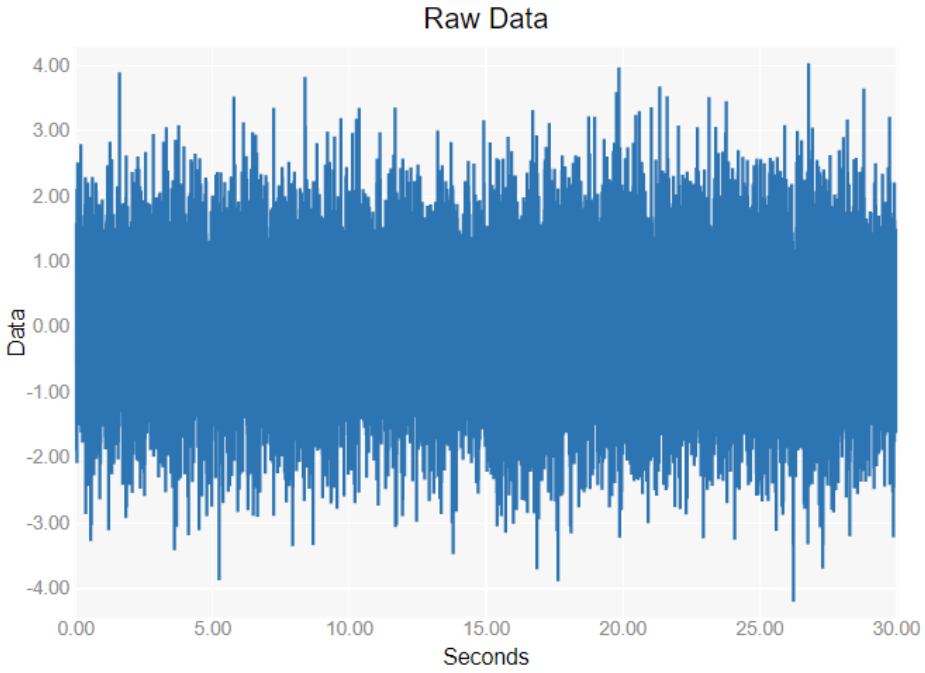
SenSIP INDUSTRY CONSORTIUM

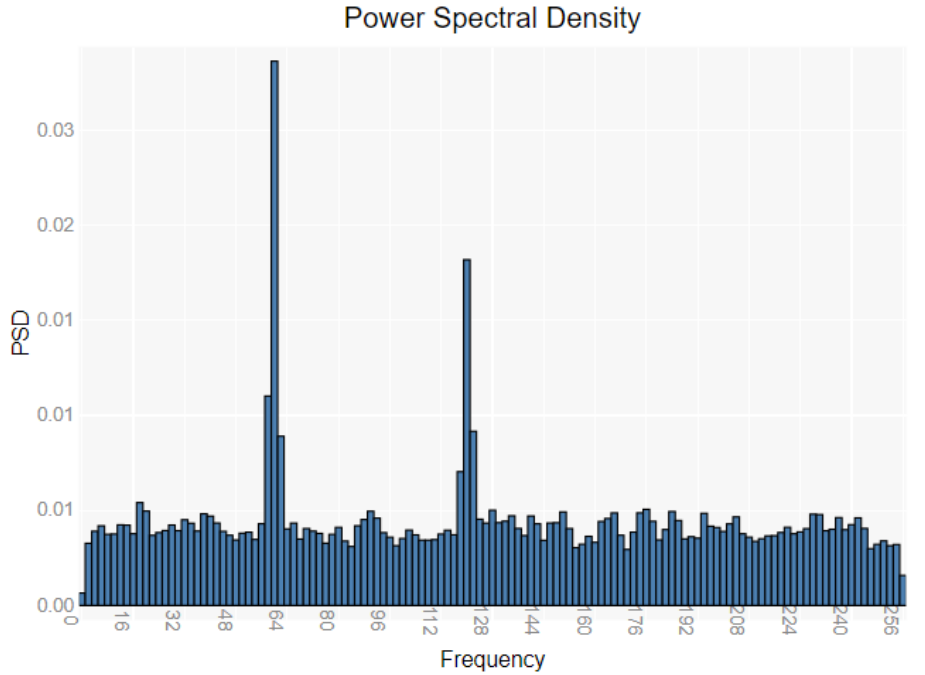
ML Modeling Example 2

- Determine Fan Speed from sensors attached passively to a fan.
- In addition to the topics explored in example 1, we will look at:
  - Features for machine learning
  - Computing features from raw data
  - Determining what features to use in your model using several different techniques
- As was done in Modeling Example 1, we'll be reviewing both Jupyter and C code in detail.

What are Features?

- Classical machine learning algorithms are trained using feature data as inputs
- Many business applications use parts of the original dataset as features. For example, zip codes, area codes, prices, quantities, etc.
- For sensor-based problems, features are often numerical descriptors or statistics which have been **computed from the raw data**. Examples include:
  - Mean, Standard Deviation/Variance, Skew Factor, Kurtosis, Shape Factor, Crest Factor, RMS
  - Minimum / Maximum / Median / Sum
  - Number of peaks, crossings, …
  - Entropy / Energy
  - Quartile value
  - Histogram coefficients
  - Slope (Linear Trend)
  - FFT / PSD / Wavelet coefficients
  - Auto-correlation, Cross-correlation
  - And many, many, **many** others…





Common statistical features

mean

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

crest factor

$$C = \frac{|x_{\text{peak}}|}{x_{\text{rms}}}$$

zero crossing rate

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$

$$\operatorname{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

standard deviation

$$\sigma \equiv \sqrt{\mathrm{E}\left[(X-\mu)^2\right]}$$

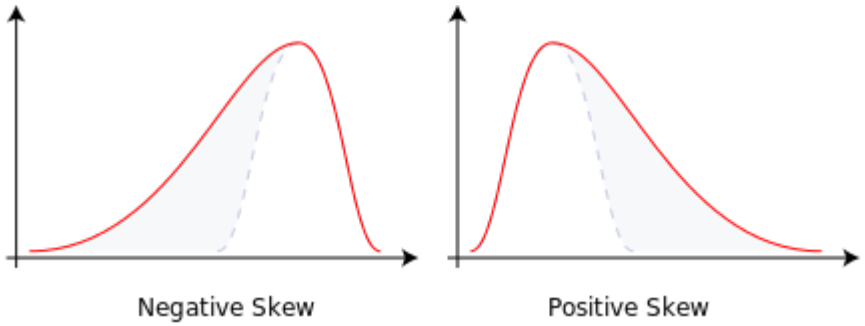
entropy

$$S = -k_{\mathrm{B}} \sum_i p_i \log p_i,$$

$$x_{\text{RMS}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

skewness

$$\tilde{\mu}_3 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$



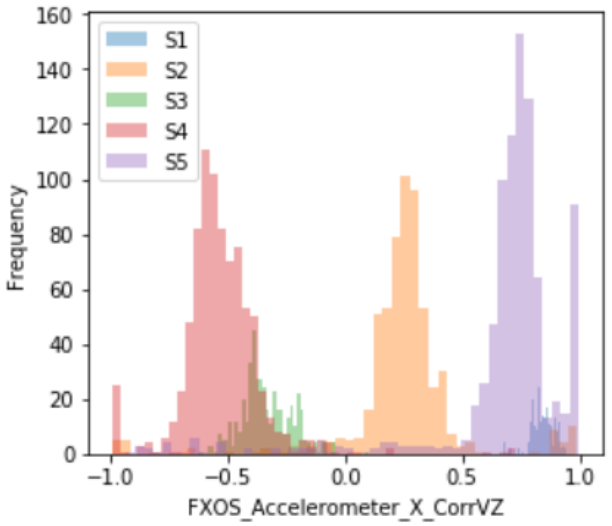
1st derivative

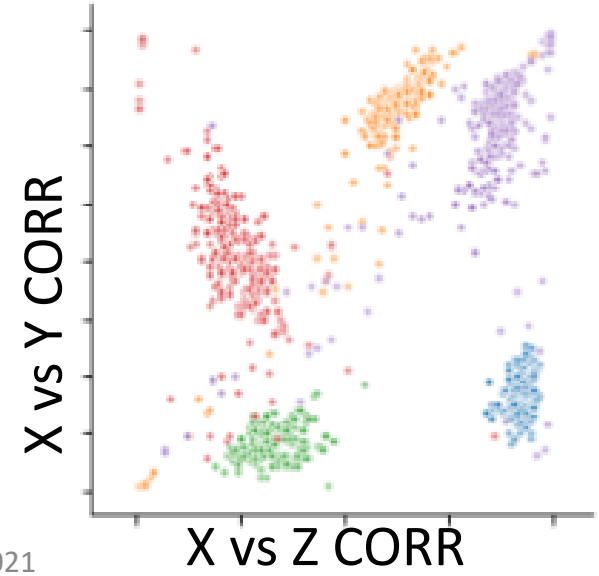
$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$\operatorname{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$

source: wikipedia

Proper Selection of Features Can Really Simplify Things

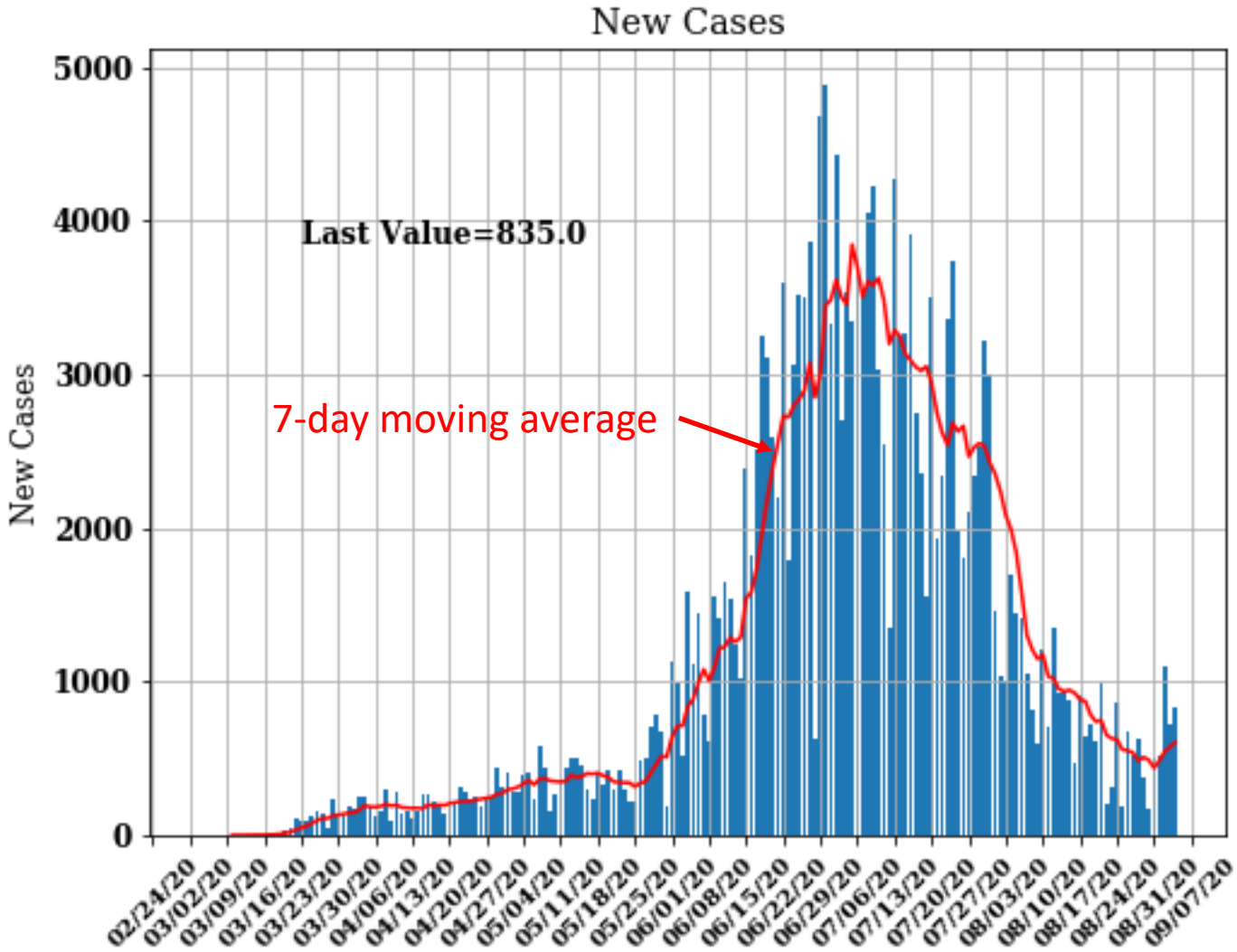




- For classical machine learning, feature engineering is one of the most difficult tasks
- Domain knowledge is advantageous when determining what features to use.
- The virtually unlimited set of possible features makes it impossible to “pre-code” all possible values in an embedded engine.
- Deep learning neural networks can determine features automatically, eliminating this task.

Example: Summer 2020 Covid-19 Cases in AZ

- New case statistics on a day-by-day basis vary tremenously
- Compute a simple moving average and the data is more meaningful.
- Take the slope of the moving average and you have trends.
- Feature engineering works in a similar fashion to extract *information* from *raw data*.



Features can be physics-based

As an example…
Meshing gears introduce three primary components in the vibration frequency domain:

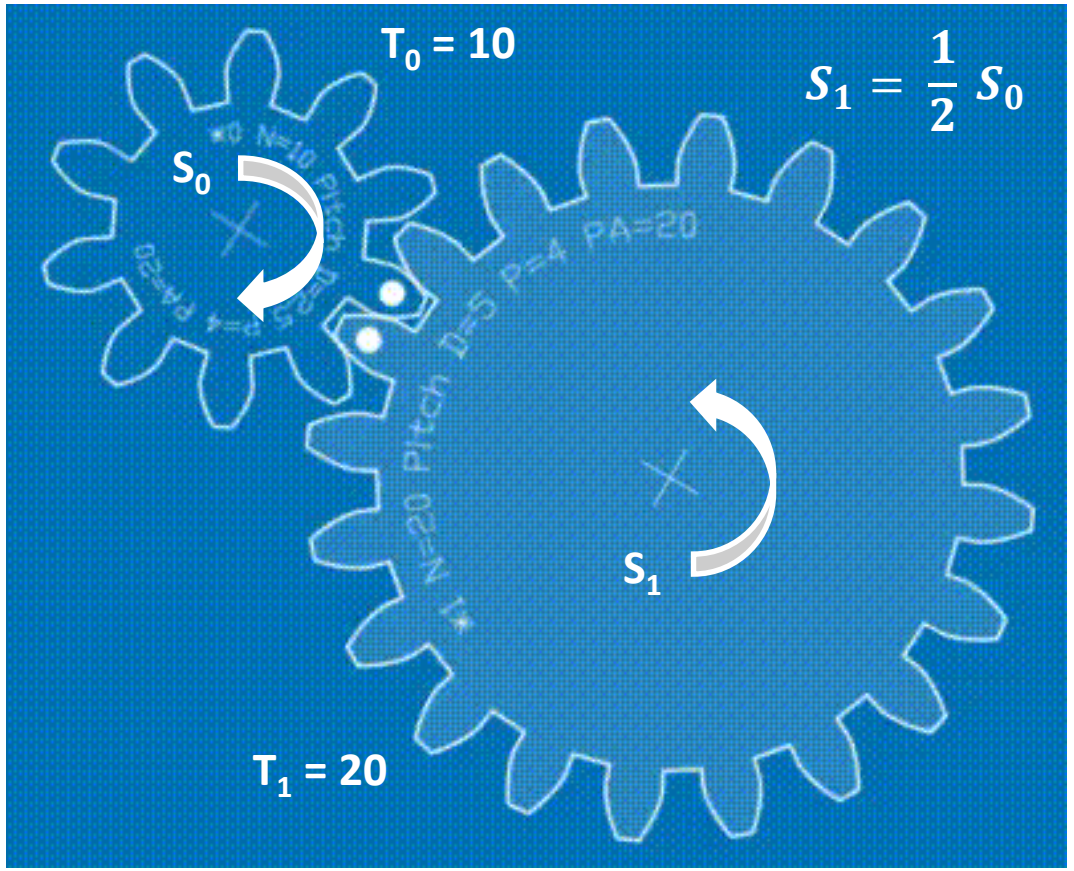
- Input gear rotation frequency
- Output gear rotation frequency
- Gear mesh frequency

Gear mesh speed = shaft speed X number of teeth

$$S_1 = \frac{T_0}{T_1} S_0$$

Where:
T_0 = number of teeth on gear #0
T_1 = number of teeth on gear #1
S_0 = rotation speed of gear #0
S_1 = rotation speed of gear #1

Defects in gear(s) affect expected noise signatures, which can be further distilled using a number of feature types (wavelets, fft, statistical, etc.)

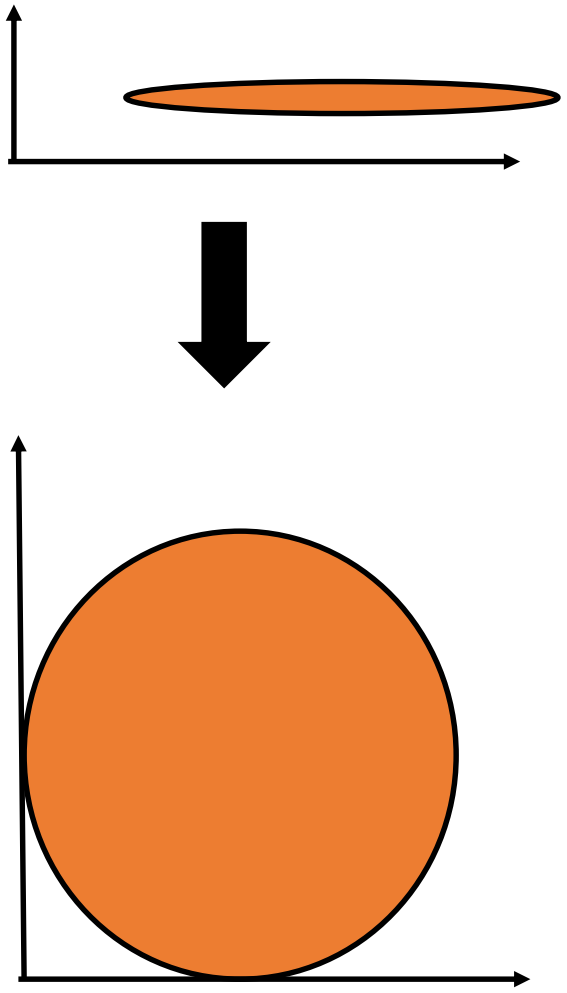


Gears designed using https://geargenerator.com/
Captured via https://www.apowersoft.com/

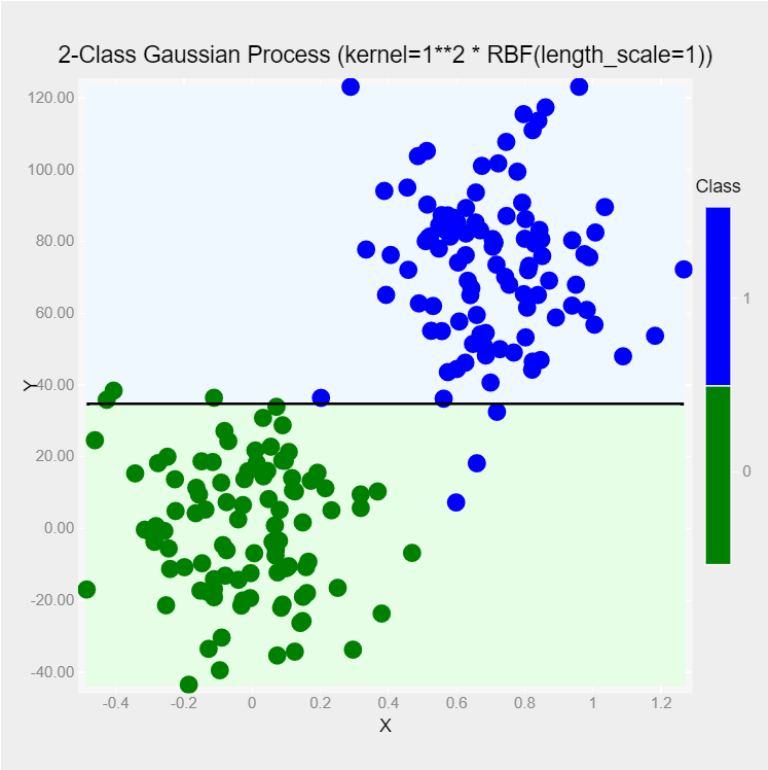
Loss of one gear tooth will introduce additional noise every time that missing tooth “meshes” with the adjacent gear.

Feature Scaling/Normalization

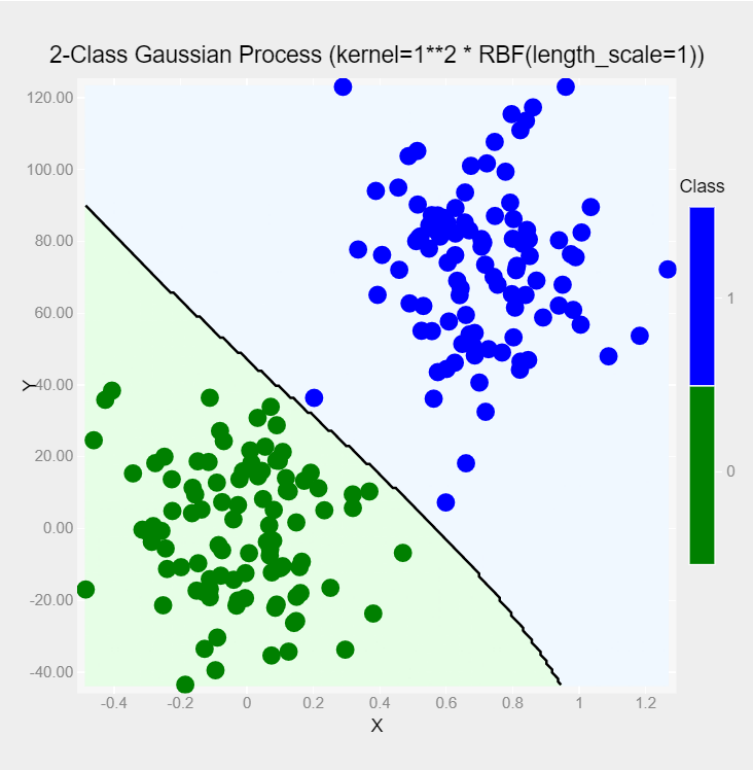
- Many machine learning algorithms will attempt to minimize output error over all input dimensions
- Large differences in signal magnitude and/or can cause the algorithms to focus on some feature dimensions while largely ignoring the error in other dimensions. This will throw off the decision boundary
- This can be rectified by normalizing all inputs prior to model training
- Normalization parameters should be computed on in-sample data, but applied to out-of-sample inputs as well.



Data Normalization



No Normalization
6/200 in-sample errors
43/1000 out-of-sample errors

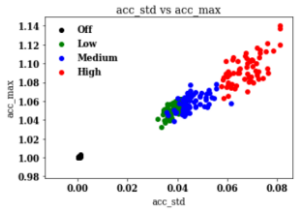


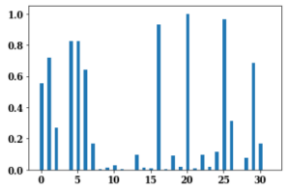
Standard Normalization
0/200 in sample errors
18/1000 out-of-sample errors

Standard and Min/Max Scaling

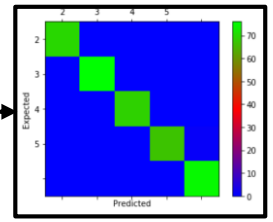
- Standard:
  - $x_{norm} = \frac{(x - \bar{x})}{\sigma_x}$
  - centers the input about zero and scales by the standard deviation
- Min/Max:
  - $\text{scale} = \frac{\text{x} - \min(x)}{\max(x) - \min(x)}$
  - maps the input into the range [0, 1.0]

Common Machine Learning Workflow





TensorFlow Model Generation



Logfiles in .csv format

Feature Extraction

features.csv

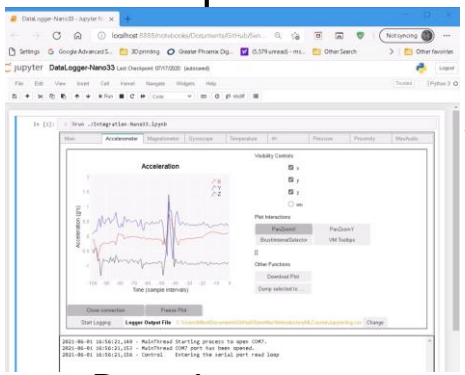
Feature Selection

Classical Model Generation

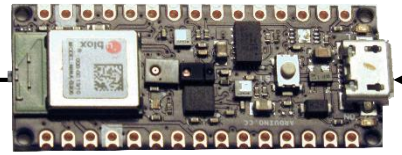
Deploy to cloud

ML Framework

Model files



Data Logger



Development Board

Embedded IDE

Fan speeds:

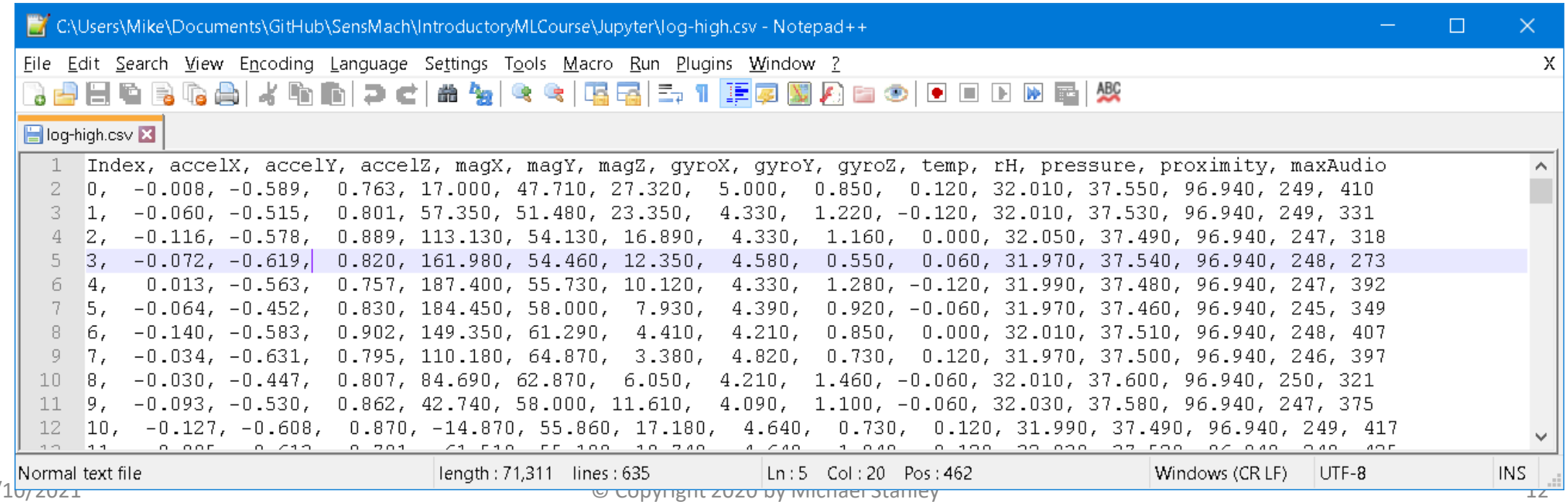
- off
- low
- medium
- high

We used a variant of the datalogger we looked at in the Software lecture for this example

Raw Data Files:

- log-off.csv
- log-low.csv
- log-medium.csv
- log-high.csv

This logger records a lot more data!



Time

Index	accelX	accelY	accelZ	magX	magY	magZ	gyroX	gyroY	gyroZ	temp	rH	pressure	proximity	maxAudio
0	-0.08	-0.633	0.814	-65.76	52.43	25.54	4.7	0.92	0.12	31.87	37.75	96.94	248	191
1	-0.044	-0.569	0.749	-12.16	47.17	28.63	4.7	0.92	0.12	31.87	37.67	96.94	248	223
2	-0.098	-0.512	0.823	10.96	46.75	27.4	4.76	0.85	0.06	31.89	37.72	96.94	247	254
3	-0.12	-0.58	0.849	43.51	51.01	23.35	4.82	0.61	0.12	31.87	37.7	96.94	248	199
4	-0.056	-0.595	0.794	94.74	53.77	18.92	4.94	0.61	0.06	31.89	37.69	96.94	248	223
5	-0.062	-0.542	0.766	144.03	55.53	14.43	4.7	0.73	0.12	31.83	37.78	96.94	247	303
6	-0.111	-0.532	0.849	172.52	56.51	10.8	4.33	0.92	-0.12	31.83	37.66	96.94	245	355
7	-0.115	-0.595	0.874	172.01	58.03	7.82	4.7	0.85	-0.06	31.85	37.73	96.94	248	368
8	-0.037	-0.576	0.776	141.25	61.96	5.92	4.52	0.67	0.12	31.87	37.69	96.94	251	172
9	-0.062	-0.477	0.81	106.34	65.33	4.75	4.7	0.85	0.06	31.83	37.73	96.94	248	257
10	-0.127	-0.563	0.863	84.2	62.79	6.69	4.39	0.79	0	31.89	37.74	96.94	249	302
11	-0.091	-0.575	0.852	46.63	58.01	10.4	4.64	0.61	0.12	31.85	37.8	96.94	248	385
12	-0.041	-0.573	0.762	-7.3	55.83	16.77	4.58	0.61	0	31.81	37.67	96.94	247	317
13	-0.079	-0.474	0.814	-51.42	54.97	20.25	4.58	0.61	0	31.8	37.77	96.94	246	193
14	-0.117	-0.575	0.865	-71.89	54.47	23.18	4.58	0.85	0.18	31.87	37.75	96.94	248	175
15	-0.08	-0.598	0.825	-59.86	51.7	25.96	4.52	0.79	0.18	31.87	37.7	96.94	247	301
16	-0.04	-0.566	0.757	-22.74	47.67	27.33	4.58	0.79	-0.06	31.85	37.78	96.94	248	289
17	-0.103	-0.503	0.827	4.38	46.83	28.69	4.39	0.85	0.06	31.87	37.67	96.94	248	244
18	-0.117	-0.584	0.862	31.57	49.91	24.15	4.39	0.61	0.12	31.85	37.81	96.94	245	186
19	-0.049	-0.623	0.787	78.06	54.08	20.13	4.76	0.67	0	31.83	37.69	96.94	247	204
20	-0.051	-0.497	0.796	128.54	54.83	14.38	4.88	0.92	0.24	31.89	37.77	96.94	249	219
21	-0.105	-0.532	0.83	165.83	55.81	11.51	4.39	0.73	0.12	31.85	37.66	96.94	247	263
22	-0.102	-0.616	0.849	176.42	56.95	9.23	4.58	0.85	-0.06	31.9	37.75	96.94	247	196
23	-0.04	-0.613	0.77	157.37	59.95	6.92	4.52	0.73	0	31.9	37.76	96.94	250	205
24	-0.066	-0.488	0.795	118.33	63.65	4.31	4.7	0.73	-0.12	31.89	37.73	96.94	246	201
25	-0.117	-0.555	0.837	96.17	64.07	4.21	4.76	0.67	-0.06	31.83	37.73	96.94	247	223
26	-0.093	-0.631	0.831	65.56	60.41	7.78	4.76	0.67	0.24	31.87	37.62	96.94	248	174
27	-0.031	-0.595	0.756	17.02	56.92	14.03	4.82	0.79	0.12	31.87	37.74	96.94	249	322
28	-0.082	-0.504	0.803	-31.56	55.2	19.97	4.58	0.61	-0.06	31.87	37.71	96.94	248	251
29	-0.119	-0.578	0.849	-64.34	54.65	22.19	4.82	0.67	0	31.85	37.64	96.94	246	216

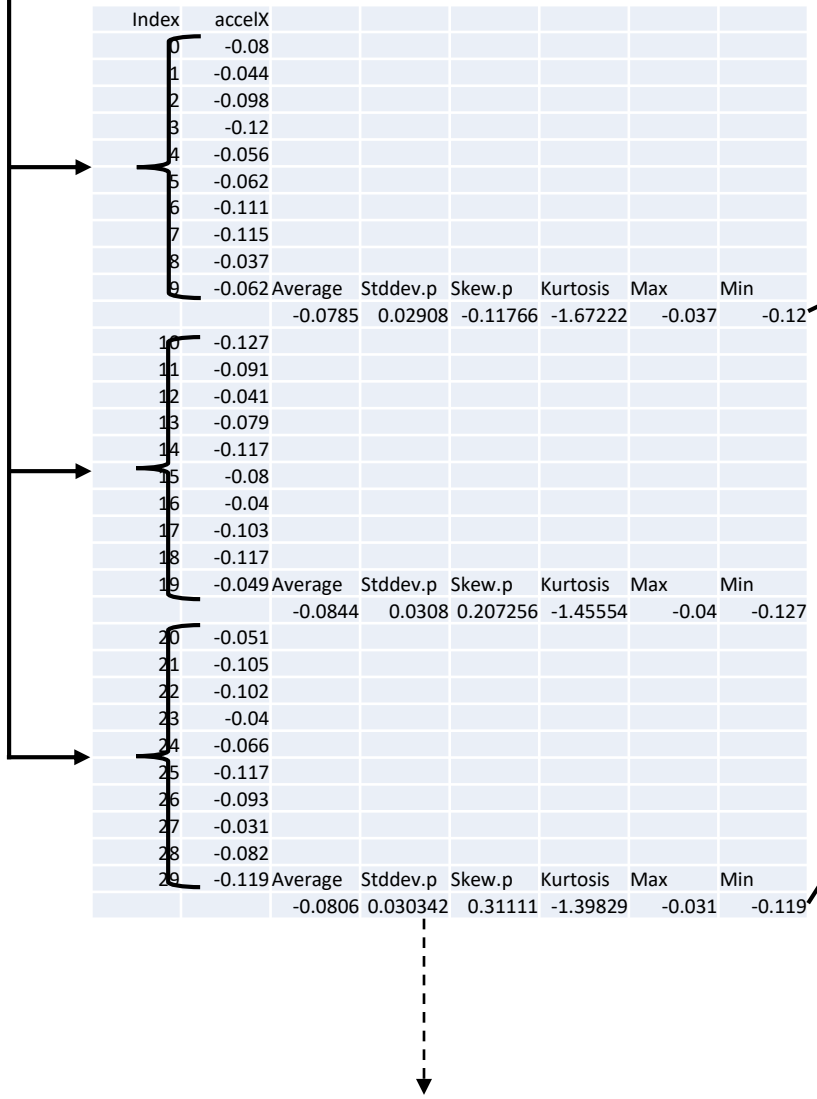
Let’s look at the feature extraction process

Index	accelX
0	-0.08
1	-0.044
2	-0.098
3	-0.12
4	-0.056
5	-0.062
6	-0.111
7	-0.115
8	-0.037
9	-0.062
10	-0.127
11	-0.091
12	-0.041
13	-0.079
14	-0.117
15	-0.08
16	-0.04
17	-0.103
18	-0.117
19	-0.049
20	-0.051
21	-0.105
22	-0.102
23	-0.04
24	-0.066
25	-0.117
26	-0.093
27	-0.031
28	-0.082
29	-0.119

Process one set of sensor measurements at a time

Let’s look at the feature extraction process

Each feature is computed from a window (often in time) of samples



Index	accelX						
0	-0.08						
1	-0.044						
2	-0.098						
3	-0.12						
4	-0.056						
5	-0.062						
6	-0.111						
7	-0.115						
8	-0.037						
9	-0.062	Average	Stddev.p	Skew.p	Kurtosis	Max	Min
		-0.0785	0.02908	-0.11766	-1.67222	-0.037	-0.12
10	-0.127						
11	-0.091						
12	-0.041						
13	-0.079						
14	-0.117						
15	-0.08						
16	-0.04						
17	-0.103						
18	-0.117						
19	-0.049	Average	Stddev.p	Skew.p	Kurtosis	Max	Min
		-0.0844	0.0308	0.207256	-1.45554	-0.04	-0.127
20	-0.051						
21	-0.105						
22	-0.102						
23	-0.04						
24	-0.066						
25	-0.117						
26	-0.093						
27	-0.031						
28	-0.082						
29	-0.119	Average	Stddev.p	Skew.p	Kurtosis	Max	Min
		-0.0806	0.030342	0.31111	-1.39829	-0.031	-0.119

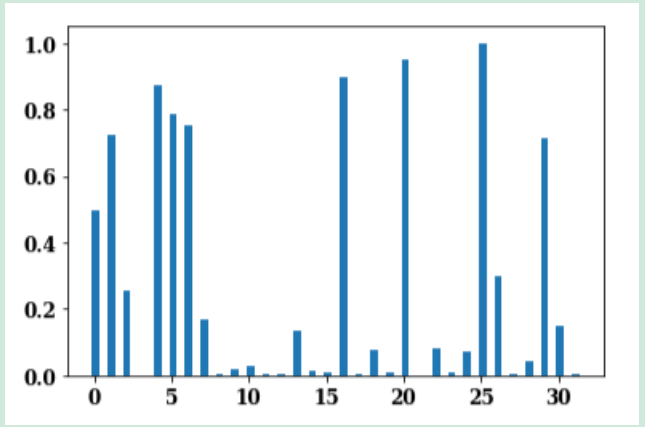
Average	Stddev.p	Skew.p	Kurtosis	Max	Min
-0.0785	0.02908	-0.11766	-1.67222	-0.037	-0.12
-0.0844	0.0308	0.207256	-1.45554	-0.04	-0.127
-0.0806	0.030342	0.31111	-1.39829	-0.031	-0.119
...	...	...	...	...	...
-0.0743	0.027288	0.043038	-1.56928	-0.032	-0.11

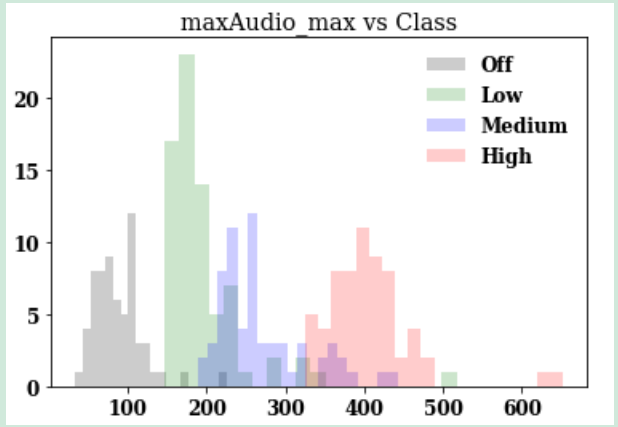
The original column of data was 624 X 1
Features after extraction = 62 X 6

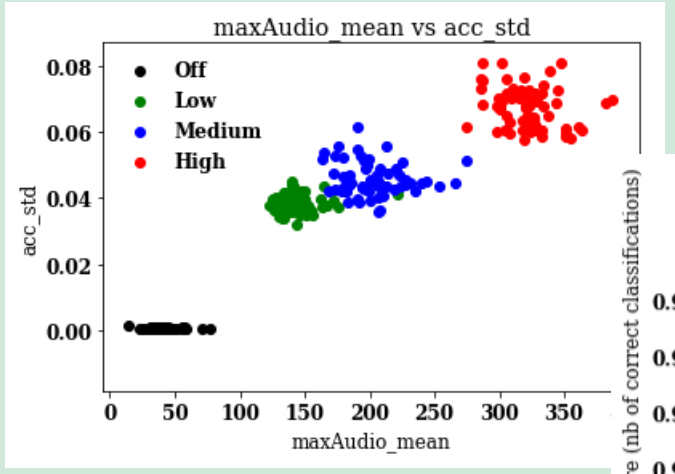
- Repeat this process for all columns, appending new features to the right.
- Repeat the entire process for each data file, appending new rows to the bottom.

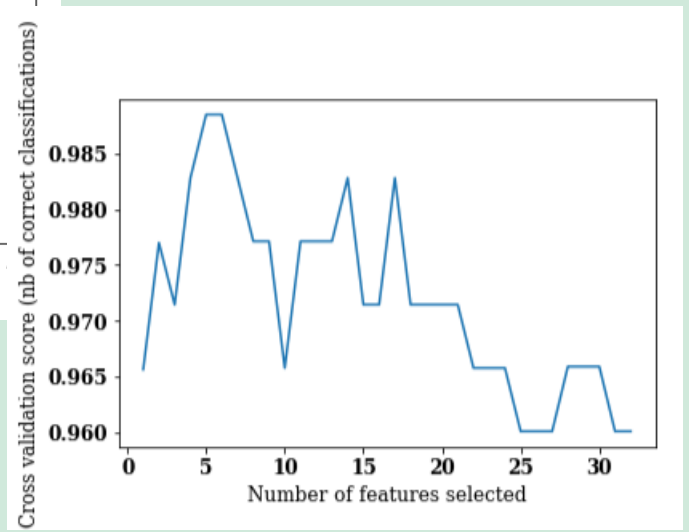
Walkthroughs

1. Jupyter Notebook for Feature Extraction
2. Jupyter Notebook for Feature Selection
3. Simple_Classification Jupyter Notebook
4. Keras_example Jupyter Notebook
5. fanDemo embedded code









Files used in this reference

Filename(s)	Description
loggerV4.html / helpV3.html	Simple HTML/Javascript based application to record data sent by Arduino. This was NOT used for this example, but you could easily do so with only minor mods to the feature extraction notebook.
nano33_data_logger_rt.ino	Embedded code for Arduino data logger
Feature_Extraction_Using_Pandas.ipynb	Jupyter notebook for feature extraction
log-*.csv	Previously recorded sensor logfiles
Feature_Selection.ipynb	Jupyter notebook for feature selection
Simple_Classification.ipynb	Jupyter notebook for classical ML model training (no model export in this one)
keras_example.ipynb	Jupyter notebook for Tensorflow model training
fanDemo/*.*	Embedded code using Tensorflow neural net