

Predictive Maintenance



by Alex Dance

Purpose -Share knowledge on Predictive Maintenance

Part 1 – the market and approach

- The market size and players
- The opportunities
- Go through types
- Go through approaches

Part 2 - Data Set on Kaggle

- Looked at the data
- Problems I had
- Solutions I borrowed
- Show my approach
- Show my results

Some machines are complicated – with multiple sensors – that are interlinked

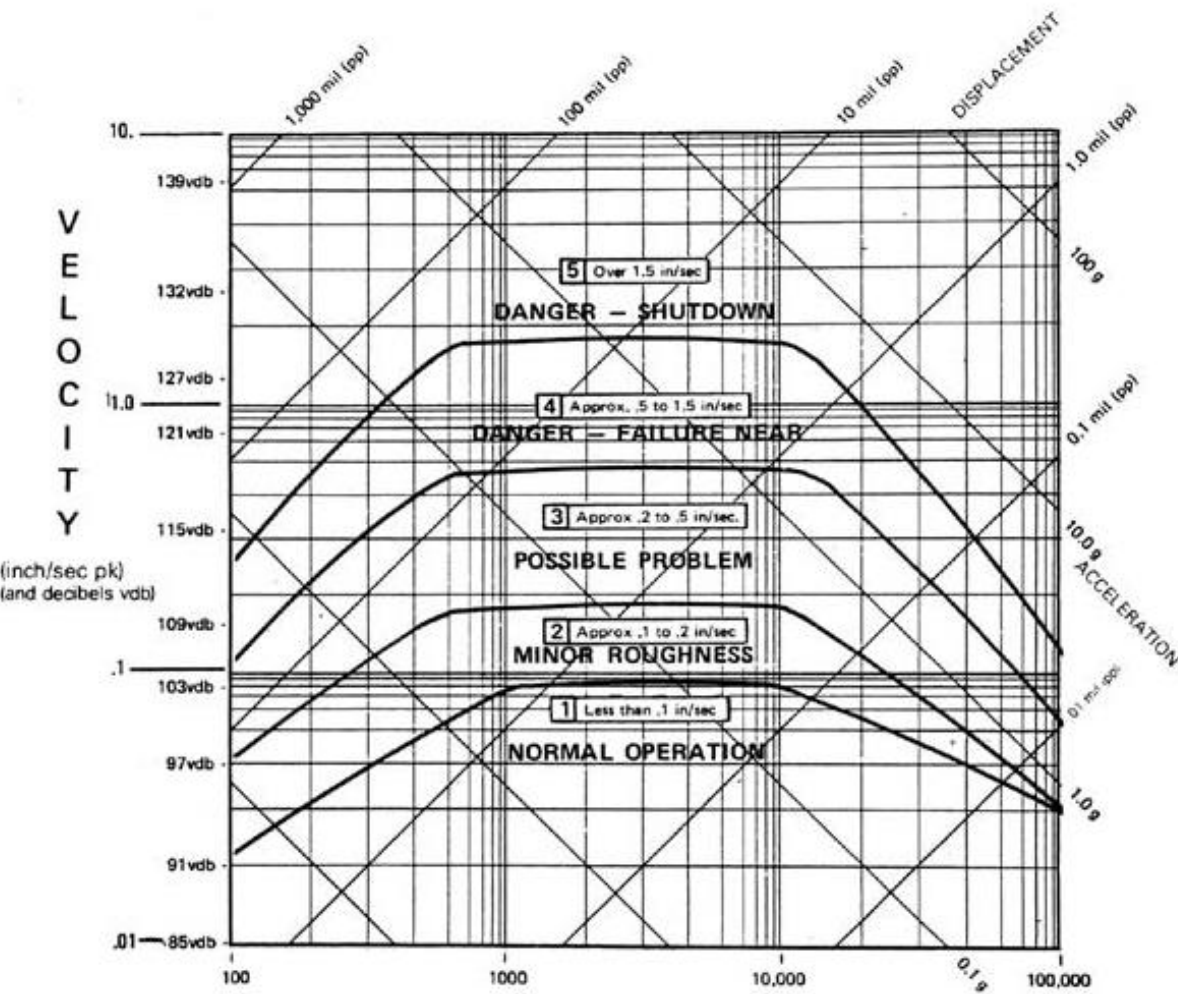
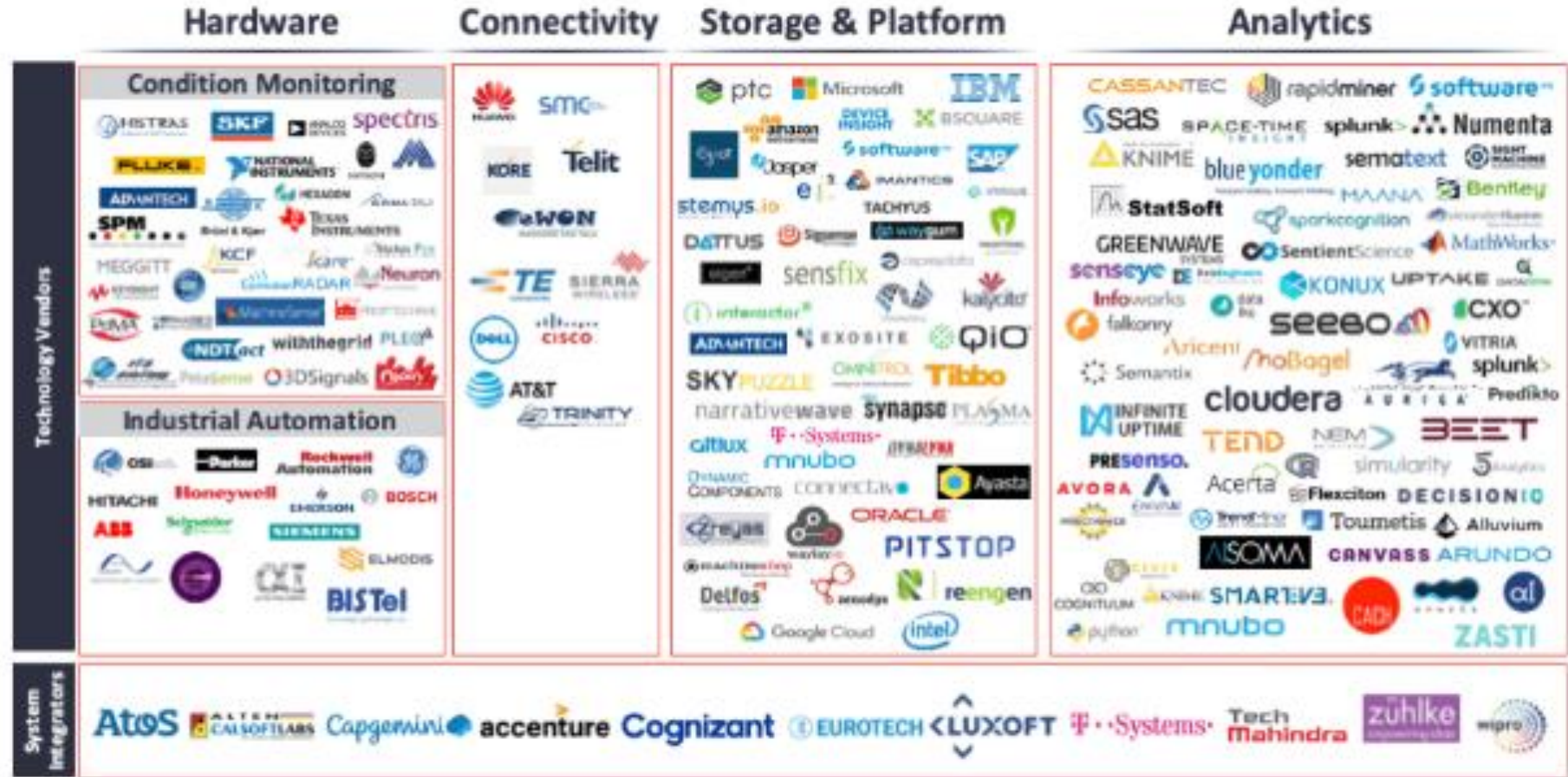

















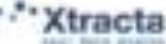


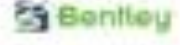












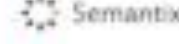




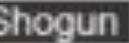


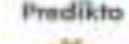






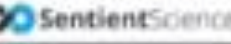
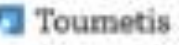

Table 2. Overall vibration velocity guide line for various motor pump assemblies and speeds

Machine Type			
1000+ RPM	ALERT	FAULT	ADVANCED. FAULT
Motor/Pump Horizontal Centrifugal	0.3	0.45	0.6
Motor/Pump Horiz. Belt Driven Centrifugal	0.4	0.6	0.8
Motor /Pump Vertical Centrifugal (<5')	0.3	0.45	0.6
Motor/Pump Vertical Centrifugal (5'<8')	0.4	0.6	0.8
Motor/Pump Vertical Centrifugal (8'<12')	0.5	0.75	1
Motor/Pump Horizontal Hydraulic	0.2	0.3	0.4

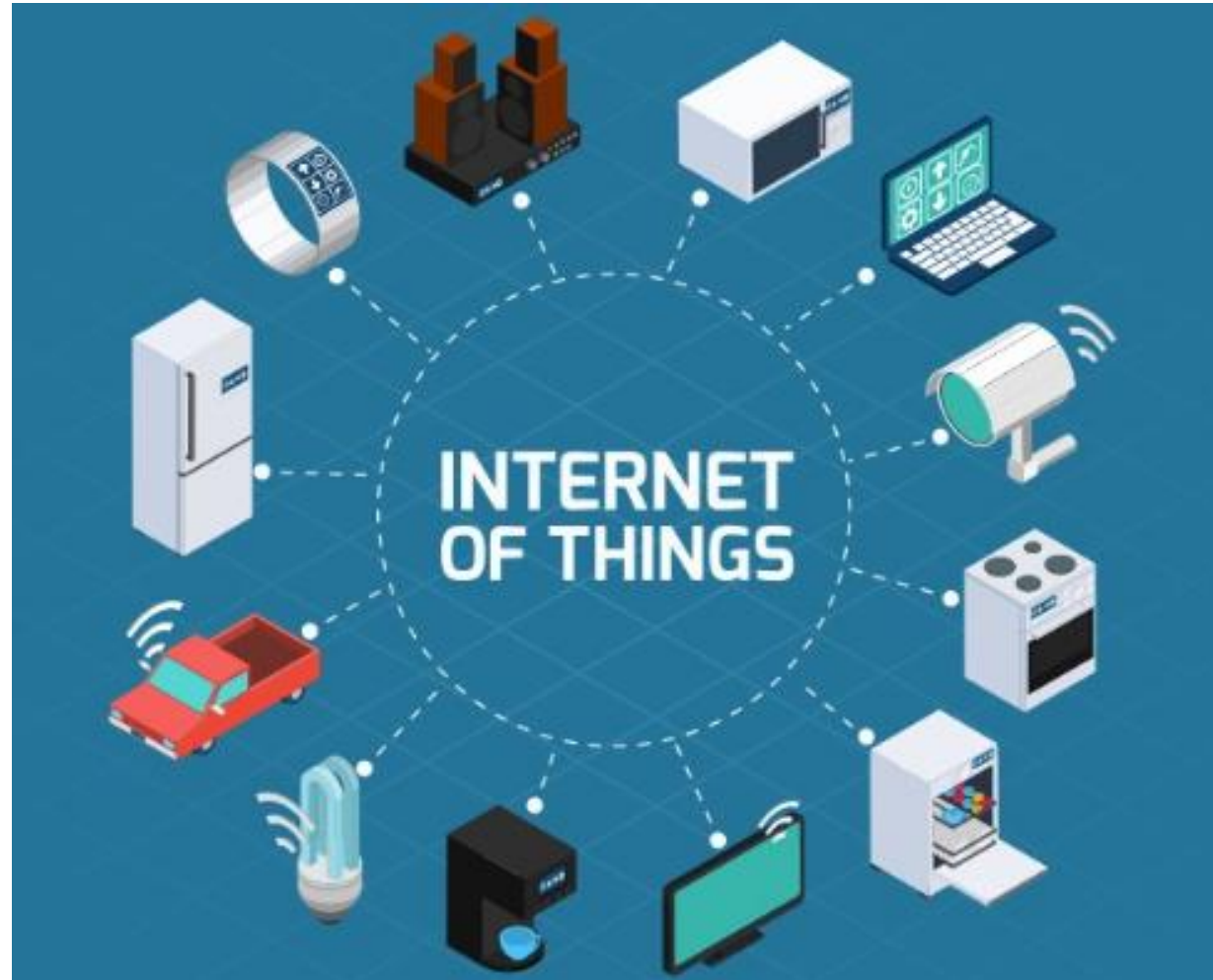
There are many players in the \$38B predictive maintenance market



There are 9 types of maintenance analytics providers

Category		Definition	Specialized Analytics company examples ¹	General Analytics companies ²
Basic Analytics	1 Data Visualisation	Software for creation of dashboards, graphs, etc.	 	   
	2 Analytics Libraries	Statistical and mathematical packages that can be used to build PdM models/software	   	  
Data Engineering	3 Big data Automation	Service/product for automation of data acquisition/data entry process	  	  
	4 Data Engineering	Service/product for data engineering process		  
General Data Science	5 Statistical Analysis	Software for sophisticated statistical analysis	 	  
	6 Data mining	Software for examining large pre-existing databases in order to generate new information	  	  
	7 Machine Learning	Software created specifically for ML applications	  	 
PdM Tailored Analytics	8 Predictive Analytics	Service/product that generates predictions based on machine data	  	  
	9 Anomaly detection	Service/product that detects anomalies in real time	 	 

The internet of things (IoT) highlights massive opportunities for big data and monitoring things



Types of maintenance problems

Expensive machines that have a big impact when break down

Standard things that have a natural life (laptop / fridge)

Cars

Predictable – petrol

Less predictable – fan belt

Buy new milk

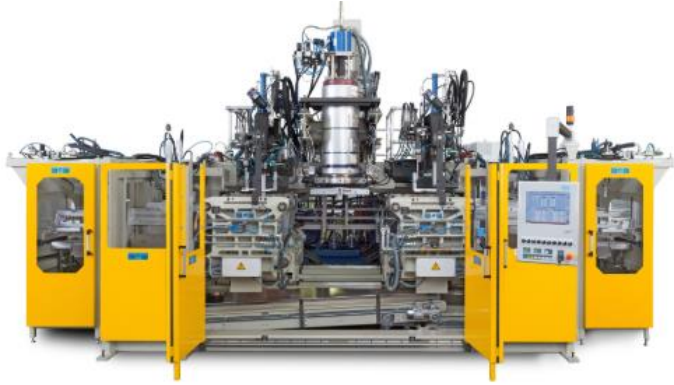
Get new petrol +
Tesla maintenance (car gets
better/faster with software upgrades)

Maintenance agreements (IT equipment)

Computer software upgrades

Coke machine that needs fixing + filling up

Types of maintenance

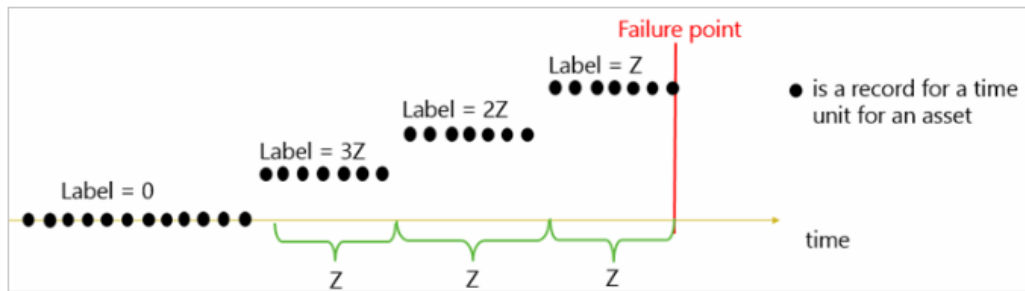


Sensors (often multiple)

Monitor

No -Action

Action



Questions to determine types of failures

Is the failure a sudden event, or is there a slow decline before complete malfunction?

Is every recorded event labelled, i.e. which measurements correspond to good functioning and which ones correspond to failure?

When labelled events are available, what is the proportion of the number of events of each type of failure and events of well functioning?

How long in advance should the model be able to indicate that a failure will occur?

What is the consequence of not predicting a failure or predicting a failure that will not happen?

Which components are typically associated with this type of failure?

Which parameters should be measured that most signify the state of component/machine health?

What is the required accuracy and frequency of the measurements needed?

There are 4 main maintenance challenges

STRATEGY 1: Regression models to predict remaining useful lifetime

How many days/cycles are left before the system fails? Static and historical data are available, and every event is labelled. Several events of each type of failure are present in the dataset.

STRATEGY 2: Classification models to predict failure within a given time window

Often the maintenance team only needs to know if the machine will fail 'soon'



My Dataset

STRATEGY 3: Flagging anomalous behaviour

If you have mission critical systems (eg Plane Crash), in which acute repairs are difficult, there are often only limited, or no examples of failures at all

STRATEGY 4: Survival models for the prediction of failure probability over time

If you however are interested in the degradation process itself and the resulting failure probability, this last strategy suits you best

Predictive maintenance can be formulated in one of the two ways

Classification approach

- predicts whether there is a possibility of failure in next n-steps.

Regression approach

- predicts how much time is left before the next failure.
- known as Remaining Useful Life.

Dataset from Kaggle

- 5 months
- Every 1 minute
- 50 sensors
- 7 break downs

:	NORMAL	205836
	RECOVERING	14477
	BROKEN	7

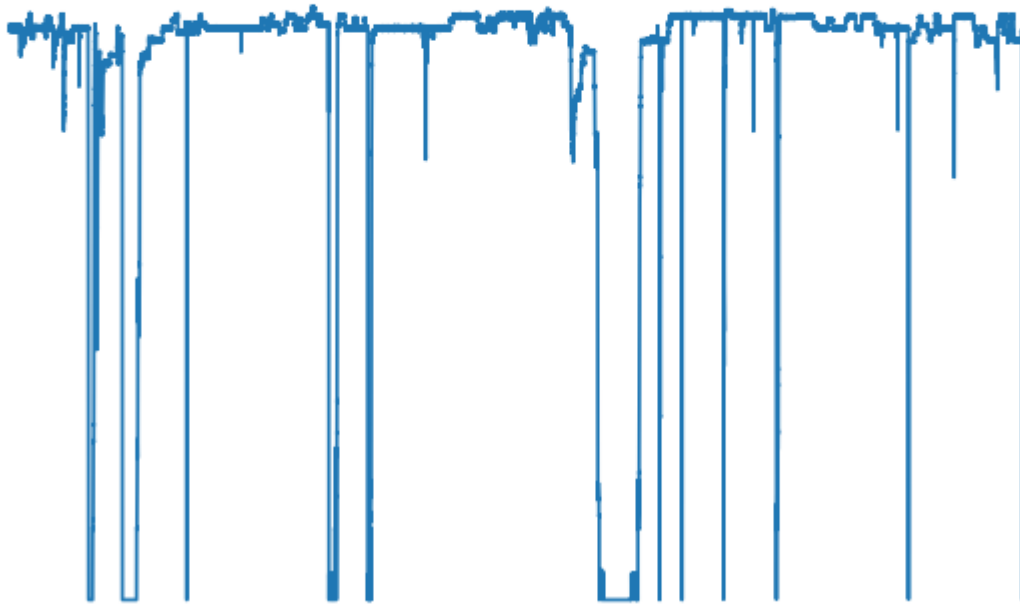
Looking at data before the first failure it was easy to pick

	timestamp	sensor_0 0	sensor_0 1	sensor_0 2	sensor_0 3	sensor_0 4	sensor_0 5	sensor_0 6	sensor_0 7	sensor_0 8	sensor_0 9	sensor_1 0	sensor_1 1	sensor_1 2	sensor_1 3
17147	12/04/2018 21:47	2.28	52.13	52.99	43.40	133.31	55.73	13.23	16.65	15.70	15.12	42.17	25.34	15.21	3.35
17148	12/04/2018 21:48	1.86	52.04	52.95	43.40	131.92	51.53	13.25	16.76	15.78	15.05	41.59	25.81	15.24	3.34
17149	12/04/2018 21:49	1.83	51.95	52.95	43.40	133.54	49.47	13.32	16.70	15.53	15.08	40.37	26.19	15.17	3.35
17150	12/04/2018 21:50	1.53	52.04	52.95	43.40	110.98	46.92	13.17	16.65	15.66	15.09	39.40	28.72	15.24	3.46
17151	12/04/2018 21:51	1.35	52.30	52.95	43.40	110.51	45.22	13.20	16.70	15.89	15.12	41.03	28.34	15.19	3.51
17152	12/04/2018 21:52	1.29	52.52	52.91	43.40	112.60	47.37	13.50	16.65	15.65	15.12	43.65	28.64	15.26	3.52
17153	12/04/2018 21:53	1.22	52.73	52.82	43.40	113.17	47.84	13.25	16.65	15.73	15.05	45.39	28.35	15.24	3.40
17154	12/04/2018 21:54	-	52.95	52.82	43.40	117.11	48.17	5.76	16.70	15.53	14.98	17.73	16.55	6.38	0.22
17155	12/04/2018 21:55	-	53.34	52.82	43.40	202.53	49.79	3.22	16.89	16.87	15.08	35.53	3.63	1.60	0.24

Often machine did not die when Sensor 00 dropped

```
In [100]: 1 df["sensor_00"].plot(figsize=(10,6))  
          2 plt.xticks(color="white")  
          3 plt.yticks(color="white")  
          4 plt.xlabel("Timestamp", color="white")
```

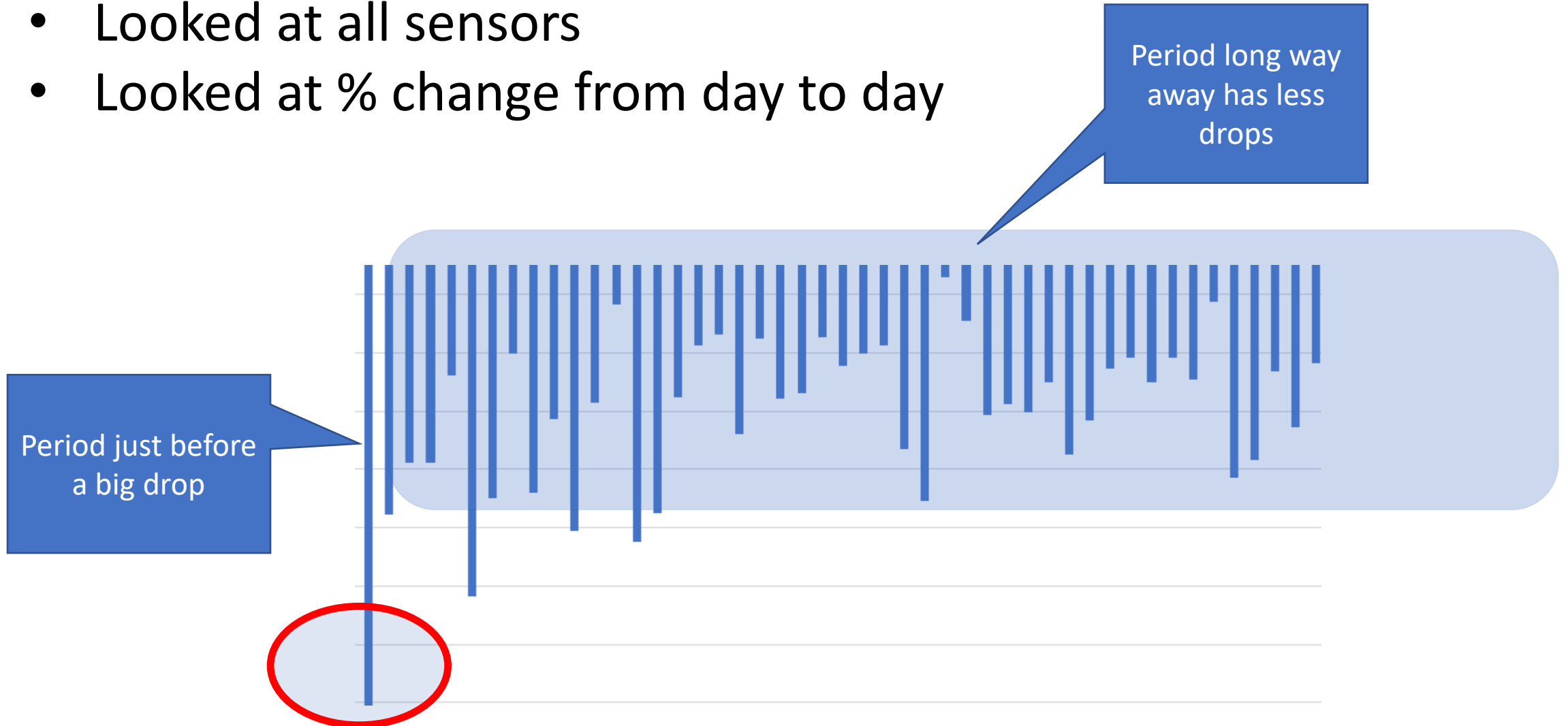
```
Out[100]: Text(0.5, 0, 'Timestamp')
```



Machine only stopped
twice when sensor 00
failed

The average of the sensors

- Looked at all sensors
- Looked at % change from day to day



Fails 1, 6 and to a lesser extent 3 were easy to pick but the others were not

Failure	1	2	3	4	5	6	7
sensor_00	-100%	-1%	0%	-3%	0%	-100%	0%
sensor_01	0%	0%	0%	0%	0%	0%	0%
sensor_02	0%	0%	0%	0%	0%	0%	0%
sensor_03	0%	0%	0%	0%	0%	0%	0%
sensor_04	3%	1%	-57%	-1%	0%	-11%	0%
sensor_05	1%	-6%	-5%	1%	0%	-85%	-2%
sensor_06	-57%	-1%	1%	1%	0%	0%	0%
sensor_07	0%	0%	-1%	0%	0%	0%	-1%
sensor_08	-1%	-1%	2%	1%	0%	0%	-1%
sensor_09	0%	-1%	1%	-1%	-1%	0%	0%
sensor_10	-61%	0%	-2%	-1%	1%	-11%	-2%
sensor_11	-42%	-2%	-2%	1%	2%	-52%	0%
sensor_12	-58%	-2%	-1%	2%	0%	-37%	2%
sensor_13	-93%	-2%	-10%	-5%	0%	122%	0%
sensor_14	0%	-1%	-1%	0%	1%	0%	0%
sensor_15	0%	0%	0%	0%	0%	0%	0%
sensor_16	0%	-1%	-1%	0%	-3%	0%	0%
sensor_17	2%	1%	-2%	-1%	1%	-2%	-1%
sensor_18	5%	2%	-5%	-1%	2%	-4%	-3%
sensor_19	0%	0%	0%	0%	5%	0%	0%
sensor_20	0%	0%	0%	0%	1%	0%	0%
sensor_21	1%	0%	0%	0%	3%	-1%	0%
sensor_22	0%	0%	0%	0%	4%	0%	0%
sensor_23	0%	0%	0%	0%	1%	0%	0%
sensor_24	0%	0%	0%	1%	-3%	0%	0%
sensor_25	0%	0%	0%	0%	-4%	0%	0%
sensor_26	0%	0%	1%	0%	-3%	0%	0%
sensor_27	0%	-1%	7%	4%	10%	0%	5%
sensor_28	2%	1%	0%	0%	-1%	0%	2%
sensor_29	0%	1%	2%	3%	-1%	1%	0%
sensor_30	1%	1%	2%	-1%	2%	-1%	10%
sensor_31	-7%	3%	3%	1%	11%	-1%	1%
sensor_32	1%	-3%	-1%	2%	-6%	2%	-2%
sensor_33	0%	0%	-1%	4%	4%	1%	-1%
sensor_34	-1%	-1%	-1%	-3%	0%	-3%	1%
sensor_35	2%	0%	0%	1%	0%	-1%	0%
sensor_36	2%	-2%	0%	-6%	1%	0%	0%
sensor_37	2%	14%	18%	-2%	62%	18%	37%
sensor_38	4%	-1%	-1%	-2%	0%	-3%	-2%
sensor_39	0%	-2%	-2%	2%	0%	-4%	1%
sensor_40	-3%	1%	1%	3%	-1%	-4%	4%
sensor_41	2%	0%	-1%	2%	0%	-2%	0%
sensor_42	1%	-1%	-1%	3%	0%	-2%	0%
sensor_43	-1%	4%	1%	2%	3%	-3%	2%
sensor_44	-4%	-1%	3%	-5%	0%	-2%	1%
sensor_45	2%	0%	2%	-5%	-1%	-1%	2%
sensor_46	0%	2%	0%	15%	0%	-1%	3%
sensor_47	1%	0%	-1%	20%	0%	-1%	0%
sensor_48	2%	-2%	-2%	1%	0%	1%	-2%
sensor_49	1%	0%	-1%	7%	0%	-1%	0%
sensor_50	4%	-1%	0%	3%	0%	-1%	0%
sensor_51	5%	0%	0%	8%	0%	1%	1%

Failure	1	2	3	4	5	6	7
sensor_00	-100%	-1%	0%	-3%	0%	-100%	0%
sensor_01	0%	0%	0%	0%	0%	0%	0%
sensor_02	0%	0%	0%	0%	0%	0%	0%
sensor_03	0%	0%	0%	0%	0%	0%	0%
sensor_04	3%	1%	-57%	-1%	0%	-11%	0%
sensor_05	1%	-6%	-5%	1%	0%	-85%	-2%
sensor_06	-57%	-1%	1%	1%	0%	0%	0%
sensor_07	0%	0%	-1%	0%	0%	0%	-1%
sensor_08	-1%	-1%	2%	1%	0%	0%	-1%
sensor_09	0%	-1%	1%	-1%	-1%	0%	0%
sensor_10	-61%	0%	-2%	-1%	1%	-11%	-2%
sensor_11	-42%	-2%	-2%	1%	2%	-52%	0%
sensor_12	-58%	-2%	-1%	2%	0%	-37%	2%
sensor_13	-93%	-2%	-10%	-5%	0%	122%	0%
sensor_14	0%	-1%	-1%	0%	1%	0%	0%

NEW APPROACH

Problem 1

- Split the data 80 /20 in a model
- But only 7 fails
- In the test there were no fails

Split 50 /50

Split 2 months Vs 3 months

Problem 2

- Indexing by i in range when the index has been changed to a date format did not work

Do some indexing by original index

Problem 3

- Every time it went down there were different sensors that were 'lead indicators'

Problem 4

- Sometimes sensors just stopped working

Problem 5 and part of the Solution

After the machine was turned on it was on 'Recovery' mode and no problems happened in that period

Ignore Recovery Mode

Problem 6

- All sensors had different variability
- All sensors had different values

1	df.nunique()
sensor_00	1253
sensor_01	831
sensor_02	831
sensor_03	588
sensor_04	7844
sensor_05	190751
sensor_06	812
sensor_07	531
sensor_08	626
sensor_09	565
sensor_10	198804
sensor_11	196368
sensor_12	187145
sensor_13	191983
sensor_14	94564
sensor_16	110522
sensor_17	148000
sensor_18	152603
sensor_19	100422
sensor_20	92129
sensor_21	131083
sensor_22	126401
sensor_23	119286
sensor_24	133778
sensor_25	165999
sensor_26	179718
sensor_27	203198

Col	Max	Min	q_0.25	q_0.50	q_0.75
sensor_00	2.549016	0.000000	2.438831	2.456539	2.499826
sensor_01	56.727430	0.000000	46.310760	48.133678	49.479160
sensor_02	56.032990	33.159720	50.390620	51.649300	52.777770
sensor_03	48.220490	31.640620	42.838539	44.227428	45.312500
sensor_04	800.000000	2.798032	626.620400	632.638916	637.615723
sensor_05	99.999880	0.000000	69.976260	75.576790	80.912150
sensor_06	22.251160	0.014468	13.346350	13.642940	14.539930
sensor_07	23.596640	0.000000	15.907120	16.167530	16.427950
sensor_08	24.348960	0.028935	15.183740	15.494790	15.697340
sensor_09	25.000000	0.000000	15.053530	15.082470	15.118630
sensor_10	76.106860	0.000000	40.705260	44.291340	47.463760
sensor_11	60.000000	0.000000	38.856420	45.363140	49.656540
sensor_12	45.000000	0.000000	28.686810	32.515830	34.939730
sensor_13	31.187550	0.000000	1.538516	2.929809	12.859520
sensor_14	500.000000	32.409550	418.103250	420.106200	420.997100
sensor_16	739.741500	0.000000	459.453400	462.856100	464.302700
sensor_17	599.999939	0.000000	454.138825	462.020250	466.857075
sensor_18	4.873250	0.000000	2.447542	2.533704	2.587682
sensor_19	878.917900	0.000000	662.768975	665.672400	667.146700
--	-----	-----	-----	-----	-----

Problem 7 - Properly understanding complex systems is complicated and required more back info than my data provided

Acceleration – Gs RMS

Severity – Amplitude – In/Sec Peak

Frequency – Cycles per Minute CPM or Hz – Cycles per Second

Displacement – Mills – Peak-Peak:

- Good for determining movement of the machine
- Turning speed vibration levels
- Normally used to measure large sleeve bearing machines
- Severity requires the need to know the frequency

Velocity – In/Sec Peak:

- Used for broad frequency ranges 100 Cpm – 120,000.00 Cpm
- The most common measurement used for machine vibration analysis
- Velocity severity is independent of frequency, which is why it is used by most severity guidelines

NOW THE SOLUTIONS

Initial Cleaning

```
df.drop("Unnamed: 0",axis=1,inplace=True)
```

```
df.drop('sensor_15',axis=1,inplace=True)
```

Inplace = True causes problems
when multiple files

Replaced Nulls

Fixed Categorical Values

```
1 df = pd.get_dummies(data = oringal_df, columns = ['machine_status'], prefix = ['machine_status' ])  
2 df.head()
```

r_45	sensor_46	sensor_47	sensor_48	sensor_49	sensor_50	sensor_51	machine_status_BROKEN	machine_status_NORMAL	machine_status_RECOVERING
3287	50.92593	38.194440	157.9861	67.70834	243.0556	201.3889	0	1	0
3287	50.92593	38.194440	157.9861	67.70834	243.0556	201.3889	0	1	0

Initial Model did not work

```
: 1 confusion_matrix(y_test, y_pred)
: array([[110157,    0],
:        [    3,    0]], dtype=int64)
```

Calculations for Logistic regression

Confusion Matrix had 3 fails
from 3 errors

```
In [99]: 1 y_test.value_counts()
Out[99]: 0    110157
         1         3
         Name: machine_status_BROKEN, dtype: int64
```

```
1 accuracy_score = model.score(X, y)
2 print(accuracy_score)
0.9999682280319535
```

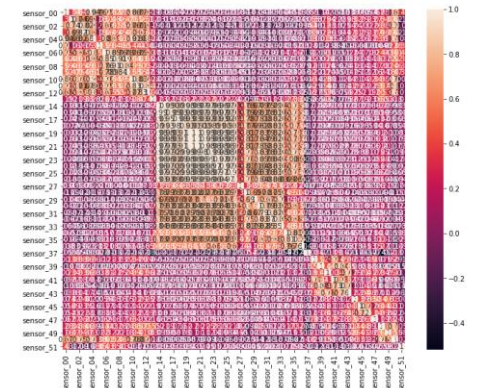
Other results were too
good/bad

```
: 1 recall_score(y_test, y_pred)
: 0.0

: 1 precision_score(y_test, y_pred)
C:\Users\alexnd\anaconda3\lib\site-package
fined and being set to 0.0 due to no pred
_warn_prf(average, modifier, msg_start,
: 0.0
```

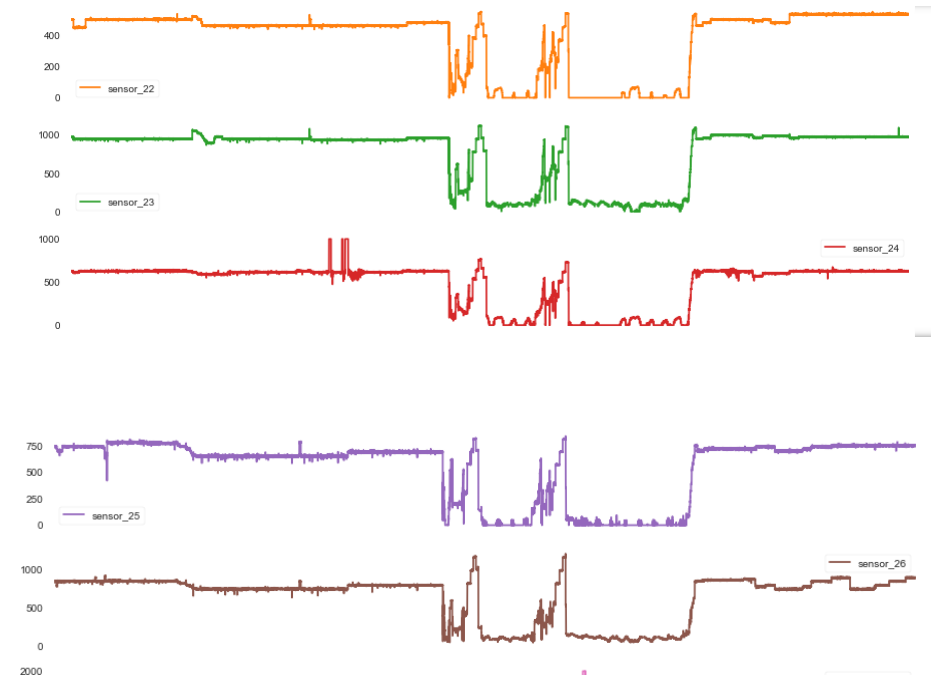
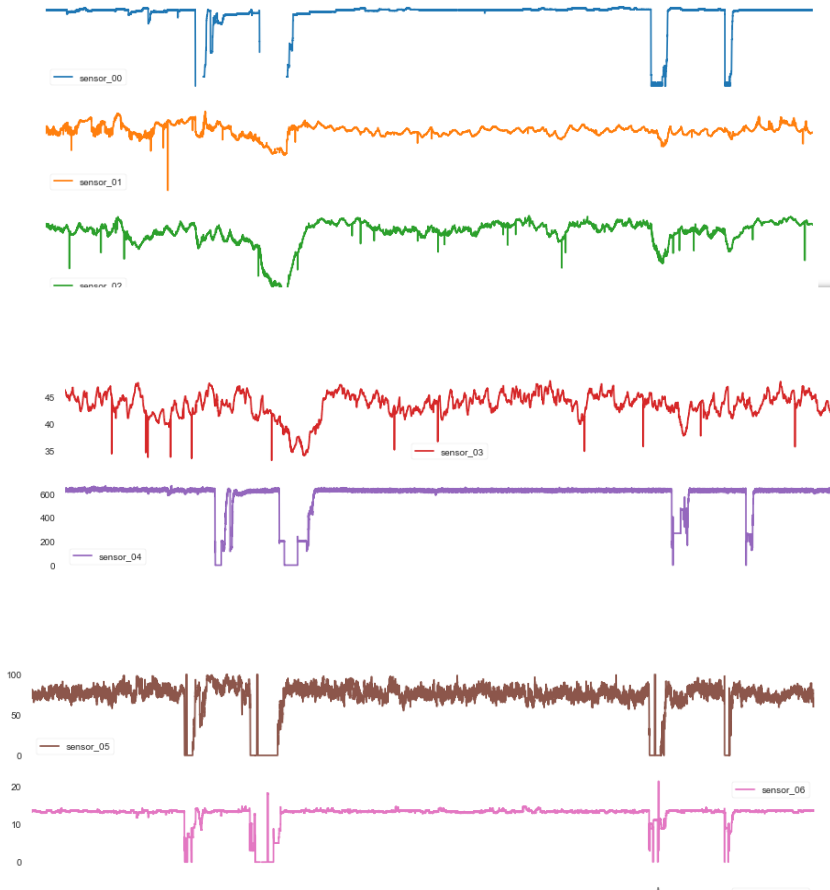

Sensors were often not strongly correlated

	sensor_00	sensor_01	sensor_02	sensor_03	\
sensor_00	1.000000	0.338561	0.641822	0.388877	
sensor_01	0.338561	1.000000	0.737403	0.693155	
sensor_02	0.641822	0.737403	1.000000	0.822339	
sensor_03	0.388877	0.693155	0.822339	1.000000	
sensor_04	0.942803	0.677455	0.820339	0.682469	
sensor_05	0.756697	0.162465	0.261234	0.145979	
sensor_06	0.750808	0.511725	0.661039	0.560201	
sensor_07	0.617905	0.483884	0.610840	0.494384	
sensor_08	0.590243	0.459991	0.553096	0.452998	
sensor_09	0.521154	0.473874	0.595187	0.482817	
sensor_10	0.861392	0.595765	0.756577	0.619876	
sensor_11	0.770751	0.564573	0.711308	0.672560	
sensor_12	0.679083	0.508968	0.686333	0.613184	
sensor_13	0.185194	0.201560	0.192426	0.325954	
sensor_14	-0.033395	0.031302	-0.125641	-0.097223	
sensor_16	-0.034044	0.030538	-0.125195	-0.097154	
sensor_17	-0.024131	0.025515	-0.115516	-0.102450	
sensor_18	-0.007720	0.025889	-0.102621	-0.090027	



Which makes sense

Solution: Print out all 50 sensors from 1 piece of code for any time period



Plus more

Solution:

- Work out when the problems occurred
- Look at a rolling period going back
 - a minute
 - 5 minutes
 - 10 minutes
 - 60 minutes
- Look for FLAGS
- Look at a pattern / correlation
- Run that across all the data
- See if that pattern had false positives

Problem – order was backwards and range going backwards had limits

```
## Warks going backwards for firs 11K or so
counter = 9999999
lenrange = 10000 #len(reduceddf) #100
for i in range(1000,1,-1):
    if reduceddf.loc[i, "machine_status_BROKEN"] > 0:
        print("aa")
        counter = 0
    if counter > 1000:
        reduceddf.loc[i, "TimeSince"] = 999
    else:
        reduceddf.loc[i, "TimeSince"] = counter
        counter = counter +1
```

This took a fair amount of EFFORT

Solution:

In the end it was 1 line of code

```
failure_idx = dfc.loc[df['machine_status'] == 'BROKEN'].index
```

17155	2018-04-12	21:55:00
24510	2018-04-18	00:30:00
69318	2018-05-19	03:18:00
77790	2018-05-25	00:30:00
128040	2018-06-28	22:00:00
141131	2018-07-08	00:11:00
166440	2018-07-25	14:00:00

Apart from 1 they
were late at night

Solution

Worked out the index for any time period beforehand (in this case 10 time periods)

```
[17155, 17154, 17153, 17152, 17151, 17150, 17149, 17148, 17147, 17146, 24510, 24509, 24508, 24507, 24506, 24505, 24504, 24503, 24502, 24501, 69318, 69317, 69316, 69315, 69314, 69313, 69312, 69311, 69310, 69309, 77790, 77789, 77788, 77787, 77786, 77785, 77784, 77783, 77782, 77781, 128040, 128039, 128038, 128037, 128036, 128035, 128034, 128033, 128032, 128031, 141131, 141130, 141129, 141128, 141127, 141126, 141125, 141124, 141123, 141122, 166440, 166439, 166438, 166437, 166436, 166435, 166434, 166433, 166432, 166431]
```

Solution

Deleted a lot of the rows that were not relevant

Thanks

Alex Dance



Background

- Maths / statistics degree
- Background in big data, strategy, analytics
- Worked at Optus, Salmat, Reuters, Pathfinder Solutions

Copy of This Presentation and code

<https://github.com/alexdance2468/>

Plus other data science projects completed

Contact Details

www.linkedin.com/in/alex-dance/

Thanks

Other Sources:

Using data from <https://www.kaggle.com/nphantawee/pump-sensor-data>

From <https://www.kaggle.com/rickantonais/pump-sensor-dataset-eda-and-arma-limits>

From <https://www.kaggle.com/garteb/eda-of-sensors-data-phase-1>