Downtime Classification Problem

January 26, 2017

1 Overview

We have been given all sensor data from an oil rig in a csv file. There are 3 years of data and 74 sensors from many different pieces of equipment on the rig. Each sensor has a reading frequency of 5 minutes so there are a little over 300K values for each sensor. The timestamps are synchronised and there is no null data. We are given two other data files. One contains a list of sensor descriptions and the other contains a list of downtime periods caused by the rig drilling system. The company have asked us to build a machine learning model from the sensor data which will predict future downtimes with high accuracy

First we read the sensor data into memory in a pandas DataFrame:

```
In [1]: import pandas as pd

In [2]: sensors  = pd.read_csv('sensor_data.csv',index_col=0)

In [3]: sensors.head()
```

Out[3]:

	1	2	3	4	5	\
01/01/2013 00:05:00	24.312992	5.871907	7.809708	11.239471	7.835118	
01/01/2013 00:10:00	24.282135	5.863674	7.817646	11.250949	7.883333	
01/01/2013 00:15:00	24.220648	5.848554	7.814390	11.246564	7.948144	
01/01/2013 00:20:00	24.146674	5.830066	7.821645	11.257110	8.173830	
01/01/2013 00:25:00	24.106516	5.820504	7.795507	11.219247	8.094501	

	6	7	8	9	10	\
01/01/2013 00:05:00	1.459977	318.410060	0.689709	2.096998	31.583410	
01/01/2013 00:10:00	1.374517	318.591824	0.667601	2.097452	31.565901	
01/01/2013 00:15:00	1.292265	316.362367	0.674543	2.088330	31.568643	
01/01/2013 00:20:00	1.294447	317.240191	0.637393	2.108070	31.577451	
01/01/2013 00:25:00	1.421029	316.967377	0.637616	2.101541	31.595215	

	...	65	66	67	68	\
01/01/2013 00:05:00	...	0.559139	0.399094	0.476946	0.536601	
01/01/2013 00:10:00	...	0.560733	0.400214	0.478327	0.538118	
01/01/2013 00:15:00	...	0.560146	0.399803	0.477825	0.537577	
01/01/2013 00:20:00	...	0.558037	0.398303	0.476035	0.535562	
01/01/2013 00:25:00	...	0.557630	0.397999	0.475688	0.535159	

```
                              69        70        71        72        73  \
01/01/2013 00:05:00  0.594613  0.481591  0.508189  0.451616  0.515066
01/01/2013 00:10:00  0.596279  0.482996  0.509664  0.452934  0.516572
01/01/2013 00:15:00  0.595655  0.482480  0.509120  0.452451  0.516010
01/01/2013 00:20:00  0.593414  0.480664  0.507197  0.450761  0.514083
01/01/2013 00:25:00  0.592952  0.480330  0.506822  0.450421  0.513682

                               74
01/01/2013 00:05:00  395.875474
01/01/2013 00:10:00  395.808283
01/01/2013 00:15:00  396.086062
01/01/2013 00:20:00  395.768737
01/01/2013 00:25:00  395.574185

[5 rows x 74 columns]
```

Next we read in and inspect the sensor description file seeing if there is anything which can help us identify the sensors related to drilling equipment:

```
In [4]: descriptions = pd.read_csv('sensor_descriptions.csv')
        descriptions
```

```
Out[4]:     ID       tag_prime_tag_sec_tag_tert_tag_quat  \
        0    1         crewqtr_hvac_primaryac_controller
        1    2               crewqtr_hvac_primaryac_vent
        2    3       crewqtr_hvac_secondaryac_controller
        3    4             crewqtr_hvac_secondaryac_vent
        4    5              crewqtr_oven_controler_power
        5    6    crewqtr_oven_secondarycontroller_power
        6    7         crewqtr_comms_internet_controller
        7    8       crewqtr_light_controller_controller
        8    9      navigation_guidance_transponder_main
        9   10         navigation_guidance_recorder_main
        10  11       navigation_guidance_transponder_aux
        11  12          navigation_guidance_recorder_aux
        12  13            navigation_thruster_power_main
        13  14             navigation_thruster_power_aux
        14  15               navigation_thruster_motor_1
        15  16               navigation_thruster_motor_2
        16  17               navigation_thruster_motor_3
        17  18               navigation_thruster_motor_4
        18  19                  drilling_mudpump_1_power
        19  20                  drilling_mudpump_2_power
        20  21                  drilling_mudpump_3_power
        21  22                  drilling_mudpump_4_power
        22  23          drilling_mudpump_1_inletpressure
        23  24         drilling_mudpump_1_outputpressure
        24  25          drilling_mudpump_2_inletpressure
```

25	26	drilling_mudpump_2_outputpressure
26	27	drilling_mudpump_3_inletpressure
27	28	drilling_mudpump_3_outputpressure
28	29	drilling_mudpump_4_inletpressure
29	30	drilling_mudpump_4_outputpressure
..	..	...
44	45	drilling_topdrive_aux_weightonbit
45	46	drilling_topdrive_aux_vibration
46	47	drilling_rotarytable_primary_power
47	48	drilling_rotarytable_primary_geartemp
48	49	drilling_rotarytable_primary_coolanttemp
49	50	drilling_rotarytable_primary_vibration
50	51	deck_crane_main_power
51	52	deck_crane_main_geartemp
52	53	deck_crane_main_vibration
53	54	deck_crane_upper_power
54	55	deck_crane_upper_geartemp
55	56	deck_crane_upper_vibration
56	57	deck_helipad_primary_light
57	58	deck_helipad_secondary_light
58	59	generator_primary_coolant_inlet1
59	60	generator_primary_coolant_inlet2
60	61	generator_primary_coolant_inlet3
61	62	generator_primary_coolant_inlet4
62	63	generator_primary_coolant_temp
63	64	controls_electrical_chair_switch1
64	65	controls_electrical_chair_switch2
65	66	controls_electrical_chair_switch3
66	67	controls_electrical_chair_switch4
67	68	controls_electrical_chair_switch5
68	69	controls_electrical_chair_switch6
69	70	controls_electrical_chair_switch7
70	71	controls_electrical_chair_switch8
71	72	controls_electrical_chair_switch9
72	73	controls_electrical_chair_switch10
73	74	productivity_personnel_scheduled_time

	description
0	This is the primary air conditioner controller...
1	This is the vent air speed on the primary system
2	This is the secondary air conditioner controll...
3	This is the vent air speed on the secondary sy...
4	This records the number of amps running throug...
5	This records the number of amps running throug...
6	This relates the number of megabits per second...
7	this records the number of amps running throug...
8	The signal amplitude of the main transponder
9	The datarate to the main recorder in kilobits ...

10	The signal amplitude of the secondary transponder
11	The datarate of the secondary recorder in kilo...
12	Power in watts of main thruster power system
13	Power in watts of the aux thruster power system
14	RPM of thruster 1
15	RPM of thruster 2
16	RPM of thruster 3
17	RPM of thruster 4
18	Amps to Mudpump 1
19	Amps to Mudpump 2
20	Amps to Mudpump 3
21	Amps to Mudpump 4
22	pascals at inlet of pump 1
23	pascals at outlet of pump 1
24	pascals at inlet of pump 2
25	pascals at outlet of pump 2
26	pascals at inlet of pump 3
27	pascals at outlet of pump 3
28	pascals at inlet of pump 4
29	pascals at outlet of pump 4
..	...
44	Weight on bit in tons for the secondary top drive
45	Amplitude of 20 hertz in dB of the secondary t...
46	The power in Amps to the rotary table
47	geartemp in C of the rotary table
48	coolant temp in C of the rotary table
49	Amplitude of 20 hertz in dB of the rotary table
50	Power to the deck crane in amps
51	geartemp in C of the main deck crane
52	Amplitude of 20 hertz in dB of the main crane
53	Power to the upper deck crane in amps
54	geartemp in C of the upper deck crane
55	Amplitude of 20 hertz in dB of the upper crane
56	Amps running through the primary helipad light
57	Amps running through the secondary helipad light
58	inlet flow 1
59	inlet flow 2
60	inlet flow 3
61	inlet flow 4
62	coolant temperature
63	switch1
64	switch2
65	switch3
66	switch4
67	switch5
68	switch6
69	switch7
70	switch8

```
71                                                 switch9
72                                                switch10
73  total minutes worked on scheduled work during ...

[74 rows x 3 columns]
```

We notice that the variable 'tag_prime_tag_sec_tag_tert_tag_quat' begins 'drilling_' for the sensors we want, we also notice a field 'ID' which we think might correspond to the column labels in the sensor data file. We check with a data expert from the company who confirms this is the case. This means we can map between these files to extract only the drilling sensors:

```
In [5]: #create a list of sensor ids which contain 'drilling'
        taglabel='tag_prime_tag_sec_tag_tert_tag_quat'
        drillingsensors =\
        list(descriptions[descriptions[taglabel].str.contains('drilling') ]['ID'])
        print(drillingsensors)

        #we also need these as strings to index the correct columns
        # in a pandas DataFrame
        drillingsensors_str = [str(x) for x in drillingsensors]
        print(drillingsensors_str)
        print( len(drillingsensors) )
```

```
[19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39
['19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '31', '32'
32
```

With this list we can select the subset of the sensor data which are the drilling equipment sensors:

```
In [6]: #selects the subset of drilling sensors
        drillingdata = sensors.loc[:,drillingsensors_str]
        #make sure pandas understands the index is a datetime quantity
        drillingdata.index = pd.DatetimeIndex(drillingdata.index)
```

```
In [7]: drillingdata.head()
```

Out[7]:

	19	20	21	22	23
2013-01-01 00:05:00	24.016932	9.234189	0.000000	21.948293	1.737437e+06
2013-01-01 00:10:00	24.000159	9.001563	0.024225	21.937836	1.724327e+06
2013-01-01 00:15:00	24.005739	8.668652	0.000000	21.934589	1.715617e+06
2013-01-01 00:20:00	24.028736	8.672117	0.000000	22.057929	1.714565e+06
2013-01-01 00:25:00	24.009215	8.289671	0.048713	22.251114	1.700305e+06

	24	25	26	
2013-01-01 00:05:00	0.000000	1.545816e+06	228823.956645	466841.7771
2013-01-01 00:10:00	0.000000	1.538575e+06	256025.737421	0.0000
2013-01-01 00:15:00	0.000000	1.558320e+06	231624.652333	1580.8948

```
2013-01-01 00:20:00  13472.237834  1.550577e+06  212466.219113       0.0000
2013-01-01 00:25:00  19371.572013  1.574187e+06  214457.018050       0.0000

                                28    ...             41          42  \
2013-01-01 00:05:00  2.227699e+06    ...      37.942339  904.554105
2013-01-01 00:10:00  0.000000e+00    ...      37.871045  903.284684
2013-01-01 00:15:00  8.555061e+05    ...      37.854591  904.202073
2013-01-01 00:20:00  0.000000e+00    ...      37.930387  904.659663
2013-01-01 00:25:00  0.000000e+00    ...      38.203840  903.980399

                            43         44         45         46          47
2013-01-01 00:05:00  62.314463  27.470602  80.039960  30.271639  252.732733
2013-01-01 00:10:00  62.260637  27.481201  80.356735  30.171009  255.431297
2013-01-01 00:15:00  62.246310  27.480470  80.350949  30.117722  258.983661
2013-01-01 00:20:00  62.262458  27.472812  80.596841  30.145673  254.731489
2013-01-01 00:25:00  62.168570  27.513777  81.157459  30.167784  252.162817

                            48         49         50
2013-01-01 00:05:00  41.087693  27.046697  49.457311
2013-01-01 00:10:00  41.125906  27.050496  49.387468
2013-01-01 00:15:00  41.090406  27.058603  49.209248
2013-01-01 00:20:00  41.137623  27.051487  49.155822
2013-01-01 00:25:00  41.161566  27.061355  48.948834

[5 rows x 32 columns]
```

The last piece of information is the file listing downtimes which we will use to label the data so lets read that in and inspect it:

```
In [8]: downtimes = pd.read_csv('downtimes.csv')
        downtimes
```

Out[8]:

	ID	Description
0	1.0	top drive gearing loose; needed repair
1	2.0	top drive failure
2	3.0	top drive coolant leak - conduit 2
3	4.0	top drive coolant leak - conduit 2
4	5.0	top drive gearing pin sheared off; led to larg...
5	6.0	top drive gearing wear
6	7.0	top drive gear pin missing
7	8.0	top drive gearing misaligned
8	9.0	top drive coolant leak - conduit 2
9	10.0	top drive gearing pin sheared off
10	11.0	top drive gearing wear
11	12.0	top drive gear pin missing
12	13.0	top drive gearing misaligned
13	14.0	top drive housing crack
14	15.0	top drive housing crack

15	16.0	top drive power failure
16	17.0	top drive power failure
17	18.0	top drive primary power failure
18	19.0	restart needed
19	20.0	restart needed
20	NaN	hidden
21	NaN	hidden
22	NaN	hidden
23	NaN	hidden
24	NaN	hidden
25	21.0	top drive coolant leak - conduit 4
26	22.0	top drive gearing pin sheared off; led to larg...
27	23.0	top drive gearing wear
28	NaN	hidden
29	24.0	top drive gear pin missing
..	...	...
135	NaN	hidden
136	NaN	hidden
137	NaN	hidden
138	NaN	hidden
139	NaN	hidden
140	NaN	hidden
141	NaN	hidden
142	NaN	hidden
143	NaN	hidden
144	NaN	hidden
145	NaN	hidden
146	NaN	hidden
147	NaN	hidden
148	NaN	hidden
149	NaN	hidden
150	NaN	hidden
151	NaN	hidden
152	NaN	hidden
153	NaN	hidden
154	NaN	hidden
155	NaN	hidden
156	NaN	hidden
157	NaN	hidden
158	NaN	hidden
159	NaN	hidden
160	NaN	hidden
161	NaN	hidden
162	NaN	hidden
163	NaN	hidden
164	NaN	hidden

	system	issue_starts	affects	end_of_issue

0	topdrive_primary	11/24/2014 08:15:00	41	11/24/2014 20:35:00	
1	topdrive_primary	04/01/2015 11:45:00	0	04/02/2015 16:45:00	
2	topdrive_primary	09/06/2015 12:30:00	39	09/07/2015 06:00:00	
3	topdrive_primary	12/24/2013 03:15:00	39	12/25/2013 02:25:00	
4	topdrive_primary	12/24/2014 21:15:00	41	12/26/2014 07:45:00	
5	topdrive_primary	01/19/2015 20:00:00	38	01/20/2015 12:30:00	
6	topdrive_primary	04/17/2014 13:45:00	38	04/18/2014 19:15:00	
7	topdrive_primary	05/08/2015 00:00:00	38	05/08/2015 17:25:00	
8	topdrive_primary	04/19/2015 17:45:00	0	04/20/2015 23:30:00	
9	topdrive_primary	05/16/2013 05:45:00	40	05/17/2013 16:00:00	
10	topdrive_primary	06/23/2015 02:00:00	40	06/24/2015 09:45:00	
11	topdrive_primary	04/23/2013 21:15:00	40	04/25/2013 02:15:00	
12	topdrive_primary	09/28/2015 10:45:00	40	09/29/2015 02:10:00	
13	topdrive_primary	11/17/2013 21:45:00	40	11/19/2013 05:45:00	
14	topdrive_primary	02/24/2013 16:00:00	0	02/26/2013 03:40:00	
15	topdrive_primary	12/08/2013 16:30:00	38	12/09/2013 11:30:00	
16	topdrive_primary	01/08/2013 01:15:00	0	01/09/2013 06:00:00	
17	topdrive_primary	07/06/2013 21:45:00	41	07/08/2013 06:25:00	
18	topdrive_primary	09/30/2014 12:00:00	39	10/01/2014 06:25:00	
19	topdrive_primary	02/14/2013 05:45:00	40	02/15/2013 05:00:00	
20	NaN	02/23/2014 17:30:00	41	02/24/2014 05:30:00	
21	NaN	03/22/2013 19:30:00	41	03/23/2013 16:45:00	
22	NaN	11/21/2013 20:45:00	38	11/23/2013 04:40:00	
23	NaN	06/08/2015 01:30:00	39	06/08/2015 23:10:00	
24	NaN	10/14/2014 22:30:00	40	10/16/2014 03:30:00	
25	topdrive_secondary	12/04/2013 10:00:00	0	12/05/2013 20:45:00	
26	topdrive_secondary	01/20/2013 09:45:00	0	01/21/2013 21:10:00	
27	topdrive_secondary	12/01/2013 22:30:00	43	12/03/2013 06:00:00	
28	NaN	12/01/2013 22:30:00	44	12/03/2013 06:00:00	
29	topdrive_secondary	07/27/2015 14:30:00	43	07/28/2015 10:15:00	
..	...	...	...	...	
135	NaN	07/06/2013 09:45:00	74	07/09/2013 04:30:00	
136	NaN	08/06/2013 23:30:00	74	08/11/2013 11:25:00	
137	NaN	09/10/2013 04:00:00	74	09/13/2013 20:10:00	
138	NaN	10/03/2013 03:00:00	74	10/07/2013 13:45:00	
139	NaN	11/02/2013 18:45:00	74	11/05/2013 15:30:00	
140	NaN	12/05/2013 22:00:00	74	12/07/2013 01:45:00	
141	NaN	01/07/2014 20:00:00	74	01/10/2014 00:30:00	
142	NaN	02/03/2014 22:00:00	74	02/06/2014 16:50:00	
143	NaN	03/04/2014 03:15:00	74	03/06/2014 09:50:00	
144	NaN	04/02/2014 10:45:00	74	04/06/2014 08:50:00	
145	NaN	05/07/2014 15:30:00	74	05/08/2014 13:25:00	
146	NaN	06/08/2014 02:00:00	74	06/11/2014 15:30:00	
147	NaN	07/09/2014 08:30:00	74	07/13/2014 04:50:00	
148	NaN	08/03/2014 00:00:00	74	08/03/2014 22:00:00	
149	NaN	09/09/2014 12:45:00	74	09/13/2014 03:55:00	
150	NaN	10/05/2014 08:30:00	74	10/08/2014 23:05:00	
151	NaN	11/05/2014 05:15:00	74	11/08/2014 07:55:00	

```
152                NaN  12/07/2014 12:15:00       74  12/10/2014 16:45:00
153                NaN  01/02/2015 18:45:00       74  01/03/2015 17:55:00
154                NaN  02/09/2015 21:15:00       74  02/13/2015 18:15:00
155                NaN  03/03/2015 16:30:00       74  03/04/2015 06:55:00
156                NaN  04/07/2015 03:00:00       74  04/09/2015 08:20:00
157                NaN  05/07/2015 05:00:00       74  05/09/2015 22:40:00
158                NaN  06/06/2015 08:00:00       74  06/10/2015 04:30:00
159                NaN  07/06/2015 11:45:00       74  07/09/2015 16:45:00
160                NaN  08/01/2015 08:15:00       74  08/03/2015 20:10:00
161                NaN  09/02/2015 14:45:00       74  09/03/2015 11:35:00
162                NaN  10/05/2015 22:30:00       74  10/09/2015 08:00:00
163                NaN  11/07/2015 18:00:00       74  11/09/2015 06:05:00
164                NaN  12/09/2015 16:30:00       74  12/13/2015 11:15:00
```

```
          issue_reported
0    11/24/2014 20:15:00
1    04/01/2015 23:45:00
2    09/07/2015 00:30:00
3    12/24/2013 15:15:00
4    12/25/2014 09:15:00
5    01/20/2015 08:00:00
6    04/18/2014 01:45:00
7    05/08/2015 12:00:00
8    04/20/2015 05:45:00
9    05/16/2013 17:45:00
10   06/23/2015 14:00:00
11   04/24/2013 09:15:00
12   09/28/2015 22:45:00
13   11/18/2013 09:45:00
14   02/25/2013 04:00:00
15   12/09/2013 04:30:00
16   01/08/2013 13:15:00
17   07/07/2013 09:45:00
18   10/01/2014 00:00:00
19   02/14/2013 17:45:00
20   02/24/2014 05:30:00
21   03/23/2013 07:30:00
22   11/22/2013 08:45:00
23   06/08/2015 13:30:00
24   10/15/2014 10:30:00
25   12/04/2013 22:00:00
26   01/20/2013 21:45:00
27   12/02/2013 10:30:00
28   12/02/2013 10:30:00
29   07/28/2015 02:30:00
..                   ...
135  07/06/2013 21:45:00
136  08/07/2013 11:30:00
```

```
137  09/10/2013 16:00:00
138  10/03/2013 15:00:00
139  11/03/2013 06:45:00
140  12/06/2013 10:00:00
141  01/08/2014 08:00:00
142  02/04/2014 10:00:00
143  03/04/2014 15:15:00
144  04/02/2014 22:45:00
145  05/08/2014 03:30:00
146  06/08/2014 14:00:00
147  07/09/2014 20:30:00
148  08/03/2014 12:00:00
149  09/10/2014 00:45:00
150  10/05/2014 20:30:00
151  11/05/2014 17:15:00
152  12/08/2014 00:15:00
153  01/03/2015 06:45:00
154  02/10/2015 09:15:00
155  03/04/2015 04:30:00
156  04/07/2015 15:00:00
157  05/07/2015 17:00:00
158  06/06/2015 20:00:00
159  07/06/2015 23:45:00
160  08/01/2015 20:15:00
161  09/03/2015 02:45:00
162  10/06/2015 10:30:00
163  11/08/2015 06:00:00
164  12/10/2015 04:30:00

[165 rows x 7 columns]
```

We want to label all timestamps between 'issue_starts' and 'end_of_issue' as downtime. We are told by the company data expert that the column 'affects' corresponds to the sensor ID, so we need to extract the subset of downtimes corresponding to drilling sensors. We can use our integer list of drilling sensor ids to do this:

```
In [9]: #some pandas trickry to give us a dataframe with start
        #and end times only for drilling related downtimes
        periodstolabel = \
        downtimes[downtimes['affects'].isin(drillingsensors)]\
        .loc[:,['issue_starts','end_of_issue']]

In [10]: periodstolabel

Out[10]:            issue_starts         end_of_issue
         0    11/24/2014 08:15:00  11/24/2014 20:35:00
         2    09/06/2015 12:30:00  09/07/2015 06:00:00
         3    12/24/2013 03:15:00  12/25/2013 02:25:00
         4    12/24/2014 21:15:00  12/26/2014 07:45:00
```

5	01/19/2015 20:00:00	01/20/2015 12:30:00
6	04/17/2014 13:45:00	04/18/2014 19:15:00
7	05/08/2015 00:00:00	05/08/2015 17:25:00
9	05/16/2013 05:45:00	05/17/2013 16:00:00
10	06/23/2015 02:00:00	06/24/2015 09:45:00
11	04/23/2013 21:15:00	04/25/2013 02:15:00
12	09/28/2015 10:45:00	09/29/2015 02:10:00
13	11/17/2013 21:45:00	11/19/2013 05:45:00
15	12/08/2013 16:30:00	12/09/2013 11:30:00
17	07/06/2013 21:45:00	07/08/2013 06:25:00
18	09/30/2014 12:00:00	10/01/2014 06:25:00
19	02/14/2013 05:45:00	02/15/2013 05:00:00
20	02/23/2014 17:30:00	02/24/2014 05:30:00
21	03/22/2013 19:30:00	03/23/2013 16:45:00
22	11/21/2013 20:45:00	11/23/2013 04:40:00
23	06/08/2015 01:30:00	06/08/2015 23:10:00
24	10/14/2014 22:30:00	10/16/2014 03:30:00
27	12/01/2013 22:30:00	12/03/2013 06:00:00
28	12/01/2013 22:30:00	12/03/2013 06:00:00
29	07/27/2015 14:30:00	07/28/2015 10:15:00
30	07/27/2015 14:30:00	07/28/2015 10:15:00
31	06/27/2015 21:45:00	06/29/2015 03:50:00
32	10/13/2013 03:15:00	10/14/2013 13:55:00
33	10/07/2014 06:15:00	10/08/2014 02:25:00
34	08/10/2013 18:15:00	08/11/2013 17:25:00
38	10/18/2015 02:15:00	10/18/2015 23:05:00
..	...	...
94	07/09/2015 02:45:00	07/09/2015 14:55:00
96	11/10/2013 23:00:00	11/12/2013 01:00:00
97	05/18/2014 15:15:00	05/19/2014 09:55:00
98	08/02/2015 02:45:00	08/02/2015 17:20:00
99	11/09/2014 14:45:00	11/10/2014 09:55:00
100	04/24/2014 11:30:00	04/25/2014 01:10:00
101	04/17/2015 18:00:00	04/18/2015 23:10:00
103	10/31/2013 23:00:00	11/02/2013 01:35:00
104	09/22/2015 04:45:00	09/23/2015 04:20:00
105	07/01/2015 18:30:00	07/02/2015 19:35:00
106	08/27/2015 15:45:00	08/28/2015 13:15:00
107	10/19/2013 16:45:00	10/20/2013 09:15:00
109	01/14/2013 08:00:00	01/15/2013 08:55:00
110	03/11/2013 08:00:00	03/12/2013 09:50:00
111	11/04/2013 04:30:00	11/05/2013 14:10:00
112	02/26/2014 09:15:00	02/27/2014 10:35:00
113	06/19/2013 02:15:00	06/20/2013 12:05:00
114	04/10/2015 07:15:00	04/11/2015 08:15:00
117	01/19/2015 15:45:00	01/21/2015 01:10:00
118	12/18/2015 11:15:00	12/19/2015 01:20:00
119	12/25/2014 00:00:00	12/25/2014 23:00:00

```
120  02/22/2014 00:15:00  02/22/2014 18:30:00
121  05/09/2013 04:00:00  05/10/2013 05:45:00
122  06/15/2014 02:45:00  06/16/2014 12:15:00
123  07/21/2015 15:00:00  07/22/2015 19:05:00
124  12/03/2015 19:00:00  12/05/2015 03:25:00
125  05/02/2014 18:00:00  05/03/2014 15:40:00
126  08/27/2013 11:45:00  08/28/2013 16:35:00
127  06/14/2014 03:45:00  06/15/2014 13:50:00
128  07/12/2013 04:15:00  07/13/2013 01:50:00

[104 rows x 2 columns]
```

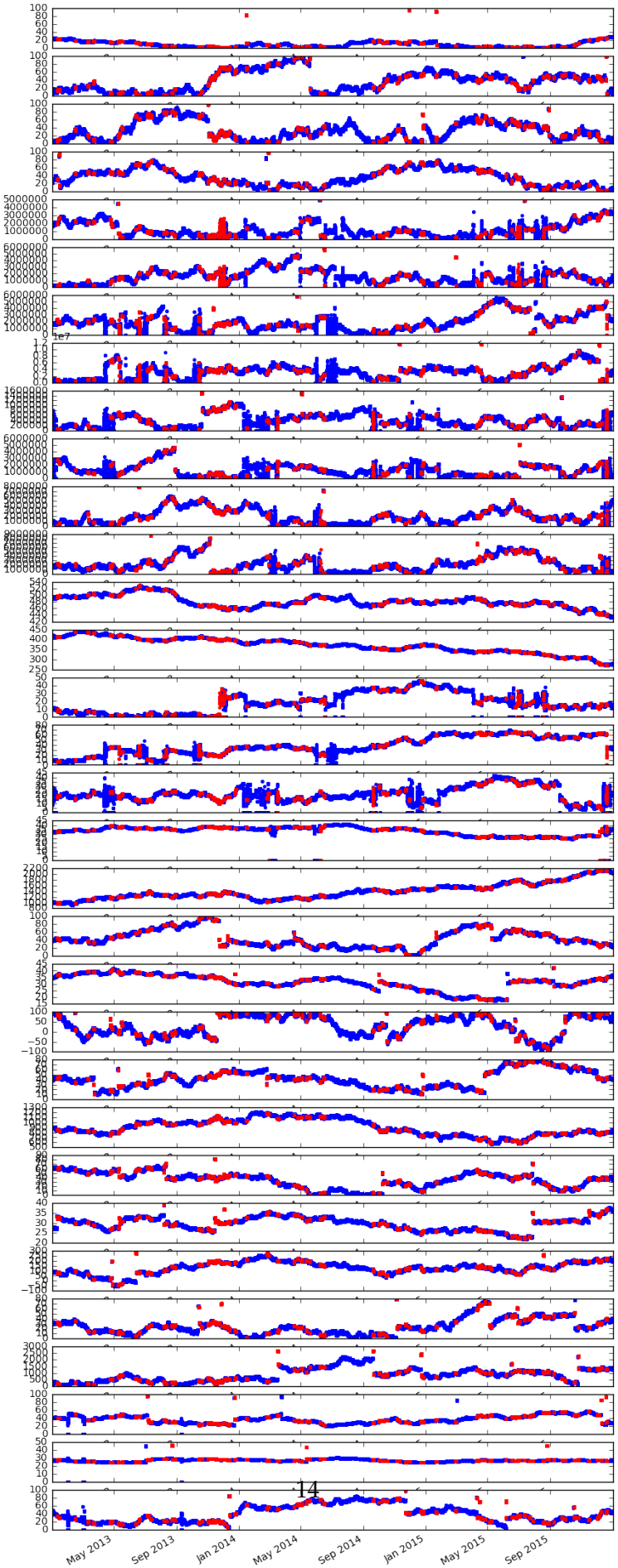
104/165 downtimes were due to the drilling system. We can add a column to the drilling sensor dataframe based on the above time windows:

```
In [11]: #set all labels to 0 initially
         drillingdata['downtime_label']=0


         start_lbl='issue_starts'
         end_lbl='end_of_issue'
         #for each downtime range create date_range object (5min freq)
         #and set downtime_label to 1 for those times
         for i,row in periodstolabel.iterrows():
             tseries = pd.date_range(start=pd.Timestamp(row[start_lbl]),\
                                     end=pd.Timestamp(row[end_lbl]),freq='5t')
             drillingdata.loc[drillingdata.index.normalize().isin(tseries),\
                              'downtime_label']=1

In [12]: print(len(drillingdata[drillingdata['downtime_label']==0]))
         print(len(drillingdata[drillingdata['downtime_label']==1]))

286271
29088
```

About 10% of the timestamps are labeled as downtime caused by drilling equipment. Before we build a model to predict those downtimes let's do a visual inspection of the sensor data with the labels applied. For this we will use matplotlib.pyplot:

```
In [13]: import matplotlib.pyplot as plt
         %matplotlib inline

         fig,axes = plt.subplots(32,1,figsize=(10,30))
         iaxis=0
         for ilabel in drillingsensors_str:
             drillingdata[drillingdata['downtime_label']==0][ilabel]\
             .plot(style='.b',ax=axes[iaxis])
             drillingdata[drillingdata['downtime_label']==1][ilabel]\
```

```
.plot(style='.r',ax=axes[iaxis])
iaxis+=1
```


May 2013
Sep 2013
Jan 2014
May 2014
Sep 2014
Jan 2015
May 2015
Sep 2015

There may be some interesting features we can see related to downtime in these plots and there may be some other visualisations we would like to do with more time (look up pandas.tools.plotting.scatter_matrix if you have time). But for now let's do a quick iteration for building a classification model. We will use scikit-learn random forest essentially out of the box as a first iteration. We will try two training strategies. First we will split the data randomly into train and test samples. Second we will use the first 2/3 of the data to train and then the remaining 1/3 to test.

Let's extraxt the input features and output classes of the model we want from the dataframe and normalized the input features:

```
In [14]: from sklearn.preprocessing import normalize

In [15]: # extract dataframe to numpy format
         Xdata = drillingdata.loc[:,drillingsensors_str].as_matrix()
         Ydata = drillingdata['downtime_label'].values

         #normalize the Xdata in place
         normalize(Xdata,copy=False)
         pass
```

For the first training strategy we can use scikit to do the data random split for test and train samples:

```
In [16]: from sklearn.model_selection import train_test_split

In [17]: Xtrain_rdm, Xtest_rdm, Ytrain_rdm, Ytest_rdm = \
         train_test_split(Xdata,Ydata,test_size=0.33,random_state=42)
```

Now lets book and train the random forest:

```
In [18]: from sklearn.ensemble import RandomForestClassifier

         rf_rdm = RandomForestClassifier()
         rf_rdm.fit(Xtrain_rdm,Ytrain_rdm)

Out[18]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini'
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_split=1e-07, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=10, n_jobs=1, oob_score=False, random_state=None,
                     verbose=0, warm_start=False)
```

Now we can predict with the test data. Before assessing the model performance let's also do the same thing for our sequence wise split of the test and train data:

```
In [19]: #predict classes for the random selected test data
         Ypred_rdm = rf_rdm.predict(Xtest_rdm)
```

```
#also train then get predicted classes for the sequence-wise split
Xtrain_split = Xdata[:211290,:]
Xtest_split = Xdata[211290:,:]
Ytrain_split = Ydata[:211290]
Ytest_split = Ydata[211290:]

rf_split = RandomForestClassifier()
rf_split.fit(Xtrain_split,Ytrain_split)
Ypred_split = rf_split.predict(Xtest_split)
```

Now we have two sets of predictions based on our two training strategies let's assess their performance by seeing how well they predicted the respective test data. We will print out the confusion matrix and make a report in each case.

```
In [20]: from sklearn.metrics import confusion_matrix,classification_report

In [21]: # confusion matrix and report for model with random test/train split
         print(confusion_matrix(Ytest_rdm,Ypred_rdm))

[[94428    48]
 [  102  9491]]


In [22]: print(classification_report(Ytest_rdm,Ypred_rdm))

             precision    recall  f1-score   support

          0       1.00      1.00      1.00     94476
          1       0.99      0.99      0.99      9593

avg / total       1.00      1.00      1.00    104069



In [23]: # confusion matrix and report for model with sequence-wise
         # test/train split
         print(confusion_matrix(Ytest_split,Ypred_split))

[[90726  3551]
 [ 9659   133]]


In [24]: print(classification_report(Ytest_split,Ypred_split))

             precision    recall  f1-score   support

          0       0.90      0.96      0.93     94277
          1       0.04      0.01      0.02      9792
```

```
avg / total       0.82      0.87      0.85    104069
```

What conclusions can we draw about our training strategies?