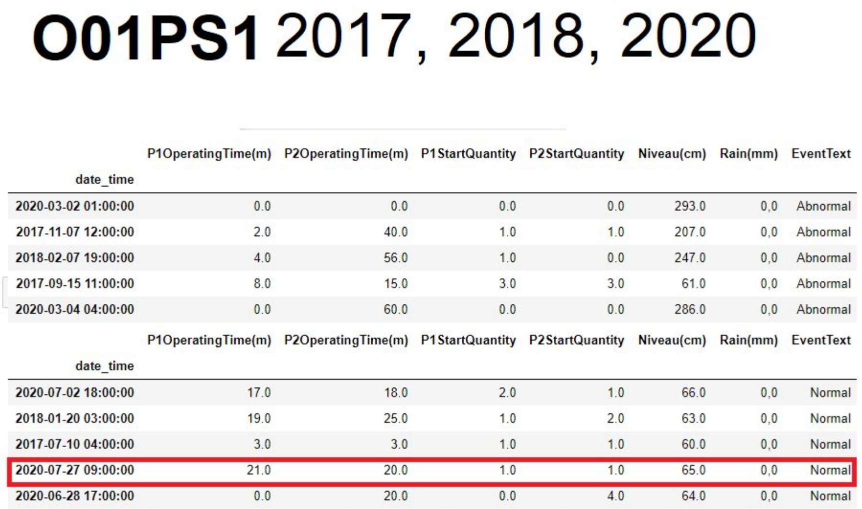
Black Box Testing:

Labeling pump data test

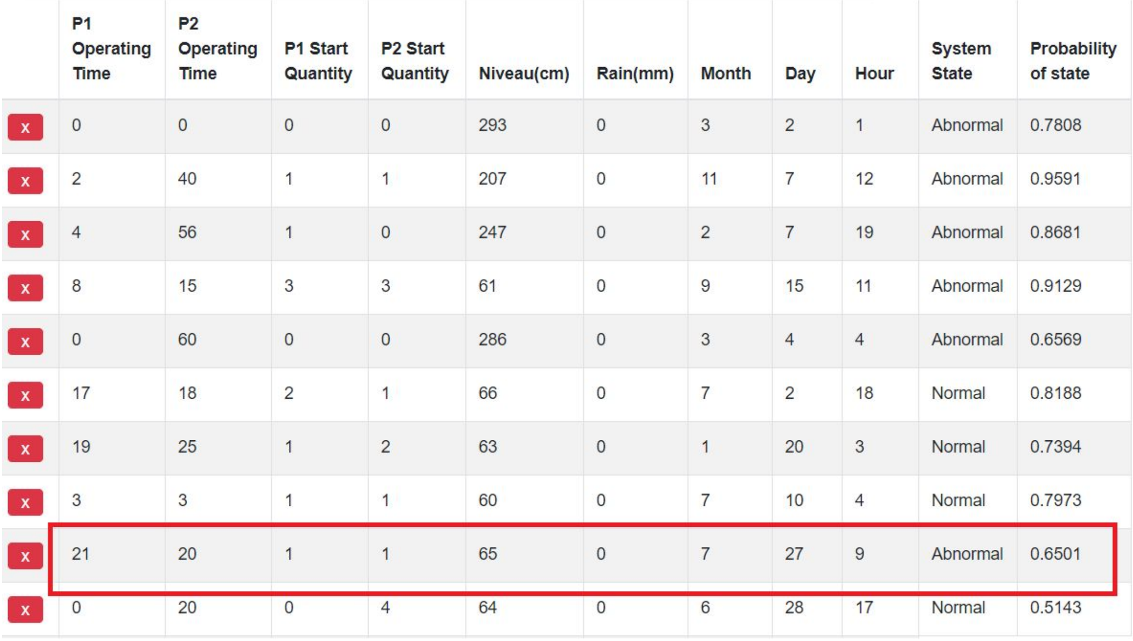
To test the labeling functionality of the system, an unused dataset from pump (O01PS1) was prepared. Ten random rows from this dataset were selected, 5 for normal pump behaviour and 5 for abnormal pump behaviour. The attributes of these rows were used as an input in the web application and the results of the test can be seen on the figure below. Out of the ten cases, 9 got labeled correctly and 1 got mislabeled by a 15% margin. The results were successfully stored in the database by the system on receiving the response.



O01PS1 2017, 2018, 2020

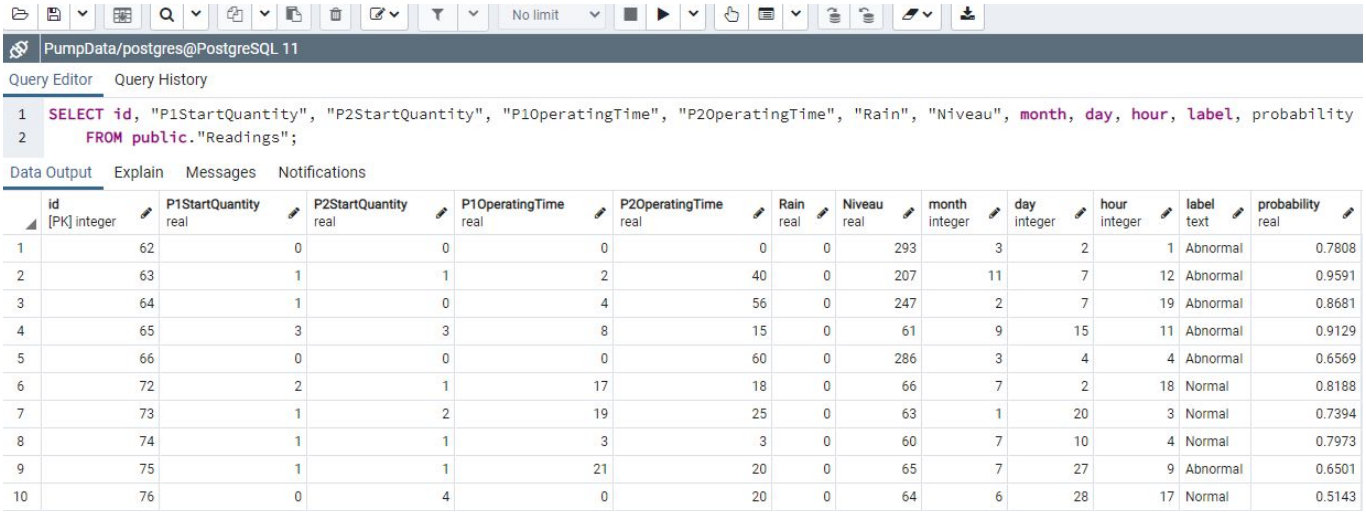
date_time	P1OperatingTime(m)	P2OperatingTime(m)	P1StartQuantity	P2StartQuantity	Niveau(cm)	Rain(mm)	EventText
2020-03-02 01:00:00	0.0	0.0	0.0	0.0	293.0	0,0	Abnormal
2017-11-07 12:00:00	2.0	40.0	1.0	1.0	207.0	0,0	Abnormal
2018-02-07 19:00:00	4.0	56.0	1.0	0.0	247.0	0,0	Abnormal
2017-09-15 11:00:00	8.0	15.0	3.0	3.0	61.0	0,0	Abnormal
2020-03-04 04:00:00	0.0	60.0	0.0	0.0	286.0	0,0	Abnormal

date_time	P1OperatingTime(m)	P2OperatingTime(m)	P1StartQuantity	P2StartQuantity	Niveau(cm)	Rain(mm)	EventText
2020-07-02 18:00:00	17.0	18.0	2.0	1.0	66.0	0,0	Normal
2018-01-20 03:00:00	19.0	25.0	1.0	2.0	63.0	0,0	Normal
2017-07-10 04:00:00	3.0	3.0	1.0	1.0	60.0	0,0	Normal
2020-07-27 09:00:00	21.0	20.0	1.0	1.0	65.0	0,0	Normal
2020-06-28 17:00:00	0.0	20.0	0.0	4.0	64.0	0,0	Normal



	P1 Operating Time	P2 Operating Time	P1 Start Quantity	P2 Start Quantity	Niveau(cm)	Rain(mm)	Month	Day	Hour	System State	Probability of state
x	0	0	0	0	293	0	3	2	1	Abnormal	0.7808
x	2	40	1	1	207	0	11	7	12	Abnormal	0.9591
x	4	56	1	0	247	0	2	7	19	Abnormal	0.8681
x	8	15	3	3	61	0	9	15	11	Abnormal	0.9129
x	0	60	0	0	286	0	3	4	4	Abnormal	0.6569
x	17	18	2	1	66	0	7	2	18	Normal	0.8188
x	19	25	1	2	63	0	1	20	3	Normal	0.7394
x	3	3	1	1	60	0	7	10	4	Normal	0.7973
x	21	20	1	1	65	0	7	27	9	Abnormal	0.6501
x	0	20	0	4	64	0	6	28	17	Normal	0.5143

(Ten random rows from an unused dataset, compared to system output)



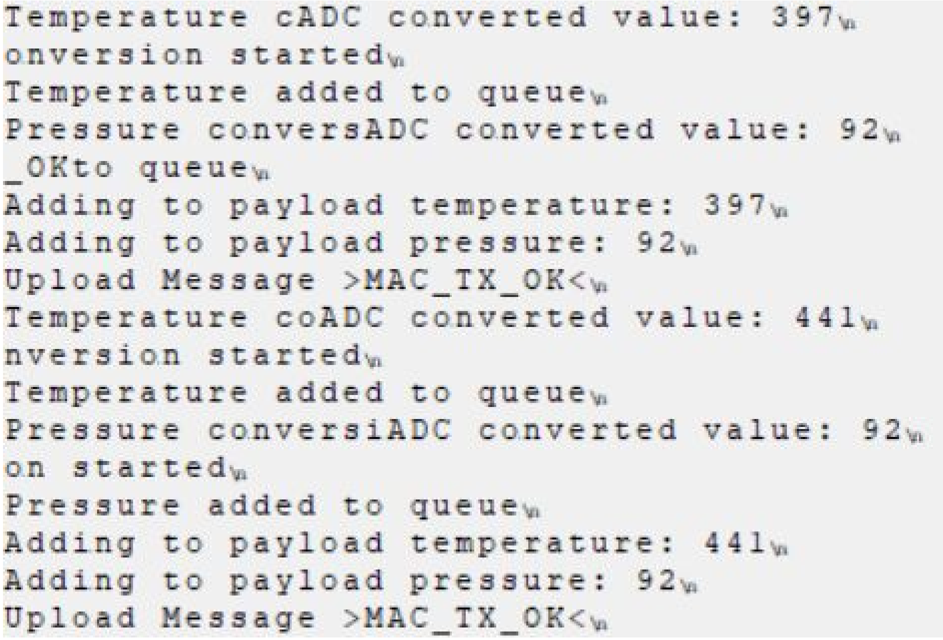
```sql
SELECT id, "P1StartQuantity", "P2StartQuantity", "P1OperatingTime", "P2OperatingTime", "Rain", "Niveau", month, day, hour, label, probability
    FROM public."Readings";
```

	id [PK] integer	P1StartQuantity real	P2StartQuantity real	P1OperatingTime real	P2OperatingTime real	Rain real	Niveau real	month integer	day integer	hour integer	label text	probability real
1	62	0	0	0	0	0	293	3	2	1	Abnormal	0.7808
2	63	1	1	2	40	0	207	11	7	12	Abnormal	0.9591
3	64	1	0	4	56	0	247	2	7	19	Abnormal	0.8681
4	65	3	3	8	15	0	61	9	15	11	Abnormal	0.9129
5	66	0	0	0	60	0	286	3	4	4	Abnormal	0.6569
6	72	2	1	17	18	0	66	7	2	18	Normal	0.8188
7	73	1	2	19	25	0	63	1	20	3	Normal	0.7394
8	74	1	1	3	3	0	60	7	10	4	Normal	0.7973
9	75	1	1	21	20	0	65	7	27	9	Abnormal	0.6501
10	76	0	4	0	20	0	64	6	28	17	Normal	0.5143

(The same 10 results stored in the database readings table after being labeled)

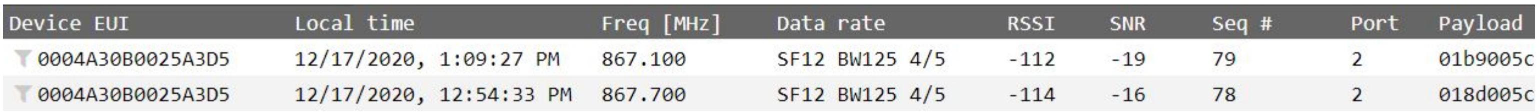
LoRa socket listener test
To test the receiving of sensor signals, the arduino application has to be running, actively sending signals that can be picked up by the listening socket. The arduino application was initiated and two signals were awaited to be forwarded to the LoRa server.

(Arduino app prints of the two sensor signals sent to the LoRa server)



```
Temperature cADC converted value: 397\n
onversion started\n
Temperature added to queue\n
Pressure conversADC converted value: 92\n
_OKto queue\n
Adding to payload temperature: 397\n
Adding to payload pressure: 92\n
Upload Message >MAC_TX_OK<\n
Temperature coADC converted value: 441\n
nversion started\n
Temperature added to queue\n
Pressure conversiADC converted value: 92\n
on started\n
Pressure added to queue\n
Adding to payload temperature: 441\n
Adding to payload pressure: 92\n
Upload Message >MAC_TX_OK<\n
```

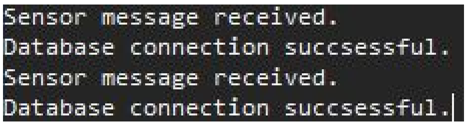
The signals were successfully picked up by the LoRa server.



Device EUI	Local time	Freq [MHz]	Data rate	RSSI	SNR	Seq #	Port	Payload
0004A30B0025A3D5	12/17/2020, 1:09:27 PM	867.100	SF12 BW125 4/5	-112	-19	79	2	01b9005c
0004A30B0025A3D5	12/17/2020, 12:54:33 PM	867.700	SF12 BW125 4/5	-114	-16	78	2	018d005c

(Figure of received messages by the LoRa server)

And respectively by the listening socket in the web application.

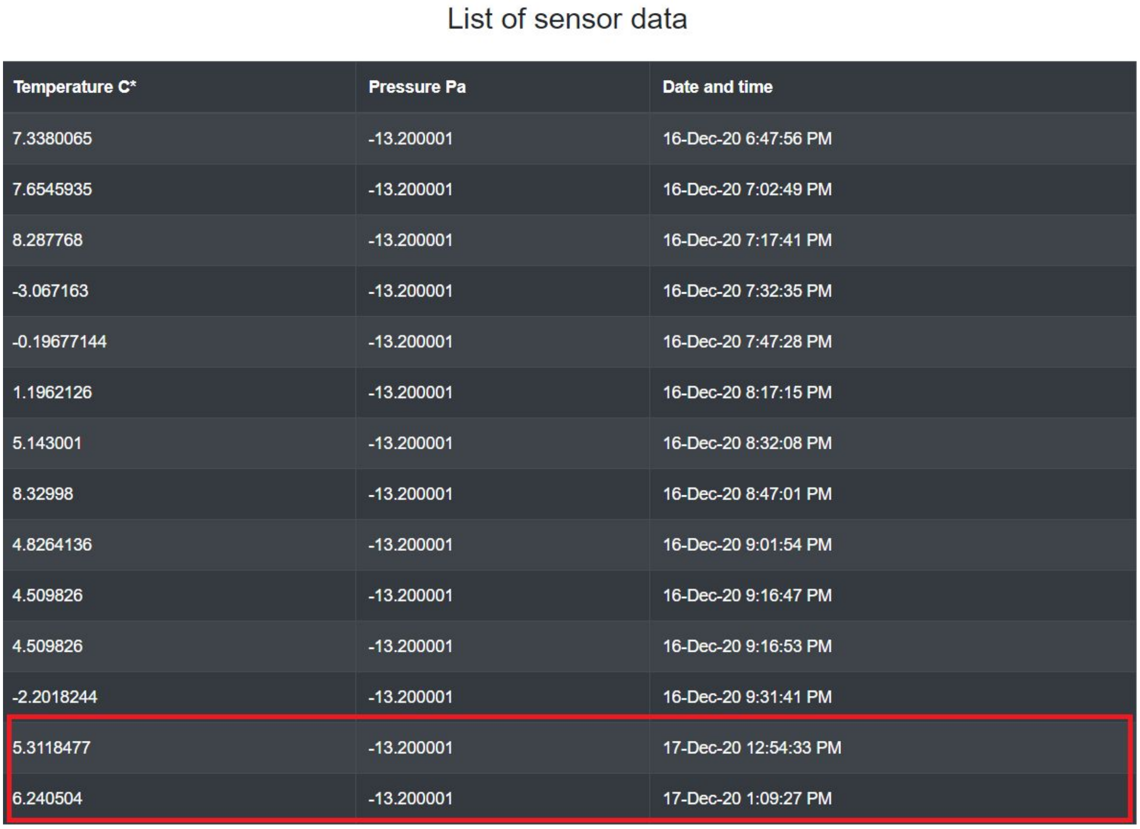


```
Sensor message received.
Database connection succsessful.
Sensor message received.
Database connection succsessful.
```

(Web application prints of received signals)

Both of the signals were successfully displayed to the web application’s view and stored in the database on retrieval.

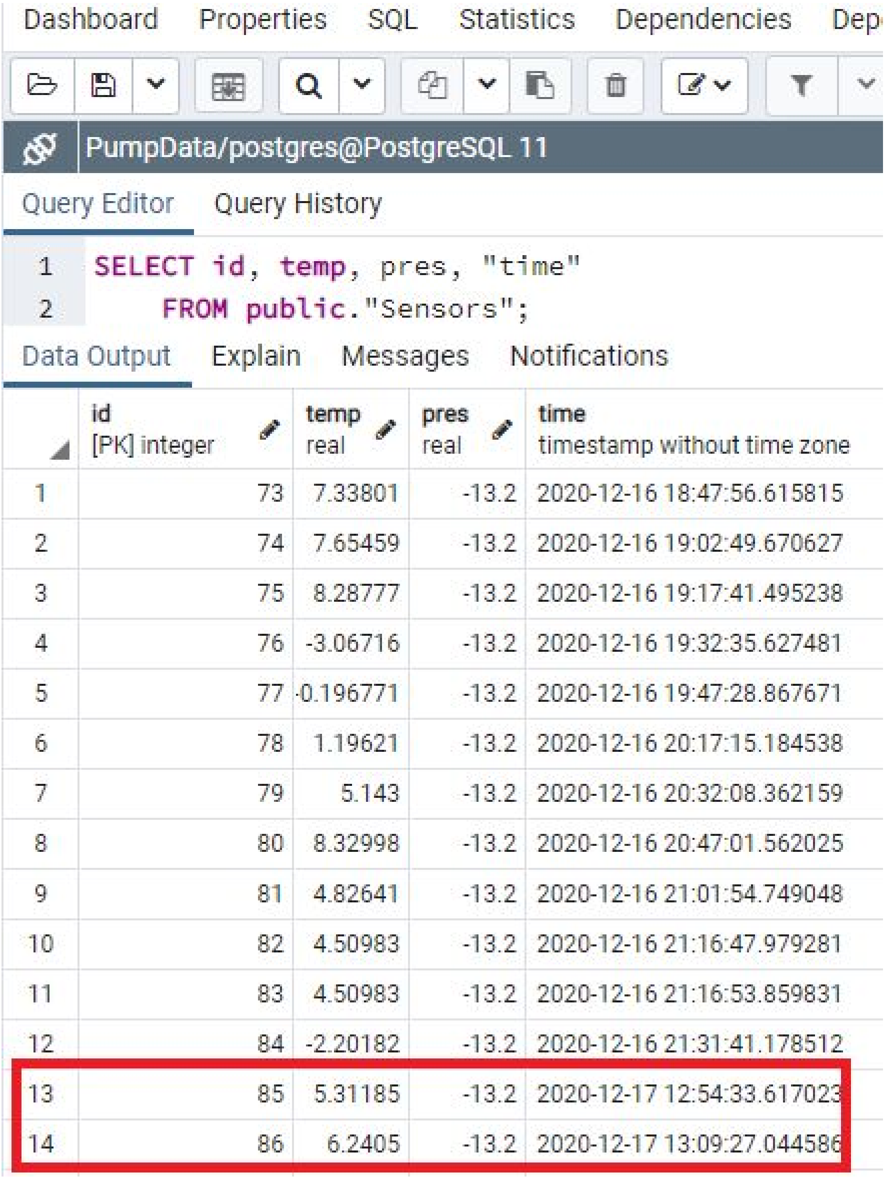
(Sensor view in the web application)



List of sensor data

Temperature C*	Pressure Pa	Date and time
7.3380065	-13.200001	16-Dec-20 6:47:56 PM
7.6545935	-13.200001	16-Dec-20 7:02:49 PM
8.287768	-13.200001	16-Dec-20 7:17:41 PM
-3.067163	-13.200001	16-Dec-20 7:32:35 PM
-0.19677144	-13.200001	16-Dec-20 7:47:28 PM
1.1962126	-13.200001	16-Dec-20 8:17:15 PM
5.143001	-13.200001	16-Dec-20 8:32:08 PM
8.32998	-13.200001	16-Dec-20 8:47:01 PM
4.8264136	-13.200001	16-Dec-20 9:01:54 PM
4.509826	-13.200001	16-Dec-20 9:16:47 PM
4.509826	-13.200001	16-Dec-20 9:16:53 PM
-2.2018244	-13.200001	16-Dec-20 9:31:41 PM
5.3118477	-13.200001	17-Dec-20 12:54:33 PM
6.240504	-13.200001	17-Dec-20 1:09:27 PM

(Sensor signal table from the database)



```
SELECT id, temp, pres, "time"
    FROM public."Sensors";
```

	id [PK] integer	temp real	pres real	time timestamp without time zone
1	73	7.33801	-13.2	2020-12-16 18:47:56.615815
2	74	7.65459	-13.2	2020-12-16 19:02:49.670627
3	75	8.28777	-13.2	2020-12-16 19:17:41.495238
4	76	-3.06716	-13.2	2020-12-16 19:32:35.627481
5	77	-0.196771	-13.2	2020-12-16 19:47:28.867671
6	78	1.19621	-13.2	2020-12-16 20:17:15.184538
7	79	5.143	-13.2	2020-12-16 20:32:08.362159
8	80	8.32998	-13.2	2020-12-16 20:47:01.562025
9	81	4.82641	-13.2	2020-12-16 21:01:54.749048
10	82	4.50983	-13.2	2020-12-16 21:16:47.979281
11	83	4.50983	-13.2	2020-12-16 21:16:53.859831
12	84	-2.20182	-13.2	2020-12-16 21:31:41.178512
13	85	5.31185	-13.2	2020-12-17 12:54:33.617023
14	86	6.2405	-13.2	2020-12-17 13:09:27.044586

Results
According to the test specifications derived from requirements, the results of the black box testing are as follows.

Test Case	Test Requirement	Test Result
1.Send input towards the python module and compare results.	Accurately label pump state by given pump attributes	Manually entered data is returned with a label of “normal” or “abnormal”
2.Await sensor signals to be delivered to the web application.	Store and display continuously collected sensor signals	Signals are stored in the database and displayed in the view.
3.Make an API client request to the python server.	Expose an API in python.	Manually entered data receives the expected response from the python server
4.Store readings and sensor signals in the database.	Store information in a database	The sensor signals and pump readings are successfully stored in the database.
5.Manage user input and views through a web application.	Host a web application	The web application loads, takes input and displays expected results