a new wave of **digital transformation**

RFID Data Applied in AI Methods @ MSGI'2021

Vorakit, Suhanya, Zhe & Tianshu

Agenda

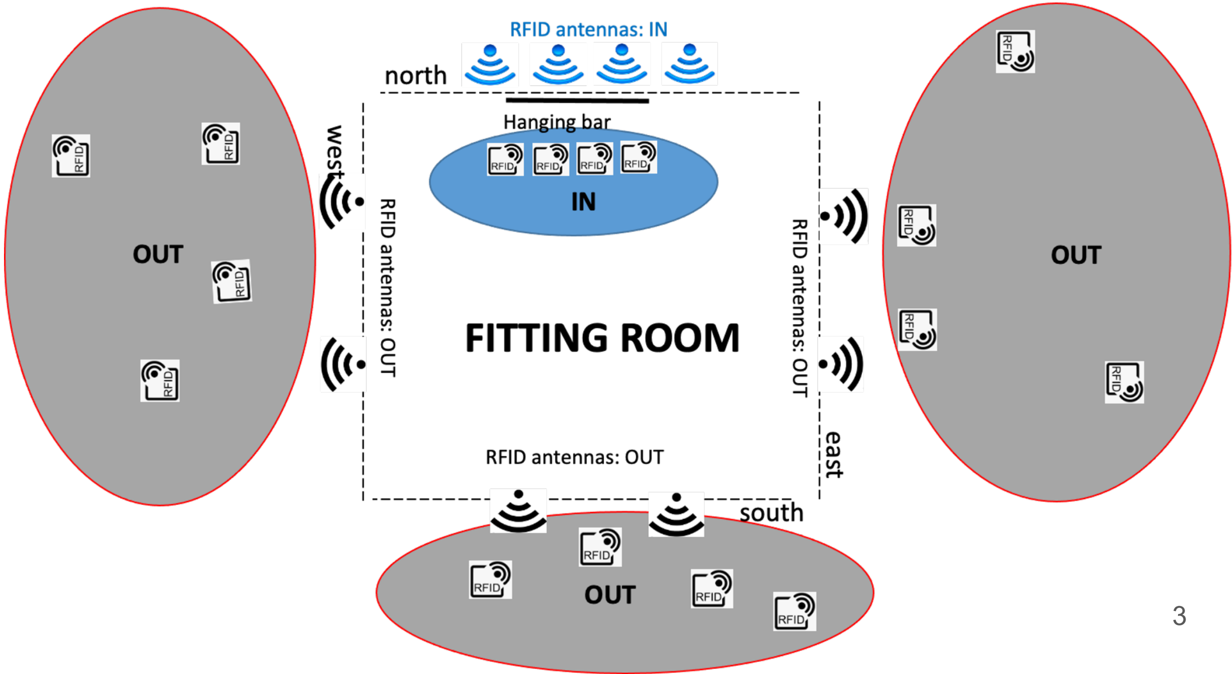
1. Context & Problem Statement
2. Input Source and Dataset
3. Feature Engineering
    a. Additional statistical features
    b. Scalarization of data
    c. Feature importances
    d. Resampling based on rrf
    e. Train test split based on motion
4. ML and DL (windows=1)
    a. Different ideas on treating the data
    b. Applied Machine Learning
    c. Applied Deep Learning Models
5. Conclusion & Future work
    a. Time-series
    b. CNN (windows>1)

Context & Problem Statement

Objective : Building a classification model which is able to predict the position (inside/outside) of RFID tags with extremely high accuracy (over 99%)

Challenges :

- very high accuracy model
- lost of signal during running
- multiple RFID moving at the same time
- Real time prediction



Input source and dataset - Source

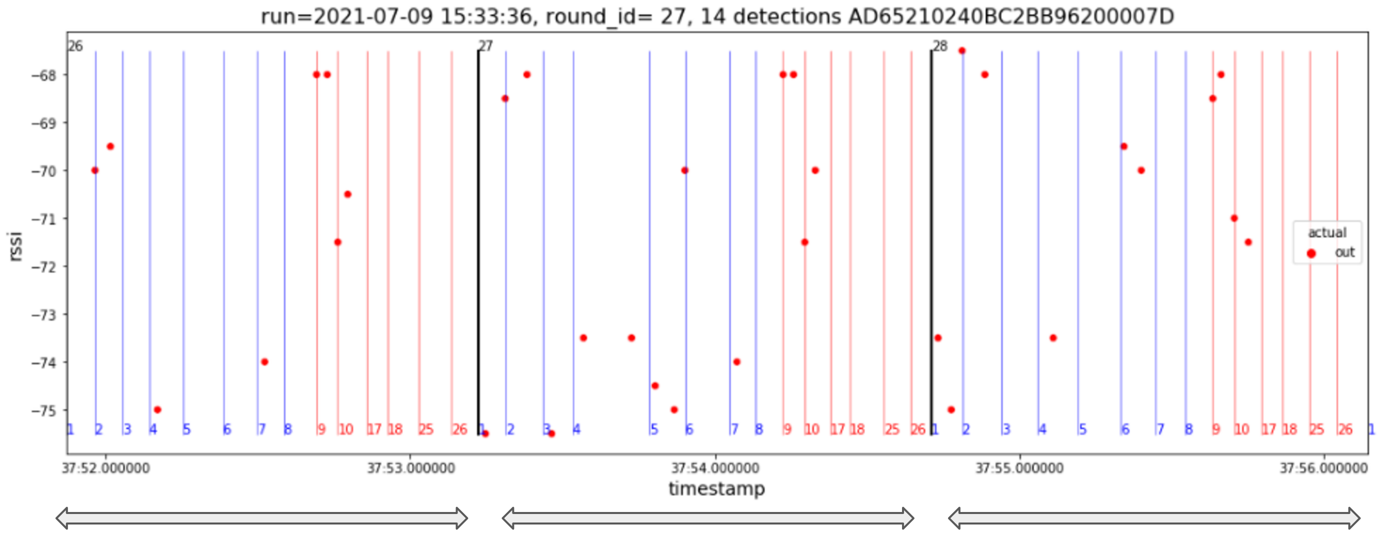
Supervised classification

Raw data are tagged with :

- “in” or ”out” position
- exact timestamp when receives signal
- the antenna which receives signal
- rssi
- motion scenario

Features are extracted and built from raw dataset and we will discuss later…

Input source and dataset - Example



40 tags exist and move at the same time during the experiment
However, if a tag is outside, it will never move inside.
14 antennas are activated one by one
One round ends when all antennas have been activated
An antenna may receive several rssi values during one round
A row in the dataset, represents the rssi values received by antennas, for one tag, during one round

Feature Engineering - Statistical features

The number of rssi values received by antennas is not fixed
To fix the number of columns, we take the statistical results of rssi values received in one round

- Per inside/outside antennas (inside fitting room or outside fitting room)
- **Per antenna position (north, south, west or east)**
- Per antenna

The following indicators are chosen:

- Max rssi values
- Min rssi values
- **Average rssi values**
- Number of antennas that has received rssi
- Number of rssi received

	epc	run	round_id	rssimax_ain	rssimax_aout	rssimin_ain	rssimin_aout	rssiavg_ain
0	AD65210240BAE1B45D00005B	2021-07-09 13:41:21	0	-80.0	-68.5	-80.0	-69.0	-80.0
2	AD65210240BAE1B45D00005B	2021-07-09 13:41:21	1	-80.0	-68.0	-80.0	-69.0	-80.0
4	AD65210240BAE1B45D00005B	2021-07-09 13:41:21	2	-80.0	-68.5	-80.0	-69.0	-80.0

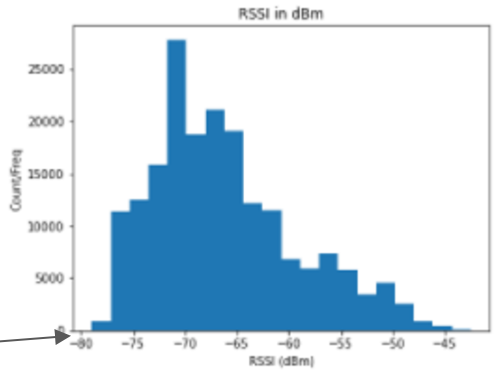
In addition, **difference** between indicators per inside and per outside is calculated

Feature Transformation and Scaling

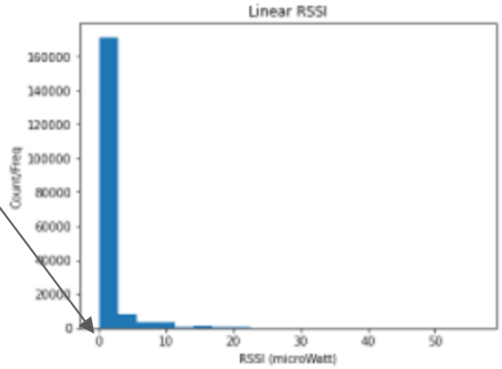
Findings: Results not affected much by scaling for all ML techniques

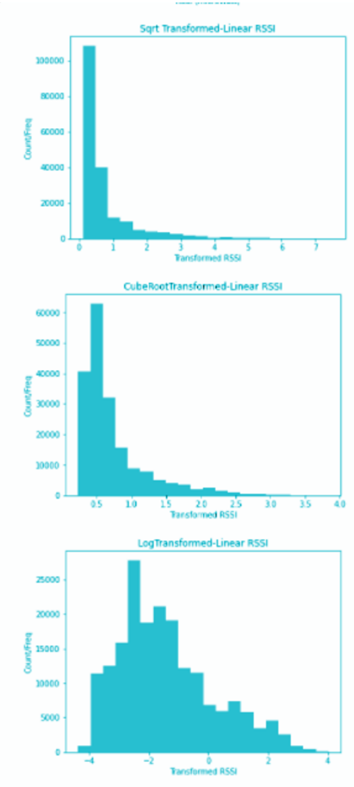
Feature Categories:

- **rssi values (max, min average) by antenna and zone - continuous**
- Number of antennas that has received rssi for the tag- ordinal: onehot encoding

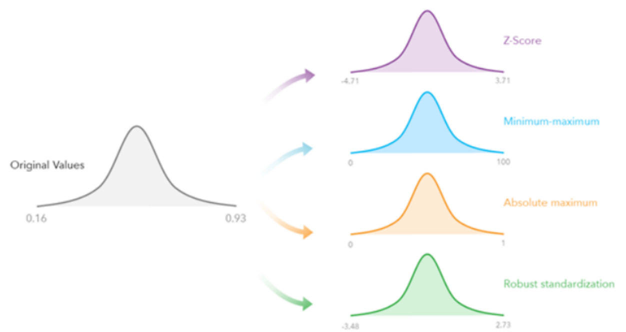


Impute NaN values here

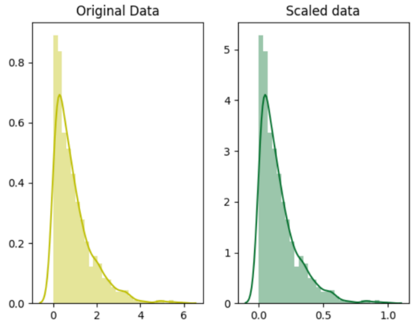




Gaussian distribution- normalize



other distribution- normalize



Feature Engineering - Feature importances

Feature importances with a forest of trees

Objective :

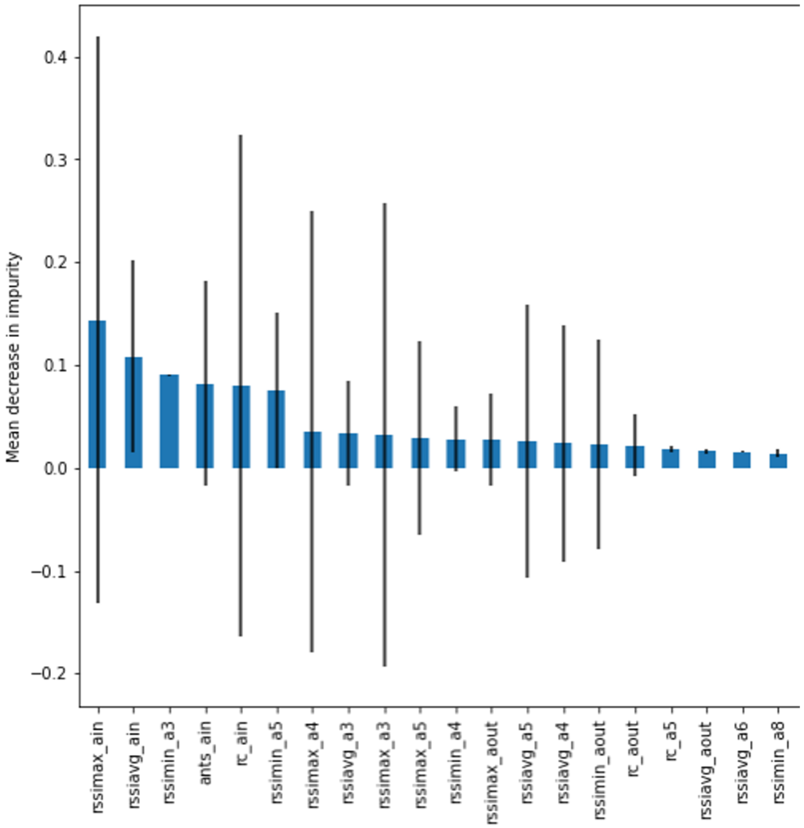
- To use a forest of trees to evaluate the importance of features.
- As expected, the outputted plot can suggest the informative features.

Steps :

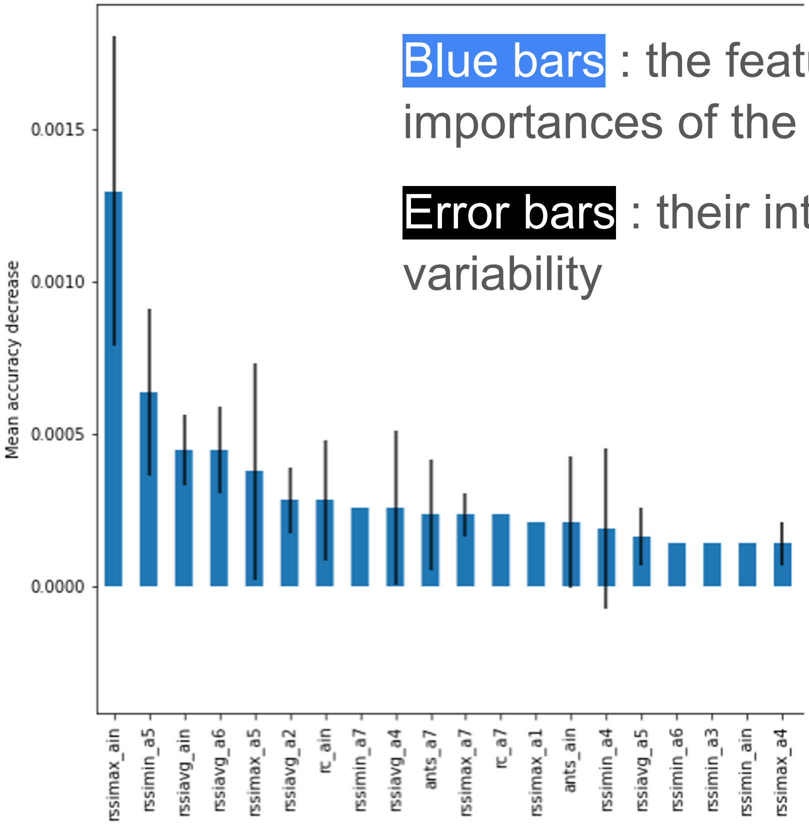
1. Calculate the importance based on mean decrease in impurity
2. Calculate the importance based on feature permutation
3. Analyses

Feature Engineering - Feature importances

mean decrease in impurity



feature permutation



Blue bars : the feature importances of the forest

Error bars : their inter-trees variability

Feature Engineering - Feature importances

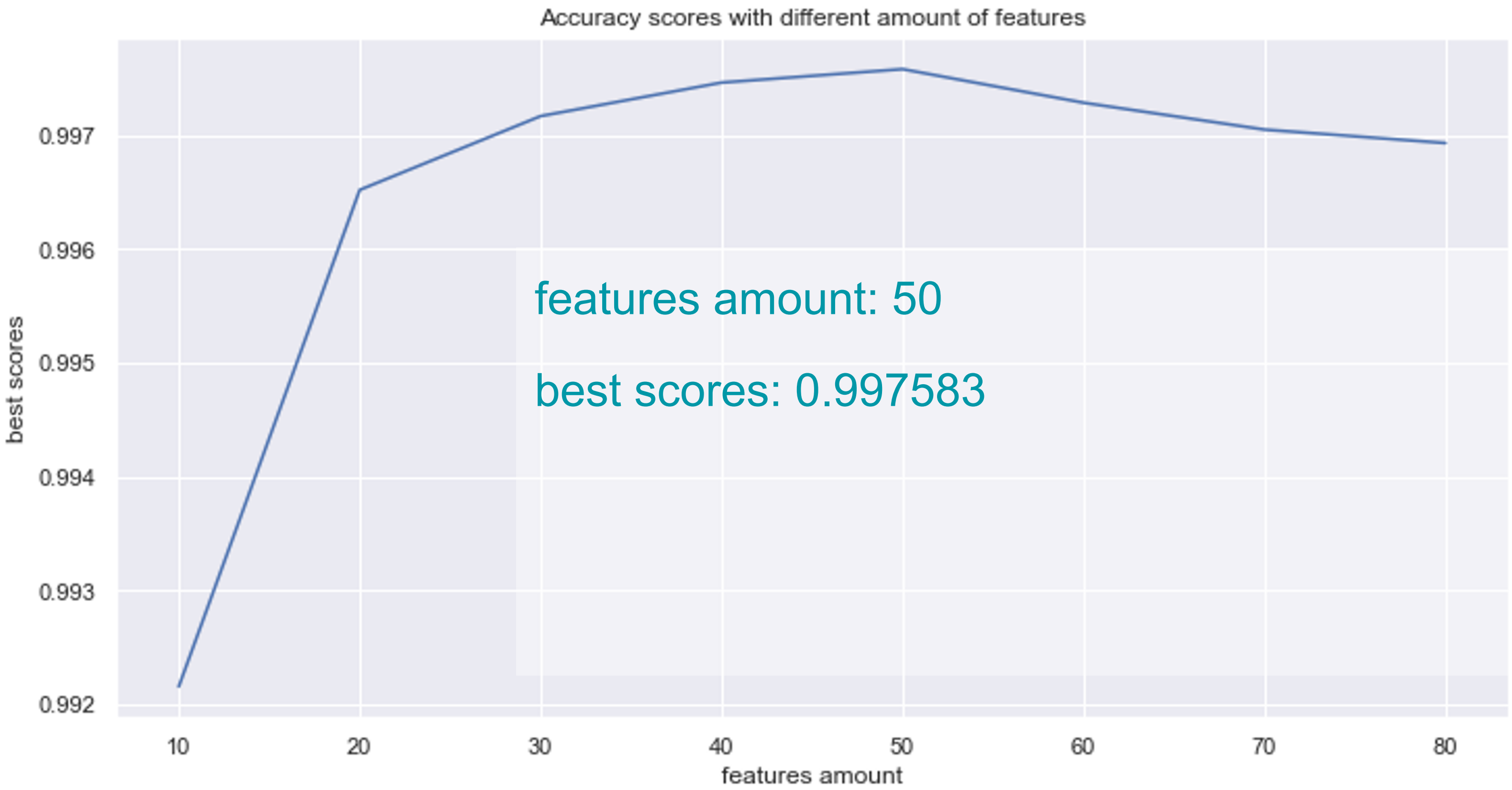
The most important features in common:

['ants_ain'
'rc_ain'
'rssiavg_a4'
'rssiavg_a5'
'rssiavg_a6'
'rssiavg_ain'
'rssimax_a4'
'rssimax_a5'
'rssimax_ain'
'rssimin_a3'
'rssimin_a4'
'rssimin_a5']

comments

- Number of antennas_in that receives rssi values
- No. of times that antenna_in receive signals
- the average of the rssi of the north side inside antenna

Feature Engineering - Feature importances

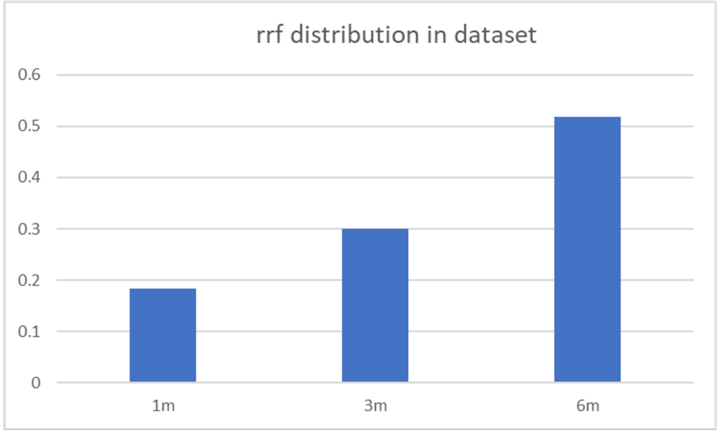


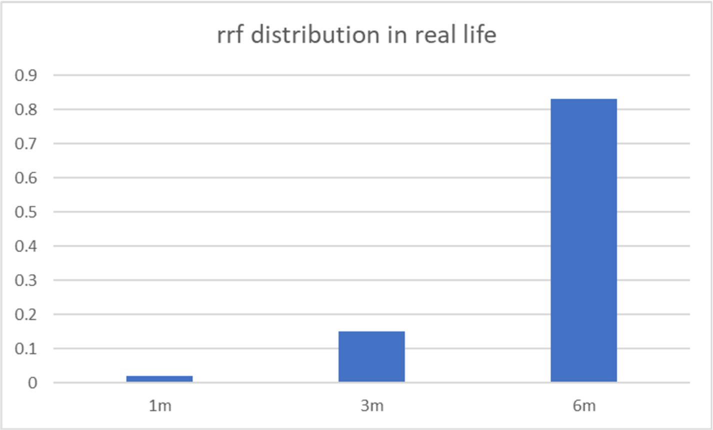
Resampling

Column rrf: indicator that represents tag performance.

E.g. from how far away a tag can be detected

Rows in dataset are subsampled to meet the real rrf distribution



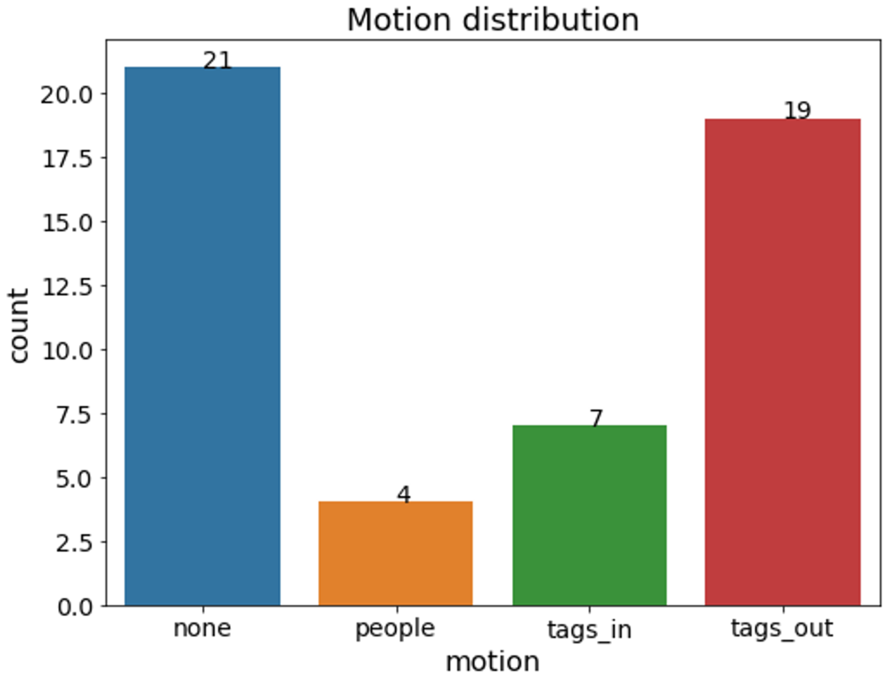


Train-Test Split

80% training data; 20% test data

Prediction on non motion is actually easier than on other motions

We split the train-test dataset while maintain the same motion distribution



Different ideas about treating data

- Framing the rounds
  - One round lasts 1.5 sec.
  - Proposed data arrangement is conducted at 4 rounds per an input.
- Non time-series (=Sequnencial data) vs. Time-series
  - if non time-series, sequence data as it is would be fine?
  - If time-series, we need to extend the multiple rounds as a single input. (Current setting)

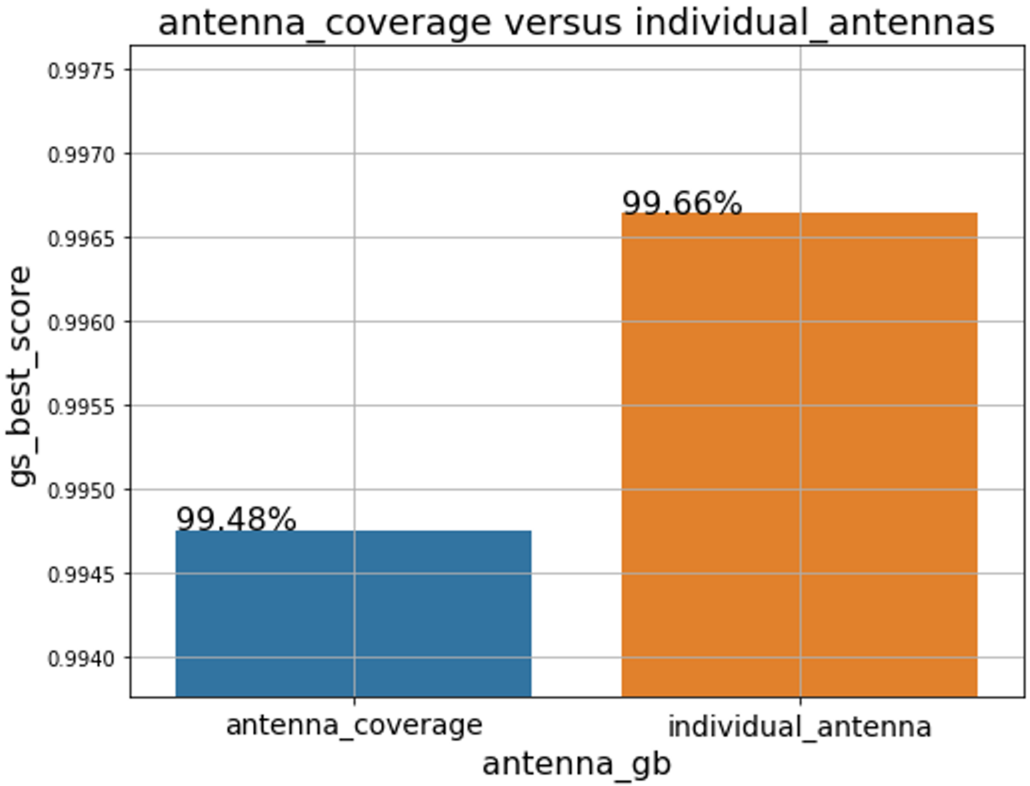
Classical ML models

Baseline result from MOJIX:

- antenna_converage (8 features): only uses statistical features per inside/outside
- individual_antenna (56 features): uses statistical features per each antenna

RandomForest classifier with entropy criterion

Best score for 30 times cross validations



Classical ML models

Our results with 13 statistical features per inside/outside antennas:

- max, min, **average** rssi values
- number of antennas that received rssi
- number of detections
- and their **differences**

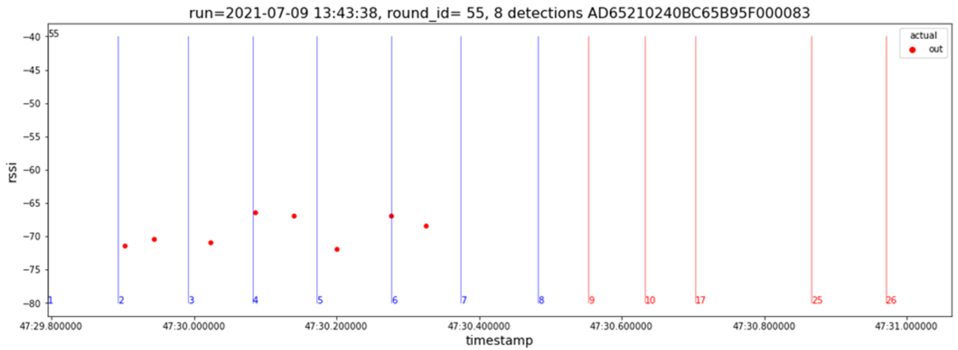
Classifier	CV best score	Accuracy (train)	Accuracy (test)	Confusion matrix (train)		Confusion matrix (test)	
Random Forest	99.506%	99.985%	99.619%	7080	**0**	1783	**13**
				2	6472	**0**	1613
KNN	99.565%	99.616%	99.765%	7062	**18**	1788	**8**
				34	6440	**0**	1613
Logistic Regression	99.565%	99.572%	99.648%	7049	**31**	1784	**12**
				27	6447	**0**	1613
GaussianNB	98.510%	98.517%	98.416%	6899	**181**	1742	**54**
				20	6454	**0**	1613
SVC	99.594%	99.594%	99.648%	7053	**27**	1784	**12**
				28	6446	**0**	1613

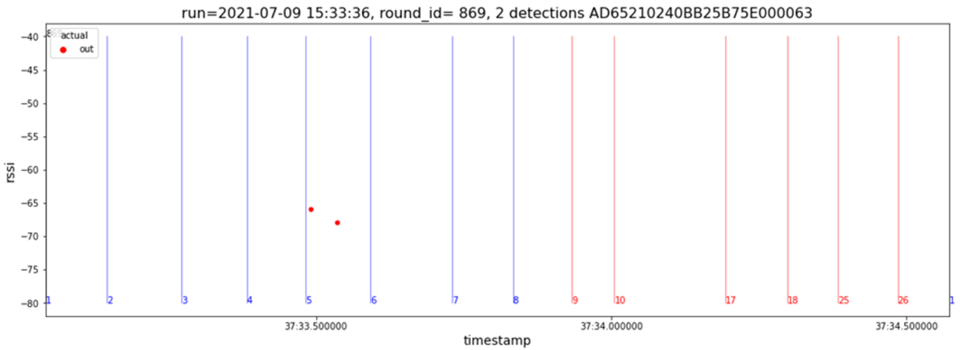
Classical ML models

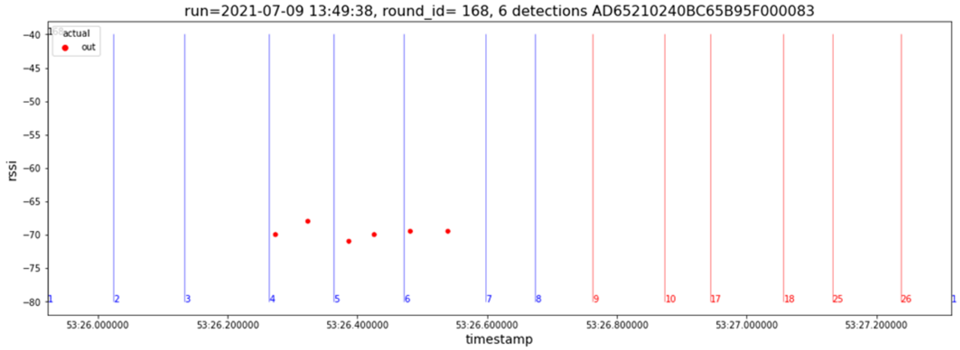
Our results with all 100 statistical features:

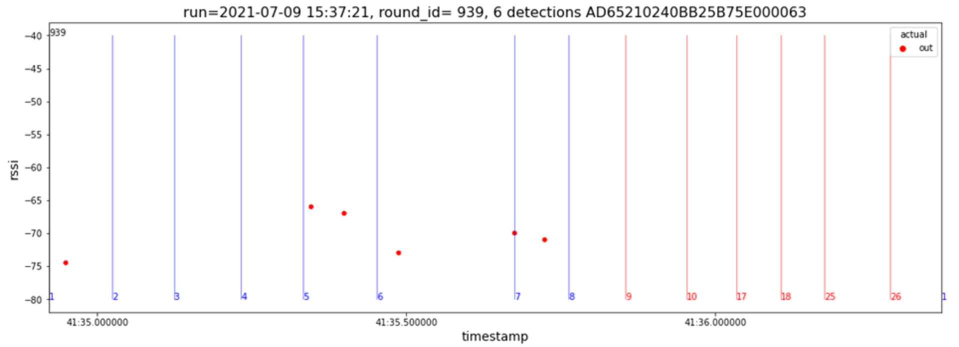
- per inside/outside
- per individual antenna
- per antenna position

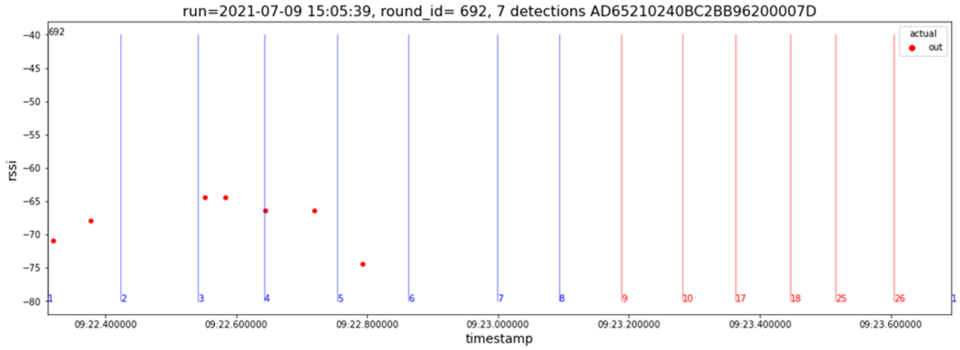
Classifier	CV best score	Accuracy (train)	Accuracy (test)	Confusion matrix (train)		Confusion matrix (test)	
Random Forest	99.727%	99.993%	99.736%	7080	0	1787	9
				1	6473	0	1613
KNN	99.683%	99.742%	99.795%	7065	15	1790	6
				20	6454	1	1612
Logistic Regression	99.668%	99.793%	99.736%	7066	14	1789	7
				14	6460	2	1611
GaussianNB	95.824%	95.846%	95.923%	6549	531	1662	134
				32	6442	5	1608
SVC	99.661%	99.764%	99.765%	7065	15	1789	7
				17	6457	1	1612

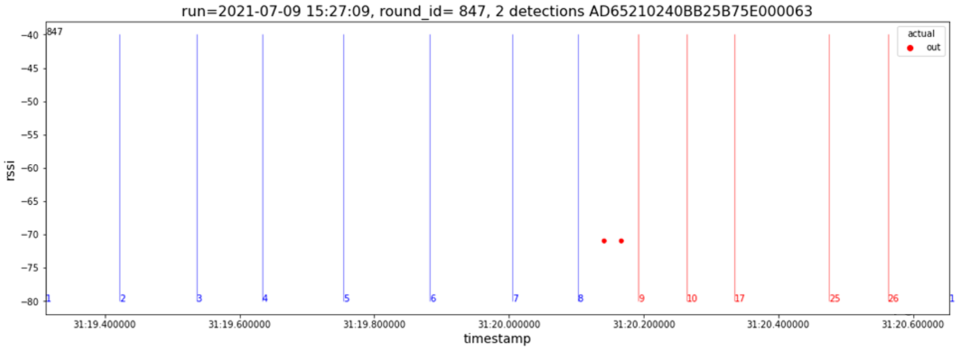

run=2021-07-09 13:43:38, round_id= 55, 8 detections AD65210240BC65B95F000083
55
actual
out
rssi
timestamp
47:29.800000
47:30.000000
47:30.200000
47:30.400000
47:30.600000
47:30.800000
47:31.000000


run=2021-07-09 15:33:36, round_id= 869, 2 detections AD65210240BB25B75E000063
actual
out
rssi
timestamp
37:33.500000
37:34.000000
37:34.500000


run=2021-07-09 13:49:38, round_id= 168, 6 detections AD65210240BC65B95F000083
actual
out
rssi
timestamp
53:26.000000
53:26.200000
53:26.400000
53:26.600000
53:26.800000
53:27.000000
53:27.200000


run=2021-07-09 15:37:21, round_id= 939, 6 detections AD65210240BB25B75E000063
939
actual
out
rssi
timestamp
41:35.000000
41:35.500000
41:36.000000


run=2021-07-09 15:05:39, round_id= 692, 7 detections AD65210240BC2BB96200007D
692
actual
out
rssi
timestamp
09:22.400000
09:22.600000
09:22.800000
09:23.000000
09:23.200000
09:23.400000
09:23.600000


run=2021-07-09 15:27:09, round_id= 847, 2 detections AD65210240BB25B75E000063
847
actual
out
rssi
timestamp
31:19.400000
31:19.600000
31:19.800000
31:20.000000
31:20.200000
31:20.400000
31:20.600000

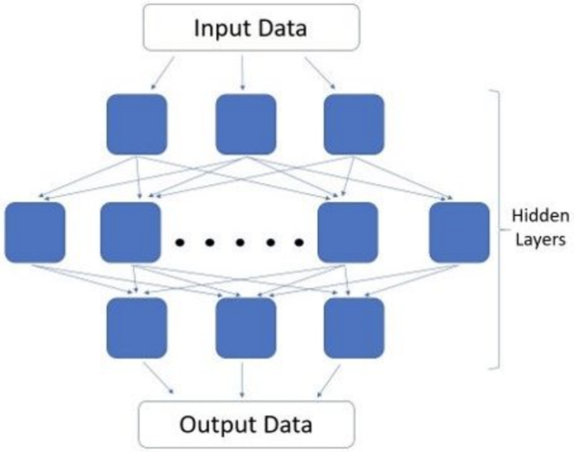
Classical ML models

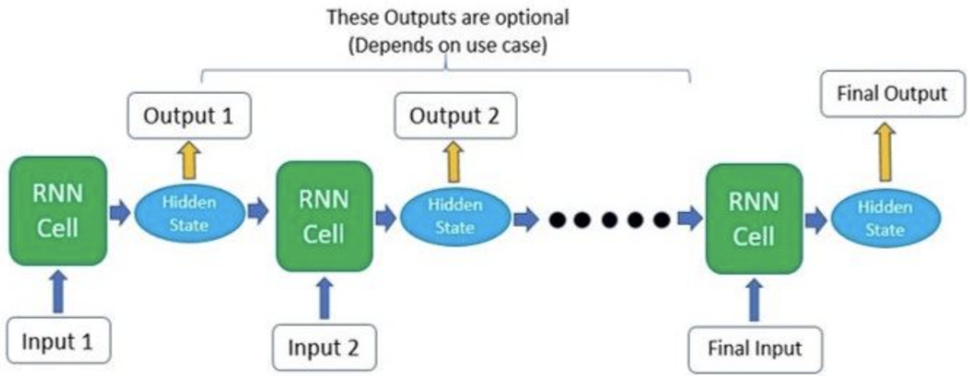
- With 13 most important features, a high accuracy can be reached
- Using all 100 features slightly improve the performance, however, it introduces some false negative (actually inside fitting room but predicted as outside)
- Wrong predictions always happen on 3 tags:
  - AD65210240BB25B75E000063
  - AD65210240BC2BB96200007D
  - AD65210240BC65B95F000083

Deep Learning Models

- Treating engineered data as sequence data.
- 14 Antennas run in order to read strength of signals from sensors.
- Possible DL methods to sequence data are:

Multi-Layer Perceptron (MLP) vs. Recurrent Neural Networks (RNN)





(params acc confusion matrix and error illustration)

Deep Learning Models (cont.)

RNN:

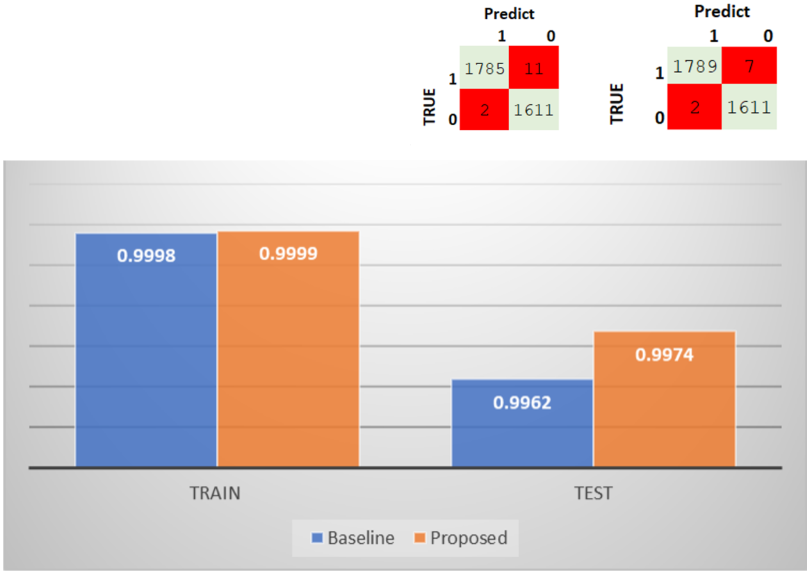
```
classification_report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1796
         1.0       1.00      1.00      1.00      1613

    accuracy                           1.00      3409
   macro avg       1.00      1.00      1.00      3409
weighted avg       1.00      1.00      1.00      3409

accuracy_score: 0.9964799061308302
confusion_matrix:
 [[1788    8]
 [   4 1609]]
```

Best performance at

`n_layer=3, epoch=100,`

`Accuracy=0.99736`

MLP:



Best performance at

`n_layer=2, epoch=100,`

`Accuracy=0.99736`

Conclusion & Hints

- **Results improved by adding additional statistical information**
- **Transforming and scaling data did not offer better results**
- **Further investigation on wrong predictions**
  - mislabeled RFID due to humain mistakes?
  - inconsistent rssi due to equipment?
- **Be careful for overfitting problem**
- **Complete the experiments with more complicated senarios and more tags**

Future Work

- RFID data can be considered also as a sequence of time-series (the order of read-signals from antennas are in the same order every round) could also help improving the results: **Long-Short-Term-Memory (LSTM), Bi-directional LSTM**

- CNN

CNN to capture Positional Information

Transform Input Data Representation (per EPC per run)

	RSSI min	RSSI max	RSSI ave	Count	TimeStamp
Antenna South 1					
Antenna East 1	x	x	x	x	x
Antenna East 2					
Antenna North1	x	x	x	x	x
Antenna North2	x	x	x	x	x
Antenna North3	x	x	x	x	x
Antenna North4					
Antenna North5	x	x	x	x	x
Antenna North6					
Antenna North7	x	x	x	x	x
Antenna North8					
Antenna West 1					
Antenna West 2					
Antenna South 2