Attack Detection in Water Distribution System part of SCADA

MU-EGCO623 Data Mining and
EGCO611 Programming Techniques for Advanced Application

10th December 2023

Sukkarin Ruensukont (G6538138)
Instructor: Asst. Prof. Tanasanee Phienthrakul, Ph.D.
Instructor: Asst. Prof. Vasin Suttichaya, Ph.D.

Outline

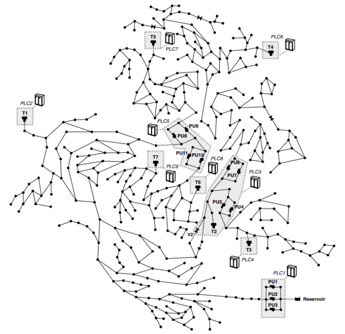


Fig. 1. C-Town water distribution system. (Adapted from Taormina et al. 2017.)

Problem and Objective

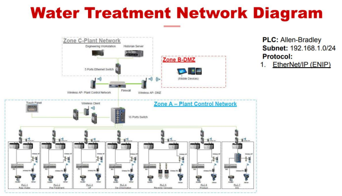
negative, missing, train_test_split, downsampling | Features | Select features

Methodologies: 3ML, 5 Ensembles, features selection

Bonus Track + Q & A

Mahidol University
Wisdom of the Land

SCADA



- Supervisory Control and Data Acquisition(SCADA)
- Stuxnet, Siemens PLC – Iran
- Water, Electricity, GAS
- Critical Infrastructure Security Showdown(CISS) 2016-2023

iTrust Centre for Research in Cyber Security

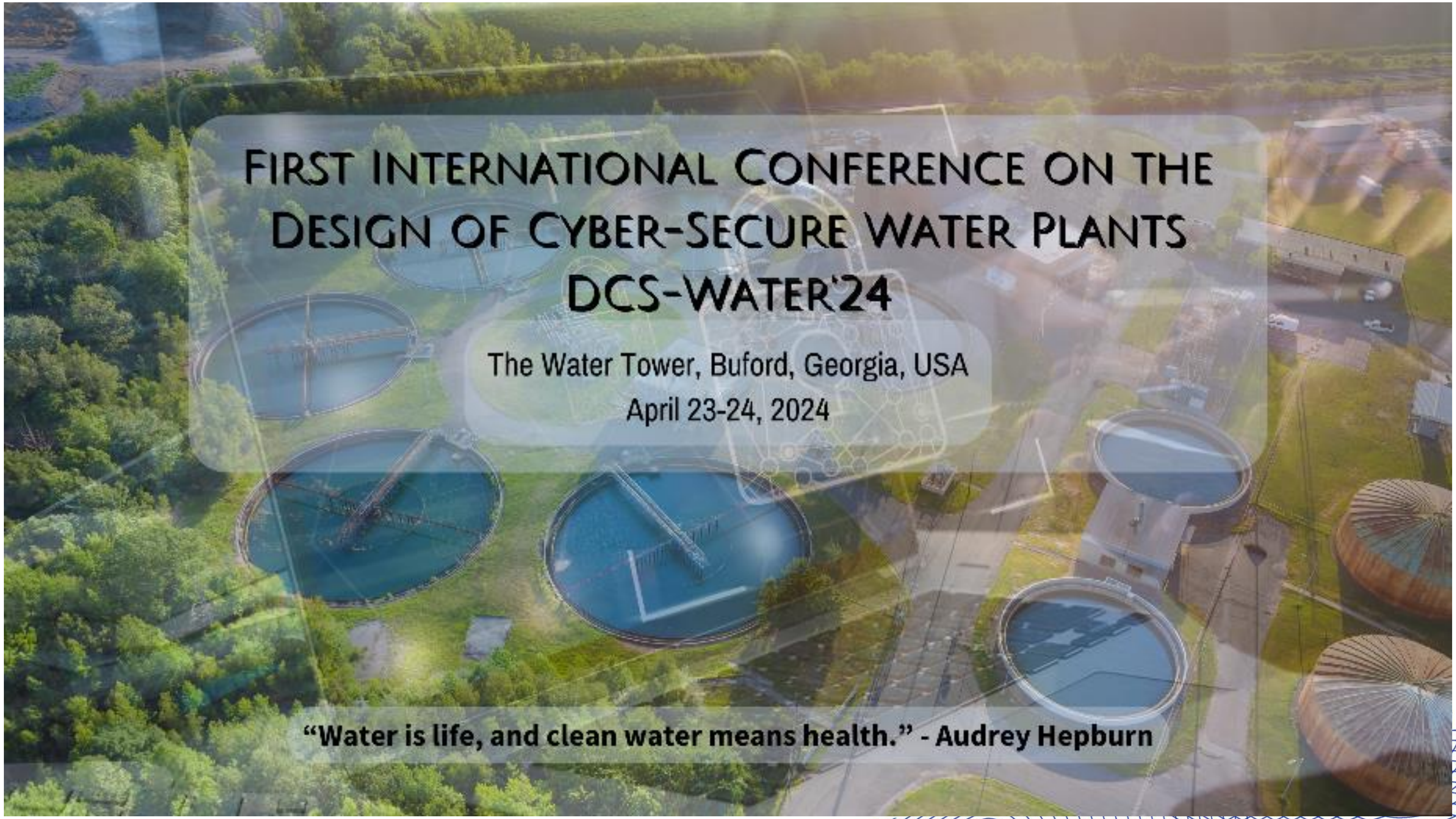
DIGITAL & INTELLIGENCE

CSA SINGAPORE

National Cybersecurity R&D Laboratory

https://itrust.sutd.edu.sg/ciss-2023/ submit request form to get labeled data

DCS – Water’24



https://itrust.sutd.edu.sg/first-international-conference-on-the-design-of-cyber-secure-water-plants-dcs-water24/programme-dcs-water24/

BATADAL2017

Features of BATADAL

- 7 **Tank water levels**, denoted L_<tank_id>
- 12 **Pressure** for actuated valve, denoted P_<junction id>
- 12 **flows** for actuated valve, denoted F_<actuator id>
- 12 **status** for actuated valve, denoted S_<actuator id>
- Total 43 features

- But we mainly **use WADI2019** which more complex

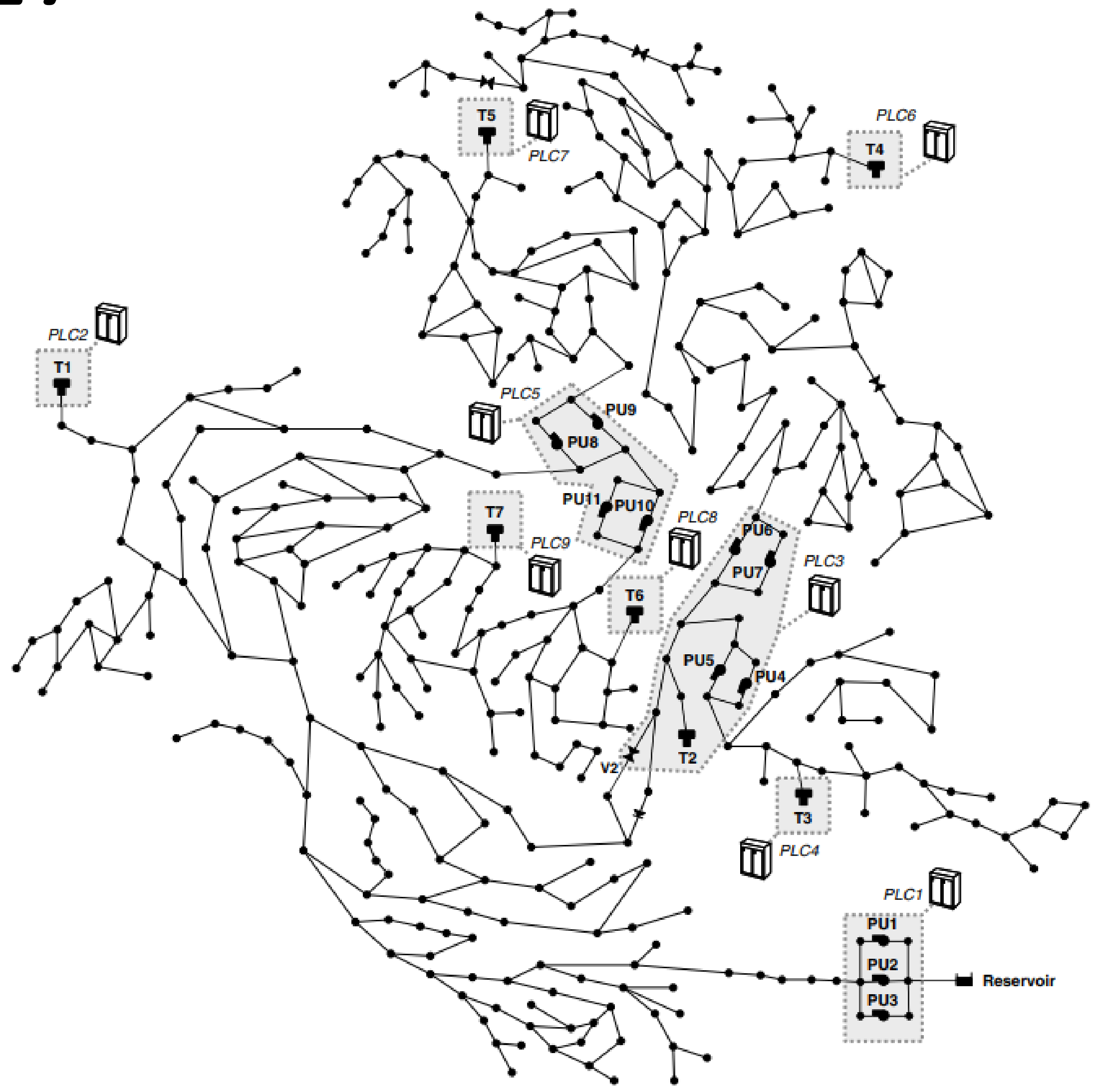
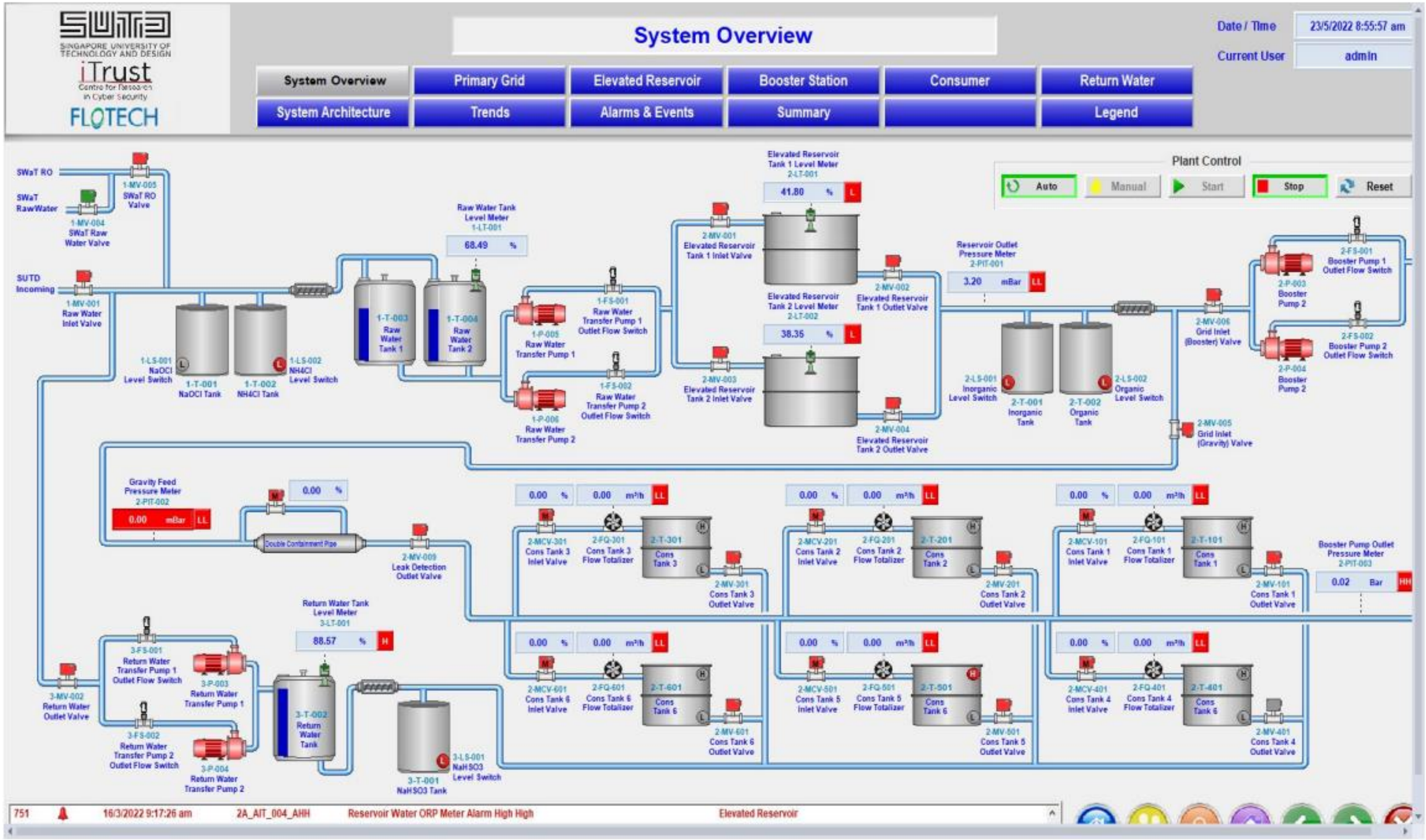


Fig. 1. C-Town water distribution system. (Adapted from Taormina et al. 2017.)

Mahidol University
Wisdom of the Land

WADI2019



Mahidol University
Wisdom of the Land

WADI2019

	0	1	2	3	4	5	6	7	8	9	129	130
0	Row	Date	Time	1_AIT_001_PV	1_AIT_002_PV	1_AIT_003_PV	1_AIT_004_PV	1_AIT_005_PV	1_FIT_001_PV	1_LS_001_AL	_FLOW	Attack LABLE (1:No Attack, -1:Attack)
1	1	10/9/17	00:00.0	164.21	0.529486	11.9972	482.48	0.331167	0.00127323	0	0.39	1
2	2	10/9/17	00:01.0	164.21	0.529486	11.9972	482.48	0.331167	0.00127323	0	0.39	1
3	3	10/9/17	00:02.0	164.21	0.529486	11.9972	482.48	0.331167	0.00127323	0	0.39	1
4	4	10/9/17	00:03.0	164.21	0.529486	11.9972	482.48	0.331167	0.00127323	0	0.39	1
...	...	...	...	...	...	...	...	...	...	...	...	...
172799	172799.0	10/11/17	59:58.0	172.915	0.583479	11.9211	466.051	0.318317	0.00126	0.0	0.0	1
172800	172800.0	10/11/17	59:59.0	172.915	0.583479	11.9211	466.051	0.318317	0.00126	0.0	0.0	1
172801	172801.0	10/11/17	00:00.0	172.915	0.583479	11.9211	466.051	0.318317	0.00126	0.0	0.0	1
172802	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1
172803	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1

172804 rows × 131 columns

Attack on WADI2019

- **Overflow** the tank
- Change **Chemical level**
- **Overflow** the tank
- **Pump malfunction** on/off repeatedly

Attack Identifier	Starting Time	Ending Time	Duration (minutes)	Attack description
1	9/10/17 19:25:00	9/10/17 19:50:16	25.16	Motorized valve 1_MV_001 is maliciously turned on, this causes an overflow on primary tank **should reflect on 1LT001 and 1FIT001**
2	10/10/17 10:24:10	10/10/17 10:34:00	9.50	Flow Indication Transmitter 1_FIT_001 is tuned off, a false reading is seen by PLC for 1_FIT_001. This will turn chemical dosing pump on while leaving the water level in primary tank constant. Consequently the attacker is increasing the level of chemicals inside water.
3-4	10/10/17 10:55:00	10/10/17 11:24:00	29.0	Stealthy attack. Attacker aims to drain elevated reservoir 2_LT_002. This is done controlling manipulating tank level draining and filling speed. 1_AIT_001 Moreover the attacker changes the reading seen by water quality sensor, this causes the raw water tank drain.

Method

- Shape = 172801 row, 130 columns
- Column contain **both positive and negative value**

	85	86	87	88	89	90
47125	1.0	NaN	NaN	23.5320	2.0	0.001922
47126	1.0	NaN	NaN	23.5320	2.0	0.001922
47127	1.0	NaN	NaN	23.5091	2.0	-0.002741
47128	1.0	NaN	NaN	23.5091	2.0	-0.002741
47129	1.0	NaN	NaN	23.5091	2.0	-0.002741
47130	1.0	NaN	NaN	23.5091	2.0	-0.002741
47131	1.0	NaN	NaN	23.5091	2.0	-0.002741
47132	1.0	NaN	NaN	23.5622	2.0	0.001947
47133	1.0	NaN	NaN	23.5622	2.0	0.001947

	89	90	90-sign
47125	2.0	0.001922	1
47126	2.0	0.001922	1
47127	2.0	0.002741	2
47128	2.0	0.002741	2
47129	2.0	0.002741	2
47130	2.0	0.002741	2
47131	2.0	0.002741	2
47132	2.0	0.001947	1
47133	2.0	0.001947	1

Method

- Find and **remove empty column**

```
The following columns have all values empty (except the header): ['50', '51', '86', '87']
         49  50  51   52
0  0.051166 NaN NaN  0.0
1  0.051166 NaN NaN  0.0
2  0.051166 NaN NaN  0.0
3  0.051166 NaN NaN  0.0
4  0.051166 NaN NaN  0.0

    85  86  87        88
0  2.0 NaN NaN  0.016157
1  2.0 NaN NaN  0.016157
2  2.0 NaN NaN  0.016157
3  2.0 NaN NaN  0.016157
4  2.0 NaN NaN  0.016157
```

```
         49   52   53   54
0  0.051166  0.0  0.0  0.0
1  0.051166  0.0  0.0  0.0
2  0.051166  0.0  0.0  0.0
3  0.051166  0.0  0.0  0.0
4  0.051166  0.0  0.0  0.0

    85        88  88-sign
0  2.0  0.016157      2.0
1  2.0  0.016157      2.0
2  2.0  0.016157      2.0
3  2.0  0.016157      2.0
4  2.0  0.016157      2.0
```

Method

- Find and drop 2 rows with missing value

```
Column '3' has missing values:
172801    NaN
172802    NaN
Name: 3, dtype: float64


Column '4' has missing values:
172801    NaN
172802    NaN
Name: 4, dtype: float64


Column '5' has missing values:
172801    NaN
172802    NaN
Name: 5, dtype: float64
```

```
Column '127' has missing values:
172801    NaN
172802    NaN
Name: 127, dtype: float64


Column '128' has missing values:
172801    NaN
172802    NaN
Name: 128, dtype: float64


Column '129' has missing values:
172801    NaN
172802    NaN
Name: 129, dtype: float64
```

```
No missing values found in the DataFrame.
```

Method

- Before Normalization

	3	4	5	6	7	8	9	10	11	12	...
count	172801.000000	172801.000000	172801.000000	172801.000000	172801.000000	172801.000000	172801.0	172801.0	172801.000000	172801.000000	...
mean	176.210422	0.648910	11.928407	453.784271	0.274574	0.542569	0.0	0.0	55.539636	1.274287	...
std	18.669165	0.351526	0.139214	18.862597	0.037848	0.862086	0.0	0.0	8.706924	0.452633	...
min	0.000000	0.000000	0.000000	0.000000	0.201966	0.000605	0.0	0.0	37.002300	0.000000	...
25%	170.866000	0.589479	11.911300	440.867000	0.241040	0.001102	0.0	0.0	47.829700	1.000000	...
50%	177.234000	0.631472	11.927600	454.977000	0.273966	0.001186	0.0	0.0	55.932900	1.000000	...
75%	179.533000	0.661469	11.952000	468.240000	0.305849	1.872090	0.0	0.0	62.489400	2.000000	...
max	634.492000	6.000000	12.109800	484.871000	0.351282	2.495160	0.0	0.0	75.216100	2.000000	...

8 rows × 127 columns

Method

- After Normalization, MinMaxScaler

	3	4	5	6	7	8	9	10	11	12	...
count	172801.000000	172801.000000	172801.000000	172801.000000	172801.000000	172801.000000	172801.0	172801.0	172801.000000	172801.000000	...
mean	0.277719	0.108152	0.985021	0.935887	0.486273	0.217259	0.0	0.0	0.485095	0.637143	...
std	0.029424	0.058588	0.011496	0.038902	0.253473	0.345587	0.0	0.0	0.227848	0.226316	...
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	...
25%	0.269296	0.098246	0.983608	0.909246	0.261687	0.000199	0.0	0.0	0.283337	0.500000	...
50%	0.279332	0.105245	0.984954	0.938346	0.482199	0.000233	0.0	0.0	0.495386	0.500000	...
75%	0.282955	0.110245	0.986969	0.965700	0.695726	0.750228	0.0	0.0	0.666961	1.000000	...
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.0	0.0	1.000000	1.000000	...

8 rows × 127 columns

Method

- **Train:test = 70:30**
- **stratify based on class(1 = No Attack, -1 = Attack)**

```
[ ]
from sklearn.model_selection import train_test_split

df_pre6 = pd.read_csv(path+'WADI_6_undersampling.csv')

feature_cols = df_pre6.columns.tolist()[:-1] # select all features(-1 = exclude class at the last column)
# feature_cols = ['109','66'] #can perform feature selection here

X_to_trainTestSplit = df_pre6[feature_cols]
Y_to_trainTestSplit = df_pre6['130']

X_train, X_test, y_train, y_test = train_test_split(X_to_trainTestSplit, Y_to_trainTestSplit, test_size=0.3, stratify=Y_to_trainTestSplit)
```

Mahidol University
Wisdom of the Land

Process

- **Top best** feature scores
- Top worst feature scores
- Select **only feature score > 100**
  count = **49** from original 130 features

```
-----------------------------------------------------
Top 10 best features selected by this method are :
66 : 12054.955263772104
16 : 4538.045647704014
18 : 4487.315854347481
8 : 4350.468063608258
37 : 3387.726424933425
89 : 2191.9522378488746
90-sign : 2179.735695068156
69 : 1703.569038191479
34 : 1546.9967666436596
129 : 1538.0722013467896

-----------------------------------------------------
Top 10 worst features selected by this method are :
94 : 0.006322622591439416
63 : 0.028701978538930605
5 : 0.03238054284976486
55 : 0.16195704792978696
116 : 0.35304909001758367
35 : 0.39265259199205804
85 : 0.4255473206577616
39 : 0.6160590566327536
49 : 0.6424503252307319
6 : 0.895688901082101
```

```
Feature score more than 100, count = 49
['66', '16', '18', '8', '37', '89', '90-sign', '69', '34', '129', '20', '64', '88', '72', '41', '43', '21', '70', '11', '12', '93', '23', '88-si
```

Method

- **Downsampling**
- Before Downsampling: No Attack = 162824, Attack = 9977
- After Downsampling   : No Attack = 9977, Attack = 9977

```
No Attack class, count =  162824
Attack class,    count =  9977
New total row          =  9977 * 2 = 19954
```

	66	16	18	8	37	89	90-sign	69	34	129	...	78	31	71	32	119	
count	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	...	19954.000000	19954.000000	19954.000000	19954.000000	19954.000000	19954.00
mean	0.054285	0.471785	0.470482	0.383091	0.497435	0.248071	0.159467	0.250370	0.452602	0.341339	...	0.625739	0.352342	0.184584	0.542774	0.212132	0.15
std	0.206885	0.499216	0.499140	0.384585	0.364434	0.431904	0.366120	0.301613	0.343617	0.278215	...	0.226874	0.271429	0.240744	0.408076	0.230422	0.23
min	0.000000	0.000000	0.000000	0.000098	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.008524	0.003520	0.00
25%	0.000000	0.000000	0.000000	0.000219	0.071429	0.000000	0.000000	0.000000	0.097561	0.133047	...	0.500000	0.073171	0.000000	0.142649	0.070816	0.00
50%	0.000000	0.000000	0.000000	0.096444	0.666667	0.000000	0.000000	0.178334	0.390244	0.296137	...	0.500000	0.365854	0.156002	0.339241	0.088087	0.00
75%	0.000000	1.000000	1.000000	0.765028	0.809524	0.000000	0.000000	0.275087	0.731707	0.510730	...	1.000000	0.560976	0.271092	1.000000	0.274820	0.23
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	0.999942	1.00

8 rows × 50 columns

Method

- replacement = False
- class1 = No Attack

```python
from sklearn.utils import resample

df_pre5 = pd.read_csv(path+"WADI_5_featureScore100Up.csv")

# Separate the two classes based on values in column '130'
class_1 = df_pre5[df_pre5['130'] == 1]
class_minus_1 = df_pre5[df_pre5['130'] == -1]

# Determine the minimum number of samples in either class
min_samples = min(len(class_1), len(class_minus_1))
print("No Attack class, count = ", class_1.shape[0])
print("Attack class,    count = ", class_minus_1.shape[0])
print("New total row          = ",min_samples,'* 2 =',min_samples*2)
print()

# Undersample both classes to have an equal number of samples
undersampled_class_1 = resample(class_1, replace=False, n_samples=min_samples, random_state=42)
undersampled_class_minus_1 = resample(class_minus_1, replace=False, n_samples=min_samples, random_state=42)

# Concatenate the undersampled classes back together
undersampled_df = pd.concat([undersampled_class_1, undersampled_class_minus_1])

undersampled_df.to_csv(path+"WADI_6_undersampling.csv", index=False)
undersampled_df.describe()
```

Method and post process

- 49 features which feature score > 100
- Train model **DT, KNN, RF**
- Ensemble 5 models
  - **Ensemble1**: Majority Votes
  - **Ensemble2**: Voting with adjusted weight, DT=2, KNN=1, RF=1
  - **Ensemble3**: Voting with adjusted weight, DT=2, KNN=2, RF=1
  - **Ensemble4**: Voting with adjusted weight, DT=4, KNN=3, RF=2
  - **Ensemble5**: Voting with adjusted weight, DT=2, KNN=1, RF=3

Results and Evaluation

- DT vs KNN vs RF

Model	Accuracy	Precision	Recall	F1-Score
DT	**98.998**	**99.613**	**<u>98.594</u>**	**99.063**
KNN	98.680	99.077	<u>98.292</u>	98.679
RF	89.276	90.082	88.349	89.186

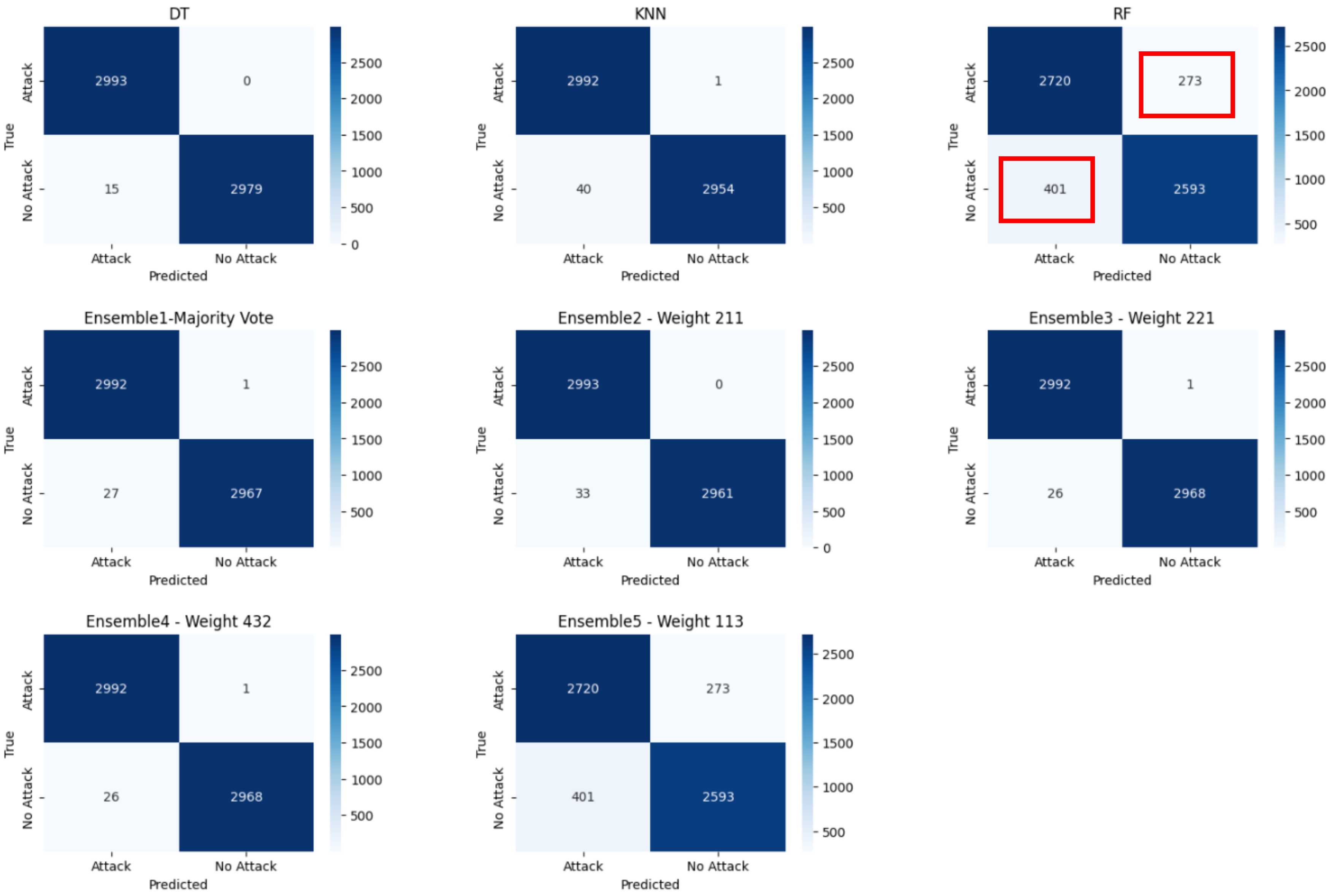
Results and Evaluation

- DT vs 5 Ensemble models

Model	Accuracy	Precision	Recall	F1-Score
DT	**98.998**	**99.613**	**98.594**	**99.063**
Ensemble1 – Maj	99.048	99.379	98.536	99.022
Ensemble2 – 211	98.931	99.757	98.205	98.973
Ensemble3 – 221	98.948	99.413	98.638	99.055
Ensemble4 – 432	98.964	99.379	96.601	99.007
Ensemble5 – 113	89.276	90.082	88.349	89.186

Results and Evaluation

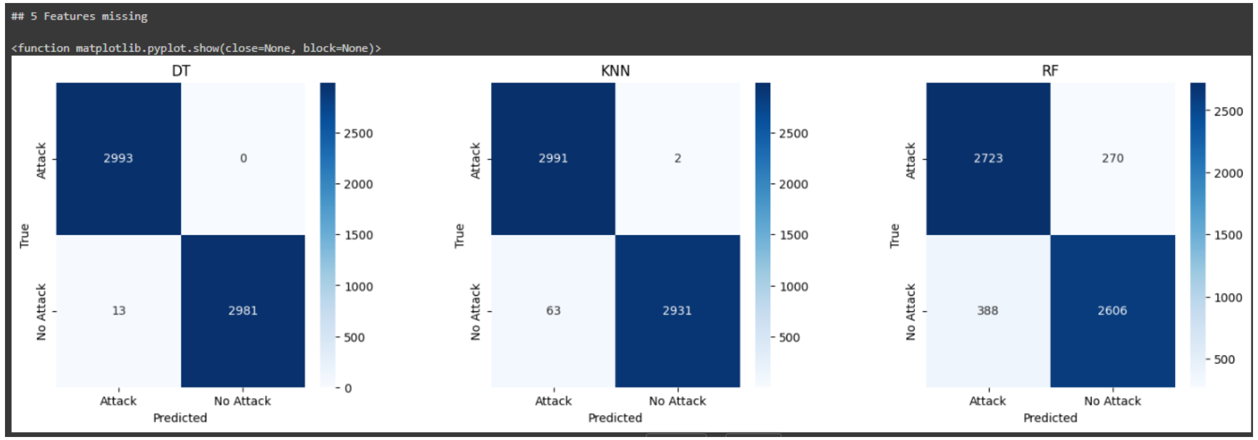
- Confusion Matrix
- False Positive > False Negative
- **RF give high FN and high FP**



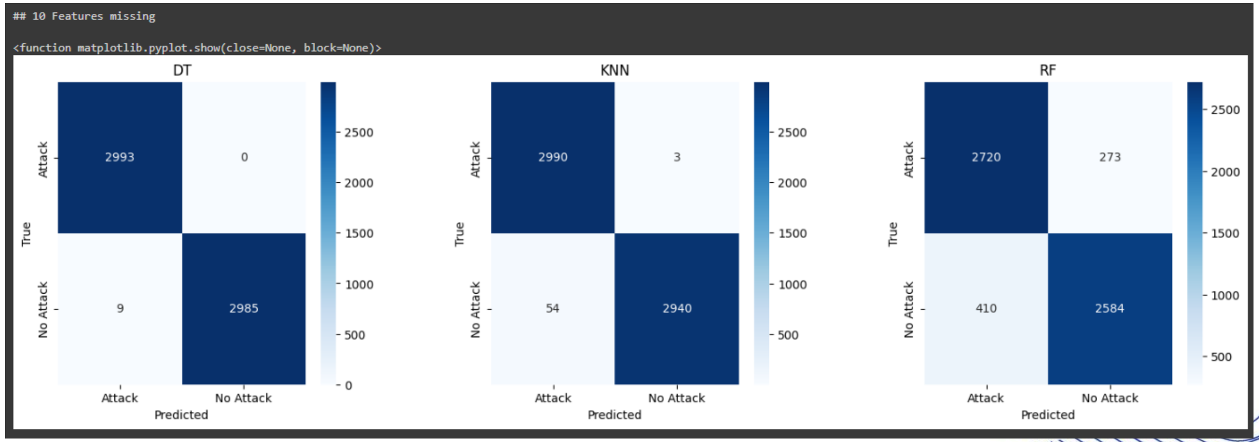
Mahidol University
Wisdom of the Land

Remove some features

- Remove 5 features
  44 features left
  DT Inspect 13 FP



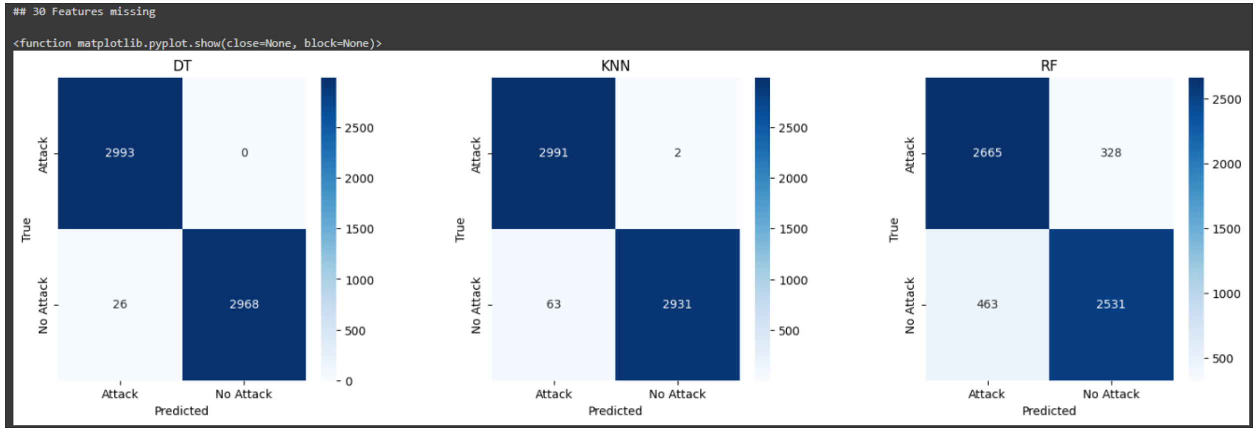
- Remove 10 features
  39 features left
  DT Inspect 9 FP



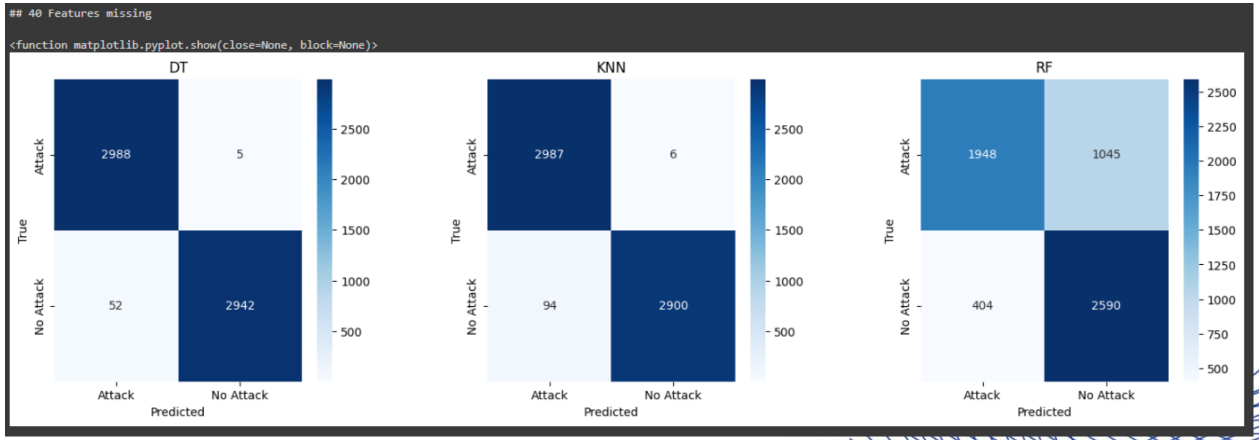
Mahidol University
Wisdom of the Land

Remove some features

- Remove 30 features
  19 features left
  DT Inspect 26 FP

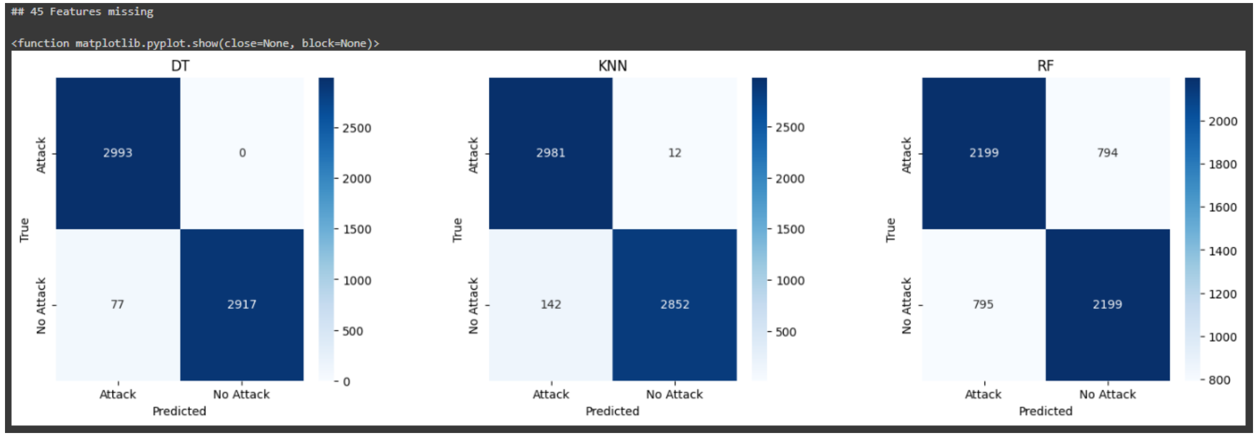


- Remove 40 features
  9 features left
  DT Inspect 52 FP, 5 FN



Remove some features

- Remove 45 features
  4 features left
- **DT Inspect 77 FP**



Analyze more

- **90-sign(6^{th}) get higher features score** than 90(23^{rd}).
- **88(12^{th}) get higher feature score** than 88-sign(22^{nd})
- This mean for feature 90, sign(positive/negative) is more significant than it's value. But for feature 88, sign(positive/negative) is less significant than it's value.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 172801 entries, 0 to 172800
Data columns (total 50 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   66        172801 non-null  float64
 1   16        172801 non-null  float64
 2   18        172801 non-null  float64
 3   8         172801 non-null  float64
 4   37        172801 non-null  float64
 5   89        172801 non-null  float64
 6   90-sign   172801 non-null  float64
 7   69        172801 non-null  float64
 8   34        172801 non-null  float64
 9   129       172801 non-null  float64
 10  20        172801 non-null  float64
 11  64        172801 non-null  float64
 12  88        172801 non-null  float64
 13  72        172801 non-null  float64
 14  41        172801 non-null  float64
 15  43        172801 non-null  float64
 16  21        172801 non-null  float64
 17  70        172801 non-null  float64
 18  11        172801 non-null  float64
 19  12        172801 non-null  float64
 20  93        172801 non-null  float64
 21  23        172801 non-null  float64
 22  88-sign   172801 non-null  float64
 23  90        172801 non-null  float64
 24  97        172801 non-null  float64
```

Bonus: BATADAL2017

- 4177 rows, 45 Features
- L_<Tank name>
  Tank level
- F_<Pump name>
  Pump flow
- S_<Pump name>
  Pump status
- P_<point>
  press at given point
- **‘ ATT_FLAG’** (1, -999)

```
1 df_BATADAL
```

	DATETIME	L_T1	L_T2	L_T3	L_T4	L_T5	L_T6	L_T7	F_PU1	S_PU1	...	P_J256	P_J289	P_J415	P_J302	P_J306	P_J307	P_J317	P_J14	P_J422	ATT_FLAG
0	04/07/16 00	2.44	5.24	3.19	4.10	2.86	5.50	4.39	93.63	1.0	...	70.00	28.22	85.87	21.69	82.72	21.58	71.99	39.33	29.64	-999
1	04/07/16 01	2.66	4.53	3.20	4.18	3.29	5.44	4.53	89.41	1.0	...	87.73	24.45	84.87	29.81	86.62	29.81	59.76	42.17	26.15	-999
2	04/07/16 02	3.11	3.66	3.66	4.21	3.87	5.15	3.22	89.88	1.0	...	89.29	23.90	87.11	29.85	87.64	29.85	58.50	42.00	25.56	-999
3	04/07/16 03	3.62	3.04	4.17	4.04	3.56	4.98	2.40	88.10	1.0	...	91.98	27.10	68.75	31.60	64.25	31.47	72.30	43.24	28.38	-999
4	04/07/16 04	4.08	2.68	4.73	3.20	3.11	5.39	3.46	87.01	1.0	...	92.11	26.76	68.74	32.30	64.23	32.17	72.53	44.00	28.04	-999
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4172	24/12/16 20	2.65	2.37	3.85	3.04	3.82	4.94	2.19	120.08	1.0	...	70.03	27.38	84.14	18.45	81.67	18.34	66.04	29.88	28.98	-999
4173	24/12/16 21	2.24	2.56	3.42	2.92	3.69	5.02	1.97	119.12	1.0	...	68.60	27.66	83.46	25.40	60.85	25.28	66.89	30.19	29.29	-999
4174	24/12/16 22	1.91	2.76	2.95	2.49	2.70	5.14	1.87	120.71	1.0	...	85.63	26.84	82.82	24.46	59.56	24.34	66.08	29.68	28.78	-999
4175	24/12/16 23	1.52	2.52	3.33	2.03	1.69	5.10	1.39	120.02	1.0	...	86.15	25.78	103.63	24.77	59.01	24.65	66.42	28.98	28.08	-999
4176	25/12/16 00	1.10	2.12	3.73	2.73	1.58	5.13	1.19	120.33	1.0	...	89.08	25.64	105.91	18.16	80.65	18.06	67.32	28.75	27.85	-999

4177 rows × 45 columns

Bonus: BATADAL

- 4177 rows, 45 Features
- L_<Tank name>
  Tank level
- **F_<Pump name>**
  Pump flow
- **S_<Pump name>**
  Pump status
- P_<point>
  press at given point
- ‘ ATT_FLAG’ (1, -999)

```
--------------------------------------------------------
Top 10 best features selected by this method are :
 F_PU6 : 695.6036391089294
 S_PU6 : 685.6517140148434
 S_PU11 : 313.42573044363075
 F_PU11 : 289.9837332248577
 F_PU7 : 17.728121058156763
 S_PU7 : 16.900190917996746
 L_T1 : 3.5553826455702273
 P_J14 : 2.0296320559887167
 S_PU2 : 1.1024679456696937
 P_J269 : 1.0682360601717962


--------------------------------------------------------
Top 10 worst features selected by this method are :
 S_PU8 : 5.033210882264993e-05
 L_T2 : 0.0014803037674457193
 L_T3 : 0.010753601483845862
 P_J289 : 0.011488111391178 55
 P_J300 : 0.011891050928399 34
 L_T4 : 0.012425482745645273
 P_J422 : 0.012583932773734138
 F_V2 : 0.014477739198065338
 P_J415 : 0.017118779622422 93
 L_T6 : 0.019690658313144066


--------------------------------------------------------
 F_PU6 : 695.6036391089294
 S_PU6 : 685.6517140148434
 S_PU11 : 313.42573044363075
 F_PU11 : 289.9837332248577
```

Mahidol University
Wisdom of the Land

Conclusion

Model	Accuracy	Precision	Recall	F1-Score
DT	**98.998**	**99.613**	**98.594**	**99.063**
KNN	98.680	99.077	98.292	98.679
RF	89.276	90.082	88.349	89.186

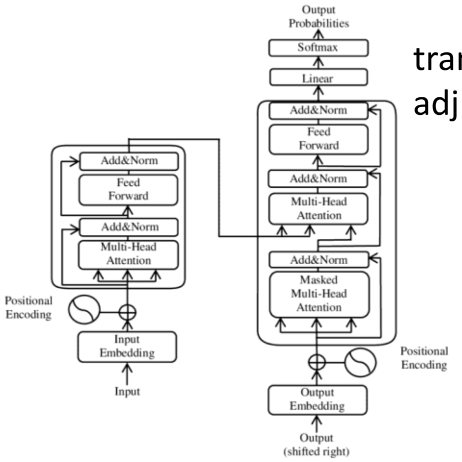
Model	Accuracy	Precision	Recall	F1-Score
DT	**98.998**	**99.613**	**98.594**	**99.063**
Ensemble1 – AVG	99.048	99.379	98.536	99.022
Ensemble2 – 211	98.931	99.757	98.205	98.973
Ensemble3 – 221	98.948	99.413	98.638	99.055
Ensemble4 – 432	98.964	99.379	96.601	99.007
Ensemble5 – 113	89.276	90.082	88.349	89.186

Conclusion

- **Decision tree** model give highest recall and F1-score
- If target to **reduce False Positive**, consider **Ensemble model** weight DT=2, KNN=1, RF=1
- Some features, such as **'90' sign is more significant** than it's numerical value
- Some features, such as **'88' it's numerical value is more significant** than it's sign
- Newer Water Distribution System(WADI2019) **is tolerant to removing top features** and still giving high recall and F1-score.
- BATADAL2017 show that features on **Pump flow and Pump status** are the most 2 important features to determines the attack.

Future Improvement

- Given **features with low score** to **Deep Learning** Model since we have 172,801 row. Already Prepare Deep Learning Model(Mine, Rock) from Aj.Wasin. Next step input dataset.
- Train model on **newer dataset** when labeled dataset available
- Train model on **Electricity system, Gas system**
- Use **transformer**(Improve from LSTM) to train model



transformer
adjust input's effect to output

linear? Tensorflow
use sigmoid
BCELoss()

```
def train_model(model, X_train, y_train, x_val, y_val, learning_rate = 1e-3, n_epochs = 250, batch_size = 10):
  # Define Loss Function
  loss_fn = torch.nn.BCELoss()
  # Define Optimizer
  optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)
  # Define batches
  batch_start = torch.arange(0, len(X_train), batch_size)
  # Define the best accuracy and best weight
  best_acc = -np.inf
  best_weights = None

  # Training loop
  for epoch in range(n_epochs):
    # Train model
    model.train()
    with tqdm(batch_start, unit = 'batch', mininterval = 0, disable = True) as bar:
      for start in bar:
        # Take a batch
        x_batch = X_train[start: start + batch_size]
        y_batch = y_train[start: start + batch_size]
        # Train model forward pass
        y_pred = model(x_batch)
        loss = loss_fn(y_pred, y_batch)
        # Train model backward pass
        optimizer.zero_grad()
        loss.backward()
        # Update Weights
        optimizer.step()
        # Progress
        acc = (y_pred.round() == y_batch).float().mean()
        bar.set_postfix(loss = float(loss), acc = float(acc))

    # Evaluate model
    model.eval()
    y_pred = model(x_val)
    acc = (y_pred.round() == y_val).float().mean()
    acc = float(acc)
    if best_acc < acc:
      best_acc = acc
      best_weights = copy.deepcopy(model.state_dict())

  # restore the best model
  model.load_state_dict(best_weights)
  return best_acc
```

```
model1 = MyModel()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle = True)
kfold = StratifiedKFold(n_splits=5,shuffle=True)
cv_scores = []

for train, val in kfold.split(X_train, y_train):
  acc = train_model(model1, X_train[train], y_train[train], X_train[val], y_train[val])
  cv_scores.append(acc)

cv_scores
```

```
[0.931034505367279,
 0.8620689511299133,
 0.6896551847457886,
 0.8620689511299133,
 0.7931034564971924]
```

Q&A

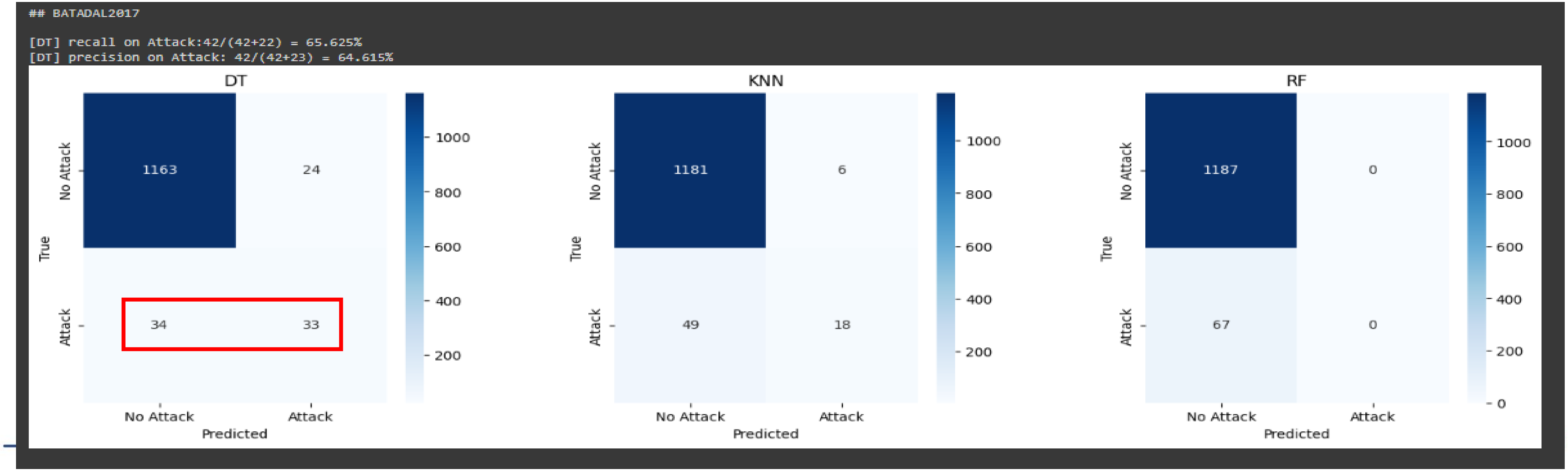
BATADAL2017

- [DT] cross validation
- Accuracy 94.898%
- Precision 61.179%
- Recall 54.460%
- F1-Score 54.632%

```
####### Features: count = 43 #######
Index([' L_T1', ' L_T2', ' L_T3', ' L_T4', ' L_T5', ' L_T6', ' L_T7', ' F_PU1',
       ' S_PU1', ' F_PU2', ' S_PU2', ' F_PU3', ' S_PU3', ' F_PU4', ' S_PU4',
       ' F_PU5', ' S_PU5', ' F_PU6', ' S_PU6', ' F_PU7', ' S_PU7', ' F_PU8',
       ' S_PU8', ' F_PU9', ' S_PU9', ' F_PU10', ' S_PU10', ' F_PU11',
       ' S_PU11', ' F_V2', ' S_V2', ' P_J280', ' P_J269', ' P_J300', ' P_J256',
       ' P_J289', ' P_J415', ' P_J302', ' P_J306', ' P_J307', ' P_J317',
       ' P_J14', ' P_J422'],
      dtype='object')
=== [ DT ] Cross validation performance ===
  Accuracy : 94.898 +-(2.596)
  Precision: 61.179 +-(20.998)
  Recall   : 54.460 +-(20.476)
      [0.85714286 0.4        0.55555556 0.5        0.83333333 0.66666667
 0.28571429 0.3        0.33333333 0.71428571]
  F1-Score : 54.632 +-(14.795)
```

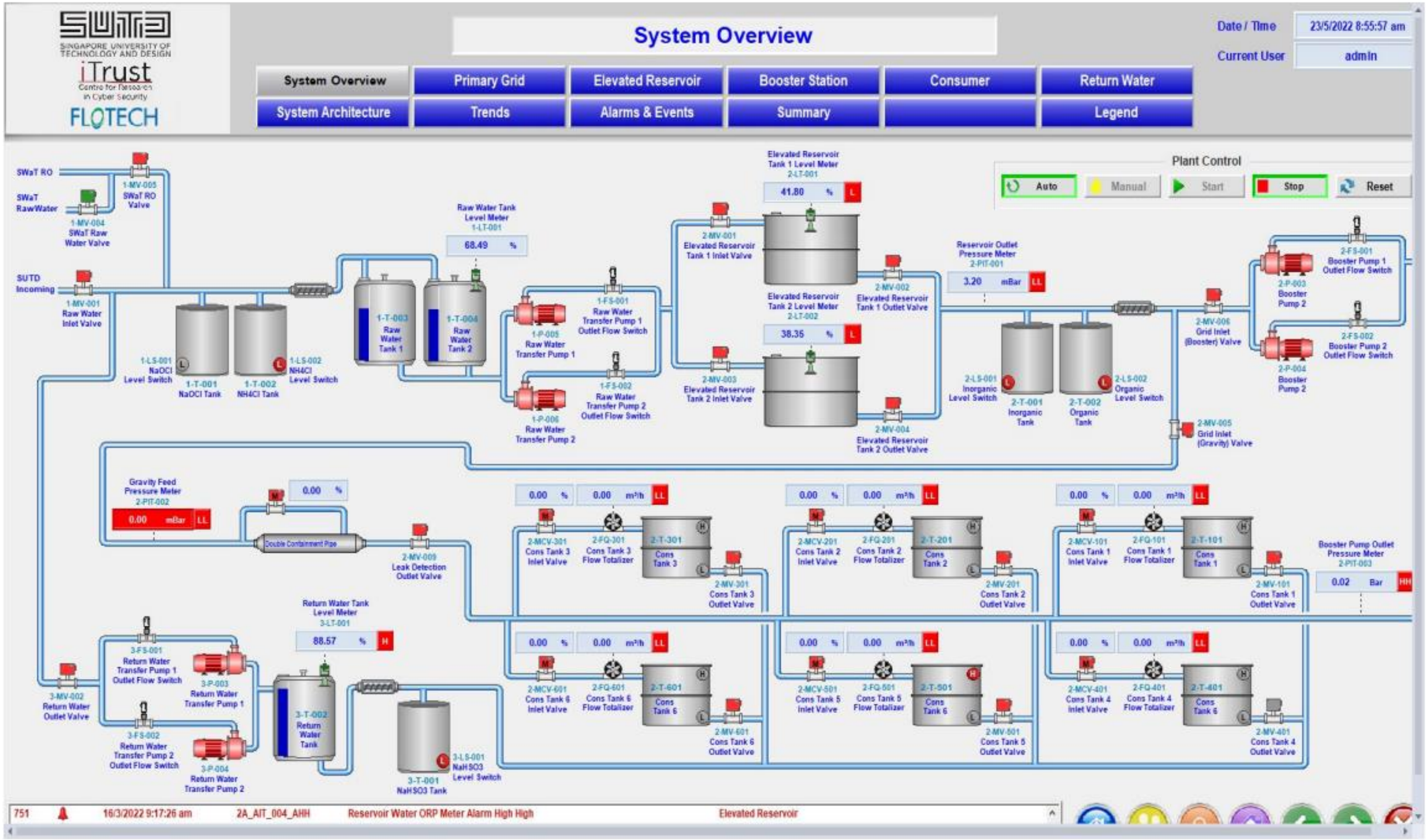
BATADAL2017

- [DT] focus on Attack
- Accuracy 95.375%
- Precision 64.625%
- Recall 65.625%
- F1-Scorre 65.121%



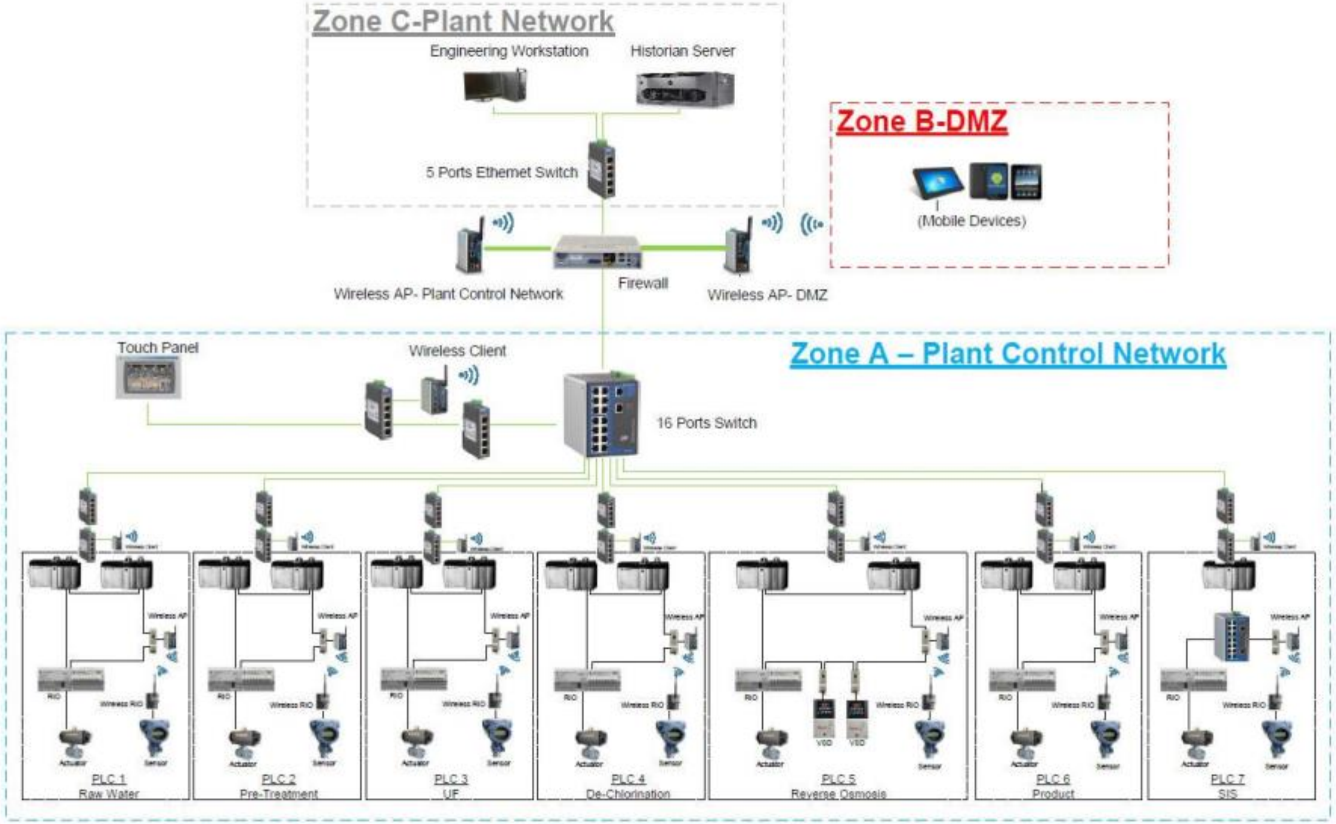
Mahidol University
Wisdom of the Land

WADI2019



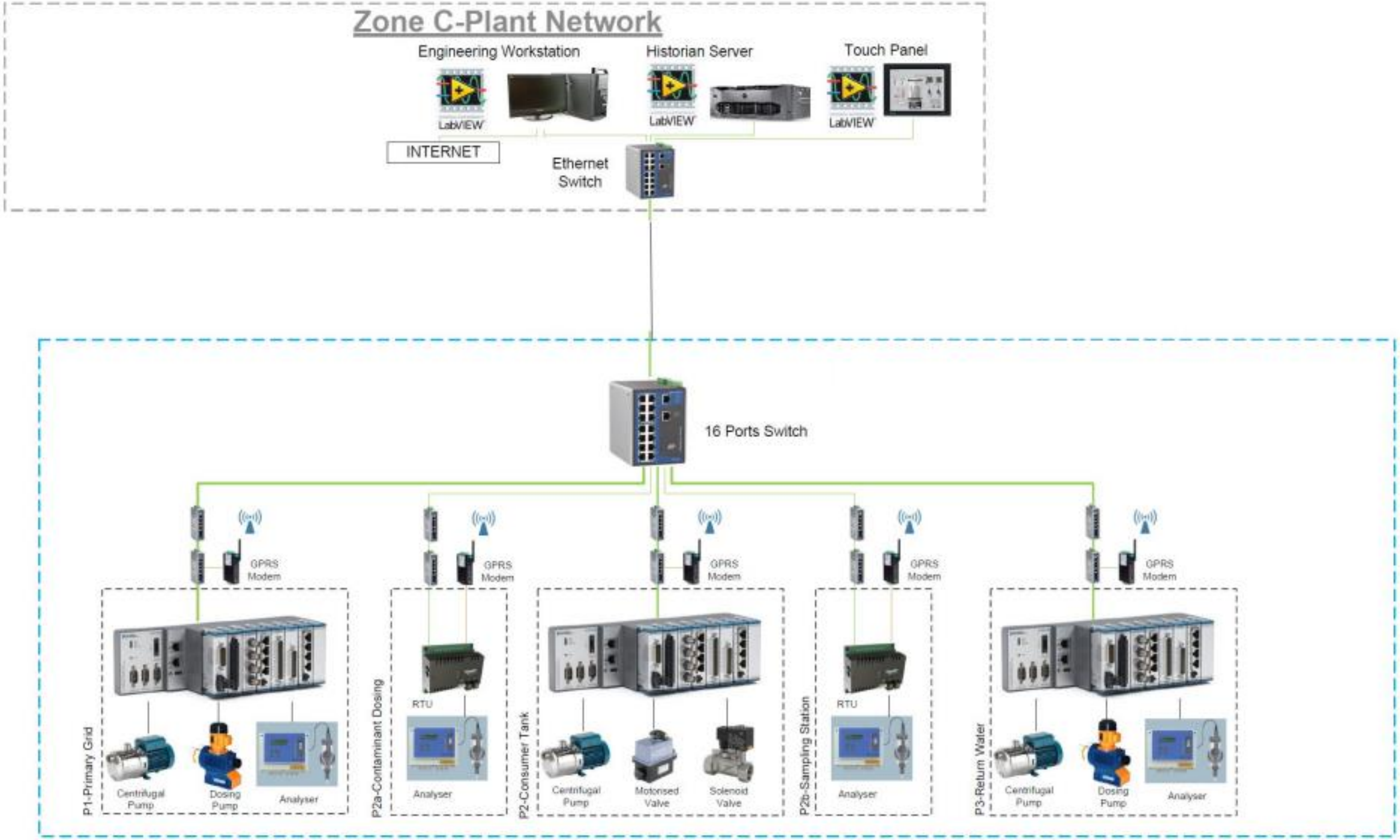
WADI2019

Water Treatment Network Diagram



PLC: Allen-Bradley
Subnet: 192.168.1.0/24
Protocol:
1. EtherNet/IP (ENIP)

WADI2019



Zone C-Plant Network
Engineering Workstation
Historian Server
Touch Panel
LabVIEW
LabVIEW
LabVIEW
INTERNET
Ethernet Switch
16 Ports Switch
GPRS Modem
GPRS Modem
GPRS Modem
GPRS Modem
GPRS Modem
P1-Primary Grid
Centrifugal Pump
Dosing Pump
Analyser
P2a-Contaminant Dosing
RTU
Analyser
P2-Consumer Tank
Centrifugal Pump
Motorised Valve
Solenoid Valve
P2b-Sampling Station
RTU
Analyser
P3-Return Water
Centrifugal Pump
Dosing Pump
Analyser