In [21]:

```
def doDiscretize(df_train, df_val, col, param_PAALen, param_Vocab, data_plot=Fals
    result = []
    # Step1 - Get train data values
    if (1):↔

    # Step2 - Get anomalies
    if (1):↔

    # Step3 - get y_predict
    if (1):↔

    if (verbose):↔

    if data_plot:↔

    return result, Y_predict

if __name__ == "__main__":
    # cols      = ['P_J280', 'P_J415', 'P_J302', 'P_J306', 'P_J307', 'P_J317']
    # cols      = ['P_J415']
    cols         = df_train.columns[:-1]
    results      = []
    Y_predicts   = []
    param_Vocab  = 'abcd'
    param_PAALen = 10
    for col in cols:
        result, Y_predict = doDiscretize(df_train, df_val, col
                                         , param_PAALen=param_PAALen, param_Vocab=
                                         , data_plot=False, verbose=False)
        results.append(result)
        Y_predicts.append(Y_predict)

    df = pd.DataFrame(results, columns=['column','TN', 'FP','FN', 'TP', 'accuracy
    display(df)

    Y_predicts = np.array(Y_predicts).T
    df_predict = pd.DataFrame(Y_predicts, columns=cols)
    df_predict.to_csv('results/03_Discretization_vocab_' + str(param_Vocab) + '_PA
```

executed in 956ms, finished 12:28:02 2019-06-02

```
 - [DEBUG] For col -  L_T1  || Anomalies :  ['bbb', 'cbd']
 - [DEBUG] For col -  L_T2  || Anomalies :  ['bab', 'ddd']
 - [DEBUG] For col -  L_T3  || Anomalies :  []
 - [DEBUG] For col -  L_T4  || Anomalies :  ['ada', 'bda', 'bdd', 'da
a', 'dad', 'dda']
 - [DEBUG] For col -  L_T5  || Anomalies :  []
 - [DEBUG] For col -  L_T6  || Anomalies :  ['aad', 'bac', 'caa', 'db
a']
 - [DEBUG] For col -  L_T7  || Anomalies :  ['aad', 'bdd', 'cda', 'da
b', 'dba', 'ddc']
 - [DEBUG] For col -  F_PU1  || Anomalies :  ['daa']
 - [DEBUG] For col -  S_PU1  || Anomalies :  []
 - [DEBUG] For col -  F_PU2  || Anomalies :  ['acd', 'cdc']
 - [DEBUG] For col -  S_PU2  || Anomalies :  []
 - [DEBUG] For col -  F_PU3  || Anomalies :  []
 - [DEBUG] For col -  S_PU3  || Anomalies :  []
 - [DEBUG] For col -  F_PU4  || Anomalies :  []
 - [DEBUG] For col -  S_PU4  || Anomalies :  []
```

```
 - [DEBUG] For col -  F_PU5  || Anomalies :  []
 - [DEBUG] For col -  S_PU5  || Anomalies :  []
 - [DEBUG] For col -  F_PU6  || Anomalies :  ['bdd', 'cbb', 'dbd', 'dc
b', 'ddc', 'ddd']
 - [DEBUG] For col -  S_PU6  || Anomalies :  ['bdd', 'cbb', 'dbd', 'dc
b', 'ddc', 'ddd']
 - [DEBUG] For col -  F_PU7  || Anomalies :  ['aaa', 'aab', 'abc', 'ba
b', 'caa']
 - [DEBUG] For col -  S_PU7  || Anomalies :  ['aaa', 'aab', 'abc', 'ca
a']
 - [DEBUG] For col -  F_PU8  || Anomalies :  []
 - [DEBUG] For col -  S_PU8  || Anomalies :  []
 - [DEBUG] For col -  F_PU9  || Anomalies :  []
 - [DEBUG] For col -  S_PU9  || Anomalies :  []
 - [DEBUG] For col -  F_PU10  || Anomalies :  ['aba', 'abb', 'bab']
 - [DEBUG] For col -  S_PU10  || Anomalies :  ['abb', 'bab']
 - [DEBUG] For col -  F_PU11  || Anomalies :  ['ddd']
 - [DEBUG] For col -  S_PU11  || Anomalies :  ['ddd']
 - [DEBUG] For col -  F_V2  || Anomalies :  ['abb', 'abd', 'bdc', 'da
b']
 - [DEBUG] For col -  S_V2  || Anomalies :  ['abb']
 - [DEBUG] For col -  P_J280  || Anomalies :  ['aac', 'bcb', 'cac', 'c
cd', 'dbc']
 - [DEBUG] For col -  P_J269  || Anomalies :  ['add']
 - [DEBUG] For col -  P_J300  || Anomalies :  ['cdc', 'dba']
 - [DEBUG] For col -  P_J256  || Anomalies :  []
 - [DEBUG] For col -  P_J289  || Anomalies :  ['cdc', 'dba']
 - [DEBUG] For col -  P_J415  || Anomalies :  ['cca']
 - [DEBUG] For col -  P_J302  || Anomalies :  ['aac', 'bdd', 'caa', 'c
dd', 'dba', 'dca', 'dcd', 'dda', 'ddc', 'ddd']
 - [DEBUG] For col -  P_J306  || Anomalies :  []
 - [DEBUG] For col -  P_J307  || Anomalies :  ['aac', 'bdd', 'caa', 'c
dd', 'dba', 'dca', 'dcd', 'dda', 'ddc', 'ddd']
 - [DEBUG] For col -  P_J317  || Anomalies :  ['bbd', 'bca', 'bdb', 'b
dd', 'dbc', 'dcc', 'ddc', 'ddd']
 - [DEBUG] For col -  P_J14  || Anomalies :  ['add', 'bdd', 'dcd', 'dd
b', 'ddc']
 - [DEBUG] For col -  P_J422  || Anomalies :  ['cba', 'dba']
```

	column	TN	FP	FN	TP	accuracy	precision	recall	f_score
0	L_T1	3898	60	219	0	93.320565	0.000000	0.000000	0.000000
1	L_T2	3898	60	219	0	93.320565	0.000000	0.000000	0.000000
2	L_T3	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
3	L_T4	3844	114	183	36	92.889634	24.000000	16.438356	19.512195
4	L_T5	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
5	L_T6	3828	130	219	0	91.644721	0.000000	0.000000	0.000000
6	L_T7	3738	220	219	0	89.490065	0.000000	0.000000	0.000000
7	F_PU1	3957	1	190	29	95.427340	96.666667	13.242009	23.293173
8	S_PU1	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
9	F_PU2	3918	40	219	0	93.799378	0.000000	0.000000	0.000000
10	S_PU2	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
11	F_PU3	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
12	S_PU3	3958	0	219	0	94.757003	0.000000	0.000000	0.000000

	column	TN	FP	FN	TP	accuracy	precision	recall	f_score
13	F_PU4	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
14	S_PU4	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
15	F_PU5	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
16	S_PU5	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
17	F_PU6	3888	70	139	80	94.996409	53.333333	36.529680	43.360434
18	S_PU6	3888	70	139	80	94.996409	53.333333	36.529680	43.360434
19	F_PU7	3856	102	141	78	94.182428	43.333333	35.616438	39.097744
20	S_PU7	3921	37	146	73	95.618865	66.363636	33.333333	44.376900
21	F_PU8	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
22	S_PU8	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
23	F_PU9	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
24	S_PU9	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
25	F_PU10	3930	28	177	42	95.092171	60.000000	19.178082	29.065744
26	S_PU10	3920	38	177	42	94.852765	52.500000	19.178082	28.093645
27	F_PU11	3940	18	177	42	95.331578	70.000000	19.178082	30.107527
28	S_PU11	3940	18	177	42	95.331578	70.000000	19.178082	30.107527
29	F_V2	3885	73	212	7	93.176921	8.750000	3.196347	4.682274
30	S_V2	3928	30	219	0	94.038784	0.000000	0.000000	0.000000
31	P_J280	3725	233	152	67	90.782859	22.333333	30.593607	25.818882
32	P_J269	3957	1	190	29	95.427340	96.666667	13.242009	23.293173
33	P_J300	3898	60	219	0	93.320565	0.000000	0.000000	0.000000
34	P_J256	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
35	P_J289	3898	60	219	0	93.320565	0.000000	0.000000	0.000000
36	P_J415	3928	30	219	0	94.038784	0.000000	0.000000	0.000000
37	P_J302	3782	176	145	74	92.315059	29.600000	33.789954	31.556503
38	P_J306	3958	0	219	0	94.757003	0.000000	0.000000	0.000000
39	P_J307	3782	176	145	74	92.315059	29.600000	33.789954	31.556503
40	P_J317	3850	108	177	42	93.176921	28.000000	19.178082	22.764228
41	P_J14	3932	26	155	64	95.666746	71.111111	29.223744	41.423948
42	P_J422	3898	60	219	0	93.320565	0.000000	0.000000	0.000000

In [ ]:

executed in 49ms, finished 12:22:33 2019-06-02

Discretization - All Sensors [...]

Final

- Discretize the sensor data using any of the methods discussed in class
  - Explain and show how it makes sense
  - Vizualize the discretization
- Apply any of the sequential data mining methods
  - What kind of anomalies can you detect using the sequential model?
  - Which sensors can be modeled effectively?

Temp

1. We attempted to use the saxpy library but it was really time consuming since they used np.append

In [ ]:

```
from saxpy.sax import *
# initialize sax parameters
window = 49
paa = 8
alpha = 3
sensor = 'L_T1'

# perform discretization for training set 1
sax1 = sax_via_window(df_train[sensor], window, paa, alpha, "none", 0.01)
print (sax1)
```

executed in 8ms, finished 23:22:10 2019-05-29

In [ ]:

```
len(df_train)//116
```

executed in 7m 25s, finished 11:42:47 2019-06-02