In [1]:
```
def concat_df(train_data,test_data):
    return pd.concat([train_data,test_data],sort=True).reset_index(drop=True)

def divide_df(all_data):
    return all_data.loc[:20630],all_data.loc[20631:]
```

In [2]:
```
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTE
# from imblearn.over_sampling import SMOTENC
# from sklearn.preprocessing import LabelEncoder
import xgboost as xgb
from xgboost.sklearn import XGBClassifier

# from sklearn import cross_validation, metrics
from sklearn import  metrics
from sklearn.model_selection import cross_validate

import types
def __iter__(self): return 0
```

In [3]:
```
train_data=pd.read_table("PM_train.txt", sep=" ", header=None)
test_data=pd.read_table("PM_test.txt", sep=" ", header=None)
```

In [4]:
```
pd.set_option('display.max_columns', None)
train_data.head(1)
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.7	1400.6	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392	23

In [5]:
```
print("train_data_contains: "+str(len(train_data))+" row and "+str(len(train_data.columns))+" columns")
print("test_data_contains: "+str(len(test_data))+" row and "+str(len(test_data.columns))+" columns")
```

```
train_data_contains: 20631 row and 28 columns
test_data_contains: 13096 row and 28 columns
```

In [6]:
```
print("first_3_row_of_train_data")
display(train_data.head(3))
```

```
print("first_3_row_of_test_data")
display(test_data.tail(3))
```

first_3_row_of_train_data

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390

first_3_row_of_test_data

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
13093	100	196	-0.0006	-0.0003	100.0	518.67	643.44	1593.15	1406.82	14.62	21.61	553.04	2388.11	9146.81	1.3	47.57	521.18	2388.04	8217.24	8.4569	0.
13094	100	197	-0.0038	0.0001	100.0	518.67	643.26	1594.99	1419.36	14.62	21.61	553.37	2388.07	9148.85	1.3	47.61	521.33	2388.08	8220.48	8.4711	0.
13095	100	198	0.0013	0.0003	100.0	518.67	642.95	1601.62	1424.99	14.62	21.61	552.48	2388.06	9155.03	1.3	47.80	521.07	2388.05	8214.64	8.4903	0.

In [7]:
```
# Subtract previous row value from the current row value in a Pandas column
train_data[26]=train_data[0] - train_data[0].shift(1)
test_data[26]=test_data[0] - test_data[0].shift(1)
```

In [8]:
```
pd.set_option('display.max_columns', None)
```

In [9]:
```
print("failure_point_of_train_data")
display(train_data)
print("failure_point_of_test_data")
display(test_data)
```

failure_point_of_train_data

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
20626	100	196	-0.0004	-0.0003	100.0	518.67	643.49	1597.98	1428.63	14.62	21.61	551.43	2388.19	9065.52	1.3	48.07	519.49	2388.26	8137.60	8.4956	0.
20627	100	197	-0.0016	-0.0005	100.0	518.67	643.54	1604.50	1433.58	14.62	21.61	550.86	2388.23	9065.11	1.3	48.04	519.68	2388.22	8136.50	8.5139	0.
20628	100	198	0.0004	0.0000	100.0	518.67	643.42	1602.46	1428.18	14.62	21.61	550.94	2388.24	9065.90	1.3	48.09	520.01	2388.24	8141.05	8.5646	0.
20629	100	199	-0.0011	0.0003	100.0	518.67	643.23	1605.26	1426.53	14.62	21.61	550.68	2388.25	9073.72	1.3	48.39	519.67	2388.23	8139.29	8.5389	0.
20630	100	200	-0.0032	-0.0005	100.0	518.67	643.85	1600.38	1432.14	14.62	21.61	550.79	2388.26	9061.48	1.3	48.20	519.30	2388.26	8137.33	8.5036	0.

20631 rows × 28 columns

failure_point_of_test_data

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	21.61	553.90	2388.04	9050.17	1.3	47.20	521.72	2388.03	8125.55	8.4052	0.
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	21.61	554.85	2388.01	9054.42	1.3	47.50	522.16	2388.06	8139.62	8.3803	0.
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	21.61	554.11	2388.05	9056.96	1.3	47.50	521.97	2388.03	8130.10	8.4441	0.
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	21.61	554.07	2388.03	9045.29	1.3	47.28	521.38	2388.05	8132.90	8.3917	0.
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	21.61	554.16	2388.01	9044.55	1.3	47.31	522.15	2388.03	8129.54	8.4031	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
13091	100	194	0.0049	0.0000	100.0	518.67	643.24	1599.45	1415.79	14.62	21.61	553.41	2388.02	9142.37	1.3	47.69	520.69	2388.00	8213.28	8.4715	0.
13092	100	195	-0.0011	-0.0001	100.0	518.67	643.22	1595.69	1422.05	14.62	21.61	553.22	2388.05	9140.68	1.3	47.60	521.05	2388.09	8210.85	8.4512	0.
13093	100	196	-0.0006	-0.0003	100.0	518.67	643.44	1593.15	1406.82	14.62	21.61	553.04	2388.11	9146.81	1.3	47.57	521.18	2388.04	8217.24	8.4569	0.
13094	100	197	-0.0038	0.0001	100.0	518.67	643.26	1594.99	1419.36	14.62	21.61	553.37	2388.07	9148.85	1.3	47.61	521.33	2388.08	8220.48	8.4711	0.
13095	100	198	0.0013	0.0003	100.0	518.67	642.95	1601.62	1424.99	14.62	21.61	552.48	2388.06	9155.03	1.3	47.80	521.07	2388.05	8214.64	8.4903	0.

13096 rows × 28 columns

In [10]:

```
train_data.rename(columns={0: 'asset_id', 1: 'run_time'}, inplace=True)
train_data.rename(columns={2: 'setting_1', 3: 'setting_2', 4: 'setting_3'}, inplace=True)
train_data.rename(columns={5: 's_1', 6: 's_2', 7: 's_3', 8: 's_4', 9: 's_5', 10: 's_6',
                           11: 's_7', 12: 's_8', 13: 's_9', 14: 's_10', 15: 's_11', 16: 's_12',
                           17: 's_13', 18: 's_14', 19: 's_15', 20: 's_16', 21: 's_17', 22: 's_18',
                           23: 's_19', 24: 's_20', 25: 's_21', 26: 'failure'}, inplace=True)

test_data.rename(columns={0: 'asset_id', 1: 'run_time'}, inplace=True)
test_data.rename(columns={2: 'setting_1', 3: 'setting_2', 4: 'setting_3'}, inplace=True)
```

```
test_data.rename(columns={5: 's_1', 6: 's_2', 7: 's_3', 8: 's_4', 9: 's_5', 10: 's_6',
                          11: 's_7', 12: 's_8', 13: 's_9', 14: 's_10', 15: 's_11', 16: 's_12',
                          17: 's_13', 18: 's_14', 19: 's_15', 20: 's_16', 21: 's_17', 22: 's_18',
                          23: 's_19', 24: 's_20', 25: 's_21', 26: 'failure'}, inplace=True)
```

In [11]:

```
pd.set_option('display.max_columns', None)
```

In [12]:

```
print("failure_point_of_train_data")
display(train_data.head(194))
print("failure_point_of_test_data")
display(test_data.tail(199))
```

failure_point_of_train_data

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13	
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	81
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	81
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	81
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	81
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	81
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	1	190	-0.0027	0.0001	100.0	518.67	643.64	1599.22	1425.95	14.62	21.61	551.29	2388.29	9040.58	1.3	48.33	520.04	2388.35	81
190	1	191	-0.0000	-0.0004	100.0	518.67	643.34	1602.36	1425.77	14.62	21.61	550.92	2388.28	9042.76	1.3	48.15	519.57	2388.30	81
191	1	192	0.0009	-0.0000	100.0	518.67	643.54	1601.41	1427.20	14.62	21.61	551.25	2388.32	9033.22	1.3	48.25	520.08	2388.32	81
192	2	1	-0.0018	0.0006	100.0	518.67	641.89	1583.84	1391.28	14.62	21.60	554.53	2388.01	9054.72	1.3	46.93	522.33	2388.06	81
193	2	2	0.0043	-0.0003	100.0	518.67	641.82	1587.05	1393.13	14.62	21.61	554.77	2387.98	9051.31	1.3	47.24	522.70	2387.98	81

194 rows × 28 columns

failure_point_of_test_data

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13
12897	99	97	0.0047	-0.0000	100.0	518.67	642.00	1585.03	1397.98	14.62	21.61	554.75	2388.01	9067.16	1.3	47.26	521.82	2388.02
12898	100	1	0.0014	0.0003	100.0	518.67	641.65	1591.50	1401.63	14.62	21.61	554.70	2388.05	9059.87	1.3	47.28	522.06	2388.02
12899	100	2	0.0031	0.0001	100.0	518.67	642.20	1588.99	1402.05	14.62	21.61	554.05	2387.99	9057.49	1.3	47.18	522.14	2388.07
12900	100	3	-0.0000	0.0001	100.0	518.67	642.27	1587.47	1396.74	14.62	21.61	554.85	2388.11	9052.23	1.3	47.11	522.54	2388.03

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13
12901	100	4	0.0011	0.0001	100.0	518.67	642.07	1579.17	1401.93	14.62	21.61	554.05	2388.01	9058.66	1.3	47.26	522.34	2388.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
13091	100	194	0.0049	0.0000	100.0	518.67	643.24	1599.45	1415.79	14.62	21.61	553.41	2388.02	9142.37	1.3	47.69	520.69	2388.00
13092	100	195	-0.0011	-0.0001	100.0	518.67	643.22	1595.69	1422.05	14.62	21.61	553.22	2388.05	9140.68	1.3	47.60	521.05	2388.09
13093	100	196	-0.0006	-0.0003	100.0	518.67	643.44	1593.15	1406.82	14.62	21.61	553.04	2388.11	9146.81	1.3	47.57	521.18	2388.04
13094	100	197	-0.0038	0.0001	100.0	518.67	643.26	1594.99	1419.36	14.62	21.61	553.37	2388.07	9148.85	1.3	47.61	521.33	2388.08
13095	100	198	0.0013	0.0003	100.0	518.67	642.95	1601.62	1424.99	14.62	21.61	552.48	2388.06	9155.03	1.3	47.80	521.07	2388.05

199 rows × 28 columns

In [13]:
```
### print("missing_values_in_train_data")
### display(train_data.isnull().sum())
### print("missing_value_in_test_data")
### display(test_data.isnull().sum())
```

In [14]:
```
### train_data.describe()
```

In [15]:
```
### test_data.describe()
```

In [16]:
```
# df_all=concat_df(train_data,test_data)
# train_data,test_data=divide_df(df_all)
```

In [17]:
```
#Subtract values from maximum value within groups
train_data[27]=train_data.groupby('asset_id').run_time.transform('max') - train_data.run_time
train_data["failure"].fillna(0, inplace = True)   ###0.0
train_data.rename(columns={27: 'remain_cycle'}, inplace=True)

#Subtract values from maximum value within groups
test_data[27]=test_data.groupby('asset_id').run_time.transform('max') - test_data.run_time
test_data["failure"].fillna(0, inplace = True)   ###0.0
test_data.rename(columns={27: 'remain_cycle'}, inplace=True)
```

In [18]:
```
test_data.head(33)
```

Out[18]:

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13	s
0	1	1	0.0023	0.0003	100.0	518.67	643.02	1585.29	1398.21	14.62	21.61	553.90	2388.04	9050.17	1.3	47.20	521.72	2388.03	812
1	1	2	-0.0027	-0.0003	100.0	518.67	641.71	1588.45	1395.42	14.62	21.61	554.85	2388.01	9054.42	1.3	47.50	522.16	2388.06	813
2	1	3	0.0003	0.0001	100.0	518.67	642.46	1586.94	1401.34	14.62	21.61	554.11	2388.05	9056.96	1.3	47.50	521.97	2388.03	813
3	1	4	0.0042	0.0000	100.0	518.67	642.44	1584.12	1406.42	14.62	21.61	554.07	2388.03	9045.29	1.3	47.28	521.38	2388.05	813
4	1	5	0.0014	0.0000	100.0	518.67	642.51	1587.19	1401.92	14.62	21.61	554.16	2388.01	9044.55	1.3	47.31	522.15	2388.03	812
5	1	6	0.0012	0.0003	100.0	518.67	642.11	1579.12	1395.13	14.62	21.61	554.22	2388.00	9050.96	1.3	47.26	521.92	2388.08	812
6	1	7	-0.0000	0.0002	100.0	518.67	642.11	1583.34	1404.84	14.62	21.61	553.89	2388.05	9051.39	1.3	47.31	522.01	2388.06	813
7	1	8	0.0006	-0.0000	100.0	518.67	642.54	1580.89	1400.89	14.62	21.61	553.59	2388.05	9052.86	1.3	47.21	522.09	2388.06	812
8	1	9	-0.0036	0.0000	100.0	518.67	641.88	1593.29	1412.28	14.62	21.61	554.49	2388.06	9048.55	1.3	47.37	522.03	2388.05	813
9	1	10	-0.0025	-0.0001	100.0	518.67	642.07	1585.25	1398.64	14.62	21.61	554.28	2388.04	9051.95	1.3	47.14	522.00	2388.06	813
10	1	11	0.0007	-0.0004	100.0	518.67	642.04	1581.03	1403.83	14.62	21.61	554.69	2388.04	9051.67	1.3	47.23	521.95	2388.06	813
11	1	12	0.0026	0.0003	100.0	518.67	642.54	1587.43	1397.82	14.62	21.61	554.35	2388.02	9050.02	1.3	47.27	522.01	2388.06	813
12	1	13	-0.0056	0.0003	100.0	518.67	641.94	1589.09	1403.94	14.62	21.61	554.04	2388.02	9045.67	1.3	47.35	522.37	2388.03	813
13	1	14	0.0017	-0.0004	100.0	518.67	642.23	1583.16	1402.88	14.62	21.61	554.66	2388.03	9045.30	1.3	47.24	521.95	2388.06	813
14	1	15	-0.0003	-0.0003	100.0	518.67	642.50	1584.81	1398.79	14.62	21.61	554.15	2388.00	9052.59	1.3	47.35	521.38	2388.00	813
15	1	16	-0.0018	0.0003	100.0	518.67	642.32	1584.51	1407.76	14.62	21.61	553.82	2388.10	9041.94	1.3	47.39	522.16	2388.10	813
16	1	17	0.0014	0.0002	100.0	518.67	642.19	1582.70	1404.12	14.62	21.61	554.42	2388.06	9045.85	1.3	47.27	522.09	2388.02	812
17	1	18	0.0035	0.0001	100.0	518.67	642.59	1586.53	1403.69	14.62	21.61	553.50	2388.04	9048.12	1.3	47.44	522.14	2388.06	813
18	1	19	0.0029	0.0001	100.0	518.67	642.43	1585.58	1402.30	14.62	21.61	553.87	2388.01	9046.90	1.3	47.25	522.06	2388.01	812
19	1	20	0.0011	-0.0001	100.0	518.67	642.61	1587.78	1400.70	14.62	21.61	554.31	2388.05	9041.12	1.3	47.46	522.28	2388.05	812
20	1	21	0.0038	-0.0002	100.0	518.67	642.70	1583.30	1399.20	14.62	21.61	554.42	2388.05	9053.73	1.3	47.36	522.05	2388.11	812
21	1	22	0.0012	0.0001	100.0	518.67	642.45	1582.78	1404.06	14.62	21.61	553.43	2388.00	9046.45	1.3	47.26	521.41	2388.04	812
22	1	23	0.0009	-0.0000	100.0	518.67	642.12	1587.51	1395.09	14.62	21.61	555.07	2388.04	9052.06	1.3	47.19	522.00	2388.06	813
23	1	24	-0.0006	-0.0001	100.0	518.67	642.32	1594.29	1400.15	14.62	21.61	553.27	2388.07	9043.32	1.3	47.29	522.06	2388.12	813
24	1	25	0.0028	-0.0003	100.0	518.67	642.25	1582.43	1400.23	14.62	21.61	553.76	2388.11	9043.80	1.3	47.37	522.26	2388.08	812
25	1	26	0.0047	-0.0005	100.0	518.67	642.48	1583.28	1408.07	14.62	21.61	554.59	2388.08	9053.43	1.3	47.33	521.95	2388.07	812
26	1	27	-0.0007	0.0001	100.0	518.67	642.08	1586.65	1400.31	14.62	21.61	554.35	2388.09	9046.10	1.3	47.34	521.82	2388.02	812
27	1	28	0.0022	0.0005	100.0	518.67	641.93	1594.25	1401.29	14.62	21.61	553.56	2388.07	9055.56	1.3	47.05	521.84	2388.07	813
28	1	29	0.0014	0.0001	100.0	518.67	641.95	1587.15	1398.11	14.62	21.61	554.15	2388.08	9046.11	1.3	47.42	522.39	2388.07	813

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13	s
29	1	30	-0.0025	0.0004	100.0	518.67	642.79	1585.72	1400.97	14.62	21.61	554.10	2388.09	9047.45	1.3	47.40	521.78	2388.10	813
30	1	31	-0.0006	0.0004	100.0	518.67	642.58	1581.22	1398.91	14.62	21.61	554.42	2388.08	9056.40	1.3	47.23	521.79	2388.06	813
31	2	1	-0.0009	0.0004	100.0	518.67	642.66	1589.30	1407.16	14.62	21.61	553.14	2388.10	9040.20	1.3	47.43	521.62	2388.14	812
32	2	2	-0.0011	0.0002	100.0	518.67	642.51	1588.43	1405.47	14.62	21.61	553.53	2388.07	9053.77	1.3	47.45	522.02	2388.08	812

In [19]:
```
train_data.head(194)
```

Out[19]:

	asset_id	run_time	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_10	s_11	s_12	s_13	
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	21.61	554.36	2388.06	9046.19	1.3	47.47	521.66	2388.02	81
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	21.61	553.75	2388.04	9044.07	1.3	47.49	522.28	2388.07	81
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	21.61	554.26	2388.08	9052.94	1.3	47.27	522.42	2388.03	81
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	21.61	554.45	2388.11	9049.48	1.3	47.13	522.86	2388.08	81
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	21.61	554.00	2388.06	9055.15	1.3	47.28	522.19	2388.04	81
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	1	190	-0.0027	0.0001	100.0	518.67	643.64	1599.22	1425.95	14.62	21.61	551.29	2388.29	9040.58	1.3	48.33	520.04	2388.35	81
190	1	191	-0.0000	-0.0004	100.0	518.67	643.34	1602.36	1425.77	14.62	21.61	550.92	2388.28	9042.76	1.3	48.15	519.57	2388.30	81
191	1	192	0.0009	-0.0000	100.0	518.67	643.54	1601.41	1427.20	14.62	21.61	551.25	2388.32	9033.22	1.3	48.25	520.08	2388.32	81
192	2	1	-0.0018	0.0006	100.0	518.67	641.89	1583.84	1391.28	14.62	21.60	554.53	2388.01	9054.72	1.3	46.93	522.33	2388.06	81
193	2	2	0.0043	-0.0003	100.0	518.67	641.82	1587.05	1393.13	14.62	21.61	554.77	2387.98	9051.31	1.3	47.24	522.70	2387.98	81

194 rows × 28 columns

In [20]:
```
train_data['coming'] = np.where(((train_data.remain_cycle < 21) & ((train_data.remain_cycle>=0))), 1, 0)
tips_summed = train_data.groupby(['coming'])['s_4'].count()
tips_summed
```

Out[20]:
```
coming
0    18531
1     2100
Name: s_4, dtype: int64
```

In [21]:
```
test_data['coming'] = np.where(((test_data.remain_cycle < 21) & ((test_data.remain_cycle>=0))), 1, 0)
```

```
tips_summed = test_data.groupby(['coming'])['s_5'].count()
tips_summed
```

Out[21]:
```
coming
0    10996
1     2100
Name: s_5, dtype: int64
```

In [22]:
```
print(train_data['coming'].mean())
print(test_data['coming'].mean())
```

```
0.10178857059764432
0.16035430665852168
```

In [23]:
```
df_all=concat_df(train_data,test_data)
#train_data,test_data=divide_df(df_all)
```

In [24]:
```
df_all
```

Out[24]:

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
0	1	0	0.0	191	1	518.67	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392	2388	100.0	641.82	39.06	23.4190
1	1	0	0.0	190	2	518.67	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392	2388	100.0	642.15	39.00	23.4236
2	1	0	0.0	189	3	518.67	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390	2388	100.0	642.35	38.95	23.3442
3	1	0	0.0	188	4	518.67	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.03	392	2388	100.0	642.35	38.88	23.3739
4	1	0	0.0	187	5	518.67	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.03	393	2388	100.0	642.37	38.90	23.4044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
33722	100	1	0.0	4	194	518.67	1.3	47.69	520.69	2388.00	8213.28	8.4715	0.03	394	2388	100.0	643.24	38.65	23.1974
33723	100	1	0.0	3	195	518.67	1.3	47.60	521.05	2388.09	8210.85	8.4512	0.03	395	2388	100.0	643.22	38.57	23.2771
33724	100	1	0.0	2	196	518.67	1.3	47.57	521.18	2388.04	8217.24	8.4569	0.03	395	2388	100.0	643.44	38.62	23.2051
33725	100	1	0.0	1	197	518.67	1.3	47.61	521.33	2388.08	8220.48	8.4711	0.03	395	2388	100.0	643.26	38.66	23.2699
33726	100	1	0.0	0	198	518.67	1.3	47.80	521.07	2388.05	8214.64	8.4903	0.03	396	2388	100.0	642.95	38.70	23.1855

33727 rows × 29 columns

In [25]:
```
import numpy as np
# Create a running mean, max, min, and median for the sensor variables
```

```
dfx=df_all
### dfx=dfx.sort_values(by=['asset_id', 'run_time'], ascending=[True, True])

dfx['setting_1_mean'] = np.where((df_all.coming == 0),(df_all['setting_1'].rolling(min_periods=1, window=21).mean()) , df_all.set
dfx['setting_2_mean'] = np.where((df_all.coming == 0),(df_all['setting_2'].rolling(min_periods=1, window=21).mean()) , df_all.set

dfx['s_2_mean'] = np.where((df_all.coming == 0),(df_all['s_2'].rolling(min_periods=1, window=21).mean()) , df_all.s_2)
dfx['s_2_median'] = np.where((df_all.coming == 0),(df_all['s_2'].rolling(min_periods=1, window=21).median()) , df_all.s_2)
dfx['s_2_max'] = np.where((df_all.coming == 0),(df_all['s_2'].rolling(min_periods=1, window=21).max()) , df_all.s_2)
dfx['s_2_min'] = np.where((df_all.coming == 0),(df_all['s_2'].rolling(min_periods=1, window=21).min()) , df_all.s_2)

dfx['s_3_mean'] = np.where((df_all.coming == 0),(df_all['s_3'].rolling(min_periods=1, window=21).mean()) , df_all.s_3)
dfx['s_3_median'] = np.where((df_all.coming == 0),(df_all['s_3'].rolling(min_periods=1, window=21).median()) , df_all.s_3)
dfx['s_3_max'] = np.where((df_all.coming == 0),(df_all['s_3'].rolling(min_periods=1, window=21).max()) , df_all.s_3)
dfx['s_3_min'] = np.where((df_all.coming == 0),(df_all['s_3'].rolling(min_periods=1, window=21).min()) , df_all.s_3)

dfx['s_4_mean'] = np.where((df_all.coming == 0),(df_all['s_4'].rolling(min_periods=1, window=21).mean()) , df_all.s_4)
dfx['s_4_median'] = np.where((df_all.coming == 0),(df_all['s_4'].rolling(min_periods=1, window=21).median()) , df_all.s_4)
dfx['s_4_max'] = np.where((df_all.coming == 0),(df_all['s_4'].rolling(min_periods=1, window=21).max()) , df_all.s_4)
dfx['s_4_min'] = np.where((df_all.coming == 0),(df_all['s_4'].rolling(min_periods=1, window=21).min()) , df_all.s_4)

dfx['s_7_mean'] = np.where((df_all.coming == 0),(df_all['s_7'].rolling(min_periods=1, window=21).mean()) , df_all.s_7)
dfx['s_7_median'] = np.where((df_all.coming == 0),(df_all['s_7'].rolling(min_periods=1, window=21).median()) , df_all.s_7)
dfx['s_7_max'] = np.where((df_all.coming == 0),(df_all['s_7'].rolling(min_periods=1, window=21).max()) , df_all.s_7)
dfx['s_7_min'] = np.where((df_all.coming == 0),(df_all['s_7'].rolling(min_periods=1, window=21).min()) , df_all.s_7)

dfx['s_8_mean'] = np.where((df_all.coming == 0),(df_all['s_8'].rolling(min_periods=1, window=21).mean()) , df_all.s_8)
dfx['s_8_median'] = np.where((df_all.coming == 0),(df_all['s_8'].rolling(min_periods=1, window=21).median()) , df_all.s_8)
dfx['s_8_max'] = np.where((df_all.coming == 0),(df_all['s_8'].rolling(min_periods=1, window=21).max()) , df_all.s_8)
dfx['s_8_min'] = np.where((df_all.coming == 0),(df_all['s_8'].rolling(min_periods=1, window=21).min()) , df_all.s_8)

dfx['s_9_mean'] = np.where((df_all.coming == 0),(df_all['s_9'].rolling(min_periods=1, window=21).mean()) , df_all.s_9)
dfx['s_9_median'] = np.where((df_all.coming == 0),(df_all['s_9'].rolling(min_periods=1, window=21).median()) , df_all.s_9)
dfx['s_9_max'] = np.where((df_all.coming == 0),(df_all['s_9'].rolling(min_periods=1, window=21).max()) , df_all.s_9)
dfx['s_9_min'] = np.where((df_all.coming == 0),(df_all['s_9'].rolling(min_periods=1, window=21).min()) , df_all.s_9)

dfx['s_11_mean'] = np.where((df_all.coming == 0),(df_all['s_11'].rolling(min_periods=1, window=21).mean()) , df_all.s_11)
dfx['s_11_median'] = np.where((df_all.coming == 0),(df_all['s_11'].rolling(min_periods=1, window=21).median()) , df_all.s_11)
dfx['s_11_max'] = np.where((df_all.coming == 0),(df_all['s_11'].rolling(min_periods=1, window=21).max()) , df_all.s_11)
dfx['s_11_min'] = np.where((df_all.coming == 0),(df_all['s_11'].rolling(min_periods=1, window=21).min()) , df_all.s_11)

dfx['s_12_mean'] = np.where((df_all.coming == 0),(df_all['s_12'].rolling(min_periods=1, window=21).mean()) , df_all.s_12)
dfx['s_12_median'] = np.where((df_all.coming == 0),(df_all['s_12'].rolling(min_periods=1, window=21).median()) , df_all.s_12)
dfx['s_12_max'] = np.where((df_all.coming == 0),(df_all['s_12'].rolling(min_periods=1, window=21).max()) , df_all.s_12)
dfx['s_12_min'] = np.where((df_all.coming == 0),(df_all['s_12'].rolling(min_periods=1, window=21).min()) , df_all.s_12)

dfx['s_13_mean'] = np.where((df_all.coming == 0),(df_all['s_13'].rolling(min_periods=1, window=21).mean()) , df_all.s_13)
dfx['s_13_median'] = np.where((df_all.coming == 0),(df_all['s_13'].rolling(min_periods=1, window=21).median()) , df_all.s_13)
```

```
dfx['s_13_max'] = np.where((df_all.coming == 0),(df_all['s_13'].rolling(min_periods=1, window=21).max()) , df_all.s_13)
dfx['s_13_min'] = np.where((df_all.coming == 0),(df_all['s_13'].rolling(min_periods=1, window=21).min()) , df_all.s_13)

dfx['s_14_mean'] = np.where((df_all.coming == 0),(df_all['s_14'].rolling(min_periods=1, window=21).mean()) , df_all.s_14)
dfx['s_14_median'] = np.where((df_all.coming == 0),(df_all['s_14'].rolling(min_periods=1, window=21).median()) , df_all.s_14)
dfx['s_14_max'] = np.where((df_all.coming == 0),(df_all['s_14'].rolling(min_periods=1, window=21).max()) , df_all.s_14)
dfx['s_14_min'] = np.where((df_all.coming == 0),(df_all['s_14'].rolling(min_periods=1, window=21).min()) , df_all.s_14)

dfx['s_15_mean'] = np.where((df_all.coming == 0),(df_all['s_15'].rolling(min_periods=1, window=21).mean()) , df_all.s_15)
dfx['s_15_median'] = np.where((df_all.coming == 0),(df_all['s_15'].rolling(min_periods=1, window=21).median()) , df_all.s_15)
dfx['s_15_max'] = np.where((df_all.coming == 0),(df_all['s_15'].rolling(min_periods=1, window=21).max()) , df_all.s_15)
dfx['s_15_min'] = np.where((df_all.coming == 0),(df_all['s_15'].rolling(min_periods=1, window=21).min()) , df_all.s_15)

dfx['s_17_mean'] = np.where((df_all.coming == 0),(df_all['s_17'].rolling(min_periods=1, window=21).mean()) , df_all.s_17)
dfx['s_17_median'] = np.where((df_all.coming == 0),(df_all['s_17'].rolling(min_periods=1, window=21).median()) , df_all.s_17)
dfx['s_17_max'] = np.where((df_all.coming == 0),(df_all['s_17'].rolling(min_periods=1, window=21).max()) , df_all.s_17)
dfx['s_17_min'] = np.where((df_all.coming == 0),(df_all['s_17'].rolling(min_periods=1, window=21).min()) , df_all.s_17)

dfx['s_20_mean'] = np.where((df_all.coming == 0),(df_all['s_20'].rolling(min_periods=1, window=21).mean()) , df_all.s_20)
dfx['s_20_median'] = np.where((df_all.coming == 0),(df_all['s_20'].rolling(min_periods=1, window=21).median()) , df_all.s_20)
dfx['s_20_max'] = np.where((df_all.coming == 0),(df_all['s_20'].rolling(min_periods=1, window=21).max()) , df_all.s_20)
dfx['s_20_min'] = np.where((df_all.coming == 0),(df_all['s_20'].rolling(min_periods=1, window=21).min()) , df_all.s_20)

dfx['s_21_mean'] = np.where((df_all.coming == 0),(df_all['s_21'].rolling(min_periods=1, window=21).mean()) , df_all.s_21)
dfx['s_21_median'] = np.where((df_all.coming == 0),(df_all['s_21'].rolling(min_periods=1, window=21).median()) , df_all.s_21)
dfx['s_21_max'] = np.where((df_all.coming == 0),(df_all['s_21'].rolling(min_periods=1, window=21).max()) , df_all.s_21)
dfx['s_21_min'] = np.where((df_all.coming == 0),(df_all['s_21'].rolling(min_periods=1, window=21).min()) , df_all.s_21)
```

In [26]:

```
# Another useful transformation is to look for sudden spikes in sensor values
# This code creates a value indicating how far the current value is from the immediate norm

dfx['setting_1_chg'] = np.where((df_all.setting_1_mean == 0),0 , df_all.setting_1/df_all.setting_1_mean)
dfx['setting_2_chg'] = np.where((df_all.setting_2_mean == 0),0 , df_all.setting_2/df_all.setting_2_mean)

dfx['s_2_chg'] = np.where((df_all.s_2_mean == 0),0 , df_all.s_2/df_all.s_2_mean)
dfx['s_3_chg'] = np.where((df_all.s_3_mean == 0),0 , df_all.s_3/df_all.s_3_mean)
dfx['s_4_chg'] = np.where((df_all.s_4_mean == 0),0 , df_all.s_4/df_all.s_4_mean)

dfx['s_7_chg'] = np.where((df_all.s_7_mean == 0),0 , df_all.s_7/df_all.s_7_mean)
dfx['s_8_chg'] = np.where((df_all.s_8_mean == 0),0 , df_all.s_8/df_all.s_8_mean)
dfx['s_9_chg'] = np.where((df_all.s_9_mean == 0),0 , df_all.s_9/df_all.s_9_mean)

dfx['s_11_chg'] = np.where((df_all.s_11_mean == 0),0 , df_all.s_11/df_all.s_11_mean)
dfx['s_12_chg'] = np.where((df_all.s_12_mean == 0),0 , df_all.s_12/df_all.s_12_mean)
dfx['s_13_chg'] = np.where((df_all.s_13_mean == 0),0 , df_all.s_13/df_all.s_13_mean)

dfx['s_14_chg'] = np.where((df_all.s_14_mean == 0),0 , df_all.s_14/df_all.s_14_mean)
dfx['s_15_chg'] = np.where((df_all.s_15_mean == 0),0 , df_all.s_15/df_all.s_15_mean)
```

```
dfx['s_17_chg'] = np.where((df_all.s_17_mean == 0),0 , df_all.s_17/df_all.s_17_mean)

dfx['s_20_chg'] = np.where((df_all.s_20_mean == 0),0 , df_all.s_20/df_all.s_20_mean)
dfx['s_21_chg'] = np.where((df_all.s_21_mean == 0),0 , df_all.s_21/df_all.s_21_mean)

df_all=dfx
```

In [27]:

```
df_all.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33727 entries, 0 to 33726
Data columns (total 103 columns):
 #    Column          Dtype
---   ------          -----
 0    asset_id        int64
 1    coming          int64
 2    failure         float64
 3    remain_cycle    int64
 4    run_time        int64
 5    s_1             float64
 6    s_10            float64
 7    s_11            float64
 8    s_12            float64
 9    s_13            float64
 10   s_14            float64
 11   s_15            float64
 12   s_16            float64
 13   s_17            int64
 14   s_18            int64
 15   s_19            float64
 16   s_2             float64
 17   s_20            float64
 18   s_21            float64
 19   s_3             float64
 20   s_4             float64
 21   s_5             float64
 22   s_6             float64
 23   s_7             float64
 24   s_8             float64
 25   s_9             float64
 26   setting_1       float64
 27   setting_2       float64
 28   setting_3       float64
 29   setting_1_mean  float64
 30   setting_2_mean  float64
 31   s_2_mean        float64
 32   s_2_median      float64
 33   s_2_max         float64
 34   s_2_min         float64
 35   s_3_mean        float64
```

```
36   s_3_median      float64
37   s_3_max         float64
38   s_3_min         float64
39   s_4_mean        float64
40   s_4_median      float64
41   s_4_max         float64
42   s_4_min         float64
43   s_7_mean        float64
44   s_7_median      float64
45   s_7_max         float64
46   s_7_min         float64
47   s_8_mean        float64
48   s_8_median      float64
49   s_8_max         float64
50   s_8_min         float64
51   s_9_mean        float64
52   s_9_median      float64
53   s_9_max         float64
54   s_9_min         float64
55   s_11_mean       float64
56   s_11_median     float64
57   s_11_max        float64
58   s_11_min        float64
59   s_12_mean       float64
60   s_12_median     float64
61   s_12_max        float64
62   s_12_min        float64
63   s_13_mean       float64
64   s_13_median     float64
65   s_13_max        float64
66   s_13_min        float64
67   s_14_mean       float64
68   s_14_median     float64
69   s_14_max        float64
70   s_14_min        float64
71   s_15_mean       float64
72   s_15_median     float64
73   s_15_max        float64
74   s_15_min        float64
75   s_17_mean       float64
76   s_17_median     float64
77   s_17_max        float64
78   s_17_min        float64
79   s_20_mean       float64
80   s_20_median     float64
81   s_20_max        float64
82   s_20_min        float64
83   s_21_mean       float64
84   s_21_median     float64
85   s_21_max        float64
86   s_21_min        float64
87   setting_1_chg   float64
```

```
 88   setting_2_chg   float64
 89   s_2_chg         float64
 90   s_3_chg         float64
 91   s_4_chg         float64
 92   s_7_chg         float64
 93   s_8_chg         float64
 94   s_9_chg         float64
 95   s_11_chg        float64
 96   s_12_chg        float64
 97   s_13_chg        float64
 98   s_14_chg        float64
 99   s_15_chg        float64
 100  s_17_chg        float64
 101  s_20_chg        float64
 102  s_21_chg        float64
dtypes: float64(97), int64(6)
memory usage: 26.5 MB
```

In [28]:
```
df_all['s_17'] = df_all.s_17.astype(float)
```

In [29]:
```
df_all['s_18'] = df_all.s_18.astype(float)
```

In [30]:
```
df_all.head(196)
```

Out[30]:

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
0	1	0	0.0	191	1	518.67	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392.0	2388.0	100.0	641.82	39.06	23.4190
1	1	0	0.0	190	2	518.67	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392.0	2388.0	100.0	642.15	39.00	23.4236
2	1	0	0.0	189	3	518.67	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390.0	2388.0	100.0	642.35	38.95	23.3442
3	1	0	0.0	188	4	518.67	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.03	392.0	2388.0	100.0	642.35	38.88	23.3739
4	1	0	0.0	187	5	518.67	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.03	393.0	2388.0	100.0	642.37	38.90	23.4044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
191	1	1	0.0	0	192	518.67	1.3	48.25	520.08	2388.32	8110.93	8.5113	0.03	396.0	2388.0	100.0	643.54	38.48	22.9649
192	2	0	1.0	286	1	518.67	1.3	46.93	522.33	2388.06	8137.72	8.3905	0.03	391.0	2388.0	100.0	641.89	38.94	23.4585
193	2	0	0.0	285	2	518.67	1.3	47.24	522.70	2387.98	8131.09	8.4167	0.03	392.0	2388.0	100.0	641.82	39.06	23.4085
194	2	0	0.0	284	3	518.67	1.3	47.22	522.58	2387.99	8140.58	8.3802	0.03	391.0	2388.0	100.0	641.55	39.11	23.4250
195	2	0	0.0	283	4	518.67	1.3	47.10	522.49	2387.93	8140.44	8.4018	0.03	391.0	2388.0	100.0	641.68	39.13	23.5027

196 rows × 103 columns

In [31]:
```
#df_all=concat_df(train_data,test_data)
train_data,test_data=divide_df(df_all)
```

In [32]:
```
### train_data.info(verbose=True)
```

In [33]:
```
### test_data.info(verbose=True)
```

In [34]:
```
train_data#.head(196)
```

Out[34]:

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
0	1	0	0.0	191	1	518.67	1.3	47.47	521.66	2388.02	8138.62	8.4195	0.03	392.0	2388.0	100.0	641.82	39.06	23.4190
1	1	0	0.0	190	2	518.67	1.3	47.49	522.28	2388.07	8131.49	8.4318	0.03	392.0	2388.0	100.0	642.15	39.00	23.4236
2	1	0	0.0	189	3	518.67	1.3	47.27	522.42	2388.03	8133.23	8.4178	0.03	390.0	2388.0	100.0	642.35	38.95	23.3442
3	1	0	0.0	188	4	518.67	1.3	47.13	522.86	2388.08	8133.83	8.3682	0.03	392.0	2388.0	100.0	642.35	38.88	23.3739
4	1	0	0.0	187	5	518.67	1.3	47.28	522.19	2388.04	8133.80	8.4294	0.03	393.0	2388.0	100.0	642.37	38.90	23.4044
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20626	100	1	0.0	4	196	518.67	1.3	48.07	519.49	2388.26	8137.60	8.4956	0.03	397.0	2388.0	100.0	643.49	38.49	22.9735
20627	100	1	0.0	3	197	518.67	1.3	48.04	519.68	2388.22	8136.50	8.5139	0.03	395.0	2388.0	100.0	643.54	38.30	23.1594
20628	100	1	0.0	2	198	518.67	1.3	48.09	520.01	2388.24	8141.05	8.5646	0.03	398.0	2388.0	100.0	643.42	38.44	22.9333
20629	100	1	0.0	1	199	518.67	1.3	48.39	519.67	2388.23	8139.29	8.5389	0.03	395.0	2388.0	100.0	643.23	38.29	23.0640
20630	100	1	0.0	0	200	518.67	1.3	48.20	519.30	2388.26	8137.33	8.5036	0.03	396.0	2388.0	100.0	643.85	38.37	23.0522

20631 rows × 103 columns

In [35]:
```
test_data.head(33)
```

Out[35]:

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
20631	1	0	0.0	30	1	518.67	1.3	47.20	521.72	2388.03	8125.55	8.4052	0.03	392.0	2388.0	100.0	643.02	38.86	23.3735
20632	1	0	0.0	29	2	518.67	1.3	47.50	522.16	2388.06	8139.62	8.3803	0.03	393.0	2388.0	100.0	641.71	39.02	23.3916
20633	1	0	0.0	28	3	518.67	1.3	47.50	521.97	2388.03	8130.10	8.4441	0.03	393.0	2388.0	100.0	642.46	39.08	23.4166

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
20634	1	0	0.0	27	4	518.67	1.3	47.28	521.38	2388.05	8132.90	8.3917	0.03	391.0	2388.0	100.0	642.44	39.00	23.3737
20635	1	0	0.0	26	5	518.67	1.3	47.31	522.15	2388.03	8129.54	8.4031	0.03	390.0	2388.0	100.0	642.51	38.99	23.4130
20636	1	0	0.0	25	6	518.67	1.3	47.26	521.92	2388.08	8127.46	8.4238	0.03	392.0	2388.0	100.0	642.11	38.91	23.3467
20637	1	0	0.0	24	7	518.67	1.3	47.31	522.01	2388.06	8134.97	8.3914	0.03	391.0	2388.0	100.0	642.11	38.85	23.3952
20638	1	0	0.0	23	8	518.67	1.3	47.21	522.09	2388.06	8125.93	8.4213	0.03	393.0	2388.0	100.0	642.54	39.05	23.3224
20639	1	0	0.0	22	9	518.67	1.3	47.37	522.03	2388.05	8134.15	8.4353	0.03	391.0	2388.0	100.0	641.88	39.10	23.4521
20640	1	0	0.0	21	10	518.67	1.3	47.14	522.00	2388.06	8134.08	8.4093	0.03	391.0	2388.0	100.0	642.07	38.87	23.3820
20641	1	1	0.0	20	11	518.67	1.3	47.23	521.95	2388.06	8132.38	8.3919	0.03	391.0	2388.0	100.0	642.04	39.06	23.3609
20642	1	1	0.0	19	12	518.67	1.3	47.27	522.01	2388.06	8132.33	8.3984	0.03	391.0	2388.0	100.0	642.54	39.11	23.3845
20643	1	1	0.0	18	13	518.67	1.3	47.35	522.37	2388.03	8131.12	8.4166	0.03	392.0	2388.0	100.0	641.94	39.08	23.3677
20644	1	1	0.0	17	14	518.67	1.3	47.24	521.95	2388.06	8130.30	8.4293	0.03	392.0	2388.0	100.0	642.23	39.03	23.4572
20645	1	1	0.0	16	15	518.67	1.3	47.35	521.38	2388.00	8133.62	8.4163	0.03	392.0	2388.0	100.0	642.50	39.04	23.3672
20646	1	1	0.0	15	16	518.67	1.3	47.39	522.16	2388.10	8133.83	8.4300	0.03	390.0	2388.0	100.0	642.32	38.87	23.3484
20647	1	1	0.0	14	17	518.67	1.3	47.27	522.09	2388.02	8126.78	8.4577	0.03	391.0	2388.0	100.0	642.19	39.09	23.3409
20648	1	1	0.0	13	18	518.67	1.3	47.44	522.14	2388.06	8133.22	8.4323	0.03	391.0	2388.0	100.0	642.59	38.96	23.4481
20649	1	1	0.0	12	19	518.67	1.3	47.25	522.06	2388.01	8129.31	8.3892	0.03	391.0	2388.0	100.0	642.43	39.06	23.3809
20650	1	1	0.0	11	20	518.67	1.3	47.46	522.28	2388.05	8128.59	8.4099	0.03	392.0	2388.0	100.0	642.61	39.00	23.3325
20651	1	1	0.0	10	21	518.67	1.3	47.36	522.05	2388.11	8126.86	8.4174	0.03	392.0	2388.0	100.0	642.70	38.96	23.4025
20652	1	1	0.0	9	22	518.67	1.3	47.26	521.41	2388.04	8128.89	8.4557	0.03	392.0	2388.0	100.0	642.45	38.94	23.3770
20653	1	1	0.0	8	23	518.67	1.3	47.19	522.00	2388.06	8130.97	8.4116	0.03	393.0	2388.0	100.0	642.12	39.10	23.3186
20654	1	1	0.0	7	24	518.67	1.3	47.29	522.06	2388.12	8130.70	8.4074	0.03	393.0	2388.0	100.0	642.32	38.94	23.3971
20655	1	1	0.0	6	25	518.67	1.3	47.37	522.26	2388.08	8128.65	8.4007	0.03	393.0	2388.0	100.0	642.25	38.96	23.3785
20656	1	1	0.0	5	26	518.67	1.3	47.33	521.95	2388.07	8129.12	8.3949	0.03	391.0	2388.0	100.0	642.48	38.77	23.3557
20657	1	1	0.0	4	27	518.67	1.3	47.34	521.82	2388.02	8127.24	8.4494	0.03	392.0	2388.0	100.0	642.08	38.87	23.3931
20658	1	1	0.0	3	28	518.67	1.3	47.05	521.84	2388.07	8134.89	8.4470	0.03	392.0	2388.0	100.0	641.93	38.83	23.3502
20659	1	1	0.0	2	29	518.67	1.3	47.42	522.39	2388.07	8133.13	8.4212	0.03	392.0	2388.0	100.0	641.95	39.02	23.3621
20660	1	1	0.0	1	30	518.67	1.3	47.40	521.78	2388.10	8134.79	8.4110	0.03	391.0	2388.0	100.0	642.79	39.09	23.4069
20661	1	1	0.0	0	31	518.67	1.3	47.23	521.79	2388.06	8130.11	8.4024	0.03	393.0	2388.0	100.0	642.58	38.81	23.3552
20662	2	0	1.0	48	1	518.67	1.3	47.43	521.62	2388.14	8129.59	8.4283	0.03	392.0	2388.0	100.0	642.66	39.00	23.3923

	asset_id	coming	failure	remain_cycle	run_time	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21
20663	2	0	0.0	47	2	518.67	1.3	47.45	522.02	2388.08	8120.05	8.4414	0.03	393.0	2388.0	100.0	642.51	38.84	23.2902

In [36]:
```
training_features=train_data[['s_2','s_3','s_4','s_7','s_8','s_9','s_11','s_12','s_13','s_14','s_15','s_17','s_20','s_21',
                              #'setting_1','setting_2',
                              's_2_mean','s_2_median','s_2_max','s_2_min',  's_3_mean','s_3_median','s_3_max','s_3_min',
                              's_4_mean','s_4_median','s_4_max','s_4_min',  's_7_mean','s_7_median','s_7_max','s_7_min',
                              's_8_mean','s_8_median','s_8_max','s_8_min',  's_9_mean','s_9_median','s_9_max','s_9_min',
                              's_11_mean','s_11_median','s_11_max','s_11_min',  's_12_mean','s_12_median','s_12_max','s_12_mi
                              's_13_mean','s_13_median','s_13_max','s_13_min',  's_14_mean','s_14_median','s_14_max','s_14_mi
                              's_15_mean','s_15_median','s_15_max','s_15_min',  's_17_mean','s_17_median','s_17_max','s_17_mi
                              's_20_mean','s_20_median','s_20_max','s_20_min',  's_21_mean','s_21_median','s_21_max','s_21_mi
                              # 'setting_1_chg','setting_2_chg',
                              's_2_chg','s_3_chg','s_4_chg','s_7_chg','s_8_chg','s_9_chg','s_11_chg','s_12_chg',
                              's_13_chg','s_14_chg','s_15_chg','s_17_chg','s_20_chg','s_21_chg']]

target_feature=train_data[['coming']]
```

In [37]:
```
### training_features=train_data[['setting_1','setting_2','s_2','s_3','s_4','s_7','s_8','s_9','s_11','s_12','s_13','s_14','s_15'
###
### target_feature=train_data[['coming']]
```

In [38]:
```
### training_features.info(verbose=True)
```

In [39]:
```
X = training_features
```

In [40]:
```
X
```

Out[40]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392.0	39.06	23.4190	641.820000	641.820	641.82	64
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392.0	39.00	23.4236	641.985000	641.985	642.15	64
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390.0	38.95	23.3442	642.106667	642.150	642.35	64
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392.0	38.88	23.3739	642.167500	642.250	642.35	64
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393.0	38.90	23.4044	642.208000	642.350	642.37	64
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_
20626	643.49	1597.98	1428.63	551.43	2388.19	9065.52	48.07	519.49	2388.26	8137.60	8.4956	397.0	38.49	22.9735	643.490000	643.490	643.49	643
20627	643.54	1604.50	1433.58	550.86	2388.23	9065.11	48.04	519.68	2388.22	8136.50	8.5139	395.0	38.30	23.1594	643.540000	643.540	643.54	643
20628	643.42	1602.46	1428.18	550.94	2388.24	9065.90	48.09	520.01	2388.24	8141.05	8.5646	398.0	38.44	22.9333	643.420000	643.420	643.42	643
20629	643.23	1605.26	1426.53	550.68	2388.25	9073.72	48.39	519.67	2388.23	8139.29	8.5389	395.0	38.29	23.0640	643.230000	643.230	643.23	643
20630	643.85	1600.38	1432.14	550.79	2388.26	9061.48	48.20	519.30	2388.26	8137.33	8.5036	396.0	38.37	23.0522	643.850000	643.850	643.85	643

20631 rows × 84 columns

In [41]:
```
target_feature.dtypes
```

Out[41]:
```
coming    int64
dtype: object
```

In [42]:
```
y = target_feature

#y=target_feature.astype(int)
```

In [43]:
```
y
```

Out[43]:

	coming
0	0
1	0
2	0
3	0
4	0
...	...
20626	1
20627	1
20628	1
20629	1
20630	1

20631 rows × 1 columns

In [44]:
```
#train_data=train_data.drop(columns=['index'])

#train_data=train_data.sort_values(by=['asset_id', 'run_time'], ascending=[True, True])
#train_data.reset_index(inplace=True)
```

In [45]:
```
df_train_test=test_data.drop(columns=['run_time','setting_1','setting_2','setting_3','s_1','s_5','s_6','s_10','s_16','s_18','s_19

df_train_test=test_data[['s_2','s_3','s_4','s_7','s_8','s_9','s_11','s_12','s_13','s_14','s_15','s_17','s_20','s_21','coming']]

X_test=test_data[['s_2','s_3','s_4','s_7','s_8','s_9','s_11','s_12','s_13','s_14','s_15','s_17','s_20','s_21',

                  's_2_mean','s_2_median','s_2_max','s_2_min',  's_3_mean','s_3_median','s_3_max','s_3_min',
                  's_4_mean','s_4_median','s_4_max','s_4_min',  's_7_mean','s_7_median','s_7_max','s_7_min',
                  's_8_mean','s_8_median','s_8_max','s_8_min',  's_9_mean','s_9_median','s_9_max','s_9_min',
                  's_11_mean','s_11_median','s_11_max','s_11_min',  's_12_mean','s_12_median','s_12_max','s_12_min'
                  's_13_mean','s_13_median','s_13_max','s_13_min',  's_14_mean','s_14_median','s_14_max','s_14_min'
                  's_15_mean','s_15_median','s_15_max','s_15_min',  's_17_mean','s_17_median','s_17_max','s_17_min'
                  's_20_mean','s_20_median','s_20_max','s_20_min',  's_21_mean','s_21_median','s_21_max','s_21_min'

                  # 'setting_1_chg','setting_2_chg',
                  's_2_chg','s_3_chg','s_4_chg','s_7_chg','s_8_chg','s_9_chg','s_11_chg','s_12_chg',
                  's_13_chg','s_14_chg','s_15_chg','s_17_chg','s_20_chg','s_21_chg']]

y_test=test_data[['coming']]
```

In [46]:
```
### X_test=test_data[['setting_1','setting_2','s_2','s_3','s_4','s_7','s_8','s_9','s_11','s_12','s_13','s_14','s_15','s_17','s_2
###
### y_test=test_data[['coming']]
```

In [47]:
```
df_train_test=test_data.drop(columns=['run_time'])
```

In [48]:
```
df_train_test
```

Out[48]:

	asset_id	coming	failure	remain_cycle	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21	s_3
20631	1	0	0.0	30	518.67	1.3	47.20	521.72	2388.03	8125.55	8.4052	0.03	392.0	2388.0	100.0	643.02	38.86	23.3735	1585.29
20632	1	0	0.0	29	518.67	1.3	47.50	522.16	2388.06	8139.62	8.3803	0.03	393.0	2388.0	100.0	641.71	39.02	23.3916	1588.45

	asset_id	coming	failure	remain_cycle	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21	s_3
20633	1	0	0.0	28	518.67	1.3	47.50	521.97	2388.03	8130.10	8.4441	0.03	393.0	2388.0	100.0	642.46	39.08	23.4166	1586.94
20634	1	0	0.0	27	518.67	1.3	47.28	521.38	2388.05	8132.90	8.3917	0.03	391.0	2388.0	100.0	642.44	39.00	23.3737	1584.12
20635	1	0	0.0	26	518.67	1.3	47.31	522.15	2388.03	8129.54	8.4031	0.03	390.0	2388.0	100.0	642.51	38.99	23.4130	1587.19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
33722	100	1	0.0	4	518.67	1.3	47.69	520.69	2388.00	8213.28	8.4715	0.03	394.0	2388.0	100.0	643.24	38.65	23.1974	1599.45
33723	100	1	0.0	3	518.67	1.3	47.60	521.05	2388.09	8210.85	8.4512	0.03	395.0	2388.0	100.0	643.22	38.57	23.2771	1595.69
33724	100	1	0.0	2	518.67	1.3	47.57	521.18	2388.04	8217.24	8.4569	0.03	395.0	2388.0	100.0	643.44	38.62	23.2051	1593.15
33725	100	1	0.0	1	518.67	1.3	47.61	521.33	2388.08	8220.48	8.4711	0.03	395.0	2388.0	100.0	643.26	38.66	23.2699	1594.99
33726	100	1	0.0	0	518.67	1.3	47.80	521.07	2388.05	8214.64	8.4903	0.03	396.0	2388.0	100.0	642.95	38.70	23.1855	1601.62

13096 rows × 102 columns

In [49]:
```
##############################################################################

y_test['coming'].astype(int)
sum(y_test['coming'])
```

Out[49]: 2100

In [50]:
```
X_test
```

Out[50]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_
20631	643.02	1585.29	1398.21	553.90	2388.04	9050.17	47.20	521.72	2388.03	8125.55	8.4052	392.0	38.86	23.3735	643.438095	643.49	643.95	642
20632	641.71	1588.45	1395.42	554.85	2388.01	9054.42	47.50	522.16	2388.06	8139.62	8.3803	393.0	39.02	23.3916	643.364762	643.49	643.95	641
20633	642.46	1586.94	1401.34	554.11	2388.05	9056.96	47.50	521.97	2388.03	8130.10	8.4441	393.0	39.08	23.4166	643.314286	643.42	643.95	641
20634	642.44	1584.12	1406.42	554.07	2388.03	9045.29	47.28	521.38	2388.05	8132.90	8.3917	391.0	39.00	23.3737	643.271429	643.42	643.95	641
20635	642.51	1587.19	1401.92	554.16	2388.01	9044.55	47.31	522.15	2388.03	8129.54	8.4031	390.0	38.99	23.4130	643.252381	643.42	643.95	641
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
33722	643.24	1599.45	1415.79	553.41	2388.02	9142.37	47.69	520.69	2388.00	8213.28	8.4715	394.0	38.65	23.1974	643.240000	643.24	643.24	643
33723	643.22	1595.69	1422.05	553.22	2388.05	9140.68	47.60	521.05	2388.09	8210.85	8.4512	395.0	38.57	23.2771	643.220000	643.22	643.22	643
33724	643.44	1593.15	1406.82	553.04	2388.11	9146.81	47.57	521.18	2388.04	8217.24	8.4569	395.0	38.62	23.2051	643.440000	643.44	643.44	643

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_
33725	643.26	1594.99	1419.36	553.37	2388.07	9148.85	47.61	521.33	2388.08	8220.48	8.4711	395.0	38.66	23.2699	643.260000	643.26	643.26	643
33726	642.95	1601.62	1424.99	552.48	2388.06	9155.03	47.80	521.07	2388.05	8214.64	8.4903	396.0	38.70	23.1855	642.950000	642.95	642.95	642

13096 rows × 84 columns

In [51]:

```
X_test.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13096 entries, 20631 to 33726
Data columns (total 84 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   s_2          13096 non-null  float64
 1   s_3          13096 non-null  float64
 2   s_4          13096 non-null  float64
 3   s_7          13096 non-null  float64
 4   s_8          13096 non-null  float64
 5   s_9          13096 non-null  float64
 6   s_11         13096 non-null  float64
 7   s_12         13096 non-null  float64
 8   s_13         13096 non-null  float64
 9   s_14         13096 non-null  float64
 10  s_15         13096 non-null  float64
 11  s_17         13096 non-null  float64
 12  s_20         13096 non-null  float64
 13  s_21         13096 non-null  float64
 14  s_2_mean     13096 non-null  float64
 15  s_2_median   13096 non-null  float64
 16  s_2_max      13096 non-null  float64
 17  s_2_min      13096 non-null  float64
 18  s_3_mean     13096 non-null  float64
 19  s_3_median   13096 non-null  float64
 20  s_3_max      13096 non-null  float64
 21  s_3_min      13096 non-null  float64
 22  s_4_mean     13096 non-null  float64
 23  s_4_median   13096 non-null  float64
 24  s_4_max      13096 non-null  float64
 25  s_4_min      13096 non-null  float64
 26  s_7_mean     13096 non-null  float64
 27  s_7_median   13096 non-null  float64
 28  s_7_max      13096 non-null  float64
 29  s_7_min      13096 non-null  float64
 30  s_8_mean     13096 non-null  float64
 31  s_8_median   13096 non-null  float64
 32  s_8_max      13096 non-null  float64
 33  s_8_min      13096 non-null  float64
```

```
 34  s_9_mean     13096 non-null  float64
 35  s_9_median   13096 non-null  float64
 36  s_9_max      13096 non-null  float64
 37  s_9_min      13096 non-null  float64
 38  s_11_mean    13096 non-null  float64
 39  s_11_median  13096 non-null  float64
 40  s_11_max     13096 non-null  float64
 41  s_11_min     13096 non-null  float64
 42  s_12_mean    13096 non-null  float64
 43  s_12_median  13096 non-null  float64
 44  s_12_max     13096 non-null  float64
 45  s_12_min     13096 non-null  float64
 46  s_13_mean    13096 non-null  float64
 47  s_13_median  13096 non-null  float64
 48  s_13_max     13096 non-null  float64
 49  s_13_min     13096 non-null  float64
 50  s_14_mean    13096 non-null  float64
 51  s_14_median  13096 non-null  float64
 52  s_14_max     13096 non-null  float64
 53  s_14_min     13096 non-null  float64
 54  s_15_mean    13096 non-null  float64
 55  s_15_median  13096 non-null  float64
 56  s_15_max     13096 non-null  float64
 57  s_15_min     13096 non-null  float64
 58  s_17_mean    13096 non-null  float64
 59  s_17_median  13096 non-null  float64
 60  s_17_max     13096 non-null  float64
 61  s_17_min     13096 non-null  float64
 62  s_20_mean    13096 non-null  float64
 63  s_20_median  13096 non-null  float64
 64  s_20_max     13096 non-null  float64
 65  s_20_min     13096 non-null  float64
 66  s_21_mean    13096 non-null  float64
 67  s_21_median  13096 non-null  float64
 68  s_21_max     13096 non-null  float64
 69  s_21_min     13096 non-null  float64
 70  s_2_chg      13096 non-null  float64
 71  s_3_chg      13096 non-null  float64
 72  s_4_chg      13096 non-null  float64
 73  s_7_chg      13096 non-null  float64
 74  s_8_chg      13096 non-null  float64
 75  s_9_chg      13096 non-null  float64
 76  s_11_chg     13096 non-null  float64
 77  s_12_chg     13096 non-null  float64
 78  s_13_chg     13096 non-null  float64
 79  s_14_chg     13096 non-null  float64
 80  s_15_chg     13096 non-null  float64
 81  s_17_chg     13096 non-null  float64
 82  s_20_chg     13096 non-null  float64
 83  s_21_chg     13096 non-null  float64
dtypes: float64(84)
memory usage: 8.4 MB
```

In [52]:
```
df_train_test.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13096 entries, 20631 to 33726
Data columns (total 102 columns):
 #    Column          Dtype
---   ------          -----
 0    asset_id        int64
 1    coming          int64
 2    failure         float64
 3    remain_cycle    int64
 4    s_1             float64
 5    s_10            float64
 6    s_11            float64
 7    s_12            float64
 8    s_13            float64
 9    s_14            float64
 10   s_15            float64
 11   s_16            float64
 12   s_17            float64
 13   s_18            float64
 14   s_19            float64
 15   s_2             float64
 16   s_20            float64
 17   s_21            float64
 18   s_3             float64
 19   s_4             float64
 20   s_5             float64
 21   s_6             float64
 22   s_7             float64
 23   s_8             float64
 24   s_9             float64
 25   setting_1       float64
 26   setting_2       float64
 27   setting_3       float64
 28   setting_1_mean  float64
 29   setting_2_mean  float64
 30   s_2_mean        float64
 31   s_2_median      float64
 32   s_2_max         float64
 33   s_2_min         float64
 34   s_3_mean        float64
 35   s_3_median      float64
 36   s_3_max         float64
 37   s_3_min         float64
 38   s_4_mean        float64
 39   s_4_median      float64
 40   s_4_max         float64
 41   s_4_min         float64
 42   s_7_mean        float64
 43   s_7_median      float64
 44   s_7_max         float64
```

```
45   s_7_min          float64
46   s_8_mean         float64
47   s_8_median       float64
48   s_8_max          float64
49   s_8_min          float64
50   s_9_mean         float64
51   s_9_median       float64
52   s_9_max          float64
53   s_9_min          float64
54   s_11_mean        float64
55   s_11_median      float64
56   s_11_max         float64
57   s_11_min         float64
58   s_12_mean        float64
59   s_12_median      float64
60   s_12_max         float64
61   s_12_min         float64
62   s_13_mean        float64
63   s_13_median      float64
64   s_13_max         float64
65   s_13_min         float64
66   s_14_mean        float64
67   s_14_median      float64
68   s_14_max         float64
69   s_14_min         float64
70   s_15_mean        float64
71   s_15_median      float64
72   s_15_max         float64
73   s_15_min         float64
74   s_17_mean        float64
75   s_17_median      float64
76   s_17_max         float64
77   s_17_min         float64
78   s_20_mean        float64
79   s_20_median      float64
80   s_20_max         float64
81   s_20_min         float64
82   s_21_mean        float64
83   s_21_median      float64
84   s_21_max         float64
85   s_21_min         float64
86   setting_1_chg    float64
87   setting_2_chg    float64
88   s_2_chg          float64
89   s_3_chg          float64
90   s_4_chg          float64
91   s_7_chg          float64
92   s_8_chg          float64
93   s_9_chg          float64
94   s_11_chg         float64
95   s_12_chg         float64
96   s_13_chg         float64
```

```
 97   s_14_chg        float64
 98   s_15_chg        float64
 99   s_17_chg        float64
 100  s_20_chg        float64
 101  s_21_chg        float64
dtypes: float64(99), int64(3)
memory usage: 10.2 MB
```

In [53]:

```
pd.set_option('display.max_columns', None)
df_train_test.head(194)
```

Out[53]:

	asset_id	coming	failure	remain_cycle	s_1	s_10	s_11	s_12	s_13	s_14	s_15	s_16	s_17	s_18	s_19	s_2	s_20	s_21	s_3
20631	1	0	0.0	30	518.67	1.3	47.20	521.72	2388.03	8125.55	8.4052	0.03	392.0	2388.0	100.0	643.02	38.86	23.3735	1585.29
20632	1	0	0.0	29	518.67	1.3	47.50	522.16	2388.06	8139.62	8.3803	0.03	393.0	2388.0	100.0	641.71	39.02	23.3916	1588.45
20633	1	0	0.0	28	518.67	1.3	47.50	521.97	2388.03	8130.10	8.4441	0.03	393.0	2388.0	100.0	642.46	39.08	23.4166	1586.94
20634	1	0	0.0	27	518.67	1.3	47.28	521.38	2388.05	8132.90	8.3917	0.03	391.0	2388.0	100.0	642.44	39.00	23.3737	1584.12
20635	1	0	0.0	26	518.67	1.3	47.31	522.15	2388.03	8129.54	8.4031	0.03	390.0	2388.0	100.0	642.51	38.99	23.4130	1587.19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
20820	3	1	0.0	16	518.67	1.3	47.59	521.37	2388.16	8125.34	8.4713	0.03	394.0	2388.0	100.0	642.60	38.88	23.2921	1582.46
20821	3	1	0.0	15	518.67	1.3	47.68	520.92	2388.13	8130.51	8.4788	0.03	393.0	2388.0	100.0	643.02	38.97	23.2875	1589.99
20822	3	1	0.0	14	518.67	1.3	47.66	520.65	2388.14	8122.26	8.4778	0.03	393.0	2388.0	100.0	643.35	38.61	23.3559	1592.61
20823	3	1	0.0	13	518.67	1.3	47.58	520.64	2388.17	8130.11	8.4495	0.03	394.0	2388.0	100.0	643.22	38.90	23.2923	1595.38
20824	3	1	0.0	12	518.67	1.3	47.82	521.74	2388.14	8131.67	8.4324	0.03	393.0	2388.0	100.0	643.17	38.81	23.2768	1592.08

194 rows × 102 columns

In [54]:

```
#df_all=df_all.drop(columns=['remain_cycle','run_time','setting_3','s_1','s_5','s_6','s_10','s_16','s_18','s_19','failure'])
```

In [55]:

```
#df_train_test['asset_id']=df_train_test.asset_id.astype(float)
#df_train_test['s_17']=df_train_test.s_17.astype(float)
```

In [56]:

```
X.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20631 entries, 0 to 20630
Data columns (total 84 columns):
```

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   s_2          20631 non-null  float64
 1   s_3          20631 non-null  float64
 2   s_4          20631 non-null  float64
 3   s_7          20631 non-null  float64
 4   s_8          20631 non-null  float64
 5   s_9          20631 non-null  float64
 6   s_11         20631 non-null  float64
 7   s_12         20631 non-null  float64
 8   s_13         20631 non-null  float64
 9   s_14         20631 non-null  float64
 10  s_15         20631 non-null  float64
 11  s_17         20631 non-null  float64
 12  s_20         20631 non-null  float64
 13  s_21         20631 non-null  float64
 14  s_2_mean     20631 non-null  float64
 15  s_2_median   20631 non-null  float64
 16  s_2_max      20631 non-null  float64
 17  s_2_min      20631 non-null  float64
 18  s_3_mean     20631 non-null  float64
 19  s_3_median   20631 non-null  float64
 20  s_3_max      20631 non-null  float64
 21  s_3_min      20631 non-null  float64
 22  s_4_mean     20631 non-null  float64
 23  s_4_median   20631 non-null  float64
 24  s_4_max      20631 non-null  float64
 25  s_4_min      20631 non-null  float64
 26  s_7_mean     20631 non-null  float64
 27  s_7_median   20631 non-null  float64
 28  s_7_max      20631 non-null  float64
 29  s_7_min      20631 non-null  float64
 30  s_8_mean     20631 non-null  float64
 31  s_8_median   20631 non-null  float64
 32  s_8_max      20631 non-null  float64
 33  s_8_min      20631 non-null  float64
 34  s_9_mean     20631 non-null  float64
 35  s_9_median   20631 non-null  float64
 36  s_9_max      20631 non-null  float64
 37  s_9_min      20631 non-null  float64
 38  s_11_mean    20631 non-null  float64
 39  s_11_median  20631 non-null  float64
 40  s_11_max     20631 non-null  float64
 41  s_11_min     20631 non-null  float64
 42  s_12_mean    20631 non-null  float64
 43  s_12_median  20631 non-null  float64
 44  s_12_max     20631 non-null  float64
 45  s_12_min     20631 non-null  float64
 46  s_13_mean    20631 non-null  float64
 47  s_13_median  20631 non-null  float64
 48  s_13_max     20631 non-null  float64
 49  s_13_min     20631 non-null  float64
```

```
 50  s_14_mean    20631 non-null  float64
 51  s_14_median  20631 non-null  float64
 52  s_14_max     20631 non-null  float64
 53  s_14_min     20631 non-null  float64
 54  s_15_mean    20631 non-null  float64
 55  s_15_median  20631 non-null  float64
 56  s_15_max     20631 non-null  float64
 57  s_15_min     20631 non-null  float64
 58  s_17_mean    20631 non-null  float64
 59  s_17_median  20631 non-null  float64
 60  s_17_max     20631 non-null  float64
 61  s_17_min     20631 non-null  float64
 62  s_20_mean    20631 non-null  float64
 63  s_20_median  20631 non-null  float64
 64  s_20_max     20631 non-null  float64
 65  s_20_min     20631 non-null  float64
 66  s_21_mean    20631 non-null  float64
 67  s_21_median  20631 non-null  float64
 68  s_21_max     20631 non-null  float64
 69  s_21_min     20631 non-null  float64
 70  s_2_chg      20631 non-null  float64
 71  s_3_chg      20631 non-null  float64
 72  s_4_chg      20631 non-null  float64
 73  s_7_chg      20631 non-null  float64
 74  s_8_chg      20631 non-null  float64
 75  s_9_chg      20631 non-null  float64
 76  s_11_chg     20631 non-null  float64
 77  s_12_chg     20631 non-null  float64
 78  s_13_chg     20631 non-null  float64
 79  s_14_chg     20631 non-null  float64
 80  s_15_chg     20631 non-null  float64
 81  s_17_chg     20631 non-null  float64
 82  s_20_chg     20631 non-null  float64
 83  s_21_chg     20631 non-null  float64
dtypes: float64(84)
memory usage: 13.2 MB
```

In [57]:

```
sm = SMOTE()
#smx = SMOTENC(random_state=12,  categorical_features=[0, 1, 2, 3])

X_res, y_res = sm.fit_resample(X,y) #.values.ravel()
```

In [58]:

```
X_res.head(194)
```

Out[58]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392.0	39.06	23.4190	641.820000	641.820	641.82	641.8
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392.0	39.00	23.4236	641.985000	641.985	642.15	641.8

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390.0	38.95	23.3442	642.106667	642.150	642.35	641.8
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392.0	38.88	23.3739	642.167500	642.250	642.35	641.8
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393.0	38.90	23.4044	642.208000	642.350	642.37	641.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	643.64	1599.22	1425.95	551.29	2388.29	9040.58	48.33	520.04	2388.35	8112.58	8.5223	398.0	38.49	23.0675	643.640000	643.640	643.64	643.6
190	643.34	1602.36	1425.77	550.92	2388.28	9042.76	48.15	519.57	2388.30	8114.61	8.5174	394.0	38.45	23.1295	643.340000	643.340	643.34	643.3
191	643.54	1601.41	1427.20	551.25	2388.32	9033.22	48.25	520.08	2388.32	8110.93	8.5113	396.0	38.48	22.9649	643.540000	643.540	643.54	643.5
192	641.89	1583.84	1391.28	554.53	2388.01	9054.72	46.93	522.33	2388.06	8137.72	8.3905	391.0	38.94	23.4585	643.410476	643.510	644.21	641.8
193	641.82	1587.05	1393.13	554.77	2387.98	9051.31	47.24	522.70	2387.98	8131.09	8.4167	392.0	39.06	23.4085	643.355714	643.510	644.21	641.8

194 rows × 84 columns

In [59]:
```
y_res.head()
```

Out[59]:

	coming
0	0
1	0
2	0
3	0
4	0

In [60]:
```
# merge the dependent and independent variables post SMOTE into dataframe
# df_balanced=pd.concat([df_y,df_X],axis=1)

df_balanced=pd.concat([X_res,y_res],axis=1)

df_balanced.head(194)
```

Out[60]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392.0	39.06	23.4190	641.820000	641.820	641.82	641.8
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392.0	39.00	23.4236	641.985000	641.985	642.15	641.8

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390.0	38.95	23.3442	642.106667	642.150	642.35	641.8
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392.0	38.88	23.3739	642.167500	642.250	642.35	641.8
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393.0	38.90	23.4044	642.208000	642.350	642.37	641.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	643.64	1599.22	1425.95	551.29	2388.29	9040.58	48.33	520.04	2388.35	8112.58	8.5223	398.0	38.49	23.0675	643.640000	643.640	643.64	643.6
190	643.34	1602.36	1425.77	550.92	2388.28	9042.76	48.15	519.57	2388.30	8114.61	8.5174	394.0	38.45	23.1295	643.340000	643.340	643.34	643.3
191	643.54	1601.41	1427.20	551.25	2388.32	9033.22	48.25	520.08	2388.32	8110.93	8.5113	396.0	38.48	22.9649	643.540000	643.540	643.54	643.5
192	641.89	1583.84	1391.28	554.53	2388.01	9054.72	46.93	522.33	2388.06	8137.72	8.3905	391.0	38.94	23.4585	643.410476	643.510	644.21	641.8
193	641.82	1587.05	1393.13	554.77	2387.98	9051.31	47.24	522.70	2387.98	8131.09	8.4167	392.0	39.06	23.4085	643.355714	643.510	644.21	641.8

194 rows × 85 columns

In [61]:
```
#df_balanced=df_balanced.drop(columns=['run_time','setting_3','s_1','s_5','s_6','s_10','s_16','s_18','s_19'])
```

In [62]:
```
###############################################################################
features = [f for f in X.columns if f not in ['coming']]#,'setting_1','setting_2','setting_1_chg','setting_2_chg']]

dependent=pd.DataFrame(df_balanced['coming'])
independent=df_balanced.drop(columns=['coming'])#

#make sure everything is numeric for simplicity
independent = independent.apply(pd.to_numeric)
df_balanced = df_balanced.apply(pd.to_numeric)
```

In [63]:
```
independent.head(194)
```

Out[63]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392.0	39.06	23.4190	641.820000	641.820	641.82	641.8
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392.0	39.00	23.4236	641.985000	641.985	642.15	641.8
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390.0	38.95	23.3442	642.106667	642.150	642.35	641.8
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392.0	38.88	23.3739	642.167500	642.250	642.35	641.8
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393.0	38.90	23.4044	642.208000	642.350	642.37	641.8

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	643.64	1599.22	1425.95	551.29	2388.29	9040.58	48.33	520.04	2388.35	8112.58	8.5223	398.0	38.49	23.0675	643.640000	643.640	643.64	643.6
190	643.34	1602.36	1425.77	550.92	2388.28	9042.76	48.15	519.57	2388.30	8114.61	8.5174	394.0	38.45	23.1295	643.340000	643.340	643.34	643.3
191	643.54	1601.41	1427.20	551.25	2388.32	9033.22	48.25	520.08	2388.32	8110.93	8.5113	396.0	38.48	22.9649	643.540000	643.540	643.54	643.5
192	641.89	1583.84	1391.28	554.53	2388.01	9054.72	46.93	522.33	2388.06	8137.72	8.3905	391.0	38.94	23.4585	643.410476	643.510	644.21	641.8
193	641.82	1587.05	1393.13	554.77	2387.98	9051.31	47.24	522.70	2387.98	8131.09	8.4167	392.0	39.06	23.4085	643.355714	643.510	644.21	641.8

194 rows × 84 columns

In [64]:
```
independent
```

Out[64]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	
0	641.820000	1589.700000	1400.600000	554.360000	2388.060000	9046.190000	47.470000	521.660000	2388.020000	8138.620000	8.419500	392.000000	39.06
1	642.150000	1591.820000	1403.140000	553.750000	2388.040000	9044.070000	47.490000	522.280000	2388.070000	8131.490000	8.431800	392.000000	39.00
2	642.350000	1587.990000	1404.200000	554.260000	2388.080000	9052.940000	47.270000	522.420000	2388.030000	8133.230000	8.417800	390.000000	38.95
3	642.350000	1582.790000	1401.870000	554.450000	2388.110000	9049.480000	47.130000	522.860000	2388.080000	8133.830000	8.368200	392.000000	38.88
4	642.370000	1582.850000	1406.220000	554.000000	2388.060000	9055.150000	47.280000	522.190000	2388.040000	8133.800000	8.429400	393.000000	38.90
...	...	...	...	...	...	...	...	...	...	...	...	...	
37057	643.631782	1597.400559	1417.823181	551.742860	2388.228083	9051.759451	47.969316	520.069441	2388.240549	8121.570664	8.514552	395.123315	38.62
37058	643.072583	1605.743251	1417.396333	551.712667	2388.205167	9061.754499	47.916833	520.398834	2388.235167	8137.419581	8.494600	395.241666	38.52
37059	643.285342	1599.342374	1419.741371	552.185342	2388.137159	9101.849103	47.934204	520.569091	2388.152841	8173.210913	8.471016	395.000000	38.53
37060	643.043355	1596.750187	1419.549142	551.419329	2388.241463	9041.994016	47.809758	519.751100	2388.253597	8116.981958	8.471300	393.640264	38.55
37061	643.233071	1599.710023	1421.089379	552.140164	2388.148921	9133.713435	48.070914	520.637386	2388.147386	8201.970856	8.491049	395.630715	38.54

37062 rows × 84 columns

In [65]:
```
# dependent.head(194)
```

In [66]:
```
# dependent
```

In [67]:
```
training_features.head(194)
```

Out[67]:

	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21	s_2_mean	s_2_median	s_2_max	s_2_mi
0	641.82	1589.70	1400.60	554.36	2388.06	9046.19	47.47	521.66	2388.02	8138.62	8.4195	392.0	39.06	23.4190	641.820000	641.820	641.82	641.8
1	642.15	1591.82	1403.14	553.75	2388.04	9044.07	47.49	522.28	2388.07	8131.49	8.4318	392.0	39.00	23.4236	641.985000	641.985	642.15	641.8
2	642.35	1587.99	1404.20	554.26	2388.08	9052.94	47.27	522.42	2388.03	8133.23	8.4178	390.0	38.95	23.3442	642.106667	642.150	642.35	641.8
3	642.35	1582.79	1401.87	554.45	2388.11	9049.48	47.13	522.86	2388.08	8133.83	8.3682	392.0	38.88	23.3739	642.167500	642.250	642.35	641.8
4	642.37	1582.85	1406.22	554.00	2388.06	9055.15	47.28	522.19	2388.04	8133.80	8.4294	393.0	38.90	23.4044	642.208000	642.350	642.37	641.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
189	643.64	1599.22	1425.95	551.29	2388.29	9040.58	48.33	520.04	2388.35	8112.58	8.5223	398.0	38.49	23.0675	643.640000	643.640	643.64	643.6
190	643.34	1602.36	1425.77	550.92	2388.28	9042.76	48.15	519.57	2388.30	8114.61	8.5174	394.0	38.45	23.1295	643.340000	643.340	643.34	643.3
191	643.54	1601.41	1427.20	551.25	2388.32	9033.22	48.25	520.08	2388.32	8110.93	8.5113	396.0	38.48	22.9649	643.540000	643.540	643.54	643.5
192	641.89	1583.84	1391.28	554.53	2388.01	9054.72	46.93	522.33	2388.06	8137.72	8.3905	391.0	38.94	23.4585	643.410476	643.510	644.21	641.8
193	641.82	1587.05	1393.13	554.77	2387.98	9051.31	47.24	522.70	2387.98	8131.09	8.4167	392.0	39.06	23.4085	643.355714	643.510	644.21	641.8

194 rows × 84 columns

In [68]:
```
# features
```

In [69]:
```
from sklearn.model_selection import cross_validate
```

In [70]:
```
import matplotlib.pylab as plt
%matplotlib inline

def evaluate_model(alg, train, target, predictors,    cv_folds=3,    early_stopping_rounds=50):  # 50

    xgb_param = alg.get_xgb_params()
    xgtrain = xgb.DMatrix(train[predictors].values, target['coming'].values)
    cvresult = xgb.cv(xgb_param, xgtrain, num_boost_round=alg.get_params()['n_estimators'], nfold=cv_folds,
        metrics='auc', early_stopping_rounds=early_stopping_rounds, verbose_eval=True)
    alg.set_params(n_estimators=cvresult.shape[0])

    #Fit the algorithm on the data
    alg.fit(train[predictors], target['coming'], eval_metric='auc')
```

```
    #Predict training set:
    dtrain_predictions = alg.predict(train[predictors])
    dtrain_predprob = alg.predict_proba(train[predictors])[:,1]

    feat_imp = pd.Series(alg.get_booster().get_fscore()).sort_values(ascending=False)
    feat_imp.plot(kind='bar', title='Feature Importance', color='g')
    plt.ylabel('Feature Importance Score')

    #Print model report:
    print("\nModel Report")
    print("Accuracy : %.4g" % metrics.accuracy_score(target['coming'].values, dtrain_predictions))
    print("AUC Score (Balanced): %f" % metrics.roc_auc_score(target['coming'], dtrain_predprob))
```

In [71]:

```
import matplotlib.pylab as plt
%matplotlib inline

def model_fit(alg, train, target, predictors,    early_stopping_rounds=100):

    #Fit the algorithm on the data
    alg.fit(train[predictors], target['coming'], eval_metric='auc')

    #Predict training set:
    dtrain_predictions = alg.predict(train[predictors])
    dtrain_predprob = alg.predict_proba(train[predictors])[:,1]

    feat_imp = pd.Series(alg.get_booster().get_fscore()).sort_values(ascending=False)
    feat_imp.plot(kind='bar', title='Feature Importance', color='g')
    plt.ylabel('Feature Importance Score')

    #Print model report:
    print("\nModel Report")
    print("Accuracy : %.4g" % metrics.accuracy_score(target['coming'].values, dtrain_predictions))
    print("AUC Score (Balanced): %f" % metrics.roc_auc_score(target['coming'], dtrain_predprob))
```

In [72]:

```
xgb1 = XGBClassifier(
 learning_rate =0.05,
 n_estimators=200,
 max_depth=20,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',

 use_label_encoder=False,
```

```
 verbosity= 0 ,

 scale_pos_weight=1,
 seed=27)
```

In [73]:

```
xgb00 = XGBClassifier(
 objective= 'binary:logistic',
 learning_rate = 0.1,
 n_estimators = 20,

 use_label_encoder=False,
 verbosity= 0 )
```

In [74]:

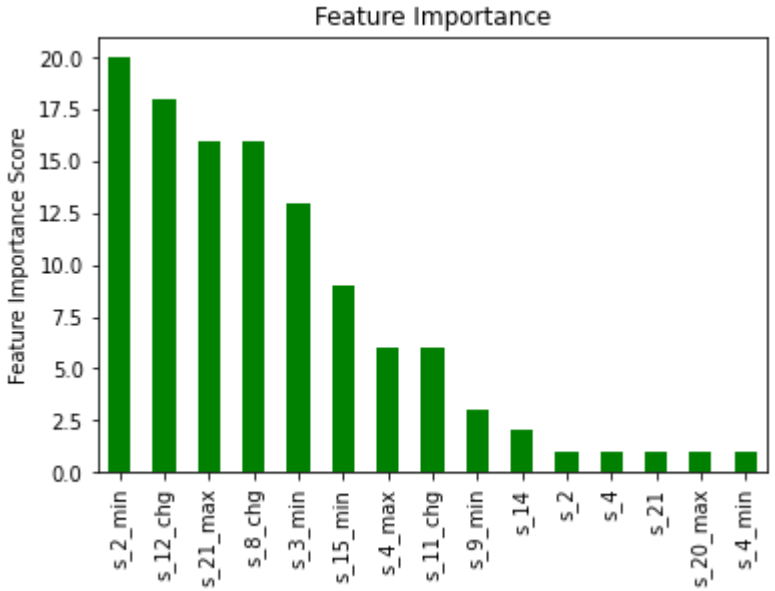
```
from sklearn.model_selection import GridSearchCV

### param_test1 = {
###  'max_depth':range(10,15,2),
###  'min_child_weight':range(1,4,2)
### }
### gsearch1 = GridSearchCV(estimator = XGBClassifier(learning_rate=0.05, n_estimators=200, max_depth=16, min_child_weight=1, ga
###                                     subsample=0.8, colsample_bytree=0.8, objective='binary:logistic', scale_pos_weight=1, se
###                           param_grid = param_test1, scoring = 'roc_auc', n_jobs = 4, cv = 5)       ###### iid=False,
###
### gsearch1.fit(independent[features],dependent)
### #gsearch1.cv_results_, gsearch1.best_params_, gsearch1.best_score_
###
### #gsearch1.grid_scores_,
### gsearch1.best_params_, gsearch1.best_score_
###
### #gsearch1.cv_results_['params'][gsearch1.best_index_]
```

In [75]:

```
evaluate_model(xgb00, independent, dependent, features)
```

```
[0]     train-auc:0.99999+0.00002       test-auc:0.99984+0.00007
[1]     train-auc:1.00000+0.00000       test-auc:0.99986+0.00010
[2]     train-auc:1.00000+0.00000       test-auc:0.99986+0.00010
[3]     train-auc:1.00000+0.00000       test-auc:0.99986+0.00010
[4]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[5]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[6]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[7]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[8]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[9]     train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[10]    train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[11]    train-auc:1.00000+0.00000       test-auc:0.99992+0.00012
[12]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
```

```
[13]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[14]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[15]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[16]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[17]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[18]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000
[19]    train-auc:1.00000+0.00000       test-auc:1.00000+0.00000

Model Report
Accuracy : 1
AUC Score (Balanced): 1.000000
```



In [76]:

```
#   from sklearn.model_selection import GridSearchCV
#
#   # updating our default model with the optimal number of estimators
#   xgb01 = XGBClassifier(
#    objective = 'binary:logistic',
#    learning_rate =0.1,
#    n_estimators=20,
#
#    use_label_encoder=False,
#    verbosity= 0)
#
#   # array of values for max_depth and min_child_weight parameters
#   param_test1 = {'max_depth':range(3,7,1),'min_child_weight':range(1,5,1)}
#
#   # grid search with cross-validation using the updated model and parameter value array
#   gsearch1 = GridSearchCV(estimator = xgb1, param_grid = param_test1, scoring='roc_auc', cv=5)    ## iid=False,
#   gsearch1.fit(independent[features],dependent['coming'])
#   gsearch1.cv_results_['params'], gsearch1.best_params_, gsearch1.best_score_
```

```
#
#
# Here Grid Search works badly, my guessing is that the train data and test data was differed so widely
# and if we don't use self created features, the model gives only 70% acc,
```

In [77]:

```
#   model_fit(xgb01, independent, dependent, features)
```

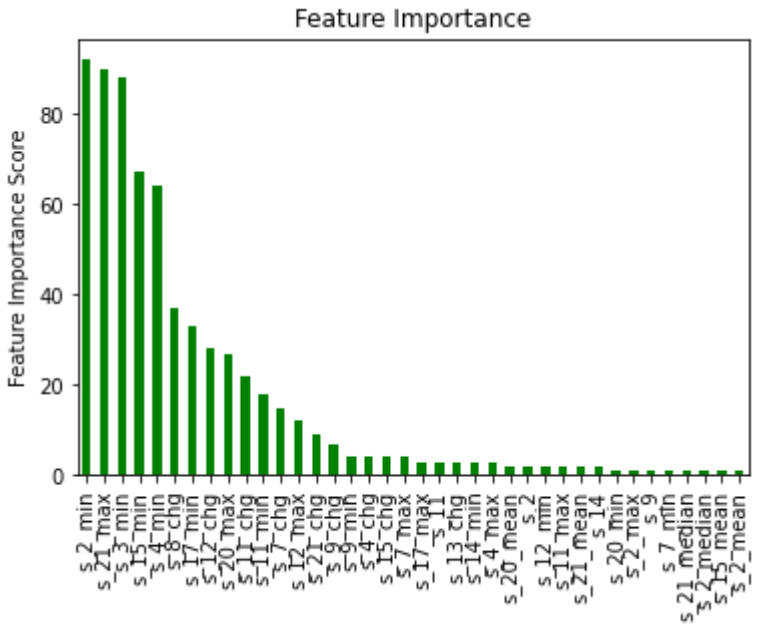
In [78]:

```
##############################################################################################################

xgb_final1 = XGBClassifier(
 learning_rate =0.05,
 n_estimators=200,  # 500 ?? TRY try
 max_depth=20,
 min_child_weight=1,
 gamma=0,
 subsample=0.8,
 colsample_bytree=0.8,
 objective= 'binary:logistic',
 #nthread=4,

 use_label_encoder=False,
 verbosity= 0 ,

 scale_pos_weight=1,
 seed=27)
#modelfit(xgb1, train, predictors)
```

In [79]:

```
model_fit(xgb_final1, independent, dependent, features)
```

```
Model Report
Accuracy : 1
AUC Score (Balanced): 1.000000
```



In [80]:
```
test_features = [x for x in X_test.columns if x not in ['coming']] #id

#Predict test set:
test_predictions = xgb_final1.predict(X_test[test_features])
test_predprob = xgb_final1.predict_proba(X_test[test_features])[:,1]

#Print model report:
print("Accuracy : %.4g" % metrics.accuracy_score(y_test['coming'].values, test_predictions))
print("AUC Score (Test): %f" % metrics.roc_auc_score(y_test['coming'].values, test_predprob))
```

```
Accuracy : 0.8988
AUC Score (Test): 0.932025
```

In [81]:
```
# df_testing['P_FAIL'].head(30)
```

In [82]:
```
# test_features
```

In [83]:
```
# X_test
```

In [84]:
```
# y_test
```

In [85]:

```
##########################################################################################

df_testing=df_train_test.copy()
```

In [86]:
```
# df_testing['P_FAIL'].head(30)
```

In [87]:
```
df_testing['P_FAIL'] = xgb_final1.predict_proba(df_testing[features])[:,1];
df_testing['Y_FAIL'] = np.where(((df_testing.P_FAIL < .000075)), 0, 1)
#Print model report:
print("Accuracy : %.4g" % metrics.accuracy_score(df_testing['coming'].values, df_testing['Y_FAIL']))
print("AUC Score (Test): %f" % metrics.roc_auc_score(df_testing['coming'], df_testing['P_FAIL']))
```

```
Accuracy : 0.9678
AUC Score (Test): 0.932025
```

In [88]:
```
df_testing['P_FAIL'].head(30)
```

Out[88]:
```
20631    0.001771
20632    0.000069
20633    0.000067
20634    0.000067
20635    0.000067
20636    0.000067
20637    0.000067
20638    0.000067
20639    0.000067
20640    0.000067
20641    0.000067
20642    0.000067
20643    0.000069
20644    0.000067
20645    0.000067
20646    0.000067
20647    0.001804
20648    0.000067
20649    0.000067
20650    0.000067
20651    0.000076
20652    0.000270
20653    0.000067
20654    0.013169
20655    0.000067
20656    0.000067
20657    0.000074
20658    0.013635
20659    0.000067
```

```
20660    0.000158
Name: P_FAIL, dtype: float32
```

In [89]:
```
Y_FAIL = df_testing['Y_FAIL']
```

In [90]:
```
# X_test

print(pd.crosstab(df_testing.Y_FAIL, df_testing.coming, dropna=False))
```

```
coming      0     1
Y_FAIL
0       10956   382
1          40  1718
```

In [91]:
```
df_testing['P_FAIL'].head(30)
```

Out[91]:
```
20631    0.001771
20632    0.000069
20633    0.000067
20634    0.000067
20635    0.000067
20636    0.000067
20637    0.000067
20638    0.000067
20639    0.000067
20640    0.000067
20641    0.000067
20642    0.000067
20643    0.000069
20644    0.000067
20645    0.000067
20646    0.000067
20647    0.001804
20648    0.000067
20649    0.000067
20650    0.000067
20651    0.000076
20652    0.000270
20653    0.000067
20654    0.013169
20655    0.000067
20656    0.000067
20657    0.000074
20658    0.013635
20659    0.000067
20660    0.000158
Name: P_FAIL, dtype: float32
```

In [92]:
```
df_testing['Y_FAIL']
```

Out[92]:
```
20631    1
20632    0
20633    0
20634    0
20635    0
        ..
33722    1
33723    1
33724    1
33725    1
33726    1
Name: Y_FAIL, Length: 13096, dtype: int64
```

In [93]:
```
sum(df_testing['Y_FAIL'])
```

Out[93]:
```
1758
```

In [94]:
```
####
output=pd.DataFrame({'asset_id':test_data.asset_id,
                     #'run_time':test_data.run_time,
                     #'P_FAIL':df_testing['P_FAIL'],
                     #'real':df_testing['coming'],
                     'coming':df_testing['Y_FAIL']})

#output['coming']=output['coming']#.astype(int)
output.to_csv('PaperCO0033.csv',index=False)
```

In [ ]:

In [ ]: