In [33]:
```
import pickle
import pandas as pd
import numpy as np
from sklearn.metrics import f1_score
import time
```

In [34]:
```
file = "median_values.pkl"
with open(file, 'rb') as file:
    median_values= pickle.load(file)

file = "truncated_SVD.pkl"
with open(file, 'rb') as file:
    T_SVD= pickle.load(file)

file = "mnmx_scaler.pkl"
with open(file, 'rb') as file:
    mnmx_scaler= pickle.load(file)

file = "clf_voting_classifier.pkl"
with open(file, 'rb') as file:
    clf= pickle.load(file)
```

In [35]:
```
df=pd.read_csv('equip_failures_training_set.csv')
df.head()
```

Out[35]:

	id	target	sensor1_measure	sensor2_measure	sensor3_measure	sensor4_measure	sensor5_measure	sensor6_measure	sensor7_histogram_bin0	sensor7_histogram_bin1	...	sensor105_histogram_bin2	sensor105_histogram_bin3	sensor105_histogram_bin4	sensor105_histogram_bin5	sensor105_histogram_bin6	sensor105_histogram_bin7	sensor105_histogram_bin8	sensor105_histogram_bin9	sensor106_measure	sensor107_meas
0	1	0	76698	na	2130706438	280	0	0	0	0	...	1240520	493384	721044	469792	339156	157956	73224	0	0	
1	2	0	33058	na	0	na	0	0	0	0	...	421400	178064	293306	245416	133654	81140	97576	1500	0	
2	3	0	41040	na	228	100	0	0	0	0	...	277378	159812	423992	409564	320746	158022	95128	514	0	
3	4	0	12	0	70	66	0	10	0	0	...	240	46	58	44	10	0	0	0	4	
4	5	0	60874	na	1368	458	0	0	0	0	...	622012	229790	405298	347188	286954	311560	433954	1218	0	

5 rows × 172 columns

In [36]:
```
def final_fun_1(X):
    zero_columns = ['sensor7_histogram_bin0','sensor7_histogram_bin1','sensor19_measure','sensor21_measure','sensor24_histogram_bin0','sensor24_histogram_bin1','sensor24_histogram_bin2','sensor24_histogram_bin3','sensor24_histogram_bin4','sensor24_histogram_bin9','sensor25_histogram_bin9','sensor64_histogram_bin0','sensor106_measure']
    null_columns = ['sensor43_measure', 'sensor42_measure', 'sensor41_measure', 'sensor40_measure', 'sensor68_measure', 'sensor2_measure', 'sensor39_measure', 'sensor38_measure']
    correlated_columns = ['sensor45_measure', 'sensor32_measure', 'sensor46_measure', 'sensor47_measure', 'sensor67_measure', 'sensor64_histogram_bin5']
    column_names = dict()

    drop_columns = zero_columns + null_columns + correlated_columns

    # Converting string nan to Numpy NaN
    for col in X.columns:
        X[col] = pd.to_numeric(X[col], errors='coerce')

    # Dropping null, zeros and correlated columns
    X = X.drop(drop_columns, axis=1)

    # Imputing with trained median values
    X = X.fillna(median_values)

    # Adding 4 Truncated SVD features
    X_SVD = T_SVD.transform(X)

    for i in range(4):
        X["SVD_"+str(i)] = X_SVD[:,i]

    # Obtaining the bin columns
    for column in X.columns:
        value = column.split("_")
        if value[1] == 'histogram':
            if value[0] not in column_names:
                column_names[value[0]] = []
            column_names[value[0]].append(column)

    # Average of each sensor bin as a new feature
    for sensor in column_names.keys():
        bins = column_names[sensor]
        temp_df = X[bins]
        X[sensor+'_bin_average'] = temp_df.mean(axis=1)

    columns = X.columns

    # Normalizing data using the trained min max scaler
    X = pd.DataFrame(mnmx_scaler.transform(X))
    X.columns = columns

    # Predicting the output
    output = clf.predict(X)

    return output
```

In [37]:
```
def final_fun_2(X,Y):
    zero_columns = ['sensor7_histogram_bin0','sensor7_histogram_bin1','sensor19_measure','sensor21_measure','sensor24_histogram_bin0','sensor24_histogram_bin1','sensor24_histogram_bin2','sensor24_histogram_bin3','sensor24_histogram_bin4','sensor24_histogram_bin9','sensor25_histogram_bin9','sensor64_histogram_bin0','sensor106_measure']
    null_columns = ['sensor43_measure', 'sensor42_measure', 'sensor41_measure', 'sensor40_measure', 'sensor68_measure', 'sensor2_measure', 'sensor39_measure', 'sensor38_measure']
    correlated_columns = ['sensor45_measure', 'sensor32_measure', 'sensor46_measure', 'sensor47_measure', 'sensor67_measure', 'sensor64_histogram_bin5']
    column_names = dict()

    drop_columns = zero_columns + null_columns + correlated_columns

    # Converting string nan to Numpy NaN
    for col in X.columns:
        X[col] = pd.to_numeric(X[col], errors='coerce')

    # Dropping null, zeros and correlated columns
    X = X.drop(drop_columns, axis=1)

    # Imputing with trained median values
    X = X.fillna(median_values)

    # Adding 4 Truncated SVD features
    X_SVD = T_SVD.transform(X)

    for i in range(4):
        X["SVD_"+str(i)] = X_SVD[:,i]

    # Obtaining the bin columns
    for column in X.columns:
        value = column.split("_")
        if value[1] == 'histogram':
            if value[0] not in column_names:
                column_names[value[0]] = []
            column_names[value[0]].append(column)

    # Average of each sensor bin as a new feature
    for sensor in column_names.keys():
        bins = column_names[sensor]
        temp_df = X[bins]
        X[sensor+'_bin_average'] = temp_df.mean(axis=1)

    columns = X.columns

    # Normalizing data using the trained min max scaler
    X = pd.DataFrame(mnmx_scaler.transform(X))
    X.columns = columns

    # Predicting the output
    output = clf.predict(X)

    # Computing the F1 score
    score_f1 = f1_score(Y, output, average="macro")

    return score_f1
```

Predicting for top 5 rows.

In [38]:
```
temp_data = df.head()
target = temp_data['target']
temp_data = temp_data.drop(['target', 'id'], axis=1)
```

In [39]:
```
start_time = time.time()
output = final_fun_1(temp_data)
print("The output value is:", output)
print("Function 1 has taken %s seconds to execute" % (time.time() - start_time))
```

```
The output value is: [0 0 0 0 0]
Function 1 has taken 0.2573113441467285 seconds to execute
```

In [40]:
```
start_time = time.time()
output = final_fun_2(temp_data, target)
print("The output metric is:", output)
print("Function 2 has taken %s seconds to execute" % (time.time() - start_time))
```

```
The output metric is: 1.0
Function 2 has taken 0.2214052677154541 seconds to execute
```