Equipment Failure Prediction.

Overview

This case study is regarding Conocophillips , multinational energy firm , that funds multiple energy projects in the US. According to them 80% of oil wells in the US are Stripper Wells(oil or gas well that is nearing the end of its economically useful life). These wells produce less volume but at an aggregate level are responsible for significant amount of oil production.

They have low operational costs and low capital intensity - ultimately providing a source of steady cash flow to fund operations that require more funds to get off the ground. Meaning less investment and relatively better outcomes.

The company requires these low cost wells to remain well maintained so that the cash flow remains steady .

But even mechanical and electronic equipment in any field have their shelf life and break down with time. It takes a lot investment of money and resources to get the repairs/replacement done and results in lost oil production .

The aim is to predict this equipment failure depending upon the data given from the sensors so that teams are pre prepared to handle failures as they occur.

Business Problem in machine learning terms:

Given the data points with their respective features, use classification to find out whether the data points belong to surface failure or downhole failure.

Metric to be used:

The formula for the standard FBeta-score is the Harmonic Mean
(
Harmonic Mean of a and b = (((a^-1) + (b^-1)) / (number of elements that is 2))^-1
)
of the precision and recall. A perfect model has an F-score of 1.

Precision =TP/(TP+FP)
Recall =TP/(TP+FN)

Formula :
Fbeta = ((1 + beta^2) * Precision * Recall) / ((beta^2 * Precision) + Recall)

What is the difference between f.5 and f2

a) f.5 = (1+.5^2) * ((precision * recall) / (.5^2 * precision)+recall)):
Means here the weight of the precision is cut to a quarter of original value. Meaning that if we divide the numerator by this lessened denominator (.25 * precision) then we get more overall value than the value we get if we divide the numerator by 1 * Recall. We get more value for the f.5 score by dividing by precision that means **more weightage given to the precision (the component containing the false positive) here.**

The only difference here is between precision and recall is that of the false positive and false negative respectively.

b) f2 = (1+2^2) * ((precision * recall) / (2^2 * precision)+recall)):
Means here the weight of the precision is more than quadrupled.

Meaning that if we divide the numerator by this increased denominator (4 * precision) then we get less overall value than the value we get if we divide the numerator by 1 * Recall. We get more value for f2 score by dividing by recall that means **more weightage given to the recall (the component containing the false negative) here.**

So if I consider downhole failures as my positive class then I do not want that I should mistake a downhole failure for a surface failure. Meaning that here, I should not have any False Negative. Meaning I should not mistake downhole for a surface failure, so I want to reduce false negative, I will consider f2 score respectively.

In this case my priority is the prevention of downhole failures.
(They are more expensive, more impactful on failure, hazardous and less accessible for repair when they occur,difficult to handle and very less in numbers).
This means I should not confuse a downhole failure for a surface failure.
Accordingly, as the data is imbalanced , I will use the f2 to give the downhole failure , my priority.

In [1]:

```
import flask
from flask import Flask, jsonify, request, render_template
import numpy as np
import pandas as pd
#import pickle5 as pickle
import pickle
import csv
import json
from flask import send_file
from sklearn.metrics import fbeta_score
import datetime
import time
```

In [2]:

```
app = Flask(__name__,template_folder='templates')
app.debug=True
```

In [3]:

```
@app.route('/',methods=['GET','POST'])
def first_page():
    return flask.render_template('first_page.html')
```

In [4]:

```
@app.route('/final_function',methods=['GET','POST'])
def final(test_original=None,y_original=None):

    """
    This function takes in either test data or both the test data and it's original
    If only test data is given, then it predicts the y values.
    If both the test data and y_values are given then it returns the predicted valu
    metric by which we measure the model. In this case the metric is F2 Score.
    """

    #file = request.files['test_original']

    #file_name=request.files['test_original'].filename

    #if not file:
        #return flask.render_template('first_page.html',csv_only_message="Enter the

    #file_extension_correct=check_extension(file_name)

    #if not file_extension_correct:
        #message = " You did not upload a csv file !!!! Please upload a csv file on
        #return flask.render_template('first_page.html',csv_only_message=message)

    #test_original = pd.read_csv(file)

    #file = request.files['y_original']
    #file_name=request.files['y_original'].filename

    #if file:

        #file_extension_correct=check_extension(file_name)

        #if not file_extension_correct:
            #message = " You did not upload a csv file !!!! Please upload a csv fil
            #return flask.render_template('first_page.html',csv_only_message=messag

        #y_original = pd.read_csv(file)


    #else :
        #y_original=None

    #Remove the id if present
    try:
        test_original.drop("id",axis=1,inplace=True)
    except KeyError:
        pass


    # ignoring SettingWithCopyWarning
    pd.options.mode.chained_assignment = None

    #Replace the string that is "na" in the features by np.NaN
    train_min=pd.read_pickle("train_min.pickle")
    train_max=pd.read_pickle("train_max.pickle")

    #train_min=pickle.load(open("train_min.pickle","rb"))
    #print("type(train_min) : ",type(train_min))
```

```
    #train_max=pickle.load(open("train_max.pickle","rb"))
    #print("type(train_max) : ",type(train_max))


    # starting time
    start = time.time()

    #Start with report generation.
    list_of_issues=[]
    for column in test_original.columns.values:
        for ind,ele in zip(test_original[column].index.values,test_original[column]
            Report_Gen(
            num=test_original[column][ind],
            minimum=train_min[column],
            maximum=train_max[column],
            column=column,
            index=ind,
            l_o_i=list_of_issues)

    end = time.time()

    #total time taken
    print(f"Time Taken by report generation :  {round(end - start, 2)}")
    print(f"list length  {len(list_of_issues)}")
    print(f"list   {list_of_issues[0:5]}")


    #Preprocessing
    pattern='|'.join(["(?i)^np.nan$", "(?i)^nan$","(?i)^na$","(?i)^NANANNA$"])

    for column in list(test_original.columns.values):
        test_original[column]=test_original[column].apply(remove_whitespaces).repla

    #sort index
    test_original.sort_index(inplace=True)

    f2_score=None
    y_predicted=None

    #Convert the rest of the non numeric elements in the format of a string to a nu
    #https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_numeric.h
    try:
        test_original=test_original.apply(pd.to_numeric,downcast='float')
    except ValueError as e:
        print(e)
        y_predicted="Issue in your sensor data check the list below."
        return y_predicted,f2_score,list_of_issues
        #return render_template('output.html',f2=f2_score,l_o_i=list_of_issues,y_pr


    #If y is given and is not None
    if y_original is not None:

        #Remove the id if present
        try:
            y_original.drop("id",axis=1,inplace=True)
        except KeyError:
            pass

        for column in y_original.columns.values:
            for ind,ele in zip(y_original[column].index.values,y_original[column]):
```

```
                Report_Gen(
                    num=y_original[column][ind],
                    minimum=0.0,
                    maximum=1.0,
                    column=column,
                    index=ind,
                    l_o_i=list_of_issues)

        for column in list(y_original.columns.values):
            y_original[column]=y_original[column].apply(remove_whitespaces).replace

        print("test_original.shape[0] : ",test_original.shape[0])
        print("y_original.shape[0] : ",y_original.shape[0])

        if (y_original.shape[0]!=test_original.shape[0]):
            y_predicted="The size of the test data must be the same as the size of
            return y_predicted,f2_score,list_of_issues
            #return flask.render_template('first_page.html',csv_only_message="The s

        #Convert the rest of the non numeric elements in the format of a string to
        #https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_numer
        try:
            y_original=y_original.apply(pd.to_numeric,downcast='float')
        except ValueError as e:
            y_predicted="Issue in your sensor/target data check the list below."
            return y_predicted,f2_score,list_of_issues
            #return render_template('output.html',f2=f2_score,l_o_i=list_of_issues,

        y_original.sort_index(inplace=True)

        #As input is being taken from a csv , the each value has become numpy array
        #So we convert it into numpy array and then to pandas Series
        y_original=pd.Series(np.array(y_original).flatten())

        #impute the datapoints with their class values in the train data only.
        test_original,y_original=impute_median_given_y(test_original,y_original)

        #Standardize the data and only include those columns that were selected dur
        y_predicted=standardize_featurize_n_predict(test_original)

        #Calculate the F2_score
        f2_score=fbeta_score(y_original, y_predicted, pos_label=1.0, beta=2)

        index=test_original.reset_index()["index"]

        #JSON stands for JavaScript Object Notation.
        #JSON is a lightweight format for storing and transporting data.
        #JSON is often used when data is sent from a server to a web page.
        #JSON is "self-describing" and easy to understand.

        index=index.tolist()
        y_original = y_original.tolist()
        y_predicted = y_predicted.tolist()
        f2_score = f2_score


        #To display on the ipython notebook
        return y_predicted,f2_score,list_of_issues

        #return the prediction and the F2_Score
        #return render_template('output.html',f2=f2_score,l_o_i=list_of_issues,y_pr
```

```
    #else
    else :

        #Load what median values to impute in this data if np.NaNs are there.
        #These values are taken previously from the train data.
        train_dataframe_median_values_smotetomek_data=\
        pickle.load(open("train_dataframe_median_values_smotetomek_data.pickle","rb

        #Fill the np.NaN in a particular feature with the median values for that pa
        test_original.fillna(train_dataframe_median_values_smotetomek_data,inplace=

        #Standardize the data and only include those columns that were selected dur
        y_predicted=standardize_featurize_n_predict(test_original)

        index=test_original.reset_index()["index"]

        #JSON stands for JavaScript Object Notation.
        #JSON is a lightweight format for storing and transporting data.
        #JSON is often used when data is sent from a server to a web page.
        #JSON is "self-describing" and easy to understand.
        index=index.tolist()
        y_predicted = y_predicted.tolist()

        #To display on the ipython notebook
        return y_predicted,f2_score,list_of_issues

        #return the prediction and the F2_Score
        #return render_template('output.html',f2=f2_score,l_o_i=list_of_issues,y_pr
```

In [5]:

```
def func(num,minimum,maximum,column,index,l_o_i):
    try:
        if (float(num) not in [np.nan]) :
            if (float(num) >= minimum) and (float(num) <= maximum):
                pass
            else:
                l_o_i.append(f"Out of range index:{index}, Sensor:{column}, value:{
        else:
            l_o_i.append(f"NaN value detected index:{index}, Sensor:{column}, value
    #We get value error when it is actually not a numeric after typecasting
    except (TypeError,ValueError):
        l_o_i.append(f"string instance index :{index}, Sensor:{column}, value:{num}
```

In [6]:

```
def Report_Gen(num,minimum,maximum,column,index,l_o_i):
    if isinstance(num,str):
        func(num,minimum,maximum,column,index,l_o_i)
    elif isinstance(num,datetime.datetime):
        l_o_i.append(f"Date time instance index :{index}, Sensor:{column}, value:{n
    elif isinstance(num,datetime.timedelta):
        l_o_i.append(f"datetime.timedelta instance index :{index}, Sensor:{column},
    elif isinstance(num,np.bool_):
        l_o_i.append(f"boolean instance index :{index}, Sensor:{column}, value:{num
    else:
        func(num,minimum,maximum,column,index,l_o_i)
```

In [7]:

```
def remove_whitespaces(num):
    try:
        num=num.strip()
        return num
    except :
        return num
```

In [8]:

```
def check_extension(name):
    name_list=name.split(".")
    print("name_list : ",name_list[-1])
    if (name_list[-1]=="csv") or (name_list[-1]=="CSV"):
        return True
    else:
        return False
```

In [9]:

```
def standardize_featurize_n_predict(dataframe):


    """
    This function to be used to standardize the particular columns of the test data
    Scalar model that was fitted on the train data columns.There is a list of fitte
    One for each column. Followed by the selection of only those features that we f
    svd/rfe and spearman corelation coefficient.Then we load the best performing mo
    model predict the y values.
    """


    #Load the StandardScalar dictionary that contains the trained StandardScalar mo
    #on that particular feature.
    scalar_dict=pickle.load(open("scalar_dict_smotetomek_data.pkl","rb"))

    #create an empty dataframe
    test_dataframe=pd.DataFrame()

    #Feature wise load the StandardScalar model for that particular feature and sta
    #particular feature with the StandardScalar model for that particular feature.
    for column in list(scalar_dict.keys()):
        #Convert into column vector that is many rows but only one column.
        #After standardizing , put the feature into another dataframe.
        test_dataframe[column] = scalar_dict[column].transform(dataframe[column].va

    #After all the process of smotetomek/SVD/RFE/Spearman correlation coefficient,
    #load the name of the reduced features that remain.
    new_x_smotetomek_df_columns=np.load('new_x_smotetomek_df_columns.npy',allow_pic

    #Take only those features in your test dataset.
    test_dataframe=test_dataframe[new_x_smotetomek_df_columns]

    #load the machine learning XGB model that performed the best, with all it's par
    Model= pickle.load(open("Model.pkl",'rb'))

    #Make the model predict the y_values for our test dataset.
    y_predicted=Model.predict(test_dataframe)

    return y_predicted
```

In [10]:

```
def impute_median_given_y(dataframe,target):


    """
    In this function I impute the median values of the 0 class as well as the 1 cla
    to those data points whose class I know to be 0 or 1 . I will impute the values
    """

    train_1_dataframe_median_values_smotetomek_data=\
    pickle.load(open("train_1_dataframe_median_values_smotetomek_data.pickle","rb")
    train_0_dataframe_median_values_smotetomek_data=\
    pickle.load(open("train_0_dataframe_median_values_smotetomek_data.pickle","rb")


    dataframe_1=dataframe.loc[target[target==1].reset_index()["index"]]
    dataframe_0=dataframe.loc[target[target==0].reset_index()["index"]]


    y_1=target[target==1]
    y_0=target[target==0]


    dataframe_1.fillna(train_1_dataframe_median_values_smotetomek_data,inplace=True
    dataframe_0.fillna(train_0_dataframe_median_values_smotetomek_data,inplace=True

    dataframe_new = pd.concat([dataframe_0,dataframe_1])
    dataframe_new.sort_index(inplace=True)

    y_new=pd.concat([y_0,y_1])
    y_new.sort_index(inplace=True)

    return dataframe_new,y_new
```

In [11]:

```
file_extension_correct=check_extension("x_test.csv")
if not file_extension_correct:
    message = " You did not upload a csv file !!!! Please upload a csv file only."
    print(message)
```

```
name_list :  csv
```

In [12]:

```
@app.route('/download_x_test')
def download_x ():
    path = "x_test_sample.csv"
    return send_file(path, as_attachment=True)
```

In [13]:

```
@app.route('/download_y_test')
def download_y ():
    path = "y_test_sample.csv"
    return send_file(path, as_attachment=True)
```

if **name** == '**main**': app.run()

1) Dataset with target as only one class(0) and only one class prediction.

In [14]:

```
test_na=pd.read_csv("test_one_class_only.csv")
y_test=pd.read_csv("y_test_one_class_only_0.csv")
```

In [15]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  0.52
list length  1298
list   ['string instance index :2, Sensor:sensor2_measure, value:na, m
inimum expected value:0.0, maximum expected value:204.0', 'string inst
ance index :3, Sensor:sensor2_measure, value:na, minimum expected valu
e:0.0, maximum expected value:204.0', 'string instance index :6, Senso
r:sensor2_measure, value:na, minimum expected value:0.0, maximum expec
ted value:204.0', 'string instance index :7, Sensor:sensor2_measure, v
alue:na, minimum expected value:0.0, maximum expected value:204.0', 's
tring instance index :8, Sensor:sensor2_measure, value:na, minimum exp
ected value:0.0, maximum expected value:204.0']
test_original.shape[0] :  100
y_original.shape[0] :  100
[20:35:34] WARNING: ../src/gbm/gbtree.cc:343: Loading from a raw memor
y buffer on CPU only machine.  Changing tree_method to hist.
[20:35:34] WARNING: ../src/learner.cc:207: No visible GPU is found, se
tting `gpu_id` to -1
CPU times: user 1.15 s, sys: 17.8 ms, total: 1.16 s
Wall time: 1.07 s

/home/a/.local/lib/python3.8/site-packages/sklearn/metrics/_classifica
tion.py:1464: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 due to no true nor predicted samples. Use `zero_division` p
arameter to control this behavior.
  _warn_prf(
```

In [16]:

```
list_of_issues[-10:-1]
```

Out[16]:

```
['string instance index :70, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :83, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :84, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :85, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :7, Sensor:sensor106_measure, value:na, minimu
m expected value:0.0, maximum expected value:482.0',
 'string instance index :31, Sensor:sensor106_measure, value:na, minim
um expected value:0.0, maximum expected value:482.0',
 'string instance index :67, Sensor:sensor106_measure, value:na, minim
um expected value:0.0, maximum expected value:482.0',
 'string instance index :7, Sensor:sensor107_measure, value:na, minimu
m expected value:0.0, maximum expected value:1146.0',
 'string instance index :31, Sensor:sensor107_measure, value:na, minim
um expected value:0.0, maximum expected value:1146.0']
```

In [17]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0]
```

In [18]:

```
pd.Series(y_predicted).value_counts()
```

Out[18]:

```
0.0    100
dtype: int64
```

In [19]:

```
f2_score
```

Out[19]:

```
0.0
```

2) Dataset with target as only one class(1) and multiple class predictions.

In [56]:

```
test_na=pd.read_csv("test_one_class_only.csv")
y_test=pd.read_csv("y_test_one_class_only_1.csv")
```

In [57]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  0.55
list length  1298
list   ['string instance index :2, Sensor:sensor2_measure, value:na, m
inimum expected value:0.0, maximum expected value:204.0', 'string inst
ance index :3, Sensor:sensor2_measure, value:na, minimum expected valu
e:0.0, maximum expected value:204.0', 'string instance index :6, Senso
r:sensor2_measure, value:na, minimum expected value:0.0, maximum expec
ted value:204.0', 'string instance index :7, Sensor:sensor2_measure, v
alue:na, minimum expected value:0.0, maximum expected value:204.0', 's
tring instance index :8, Sensor:sensor2_measure, value:na, minimum exp
ected value:0.0, maximum expected value:204.0']
test_original.shape[0] :  100
y_original.shape[0] :  100
[20:56:13] WARNING: ../src/gbm/gbtree.cc:343: Loading from a raw memor
y buffer on CPU only machine.  Changing tree_method to hist.
[20:56:13] WARNING: ../src/learner.cc:207: No visible GPU is found, se
tting `gpu_id` to -1
CPU times: user 1.08 s, sys: 4.06 ms, total: 1.08 s
Wall time: 1.07 s
```

In [58]:

```
list_of_issues[-10:-1]
```

Out[58]:

```
['string instance index :70, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :83, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :84, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :85, Sensor:sensor104_measure, value:na, minim
um expected value:0.0, maximum expected value:82806.0',
 'string instance index :7, Sensor:sensor106_measure, value:na, minimu
m expected value:0.0, maximum expected value:482.0',
 'string instance index :31, Sensor:sensor106_measure, value:na, minim
um expected value:0.0, maximum expected value:482.0',
 'string instance index :67, Sensor:sensor106_measure, value:na, minim
um expected value:0.0, maximum expected value:482.0',
 'string instance index :7, Sensor:sensor107_measure, value:na, minimu
m expected value:0.0, maximum expected value:1146.0',
 'string instance index :31, Sensor:sensor107_measure, value:na, minim
um expected value:0.0, maximum expected value:1146.0']
```

In [59]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0,
1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 0.0]
```

In [60]:

```
pd.Series(y_predicted).value_counts()
```

Out[60]:

```
1.0    85
0.0    15
dtype: int64
```

In [61]:

```
f2_score
```

Out[61]:

```
0.8762886597938145
```

1) Test data not having a lot of different strings.

In [20]:

```
test_data=pd.read_csv("equip_failures_test_set.csv")
```

In [21]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_data)
```

```
Time Taken by report generation :  87.21
list length  228814
list   ['Out of range index:5858, Sensor:sensor1_measure, value:429496
72, minimum expected value:0.0, maximum expected value:2434708.0', 'Ou
t of range index:7902, Sensor:sensor1_measure, value:42949672, minimum
expected value:0.0, maximum expected value:2434708.0', 'string instanc
e index :0, Sensor:sensor2_measure, value:na, minimum expected value:
0.0, maximum expected value:204.0', 'string instance index :1, Sensor:
sensor2_measure, value:na, minimum expected value:0.0, maximum expecte
d value:204.0', 'string instance index :2, Sensor:sensor2_measure, val
ue:na, minimum expected value:0.0, maximum expected value:204.0']
[20:37:28] WARNING: ../src/gbm/gbtree.cc:343: Loading from a raw memor
y buffer on CPU only machine.  Changing tree_method to hist.
[20:37:28] WARNING: ../src/learner.cc:207: No visible GPU is found, se
tting `gpu_id` to -1
CPU times: user 1min 34s, sys: 195 ms, total: 1min 35s
Wall time: 1min 35s
```

In [22]:

```
list_of_issues[-10:-1]
```

Out[22]:

```
['string instance index :15783, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15784, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15822, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15840, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15863, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15864, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15868, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15916, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :15931, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0']
```

In [23]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
```

In [24]:

```
f2_score
```

In [25]:

```
pd.Series(y_predicted).value_counts()
```

Out[25]:

```
0.0    15894
1.0      107
dtype: int64
```

2)Test data having a lot of different strings.

In [26]:

```
test_data=pd.read_csv("equip_failures_test_set2.csv")
```

In [27]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_data)
```

```
Time Taken by report generation :  0.04
list length  177
list   ['string instance index :0, Sensor:sensor2_measure, value:na, m
inimum expected value:0.0, maximum expected value:204.0', 'string inst
ance index :1, Sensor:sensor2_measure, value:na, minimum expected valu
e:0.0, maximum expected value:204.0', 'string instance index :2, Senso
r:sensor2_measure, value:na, minimum expected value:0.0, maximum expec
ted value:204.0', 'string instance index :4, Sensor:sensor2_measure, v
alue:na, minimum expected value:0.0, maximum expected value:204.0', 's
tring instance index :1, Sensor:sensor3_measure, value:na, minimum exp
ected value:0.0, maximum expected value:2130706816.0']
Unable to parse string "yoyoy!!!" at position 1
CPU times: user 199 ms, sys: 8.11 ms, total: 207 ms
Wall time: 202 ms
```

In [28]:

```
list_of_issues[-10:-1]
```

Out[28]:

```
['string instance index :2, Sensor:sensor100_measure, value:na, minimu
m expected value:0.0, maximum expected value:940.0',
 'string instance index :2, Sensor:sensor101_measure, value:na, minimu
m expected value:0.0, maximum expected value:3342.0',
 'string instance index :1, Sensor:sensor102_measure, value:na, minimu
m expected value:0.0, maximum expected value:1322456960.0',
 'string instance index :2, Sensor:sensor102_measure, value:na, minimu
m expected value:0.0, maximum expected value:1322456960.0',
 'string instance index :1, Sensor:sensor103_measure, value:na, minimu
m expected value:0.0, maximum expected value:106020.21875',
 'string instance index :2, Sensor:sensor103_measure, value:na, minimu
m expected value:0.0, maximum expected value:106020.21875',
 'string instance index :1, Sensor:sensor104_measure, value:na, minimu
m expected value:0.0, maximum expected value:82806.0',
 'string instance index :2, Sensor:sensor104_measure, value:na, minimu
m expected value:0.0, maximum expected value:82806.0',
 'string instance index :2, Sensor:sensor106_measure, value:na, minimu
m expected value:0.0, maximum expected value:482.0']
```

In [29]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  Issue in your sensor data check the list below.
```

In [30]:

```
pd.Series(y_predicted).value_counts()
```

Out[30]:

```
Issue in your sensor data check the list below.    1
dtype: int64
```

In [31]:

```
f2_score
```

3)Test and target data not having a lot of different strings.

In [32]:

```
test_na=pd.read_csv("test_na.csv")
y_test=pd.read_csv("y_test.csv")
```

In [33]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  64.69
list length  168535
list   ['Out of range index:3748, Sensor:sensor1_measure, value:274656
4.0, minimum expected value:0.0, maximum expected value:2434708.0', 's
tring instance index :2, Sensor:sensor2_measure, value:na, minimum exp
ected value:0.0, maximum expected value:204.0', 'string instance index
:3, Sensor:sensor2_measure, value:na, minimum expected value:0.0, maxi
mum expected value:204.0', 'string instance index :6, Sensor:sensor2_m
easure, value:na, minimum expected value:0.0, maximum expected value:2
04.0', 'string instance index :7, Sensor:sensor2_measure, value:na, mi
nimum expected value:0.0, maximum expected value:204.0']
test_original.shape[0] :  12000
y_original.shape[0] :  12000
[20:39:15] WARNING: ../src/gbm/gbtree.cc:343: Loading from a raw memor
y buffer on CPU only machine.  Changing tree_method to hist.
[20:39:15] WARNING: ../src/learner.cc:207: No visible GPU is found, se
tting `gpu_id` to -1
CPU times: user 1min 10s, sys: 67.8 ms, total: 1min 10s
Wall time: 1min 10s
```

In [34]:

```
list_of_issues[-10:-1]
```

Out[34]:

```
['string instance index :11652, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11655, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11759, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11778, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11789, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11797, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11812, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11844, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11931, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0']
```

In [35]:

```python
print("y_predicted : ",y_predicted)
```

```
y_predicted :  [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.
```

In [36]:

```python
pd.Series(y_predicted).value_counts()
```

Out[36]:

```
0.0    11797
1.0      203
dtype: int64
```

In [37]:

```python
f2_score
```

Out[37]:

```
0.9571286141575274
```

4) Test and target data with target data having a lot of different strings.

In [38]:

```python
test_na=pd.read_csv("test_na.csv")
y_test=pd.read_csv("y_test_2.csv")
```

In [39]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  69.86
list length  168535
list   ['Out of range index:3748, Sensor:sensor1_measure, value:274656
4.0, minimum expected value:0.0, maximum expected value:2434708.0', 's
tring instance index :2, Sensor:sensor2_measure, value:na, minimum exp
ected value:0.0, maximum expected value:204.0', 'string instance index
:3, Sensor:sensor2_measure, value:na, minimum expected value:0.0, maxi
mum expected value:204.0', 'string instance index :6, Sensor:sensor2_m
easure, value:na, minimum expected value:0.0, maximum expected value:2
04.0', 'string instance index :7, Sensor:sensor2_measure, value:na, mi
nimum expected value:0.0, maximum expected value:204.0']
test_original.shape[0] :  12000
y_original.shape[0] :  12000
CPU times: user 1min 15s, sys: 88 ms, total: 1min 15s
Wall time: 1min 15s
```

In [40]:

```
list_of_issues[-10:-1]
```

Out[40]:

```
['string instance index :11972, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :0, Sensor:target, value:np.Nan, minimum expec
ted value:0.0, maximum expected value:1.0',
 'NaN value detected index:1, Sensor:target, value:nan, minimum expect
ed value:0.0, maximum expected value:1.0',
 'string instance index :2, Sensor:target, value:YOYOYO, minimum expec
ted value:0.0, maximum expected value:1.0',
 'string instance index :3, Sensor:target, value:Whay?, minimum expect
ed value:0.0, maximum expected value:1.0',
 'string instance index :4, Sensor:target, value:Wwhay?hay?, minimum e
xpected value:0.0, maximum expected value:1.0',
 'string instance index :5, Sensor:target, value:NONE, minimum expecte
d value:0.0, maximum expected value:1.0',
 'string instance index :6, Sensor:target, value:TRUE, minimum expecte
d value:0.0, maximum expected value:1.0',
 'NaN value detected index:7, Sensor:target, value:nan, minimum expect
ed value:0.0, maximum expected value:1.0']
```

In [41]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  Issue in your sensor/target data check the list below.
```

In [42]:

```
pd.Series(y_predicted).value_counts()
```

Out[42]:

```
Issue in your sensor/target data check the list below.    1
dtype: int64
```

In [43]:

```
f2_score
```

5) Test and target data , with test data having a lot of different strings.

In [44]:

```
test_na=pd.read_csv("test_na_2.csv")
y_test=pd.read_csv("y_test.csv")
```

```
/home/a/.local/lib/python3.8/site-packages/IPython/core/interactiveshe
ll.py:3145: DtypeWarning: Columns (0) have mixed types.Specify dtype o
ption on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [45]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  67.02
list length  168540
list   ['string instance index :2, Sensor:sensor1_measure, value:np.na
n, minimum expected value:0.0, maximum expected value:2434708.0', 'str
ing instance index :3, Sensor:sensor1_measure, value:!!!, minimum expe
cted value:0.0, maximum expected value:2434708.0', 'string instance in
dex :4, Sensor:sensor1_measure, value:hello, minimum expected value:0.
0, maximum expected value:2434708.0', 'Out of range index:3748, Senso
r:sensor1_measure, value:2746564, minimum expected value:0.0, maximum
expected value:2434708.0', 'string instance index :2, Sensor:sensor2_m
easure, value:na, minimum expected value:0.0, maximum expected value:2
04.0']
Unable to parse string "!!!" at position 3
CPU times: user 1min 11s, sys: 31.9 ms, total: 1min 11s
Wall time: 1min 11s
```

In [46]:

```
list_of_issues[-10:-1]
```

Out[46]:

```
['string instance index :11652, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11655, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11759, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11778, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11789, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11797, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11812, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11844, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11931, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0']
```

In [47]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  Issue in your sensor data check the list below.
```

In [48]:

```
pd.Series(y_predicted).value_counts()
```

Out[48]:

```
Issue in your sensor data check the list below.    1
dtype: int64
```

In [49]:

```
f2_score
```

6) Test and target , with both test and target data having a lot of different strings.

In [50]:

```
test_na=pd.read_csv("test_na_2.csv")
y_test=pd.read_csv("y_test_2.csv")
```

```
/home/a/.local/lib/python3.8/site-packages/IPython/core/interactiveshe
ll.py:3145: DtypeWarning: Columns (0) have mixed types.Specify dtype o
ption on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

In [51]:

```
%%time
y_predicted,f2_score,list_of_issues=final(test_na,y_test)
```

```
Time Taken by report generation :  66.39
list length  168540
list   ['string instance index :2, Sensor:sensor1_measure, value:np.na
n, minimum expected value:0.0, maximum expected value:2434708.0', 'str
ing instance index :3, Sensor:sensor1_measure, value:!!!, minimum expe
cted value:0.0, maximum expected value:2434708.0', 'string instance in
dex :4, Sensor:sensor1_measure, value:hello, minimum expected value:0.
0, maximum expected value:2434708.0', 'Out of range index:3748, Senso
r:sensor1_measure, value:2746564, minimum expected value:0.0, maximum
expected value:2434708.0', 'string instance index :2, Sensor:sensor2_m
easure, value:na, minimum expected value:0.0, maximum expected value:2
04.0']
Unable to parse string "!!!" at position 3
CPU times: user 1min 10s, sys: 3.91 ms, total: 1min 10s
Wall time: 1min 10s
```

In [52]:

```
list_of_issues[-10:-1]
```

Out[52]:

```
['string instance index :11652, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11655, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11759, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11778, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11789, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11797, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11812, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11844, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0',
 'string instance index :11931, Sensor:sensor107_measure, value:na, mi
nimum expected value:0.0, maximum expected value:1146.0']
```

In [53]:

```
print("y_predicted : ",y_predicted)
```

```
y_predicted :  Issue in your sensor data check the list below.
```

In [54]:

```
pd.Series(y_predicted).value_counts()
```

Out[54]:

```
Issue in your sensor data check the list below.    1
dtype: int64
```

In [55]:

```
f2_score
```

Conclusion : For 16000 data points, it takes 87 seconds for report generation and 8 seconds for preprocessing and prediction. Total 1 minute 35 seconds.
Similarly :
For 12000 data points, it takes 64 seconds for report generation and 6 seconds for preprocessing and prediction. Total 1 minute 10 seconds.