In [81]:
```
import numpy as np
import pandas as pd
import warnings
train_file = r'D:\python_project\logistic_regression\train_data_set.csv'
test_file = r'D:\python_project\logistic_regression\test_data_set.csv'
mt_train = pd.read_csv(train_file)
mt_test = pd.read_csv(test_file)
```

In [82]:
```
mt_train.columns.nunique()
mt_test.columns.nunique()
```

Out[82]: 11

In [83]:
```
mt_test.head()
```

Out[83]:

	created_at	entry_id	current	voltage	vibration	temperature	latitude	longitude	relay_st
0	2022-04-05T13:30:41+05:30	1	2.223148	223.14752	0	37.36264	NaN	NaN	
1	2022-04-05T13:30:58+05:30	2	2.213905	224.07919	0	37.48474	NaN	NaN	
2	2022-04-05T13:31:14+05:30	3	2.333177	223.81500	0	37.60684	NaN	NaN	
3	2022-04-05T13:31:30+05:30	4	2.097085	224.11105	0	37.24054	NaN	NaN	
4	2022-04-05T13:31:47+05:30	5	2.249357	223.86400	0	37.36264	NaN	NaN	

In [84]:
```
mt_train.head()
```

Out[84]:

	created_at	entry_id	current	voltage	vibration	temperature	latitude	longitude	relay_st
0	2022-03-15T17:17:51+05:30	1	1.938089	238.64746	0	37.11844	NaN	NaN	
1	2022-03-15T17:18:07+05:30	2	2.352603	215.62500	0	37.36264	NaN	NaN	
2	2022-03-15T17:18:24+05:30	3	3.654926	152.06055	0	37.24054	NaN	NaN	
3	2022-03-15T17:18:41+05:30	4	2.420011	195.74707	0	37.11844	NaN	NaN	
4	2022-03-15T17:19:00+05:30	5	2.632627	177.10449	0	37.11844	NaN	NaN	

In [85]:
```
mt_train['data']='train'
```

```
mt_test['data']='test'
mt_all = pd.concat([mt_train,mt_test],axis=0)
```

In [86]:
```
mt_all.nunique()
```

Out[86]:
```
created_at      952
entry_id        759
current         952
voltage         943
vibration         2
temperature     142
latitude          0
longitude         0
relay_status      2
humidity         37
failure           2
data              2
dtype: int64
```

In [87]:
```
mt_all.head(1392)
```

Out[87]:

	created_at	entry_id	current	voltage	vibration	temperature	latitude	longitude	relay
0	2022-03-15T17:17:51+05:30	1	1.938089	238.64746	0	37.11844	NaN	NaN	
1	2022-03-15T17:18:07+05:30	2	2.352603	215.62500	0	37.36264	NaN	NaN	
2	2022-03-15T17:18:24+05:30	3	3.654926	152.06055	0	37.24054	NaN	NaN	
3	2022-03-15T17:18:41+05:30	4	2.420011	195.74707	0	37.11844	NaN	NaN	
4	2022-03-15T17:19:00+05:30	5	2.632627	177.10449	0	37.11844	NaN	NaN	
...	...	...	...	...	...	...	...	...	
188	2022-04-05T14:45:50+05:30	189	2.447289	211.85481	0	43.83394	NaN	NaN	
189	2022-04-05T14:46:06+05:30	190	2.425245	212.24246	0	44.07814	NaN	NaN	
190	2022-04-05T14:46:22+05:30	191	4.281932	238.45860	0	45.12915	NaN	NaN	
191	2022-04-05T14:46:38+05:30	192	3.826854	401.23560	0	47.26988	NaN	NaN	
192	2022-04-05T14:46:54+05:30	193	4.181952	250.38970	0	45.63972	NaN	NaN	

952 rows × 12 columns

In [88]:
```
mt_all.drop(['created_at','entry_id','relay_status','longitude','latitude'],axis=1,inpl
```

```
mt_all.head()
```

Out[88]:

	current	voltage	vibration	temperature	humidity	failure	data
0	1.938089	238.64746	0	37.11844	25	yes	train
1	2.352603	215.62500	0	37.36264	25	no	train
2	3.654926	152.06055	0	37.24054	46	no	train
3	2.420011	195.74707	0	37.11844	26	no	train
4	2.632627	177.10449	0	37.11844	26	no	train

In [89]:
```
mt_all['vibration']=(mt_all['vibration']==1).astype(int)
mt_all['vibration'].unique()
```

Out[89]:
```
array([0, 1])
```

In [90]:
```
mt_all['failure']=(mt_all['failure']=='yes').astype(int)
#mt_all['failure']=pd.to_numeric(mt_all['failure'],errors='coerce')
mt_all['failure'].unique()
```

Out[90]:
```
array([1, 0])
```

In [91]:
```
mt_all['failure'].unique()
```

Out[91]:
```
array([1, 0])
```

In [92]:
```
for col in mt_all.columns:
    if col=='data' or mt_all[col].isnull().sum()==0:continue
    mt_all.loc[mt_all[col].isnull(),col]=mt_all.loc[mt_all['data']=='train',col].mean()
```

In [93]:
```
mt_train = mt_all[mt_all['data']=='train']
del mt_train['data']
mt_test = mt_all[mt_all['data']=='test']
mt_test.drop(['data','failure'],axis=1,inplace=True)
```

```
C:\Users\Omkar250\anaconda3\lib\site-packages\pandas\core\frame.py:4308: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

In [94]:
```
from sklearn.linear_model import LogisticRegression
```

In [95]:
```
params = {'class_weight':['balanced'],'penalty':['l1','l2'],
          'C':np.linspace(0.0001,1000,10)}
```

In [96]:
```
model = LogisticRegression(fit_intercept=True)
```

In [97]:
```
from sklearn.model_selection import GridSearchCV
```

In [98]:
```
grid_search = GridSearchCV(model,param_grid=params,cv=10,scoring='roc_auc',n_jobs=-1)
```

In [99]:
```
x_train = mt_train.drop('failure',axis=1)
y_train = mt_train['failure']
```

In [100…
```
grid_search.fit(x_train,y_train)
```

```
C:\Users\Omkar250\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:918: Us
erWarning: One or more of the test scores are non-finite: [       nan 0.82535887
nan 0.98152174        nan 0.98244306
        nan 0.98291925        nan 0.98128364        nan 0.98271222
        nan 0.98196687        nan 0.98220497        nan 0.98312629
        nan 0.98291925]
  warnings.warn(
C:\Users\Omkar250\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[100…
```
GridSearchCV(cv=10, estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'C': array([1.000000e-04, 1.111112e+02, 2.222223e+02, 3.333334e
+02,
       4.444445e+02, 5.555556e+02, 6.666667e+02, 7.777778e+02,
       8.888889e+02, 1.000000e+03]),
                         'class_weight': ['balanced'],
                         'penalty': ['l1', 'l2']},
             scoring='roc_auc')
```

In [101…
```
grid_search.best_estimator_
```

Out[101…
```
LogisticRegression(C=888.8889, class_weight='balanced')
```

In [102…
```
mt_train['failure'].nunique()
grid_search.cv_results_
```

Out[102…
```
{'mean_fit_time': array([0.00290363, 0.05640724, 0.0026552 , 0.09638443, 0.00205066,
        0.07529125, 0.00274177, 0.07637553, 0.00223219, 0.06401577,
        0.00239375, 0.06791797, 0.00214696, 0.08126185, 0.00219271,
        0.07892537, 0.00209434, 0.07617378, 0.0023937 , 0.07993965]),
 'std_fit_time': array([0.00015297, 0.01308862, 0.00043239, 0.01876899, 0.00016699,
        0.00655309, 0.00116188, 0.00711257, 0.00039607, 0.01110674,
        0.00066166, 0.01249217, 0.00032261, 0.00944136, 0.00039957,
        0.00947944, 0.00036288, 0.00777443, 0.00091423, 0.00716464]),
 'mean_score_time': array([0.        , 0.00815263, 0.        , 0.01017141, 0.        ,
        0.00708137, 0.        , 0.00628347, 0.        , 0.00579731,
        0.        , 0.00589328, 0.        , 0.00688162, 0.        ,
```

```
       0.00804198, 0.        , 0.00602436, 0.        , 0.00563202]),
 'std_score_time': array([0.        , 0.00083781, 0.        , 0.00725848, 0.        ,
       0.00175321, 0.        , 0.0006386 , 0.        , 0.00125311,
       0.        , 0.00157448, 0.        , 0.00082822, 0.        ,
       0.00406582, 0.        , 0.00079393, 0.        , 0.00116036]),
 'param_C': masked_array(data=[0.0001, 0.0001, 111.11120000000001, 111.11120000000001,
                   222.22230000000002, 222.22230000000002, 333.3334,
                   333.3334, 444.4445, 444.4445, 555.5556, 555.5556,
                   666.6667, 666.6667, 777.7778000000001,
                   777.7778000000001, 888.8889, 888.8889, 1000.0, 1000.0],
              mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False],
       fill_value='?',
            dtype=object),
 'param_class_weight': masked_array(data=['balanced', 'balanced', 'balanced', 'balance
d',
                   'balanced', 'balanced', 'balanced', 'balanced',
                   'balanced', 'balanced', 'balanced', 'balanced',
                   'balanced', 'balanced', 'balanced', 'balanced',
                   'balanced', 'balanced', 'balanced', 'balanced'],
              mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False],
       fill_value='?',
            dtype=object),
 'param_penalty': masked_array(data=['l1', 'l2', 'l1', 'l2', 'l1', 'l2', 'l1', 'l2', 'l
1',
                   'l2', 'l1', 'l2', 'l1', 'l2', 'l1', 'l2', 'l1', 'l2',
                   'l1', 'l2'],
              mask=[False, False, False, False, False, False, False, False,
                   False, False, False, False, False, False, False, False,
                   False, False, False, False],
       fill_value='?',
            dtype=object),
 'params': [{'C': 0.0001, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 0.0001, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 111.11120000000001, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 111.11120000000001, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 222.22230000000002, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 222.22230000000002, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 333.3334, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 333.3334, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 444.4445, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 444.4445, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 555.5556, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 555.5556, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 666.6667, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 666.6667, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 777.7778000000001, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 777.7778000000001, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 888.8889, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 888.8889, 'class_weight': 'balanced', 'penalty': 'l2'},
  {'C': 1000.0, 'class_weight': 'balanced', 'penalty': 'l1'},
  {'C': 1000.0, 'class_weight': 'balanced', 'penalty': 'l2'}],
 'split0_test_score': array([       nan, 0.99585921,        nan, 0.99378882,        nan,
       0.99585921,        nan, 0.99585921,        nan, 0.99378882,
              nan, 0.99378882,        nan, 0.99585921,        nan,
       0.99585921,        nan, 0.99792961,        nan, 0.99585921]),
 'split1_test_score': array([nan,  1., nan,  1., nan,  1., nan,  1., nan,  1., nan,  1.,
nan,
        1., nan,  1., nan,  1., nan,  1.]),
 'split2_test_score': array([nan,  1., nan,  1., nan,  1., nan,  1., nan,  1., nan,  1.,
nan,
        1., nan,  1., nan,  1., nan,  1.]),
```

```
 'split3_test_score': array([       nan, 0.73809524,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ,
              nan, 1.        ,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ]),
 'split4_test_score': array([nan,  1., nan,  1., nan,  1., nan,  1., nan,  1., nan,  1.,
nan,
        1., nan,  1., nan,  1., nan,  1.]),
 'split5_test_score': array([       nan, 0.98571429,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ,
              nan, 1.        ,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ]),
 'split6_test_score': array([       nan, 0.98571429,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ,
              nan, 1.        ,        nan, 1.        ,        nan,
       1.        ,        nan, 1.        ,        nan, 1.        ]),
 'split7_test_score': array([nan,  1., nan,  1., nan,  1., nan,  1., nan,  1., nan,  1.,
nan,
        1., nan,  1., nan,  1., nan,  1.]),
 'split8_test_score': array([       nan, 0.50714286,        nan, 0.98809524,        nan,
       0.9952381 ,        nan, 1.        ,        nan, 0.98571429,
              nan, 1.        ,        nan, 0.99047619,        nan,
       0.99285714,        nan, 1.        ,        nan, 1.        ]),
 'split9_test_score': array([       nan, 0.0410628 ,        nan, 0.83333333,        nan,
       0.83333333,        nan, 0.83333333,        nan, 0.83333333,
              nan, 0.83333333,        nan, 0.83333333,        nan,
       0.83333333,        nan, 0.83333333,        nan, 0.83333333]),
 'mean_test_score': array([       nan, 0.82535887,        nan, 0.98152174,        nan,
       0.98244306,        nan, 0.98291925,        nan, 0.98128364,
              nan, 0.98271222,        nan, 0.98196687,        nan,
       0.98220497,        nan, 0.98312629,        nan, 0.98291925]),
 'std_test_score': array([       nan, 0.30438346,        nan, 0.04954152,        nan,
       0.04973443,        nan, 0.04987725,        nan, 0.04951507,
              nan, 0.04982738,        nan, 0.04963234,        nan,
       0.04967828,        nan, 0.0499348 ,        nan, 0.04987725]),
 'rank_test_score': array([20, 10, 17,  8, 15,  5, 13,  2, 19,  9, 12,  4, 14,  7, 16,
6, 18,
        1, 11,  2])}
```

In [103…

```
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.6f} (std: {1:.6f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}".format(results['params'][candidate]))
            print("")
```

In [104…

```
report(grid_search.cv_results_,3)
```

```
Model with rank: 1
Mean validation score: 0.983126 (std: 0.049935)
Parameters: {'C': 888.8889, 'class_weight': 'balanced', 'penalty': 'l2'}

Model with rank: 2
Mean validation score: 0.982919 (std: 0.049877)
Parameters: {'C': 333.3334, 'class_weight': 'balanced', 'penalty': 'l2'}

Model with rank: 2
Mean validation score: 0.982919 (std: 0.049877)
```

```
Parameters: {'C': 1000.0, 'class_weight': 'balanced', 'penalty': 'l2'}
```

In [105…
```
test_prediction = grid_search.predict_proba(mt_test)
```

In [106…
```
test_prediction
```

Out[106…
```
array([[9.81105344e-01, 1.88946561e-02],
       [9.78083905e-01, 2.19160945e-02],
       [9.74408974e-01, 2.55910259e-02],
       [9.81867360e-01, 1.81326395e-02],
       [9.76969607e-01, 2.30303933e-02],
       [9.86168930e-01, 1.38310697e-02],
       [9.81344741e-01, 1.86552589e-02],
       [9.83758099e-01, 1.62419008e-02],
       [9.83244606e-01, 1.67553938e-02],
       [9.79377946e-01, 2.06220543e-02],
       [9.89307585e-01, 1.06924149e-02],
       [9.85626923e-01, 1.43730767e-02],
       [9.72945867e-01, 2.70541334e-02],
       [9.84986309e-01, 1.50136910e-02],
       [9.76298213e-01, 2.37017868e-02],
       [9.89457039e-01, 1.05429611e-02],
       [9.80636745e-01, 1.93632551e-02],
       [9.82757467e-01, 1.72425332e-02],
       [9.84820278e-01, 1.51797220e-02],
       [9.90403553e-01, 9.59644733e-03],
       [9.87243805e-01, 1.27561952e-02],
       [9.90800646e-01, 9.19935420e-03],
       [9.93581387e-01, 6.41861326e-03],
       [9.93224753e-01, 6.77524686e-03],
       [9.88180468e-01, 1.18195321e-02],
       [9.93070183e-01, 6.92981716e-03],
       [9.76655255e-01, 2.33447450e-02],
       [9.82636101e-01, 1.73638985e-02],
       [9.87855736e-01, 1.21442644e-02],
       [9.83851686e-01, 1.61483136e-02],
       [9.87190949e-01, 1.28090515e-02],
       [9.86368830e-01, 1.36311697e-02],
       [9.87401217e-01, 1.25987833e-02],
       [9.87047485e-01, 1.29525147e-02],
       [9.88259824e-01, 1.17401764e-02],
       [9.79701706e-01, 2.02982940e-02],
       [9.79510762e-01, 2.04892376e-02],
       [9.90579723e-01, 9.42027652e-03],
       [9.93753303e-01, 6.24669661e-03],
       [9.85413350e-01, 1.45866500e-02],
       [9.93425330e-01, 6.57466969e-03],
       [9.86796229e-01, 1.32037715e-02],
       [9.86367979e-01, 1.36320205e-02],
       [9.90565718e-01, 9.43428167e-03],
       [9.86397107e-01, 1.36028930e-02],
       [9.79775898e-01, 2.02241021e-02],
       [9.82619491e-01, 1.73805088e-02],
       [9.88287059e-01, 1.17129410e-02],
       [9.83931883e-01, 1.60681174e-02],
       [9.90635576e-01, 9.36442440e-03],
       [9.78909022e-01, 2.10909781e-02],
       [9.92854627e-01, 7.14537336e-03],
       [9.83175522e-01, 1.68244783e-02],
       [9.87033997e-01, 1.29660033e-02],
       [9.93006109e-01, 6.99389147e-03],
```

```
[9.91799522e-01, 8.20047790e-03],
[9.84013681e-01, 1.59863191e-02],
[9.91268444e-01, 8.73155572e-03],
[9.91887872e-01, 8.11212824e-03],
[9.89131353e-01, 1.08686474e-02],
[9.93719288e-01, 6.28071186e-03],
[9.92410850e-01, 7.58914987e-03],
[9.89109046e-01, 1.08909541e-02],
[9.95395866e-01, 4.60413405e-03],
[9.86725962e-01, 1.32740377e-02],
[9.91026590e-01, 8.97340988e-03],
[9.95542215e-01, 4.45778541e-03],
[9.89139291e-01, 1.08607092e-02],
[9.96117939e-01, 3.88206148e-03],
[9.96164629e-01, 3.83537095e-03],
[9.90984993e-01, 9.01500695e-03],
[9.84356963e-01, 1.56430368e-02],
[9.91764190e-01, 8.23580971e-03],
[9.89382982e-01, 1.06170179e-02],
[9.90187805e-01, 9.81219523e-03],
[9.95068980e-01, 4.93102020e-03],
[9.91114377e-01, 8.88562278e-03],
[9.88534339e-01, 1.14656611e-02],
[9.88324753e-01, 1.16752467e-02],
[9.97182061e-01, 2.81793856e-03],
[9.88797002e-01, 1.12029976e-02],
[9.94188417e-01, 5.81158324e-03],
[9.89916046e-01, 1.00839538e-02],
[9.87646056e-01, 1.23539436e-02],
[9.94454136e-01, 5.54586440e-03],
[9.88588882e-01, 1.14111179e-02],
[9.88282198e-01, 1.17178017e-02],
[9.94084187e-01, 5.91581264e-03],
[9.93086048e-01, 6.91395247e-03],
[9.95378419e-01, 4.62158122e-03],
[9.87093313e-01, 1.29066865e-02],
[9.94219738e-01, 5.78026210e-03],
[9.93267171e-01, 6.73282860e-03],
[9.95982380e-01, 4.01761992e-03],
[9.95492405e-01, 4.50759458e-03],
[9.92209480e-01, 7.79052013e-03],
[9.94527855e-01, 5.47214540e-03],
[9.95283547e-01, 4.71645295e-03],
[9.94223810e-01, 5.77618974e-03],
[9.94445255e-01, 5.55474481e-03],
[9.94468159e-01, 5.53184086e-03],
[9.97129831e-01, 2.87016924e-03],
[9.96791369e-01, 3.20863147e-03],
[9.95540163e-01, 4.45983678e-03],
[9.96651931e-01, 3.34806915e-03],
[9.94667016e-01, 5.33298428e-03],
[9.96562948e-01, 3.43705152e-03],
[9.90775812e-01, 9.22418790e-03],
[9.93918748e-01, 6.08125216e-03],
[9.95458270e-01, 4.54172975e-03],
[9.92913827e-01, 7.08617276e-03],
[9.96776167e-01, 3.22383343e-03],
[9.97409940e-01, 2.59005952e-03],
[9.95617917e-01, 4.38208301e-03],
[9.94254556e-01, 5.74544357e-03],
[9.93051359e-01, 6.94864063e-03],
[2.22044605e-16, 1.00000000e+00],
[9.96384340e-01, 3.61566017e-03],
[9.98331594e-01, 1.66840640e-03],
[9.95307662e-01, 4.69233782e-03],
```

```
[9.97685863e-01, 2.31413714e-03],
[9.97705537e-01, 2.29446316e-03],
[9.95014971e-01, 4.98502912e-03],
[9.97567964e-01, 2.43203587e-03],
[9.97889388e-01, 2.11061222e-03],
[9.97489071e-01, 2.51092918e-03],
[9.95749915e-01, 4.25008526e-03],
[9.97331193e-01, 2.66880689e-03],
[9.95890556e-01, 4.10944440e-03],
[9.97949643e-01, 2.05035742e-03],
[9.96449267e-01, 3.55073327e-03],
[9.96435694e-01, 3.56430580e-03],
[9.96488826e-01, 3.51117433e-03],
[9.95192206e-01, 4.80779388e-03],
[9.96213761e-01, 3.78623878e-03],
[9.97599502e-01, 2.40049759e-03],
[9.95346024e-01, 4.65397604e-03],
[9.96701580e-01, 3.29842023e-03],
[9.93785167e-01, 6.21483260e-03],
[9.97955902e-01, 2.04409843e-03],
[9.96905882e-01, 3.09411805e-03],
[9.98077153e-01, 1.92284730e-03],
[9.97808131e-01, 2.19186868e-03],
[9.96597368e-01, 3.40263189e-03],
[9.98054081e-01, 1.94591882e-03],
[9.93123224e-01, 6.87677603e-03],
[9.97673158e-01, 2.32684192e-03],
[9.97108870e-01, 2.89113011e-03],
[9.98203159e-01, 1.79684063e-03],
[9.97680109e-01, 2.31989129e-03],
[9.97595267e-01, 2.40473280e-03],
[9.97059365e-01, 2.94063490e-03],
[9.97373248e-01, 2.62675205e-03],
[9.98011348e-01, 1.98865151e-03],
[9.98240513e-01, 1.75948743e-03],
[9.96113188e-01, 3.88681217e-03],
[9.97219663e-01, 2.78033684e-03],
[9.98032335e-01, 1.96766484e-03],
[9.98279021e-01, 1.72097936e-03],
[9.97850175e-01, 2.14982483e-03],
[9.97055925e-01, 2.94407466e-03],
[9.95758744e-01, 4.24125587e-03],
[9.98326232e-01, 1.67376789e-03],
[9.96821568e-01, 3.17843176e-03],
[9.97830206e-01, 2.16979435e-03],
[9.95488129e-01, 4.51187080e-03],
[9.98597916e-01, 1.40208404e-03],
[9.97451975e-01, 2.54802514e-03],
[9.97803081e-01, 2.19691888e-03],
[9.95502719e-01, 4.49728114e-03],
[9.96832321e-01, 3.16767877e-03],
[9.98112513e-01, 1.88748670e-03],
[9.97991001e-01, 2.00899870e-03],
[9.97659340e-01, 2.34066014e-03],
[9.97798235e-01, 2.20176533e-03],
[9.97521590e-01, 2.47840974e-03],
[9.95534449e-01, 4.46555080e-03],
[9.97073006e-01, 2.92699437e-03],
[9.97438122e-01, 2.56187842e-03],
[9.98159376e-01, 1.84062440e-03],
[9.98688231e-01, 1.31176872e-03],
[9.96234595e-01, 3.76540471e-03],
[9.96974909e-01, 3.02509095e-03],
[9.95527328e-01, 4.47267208e-03],
[9.98429268e-01, 1.57073189e-03],
```

```
       [9.95598786e-01, 4.40121444e-03],
       [9.98429939e-01, 1.57006065e-03],
       [9.97804963e-01, 2.19503739e-03],
       [9.95007686e-01, 4.99231371e-03],
       [9.96955165e-01, 3.04483489e-03],
       [9.33901878e-02, 9.06609812e-01],
       [1.55431223e-14, 1.00000000e+00],
       [1.27517096e-02, 9.87248290e-01]])
```

In [107…
```
grid_search.classes_
```

Out[107…
```
array([0, 1])
```

In [108…
```
train_score=grid_search.predict_proba(x_train)[:,1]
real = y_train
```

In [109…
```
cutoffs = np.linspace(.001,0.999, 999)
```

In [110…
```
import seaborn as sns
```

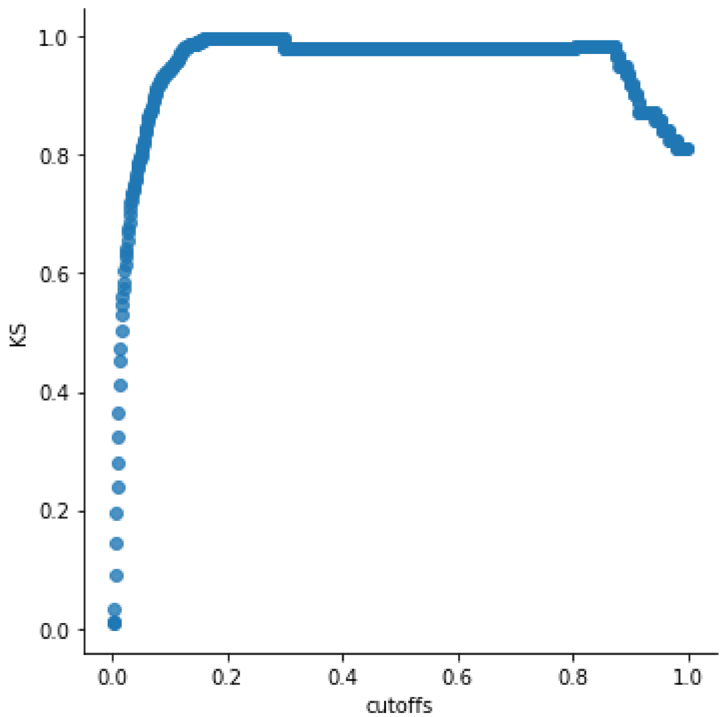
In [111…
```
 KS=[]
```

In [112…
```
for cutoff in cutoffs:
 predicted=(train_score>cutoff).astype(int)
 TP=((real==1)&(predicted==1)).sum()
 FP=((real==0)&(predicted==1)).sum()
 TN=((real==0)&(predicted==0)).sum()
 FN=((real==1)&(predicted==0)).sum()

 ks=(TP/(TP+FN))-(FP/(TN+FP))
 KS.append(ks)

temp=pd.DataFrame({'cutoffs':cutoffs,'KS':KS})
sns.lmplot(x='cutoffs',y='KS',data=temp,fit_reg=False)
```

Out[112…
```
<seaborn.axisgrid.FacetGrid at 0x2660e28f3d0>
```



In [113…

```
ks
```

Out[113…

```
0.8095238095238095
```

In [114…

```
test_hard_classes=(test_prediction>cutoffs[KS==max(KS)][0]).astype(int)
```

In [115…

```
test_hard_classes
```

Out[115…

```
array([[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],
[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],
[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],
[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, 1],
[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],
[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],
       [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, 1],
       [0, 1],
       [0, 1]])
```

In [117…

```
pd.DataFrame(test_hard_classes).to_csv("mysubmissionelogistic.csv",index=False)
```

In [ ]: