In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import graphviz
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import ShuffleSplit
from sklearn import tree
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import precision_recall_fscore_support
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import PolynomialFeatures
from sklearn import model_selection
from sklearn.feature_selection import SelectPercentile, f_classif

import random
```

In [2]:
```
df = pd.read_csv(r'D:\ML Assignment 3\df_out.csv', index_col = 0)
pd.set_option('display.max_rows', None)
percentiles = [1, 5, 10]
```

In [3]:
```
def classifying1(x):
    if x > 1:
        return 1
    else:
        return 0
```

In [4]:
```
def preprocess_percentile(X_train, X_test, y_train, y_label, per=10):
    selector = SelectPercentile(f_classif, percentile=per)
    selector.fit(X_train, y_train)
    features_train_transformed = selector.transform(X_train)
    features_test_transformed  = selector.transform(X_test)

    return features_train_transformed, features_test_transformed, y_train, y_label
```

In [5]:
```
df["classes"] = df['2015 PRICE VAR [%]'].apply(classifying1)
df.corrwith(df["2015 PRICE VAR [%]"]).sort_values(ascending = False)
```

Out[5]:
```
2015 PRICE VAR [%]                1.000000
Class                             0.652077
classes                           0.651469
EPS Diluted                       0.118955
EPS                               0.117757
Free Cash Flow per Share          0.113474
freeCashFlowPerShare              0.112033
Gross Margin                      0.097719
Capex per Share                   0.091541
Net Income per Share              0.090849
ROE                               0.075625
returnOnEquity                    0.074910
Capital Expenditure               0.066873
Earnings Yield                    0.059948
Capex to Depreciation             0.049635
Earnings Before Tax Margin        0.048110
Free Cash Flow margin             0.046746
Net Profit Margin                 0.046680
EBIT Growth                       0.045174
EBIT Margin                       0.044077
Profit Margin                     0.035585
Investing Cash flow               0.030782
EBITDA Margin                     0.024552
Operating Income Growth           0.022982
Free Cash Flow                    0.019073
SG&A Expense                      0.018816
Net cash flow / Change in cash    0.017040
Operating Cash Flow per Share     0.010894
assetTurnover                     0.007601
operatingCashFlowPerShare         0.006973
Income Quality                    0.002378
Net Income Com                    0.001504
Gross Profit                      0.001137
Operating Expenses                0.000830
Net Income                        0.000499
Cash and cash equivalents        -0.000372
Earnings before Tax              -0.001591
Financing Cash Flow              -0.001594
Consolidated Income              -0.002536
Retained earnings (deficit)      -0.002904
Revenue                          -0.003669
SG&A to Revenue                  -0.007272
Weighted Average Shs Out         -0.007875
Operating Income                 -0.009540
priceToSalesRatio                -0.010414
Total liabilities                -0.010831
EBIT                             -0.011131
Price to Sales Ratio             -0.011966
Gross Profit Growth              -0.012322
Weighted Average Shs Out (Dil)   -0.012719
Total assets                     -0.013728
EBITDA                           -0.018143
Operating Cash Flow              -0.019857
Total shareholders equity        -0.021287
Tangible Asset Value             -0.022068
Revenue per Share                -0.029813
Invested Capital                 -0.032335
Tangible Book Value per Share    -0.037217
Depreciation & Amortization      -0.048609
cashPerShare                     -0.052292
Cash per Share                   -0.052292
Shareholders Equity per Share    -0.056133
Property, Plant & Equipment Net  -0.059593
Book Value per Share             -0.097445
dtype: float64
```

In [6]:
```
df = df.drop(columns=['2015 PRICE VAR [%]','Class', 'Sector'])
nparray = df.to_numpy()
scores = ["recall",'precision','accuracy','f1']
```

In [7]:
```
features = nparray[:,0:-1]
label = nparray[:,-1]

X = features
y = label

features.shape
```

Out[7]:
```
(3788, 61)
```

In [8]:
```
resultsDF = pd.DataFrame([], columns = ['Classifier','Precision','Recall','Fscore', 'Train score', 'Test score']).set_
index('Classifier')
```

SVC Linear

In [17]:
```
resultsDF = pd.DataFrame([], columns = ['Classifier','Precision','Recall','Fscore', 'Train score', 'Test score']).set_
index('Classifier')
for i in range (0,3):
    for p in percentiles:
        for score in scores:
            X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.2)
            X_train, X_test, y_train, y_test = preprocess_percentile(X_train, X_test, y_train, y_test, p)
            param_grid = {'C': [0.1,0.5,0.7, 0.9,1,2,10,15,20], 'dual': [False]}
            SVC_GS = GridSearchCV(LinearSVC(),scoring = score, param_grid = param_grid , cv = 30,refit=True,verbose=1,
n_jobs=-1)
            SVC_GS.fit(X_train,y_train)
            y_pred = SVC_GS.predict(X_test)
            resultsSVM1 = list(precision_recall_fscore_support(y_test, y_pred, average='macro'))
            resultsSVM1.insert(0,'SVMLinear RUN ' + str(i+1) + " percentile=" + str(p) +   " With Scoring method " + s
core + " : ")
            resultsSVM1.pop(4)
            resultsSVM1.insert(4, SVC_GS.score(X_train, y_train))
            resultsSVM1.insert(5, SVC_GS.score(X_test, y_test))


            SVM1_dataframe = pd.DataFrame([resultsSVM1], columns = ['Classifier','Precision','Recall','Fscore', 'Train
score', 'Test score']).set_index('Classifier')

            resultsDF = resultsDF.append([SVM1_dataframe])
            print("The best estimator for RUN " + str(i+1) + " percentile=" + str(p) + " "+  " With Scoring method " +
score + " : "  + str(SVC_GS.best_estimator_))
            print("The Confusion matrix for RUN" + str(i+1) + "percentile=" + str(p) +  " With Scoring method " + scor
e + " : "   + " is \n")
            print(print(multilabel_confusion_matrix(y_test, y_pred)))
```

```
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1  With Scoring method recall : LinearSVC(C=2, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=1 With Scoring method recall :  is

[[[ 73 239]
  [ 78 368]]

 [[368  78]
  [239  73]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1  With Scoring method precision : LinearSVC(C=0.5, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=1 With Scoring method precision :  is

[[[ 80 262]
  [ 41 375]]

 [[375  41]
  [262  80]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=1 With Scoring method accuracy :  is

[[[ 78 218]
  [ 77 385]]

 [[385  77]
  [218  78]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1  With Scoring method f1 : LinearSVC(C=0.1, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=1 With Scoring method f1 :  is

[[[ 72 256]
  [ 58 372]]

 [[372  58]
  [256  72]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 1 percentile=5  With Scoring method recall : LinearSVC(C=10, class_weight=None, dual=Fals
e, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=5 With Scoring method recall :  is

[[[ 75 246]
  [ 56 381]]

 [[381  56]
  [246  75]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=5 With Scoring method precision :  is

[[[ 71 238]
  [ 46 403]]

 [[403  46]
  [238  71]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=5 With Scoring method accuracy :  is

[[[ 60 247]
  [ 40 411]]

 [[411  40]
  [247  60]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5  With Scoring method f1 : LinearSVC(C=0.7, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=5 With Scoring method f1 :  is

[[[ 74 217]
  [ 72 395]]

 [[395  72]
  [217  74]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 1 percentile=10  With Scoring method recall : LinearSVC(C=0.5, class_weight=None, dual=Fal
se, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=10 With Scoring method recall :  is

[[[ 72 238]
  [ 45 403]]

 [[403  45]
  [238  72]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=10 With Scoring method precision :  is

[[[ 77 223]
  [ 65 393]]

 [[393  65]
  [223  77]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=10 With Scoring method accuracy :  is

[[[ 78 218]
  [ 58 404]]

 [[404  58]
  [218  78]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10  With Scoring method f1 : LinearSVC(C=0.5, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN1percentile=10 With Scoring method f1 :  is

[[[ 70 230]
  [ 46 412]]

 [[412  46]
  [230  70]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=1  With Scoring method recall : LinearSVC(C=0.1, class_weight=None, dual=Fals
e, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=1 With Scoring method recall :  is

[[[ 74 250]
  [ 70 364]]

 [[364  70]
  [250  74]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=1  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=1 With Scoring method precision :  is

[[[ 80 225]
  [ 69 384]]

 [[384  69]
  [225  80]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=1  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=1 With Scoring method accuracy :  is

[[[ 69 247]
  [ 69 373]]

 [[373  69]
  [247  69]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=1  With Scoring method f1 : LinearSVC(C=0.7, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=1 With Scoring method f1 :  is

[[[ 78 222]
  [ 56 402]]

 [[402  56]
  [222  78]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=5  With Scoring method recall : LinearSVC(C=0.5, class_weight=None, dual=Fals
e, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=5 With Scoring method recall :  is

[[[ 66 256]
  [ 50 386]]

 [[386  50]
  [256  66]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5  With Scoring method precision : LinearSVC(C=10, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=5 With Scoring method precision :  is

[[[ 80 234]
  [ 69 375]]

 [[375  69]
  [234  80]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=5 With Scoring method accuracy :  is

[[[ 83 223]
  [ 74 378]]

 [[378  74]
  [223  83]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5  With Scoring method f1 : LinearSVC(C=0.7, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=5 With Scoring method f1 :  is

[[[ 77 255]
  [ 46 380]]

 [[380  46]
  [255  77]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=10  With Scoring method recall : LinearSVC(C=0.5, class_weight=None, dual=Fal
se, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=10 With Scoring method recall :  is

[[[ 77 234]
  [ 44 403]]

 [[403  44]
  [234  77]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=10 With Scoring method precision :  is

[[[ 74 242]
  [ 59 383]]

 [[383  59]
  [242  74]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=10 With Scoring method accuracy :  is

[[[ 61 264]
  [ 41 392]]

 [[392  41]
  [264  61]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10  With Scoring method f1 : LinearSVC(C=0.5, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN2percentile=10 With Scoring method f1 :  is

[[[ 74 240]
  [ 52 392]]

 [[392  52]
  [240  74]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 3 percentile=1  With Scoring method recall : LinearSVC(C=0.9, class_weight=None, dual=Fals
e, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=1 With Scoring method recall :  is

[[[ 75 212]
  [ 80 391]]

 [[391  80]
  [212  75]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=1  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=1 With Scoring method precision :  is

[[[ 76 220]
  [ 58 404]]

 [[404  58]
  [220  76]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=1  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=1 With Scoring method accuracy :  is

[[[ 70 241]
  [ 62 385]]

 [[385  62]
  [241  70]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=1  With Scoring method f1 : LinearSVC(C=0.7, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=1 With Scoring method f1 :  is

[[[ 75 246]
  [ 64 373]]

 [[373  64]
  [246  75]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 3 percentile=5  With Scoring method recall : LinearSVC(C=0.5, class_weight=None, dual=Fals
e, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=5 With Scoring method recall :  is

[[[ 64 249]
  [ 53 392]]

 [[392  53]
  [249  64]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=5 With Scoring method precision :  is

[[[ 60 257]
  [ 50 391]]

 [[391  50]
  [257  60]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=Fa
lse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=5 With Scoring method accuracy :  is

[[[ 70 240]
  [ 67 381]]

 [[381  67]
  [240  70]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5  With Scoring method f1 : LinearSVC(C=10, class_weight=None, dual=False, fi
t_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=5 With Scoring method f1 :  is

[[[ 81 213]
  [ 70 394]]

 [[394  70]
  [213  81]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 3 percentile=10  With Scoring method recall : LinearSVC(C=0.7, class_weight=None, dual=Fal
se, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=10 With Scoring method recall :  is

[[[ 72 224]
  [ 83 379]]

 [[379  83]
  [224  72]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10  With Scoring method precision : LinearSVC(C=0.1, class_weight=None, dual=
False, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=10 With Scoring method precision :  is

[[[ 78 228]
  [ 61 391]]

 [[391  61]
  [228  78]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10  With Scoring method accuracy : LinearSVC(C=0.1, class_weight=None, dual=F
alse, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=10 With Scoring method accuracy :  is

[[[ 76 224]
  [ 54 404]]

 [[404  54]
  [224  76]]]
None
Fitting 30 folds for each of 9 candidates, totalling 270 fits
The best estimator for RUN 3 percentile=10  With Scoring method f1 : LinearSVC(C=10, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
The Confusion matrix for RUN3percentile=10 With Scoring method f1 :  is

[[[ 77 205]
  [ 68 408]]

 [[408  68]
  [205  77]]]
None

[Parallel(n_jobs=-1)]: Done 270 out of 270 | elapsed:    0.2s finished
```

In [18]:
```
print('The parameters combination that would give best accuracy is : ')
print(SVC_GS.best_params_)
```

```
The parameters combination that would give best accuracy is :
{'C': 10, 'dual': False}
```

In [19]:
```
resultsDF
```

Out[19]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656

SVM Non-Linear

In [21]:
```
for i in range (0,1):
    for p in percentiles:
        for score in scores:
            X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.2)
            X_train, X_test, y_train, y_test = preprocess_percentile(X_train, X_test, y_train, y_test, p)
            param_grid = {'C': [0.5,0.9,1,2,10,15,20], 'degree' : [2,3] , 'gamma' : ['scale'], 'kernel' : ['poly'], 'c
oef0': [1]}
            SVM_NonLinear_GS = GridSearchCV(SVC(),scoring = score,param_grid = param_grid , cv = 2,refit=True,verbose=
1,n_jobs = -1)
            SVM_NonLinear_GS.get_params().keys()
            SVM_NonLinear_GS.fit(X_train,y_train)
            y_pred = SVM_NonLinear_GS.predict(X_test)
            resultsSVM2 = list(precision_recall_fscore_support(y_test, y_pred, average='macro',labels=np.unique(y_pred
)))
            resultsSVM2.insert(0,'SVM NON Linear RUN ' + str(i+1) + " percentile=" + str(p) + " With Scoring method "
+ score + " : ")
            resultsSVM2.pop(4)
            resultsSVM2.insert(4, SVM_NonLinear_GS.score(X_train, y_train))
            resultsSVM2.insert(5, SVM_NonLinear_GS.score(X_test, y_test))


            SVM2_dataframe = pd.DataFrame([resultsSVM2], columns = ['Classifier','Precision','Recall','Fscore', 'Train
score', 'Test score']).set_index('Classifier')

            resultsDF = resultsDF.append([SVM2_dataframe])
            print("The best estimator for RUN " + str(i+1) + " percentile=" + str(p) +   " With Scoring method " + sco
re + " : " + str(SVM_NonLinear_GS.best_estimator_))
            print("The Confusion matrix for RUN" + str(i+1) + "percentile=" + str(p) +   " With Scoring method " + sco
re + " : " + " is \n")
             print(print(multilabel_confusion_matrix(y_test, y_pred)))
```

```
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    8.1s finished

The best estimator for RUN 1 percentile=1 With Scoring method recall : SVC(C=0.5, break_ties=False, cache_size=200, c
lass_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=1 With Scoring method recall :  is

[[[ 95 231]
  [ 63 369]]

 [[369  63]
  [231  95]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:   22.0s finished

The best estimator for RUN 1 percentile=1 With Scoring method precision : SVC(C=0.5, break_ties=False, cache_size=20
0, class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=1 With Scoring method precision :  is

[[[ 85 250]
  [ 64 359]]

 [[359  64]
  [250  85]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    4.9s finished

The best estimator for RUN 1 percentile=1 With Scoring method accuracy : SVC(C=0.5, break_ties=False, cache_size=200,
class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=1 With Scoring method accuracy :  is

[[[ 80 233]
  [ 74 371]]

 [[371  74]
  [233  80]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    4.5s finished

The best estimator for RUN 1 percentile=1 With Scoring method f1 : SVC(C=15, break_ties=False, cache_size=200, class_
weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=1 With Scoring method f1 :  is

[[[ 92 235]
  [ 67 364]]

 [[364  67]
  [235  92]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:   12.3s finished
```

```
The best estimator for RUN 1 percentile=5 With Scoring method recall : SVC(C=0.5, break_ties=False, cache_size=200, c
lass_weight=None, coef0=1,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=5 With Scoring method recall :  is

[[[  0 335]
  [  0 423]]

 [[423   0]
  [335   0]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    3.0s finished

The best estimator for RUN 1 percentile=5 With Scoring method precision : SVC(C=20, break_ties=False, cache_size=200,
class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=5 With Scoring method precision :  is

[[[  1 296]
  [  0 461]]

 [[461   0]
  [296   1]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    4.6s finished

The best estimator for RUN 1 percentile=5 With Scoring method accuracy : SVC(C=0.5, break_ties=False, cache_size=200,
class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=5 With Scoring method accuracy :  is

[[[  0 308]
  [  0 450]]

 [[450   0]
  [308   0]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    5.0s finished

The best estimator for RUN 1 percentile=5 With Scoring method f1 : SVC(C=20, break_ties=False, cache_size=200, class_
weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=5 With Scoring method f1 :  is

[[[ 14 270]
  [ 16 458]]

 [[458  16]
  [270  14]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:   12.9s finished

The best estimator for RUN 1 percentile=10 With Scoring method recall : SVC(C=20, break_ties=False, cache_size=200, c
lass_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=10 With Scoring method recall :  is

[[[  0 304]
  [  0 454]]

 [[454   0]
  [304   0]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    2.4s finished
```

```
The best estimator for RUN 1 percentile=10 With Scoring method precision : SVC(C=1, break_ties=False, cache_size=200,
class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=10 With Scoring method precision :  is

[[[  1 309]
  [  1 447]]

 [[447   1]
  [309   1]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    3.0s finished

The best estimator for RUN 1 percentile=10 With Scoring method accuracy : SVC(C=20, break_ties=False, cache_size=200,
class_weight=None, coef0=1,
    decision_function_shape='ovr', degree=2, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=10 With Scoring method accuracy :  is

[[[  0 294]
  [  0 464]]

 [[464   0]
  [294   0]]]
None
Fitting 2 folds for each of 14 candidates, totalling 28 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 out of  28 | elapsed:    4.1s finished

The best estimator for RUN 1 percentile=10 With Scoring method f1 : SVC(C=20, break_ties=False, cache_size=200, class
_weight=None, coef0=1,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
The Confusion matrix for RUN1percentile=10 With Scoring method f1 :  is

[[[  0 324]
  [  0 434]]

 [[434   0]
  [324   0]]]
None
```

In [22]:
```
print('The parameters combination that would give best accuracy is : ')
print(SVM_NonLinear_GS.best_params_)
```

```
The parameters combination that would give best accuracy is :
{'C': 20, 'coef0': 1, 'degree': 3, 'gamma': 'scale', 'kernel': 'poly'}
```

In [23]:
```
resultsDF
```

Out[23]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000

KNN

In [31]:
```
for i in range (0,3):
    for p in percentiles:
        for score in scores:
            X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.2)
            X_train, X_test, y_train, y_test = preprocess_percentile(X_train, X_test, y_train, y_test, p)
            print(X_train)
            param_grid = {'n_neighbors': [3,5,10,15,50], 'n_jobs' : [-1],}
            KNN_GS = GridSearchCV(KNeighborsClassifier(),scoring = score,param_grid = param_grid , cv = 30,refit=True,
verbose=1, n_jobs=-1)
            KNN_GS.fit(X_train,y_train)
            y_pred = KNN_GS.predict(X_test)
            resultsKNN = list(precision_recall_fscore_support(y_test, y_pred, average='macro'))
            resultsKNN.insert(0,'KNN RUN ' + str(i+1) + " percentile=" + str(p) + " With Scoring method " + score)
            resultsKNN.pop(4)
            resultsKNN.insert(4, KNN_GS.score(X_train, y_train))
            resultsKNN.insert(5, KNN_GS.score(X_test, y_test))


            KNN_dataframe = pd.DataFrame([resultsKNN], columns = ['Classifier','Precision','Recall','Fscore', 'Train s
core', 'Test score']).set_index('Classifier')

            resultsDF = resultsDF.append([KNN_dataframe])
            print("The best estimator for RUN " + str(i+1) + " percentile=" + str(p) + " With Scoring method " + score
+ str(KNN_GS.best_estimator_))
            print("The Confusion matrix for RUN" + str(i+1) + "percentile=" + str(p) + " With Scoring method " + score
+ " is \n")
            print(print(multilabel_confusion_matrix(y_test, y_pred)))
```

```
[[0.4311    ]
 [1.        ]
 [1.        ]
 ...
 [0.7909    ]
 [1.        ]
 [0.86201867]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=1 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=1 With Scoring method recall is

[[[145 153]
  [163 297]]

 [[297 163]
  [153 145]]]
None
[[0.5551]
 [1.    ]
 [0.3512]
 ...
 [1.    ]
 [1.    ]
 [0.8866]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=1 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=1 With Scoring method precision is

[[[ 16 305]
  [ 17 420]]

 [[420  17]
  [305  16]]]
None
[[0.8733]
 [0.    ]
 [0.8137]
 ...
 [0.5375]
 [0.    ]
 [0.6433]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=1 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=1 With Scoring method accuracy is

[[[ 76 230]
  [ 88 364]]

 [[364  88]
  [230  76]]]
None
[[1.    ]
 [0.4831]
 [0.5845]
 ...
 [0.3615]
 [0.7968]
 [0.6739]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 1 percentile=1 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=1 With Scoring method f1 is

[[[ 77 240]
  [111 330]]

 [[330 111]
  [240  77]]]
None
[[ 1.72    1.72    0.6166]
 [ 1.77    1.74    0.5154]
 [ 7.4649  7.317   0.3343]
 ...
 [-3.6    -3.6     1.    ]
 [ 2.45    2.39    0.5906]
 [ 0.62    0.62    1.    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=5 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=5 With Scoring method recall is

[[[125 200]
  [128 305]]

 [[305 128]
  [200 125]]]
None
[[ 0.2979  1.184   0.0541]
 [ 0.1917  3.001   0.1044]
 [ 0.917  -8.389   0.0399]
 ...
 [ 0.7834  0.939   0.1459]
 [ 0.058   0.428  -0.0888]
 [ 1.      1.766   0.0673]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=5 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=5 With Scoring method precision is

[[[ 99 205]
  [ 80 374]]

 [[374  80]
  [205  99]]]
None
[[ 1.67        1.65        0.3496    ]
 [ 6.37        6.27        0.3725    ]
 [ 0.03        0.03        0.1601    ]
 ...
 [ 2.51        2.49        0.1295    ]
 [-3.9        -3.9         0.29268293]
 [ 3.9         3.88        0.3918    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 1 percentile=5 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=5 With Scoring method accuracy is

[[[ 76 243]
  [ 54 385]]

 [[385  54]
  [243  76]]]
None
[[ 2.95    2.85    0.3243]
 [-0.68   -0.68    0.    ]
 [ 1.      0.99    0.6321]
 ...
 [ 5.27    5.25    0.7183]
 [-0.52   -0.52    0.1577]
 [ 3.57    3.52    0.4941]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=5 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=5 With Scoring method f1 is

[[[135 199]
  [125 299]]

 [[299 125]
  [199 135]]]
None
[[ 7.4649e+00  7.3170e+00  6.4700e-02 -2.3100e-01 -2.3100e-01  1.2840e-01]
 [ 5.8000e-01  5.2000e-01  1.0000e+00 -3.9300e-01 -3.9300e-01  4.3800e-02]
 [ 1.8800e+00  1.8600e+00  4.9010e-01  3.0670e+00  3.0670e+00  3.9300e-02]
 ...
 [ 4.0000e-02  4.0000e-02  2.1810e-01  5.4000e-02  5.4000e-02  5.0000e-03]
 [-1.0000e-02 -1.0000e-02  1.4530e-01  2.1000e-02  2.1000e-02 -4.1000e-03]
 [ 1.0499e+00  1.0413e+00  8.1830e-01 -1.6436e+00 -1.6436e+00  4.5300e-02]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=10 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size
=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=10 With Scoring method recall is

[[[136 165]
  [157 300]]

 [[300 157]
  [165 136]]]
None
[[-1.45100e+01 -1.45100e+01  0.00000e+00 -1.06127e+01 -1.06127e+01
  -7.30328e-01]
 [-1.26000e+00 -1.26000e+00  1.44400e-01  1.14620e+00  1.14620e+00
  -2.04000e-02]
 [ 3.69000e+00  3.69000e+00  5.35900e-01  3.27400e+00  3.27400e+00
   4.46000e-02]
 ...
 [ 1.67000e+00  1.65000e+00  3.49600e-01  1.41200e+00  1.41200e+00
   3.19000e-02]
 [ 2.02000e+00  2.01000e+00  3.71000e-01  4.00900e+00  4.00900e+00
   1.39000e-02]
 [-7.60000e-01 -7.60000e-01  0.00000e+00 -7.85000e-01 -7.85000e-01
  -2.21600e-01]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 1 percentile=10 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_s
ize=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=10 With Scoring method precision is

[[[ 96 208]
  [ 78 376]]

 [[376  78]
  [208  96]]]
None
[[ 4.52    0.8188  6.222   6.222   0.0248 -2.5636]
 [ 0.6     0.5689  0.592   0.592   0.0297 -0.7318]
 [ 2.73    0.3963  2.9328  2.9328  0.0403 -1.3167]
 ...
 [ 1.37    1.      1.488   1.488   0.0677 -0.3451]
 [ 1.31    1.      2.205   2.205   0.0814 -0.0597]
 [ 0.33    0.4896  0.324   0.324   0.0776 -0.0789]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=10 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=10 With Scoring method accuracy is

[[[ 96 218]
  [ 82 362]]

 [[362  82]
  [218  96]]]
None
[[  2.47     2.47     0.4545   2.543    2.543    0.0737]
 [  2.75     2.75     0.5665   0.386    0.386    0.0377]
 [  1.32     1.32     0.0907   0.5122   0.5122   0.0832]
 ...
 [  0.35     0.35     0.4873 -13.064  -13.064    0.0488]
 [  2.04     1.96     0.4251   2.497    2.497    0.0404]
 [ -0.2324  -0.2324   0.      -0.838   -0.838   -0.0308]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 1 percentile=10 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN1percentile=10 With Scoring method f1 is

[[[121 180]
  [140 317]]

 [[317 140]
  [180 121]]]
None
[[0.    ]
 [0.4326]
 [0.    ]
 ...
 [0.9325]
 [0.2945]
 [0.4951]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 2 percentile=1 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=1 With Scoring method recall is

[[[ 80 219]
  [178 281]]

 [[281 178]
  [219  80]]]
None
[[1.    ]
 [1.    ]
 [0.168 ]
 ...
 [0.6157]
 [0.    ]
 [0.3917]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=1 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=1 With Scoring method precision is

[[[ 85 234]
  [ 58 381]]

 [[381  58]
  [234  85]]]
None
[[0.4698]
 [0.5606]
 [0.2088]
 ...
 [0.1618]
 [0.6374]
 [0.326 ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=1 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=1 With Scoring method accuracy is

[[[  9 309]
  [ 12 428]]

 [[428  12]
  [309   9]]]
None
[[0.2823]
 [0.8733]
 [0.6486]
 ...
 [1.    ]
 [0.6198]
 [1.    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 2 percentile=1 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=1 With Scoring method f1 is

[[[ 77 230]
  [118 333]]

 [[333 118]
  [230  77]]]
None
[[ 1.74        0.9718      0.1432    ]
 [ 2.4927      0.2153      0.166428  ]
 [-0.15        0.31818182  0.00760495]
 ...
 [-0.42        0.8578     -0.0271    ]
 [ 2.23        0.2523      0.0868    ]
 [ 1.51        0.3374      0.0252    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=5 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=5 With Scoring method recall is

[[[136 175]
  [134 313]]

 [[313 134]
  [175 136]]]
None
[[ 0.1502      0.349       0.349     ]
 [ 1.          1.188       1.188     ]
 [ 0.61074832  0.47931741  0.50475737]
 ...
 [ 0.1403      0.777       0.777     ]
 [ 0.2226      0.02        0.02      ]
 [ 0.3836     -0.209      -0.209     ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=5 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=5 With Scoring method precision is

[[[ 88 217]
  [ 56 397]]

 [[397  56]
  [217  88]]]
None
[[ 0.1977  2.236   2.236 ]
 [ 0.     -0.574  -0.574 ]
 [ 1.      1.864   1.864 ]
 ...
 [ 0.4646  1.7534  1.7534]
 [ 0.3183 -0.304  -0.304 ]
 [ 0.0728  1.166   1.166 ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 2 percentile=5 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=5 With Scoring method accuracy is

[[[ 82 254]
  [ 48 374]]

 [[374  48]
  [254  82]]]
None
[[-0.37    0.3353 -0.0118]
 [ 0.56    0.75    0.0391]
 [-1.182   1.     -0.0548]
 ...
 [ 0.933   0.371   0.0343]
 [-0.47    0.5875 -0.0354]
 [ 1.04    0.3769  0.0373]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=5 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=5 With Scoring method f1 is

[[[116 190]
  [134 318]]

 [[318 134]
  [190 116]]]
None
[[ 3.18        3.12        0.8579      3.943       3.943       0.0388    ]
 [ 1.3         1.3         0.3486      0.47931741  0.50475737  0.1313    ]
 [-0.1043     -0.1043      0.3261     -0.112      -0.112      -0.0115    ]
 ...
 [-0.65       -0.65        0.0765     -2.281      -2.281      -0.1094    ]
 [ 0.22        0.22        1.          1.85        1.85        0.0147    ]
 [-0.03       -0.03        0.86201867  1.47151075  1.45692504  0.05746677]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=10 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size
=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=10 With Scoring method recall is

[[[144 184]
  [141 289]]

 [[289 141]
  [184 144]]]
None
[[  0.43       0.43       0.364     -0.969     -0.969      0.0491  ]
 [  0.56       0.55       0.3331     1.574      1.574      0.035   ]
 [  0.2644     0.2644     0.223      7.858      7.858      0.0523  ]
 ...
 [-11.59     -11.59       0.        -0.103     -0.103     -0.730328]
 [ -4.2       -4.2        0.3691    -0.8921    -0.8921    -0.1321  ]
 [  0.79       0.78       0.3889     3.224      3.224      0.0161  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 2 percentile=10 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_s
ize=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=10 With Scoring method precision is

[[[102 194]
  [ 72 390]]

 [[390  72]
  [194 102]]]
None
[[ 4.0100e+00  3.9300e+00  4.6230e-01  3.4410e+00  3.4410e+00  3.8400e-02]
 [ 1.2700e+00  1.2600e+00  4.0740e-01  1.7730e+00  1.7730e+00  4.0700e-02]
 [ 2.0500e+00  2.0400e+00  3.7760e-01  2.9910e+00  2.9910e+00  2.8100e-02]
 ...
 [ 1.1100e+00  1.1000e+00  1.0000e+00  2.2250e+00  2.2250e+00  5.6800e-02]
 [ 1.5835e+00  1.5835e+00  5.4280e-01  1.4880e+00  1.4880e+00  4.9800e-02]
 [ 1.2000e-01  1.2000e-01  1.0000e+00 -2.4540e+00 -2.4540e+00  2.4000e-03]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=10 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=10 With Scoring method accuracy is

[[[ 89 226]
  [ 62 381]]

 [[381  62]
  [226  89]]]
None
[[ 8.35400000e-01  8.35400000e-01  5.06400000e-01  4.55000000e-01
   4.55000000e-01  4.92000000e-02]
 [-2.60000000e+00 -2.60000000e+00  1.69900000e-01 -1.83400000e+00
  -1.83400000e+00 -6.70000000e-02]
 [-1.75960000e+01 -1.77400000e+01  1.71200000e-01 -1.78600000e+00
  -1.78600000e+00 -2.63500000e-01]
 ...
 [ 1.68000000e+00  1.66000000e+00  1.13700000e-01  1.74850000e+00
   1.74850000e+00  6.70000000e-02]
 [ 2.28000000e+00  2.25000000e+00  6.96600000e-01  3.66000000e+00
   3.66000000e+00  6.53000000e-02]
 [-1.32000000e+00 -1.32000000e+00  6.80272109e-01  1.09411807e+00
   1.09826813e+00  4.78473469e-03]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 2 percentile=10 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN2percentile=10 With Scoring method f1 is

[[[119 193]
  [139 307]]

 [[307 139]
  [193 119]]]
None
[[0.5357]
 [0.6157]
 [0.2857]
 ...
 [0.0713]
 [0.4717]
 [0.4795]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 3 percentile=1 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=1 With Scoring method recall is

[[[147 162]
  [169 280]]

 [[280 169]
  [162 147]]]
None
[[0.1148]
 [0.9851]
 [0.2962]
 ...
 [1.    ]
 [0.7516]
 [0.1497]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=1 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=1 With Scoring method precision is

[[[ 87 256]
  [ 66 349]]

 [[349  66]
  [256  87]]]
None
[[0.        ]
 [0.5357    ]
 [0.1287    ]
 ...
 [0.40909091]
 [0.3862    ]
 [0.8654    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=1 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=1 With Scoring method accuracy is

[[[108 192]
  [ 82 376]]

 [[376  82]
  [192 108]]]
None
[[0.8033]
 [0.4313]
 [0.2364]
 ...
 [0.7621]
 [0.3289]
 [1.    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 3 percentile=1 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=1 With Scoring method f1 is

[[[167 143]
  [191 257]]

 [[257 191]
  [143 167]]]
None
[[ 0.2477   5.0991   5.0991 ]
 [ 0.5465   0.83     0.83   ]
 [ 0.1885  -1.323   -1.323  ]
 ...
 [ 0.6479  -2.46    -2.46   ]
 [ 0.2072   1.5015   1.5015 ]
 [ 0.1432  11.13476 11.551  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=5 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size=
30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=5 With Scoring method recall is

[[[132 179]
  [132 315]]

 [[315 132]
  [179 132]]]
None
[[ 1.        -1.4819033 -1.4797995]
 [ 0.3325     1.351      1.351    ]
 [ 0.5229     0.188      0.188    ]
 ...
 [ 0.6763     1.3738     1.3738   ]
 [ 1.         1.202      1.202    ]
 [ 0.7788     4.004      4.004    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=5 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=5 With Scoring method precision is

[[[ 76 218]
  [ 48 416]]

 [[416  48]
  [218  76]]]
None
[[ 1.      5.982   5.982 ]
 [ 0.382  -0.175  -0.175 ]
 [ 1.     -2.4388 -2.4388]
 ...
 [ 0.5906  1.342   1.342 ]
 [ 0.7145 -2.568  -2.568 ]
 [ 0.5946  0.1196  0.1196]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 3 percentile=5 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_siz
e=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=5 With Scoring method accuracy is

[[[ 89 231]
  [ 71 367]]

 [[367  71]
  [231  89]]]
None
[[  5.06     5.03     0.801 ]
 [  0.35     0.34     0.3186]
 [  7.4649   7.317    0.7801]
 ...
 [  0.36     0.36     0.2791]
 [-17.596  -17.74     0.    ]
 [  0.25     0.25     0.4067]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=5 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=5 With Scoring method f1 is

[[[132 156]
  [142 328]]

 [[328 142]
  [156 132]]]
None
[[ 7.4649      7.317       0.1047     11.13476    11.551       0.0742    ]
 [ 1.52        1.47        0.121       2.132       2.132       0.0552    ]
 [-0.32       -0.32        0.54271357  0.47931741  0.50475737 -0.03911515]
 ...
 [-0.75       -0.75        0.3426     -1.343      -1.343      -0.1521    ]
 [ 1.02        1.          1.          2.127       2.127       0.042     ]
 [-0.06       -0.06        0.4252      0.056       0.056      -0.0335    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=10 With Scoring method recallKNeighborsClassifier(algorithm='auto', leaf_size
=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=3, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=10 With Scoring method recall is

[[[134 174]
  [166 284]]

 [[284 166]
  [174 134]]]
None
[[-2.6000000e-01 -2.6000000e-01  5.9010000e-01  7.0700000e-01
   7.0700000e-01 -1.5200000e-02]
 [ 4.9800000e+00  4.6800000e+00  7.2340000e-01  4.3770000e+00
   4.3770000e+00  8.1900000e-02]
 [-2.2200000e+00 -2.2200000e+00  0.0000000e+00 -2.0089000e+00
  -2.0089000e+00 -5.4900000e-02]
 ...
 [ 7.1000000e-01  6.9000000e-01  7.1450000e-01 -2.5680000e+00
  -2.5680000e+00  7.2800000e-02]
 [-1.7596000e+01 -1.7740000e+01  1.0000000e+00 -2.2461712e+01
  -2.2617500e+01 -4.6770000e-01]
 [-1.7596000e+01 -1.7740000e+01  0.0000000e+00 -2.2461712e+01
  -2.2617500e+01 -3.6860000e-01]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished
```

```
The best estimator for RUN 3 percentile=10 With Scoring method precisionKNeighborsClassifier(algorithm='auto', leaf_s
ize=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=10 With Scoring method precision is

[[[ 93 209]
  [ 79 377]]

 [[377  79]
  [209  93]]]
None
[[ 1.41    1.39    0.5617  2.515   2.515   0.0517]
 [ 0.32    0.32    0.3625  0.241   0.241   0.022 ]
 [-0.25   -0.25    0.2435  0.0132  0.0132 -0.0215]
 ...
 [ 2.34    2.32    0.2736  3.146   3.146   0.0527]
 [ 1.1     1.1     0.6763  1.3738  1.3738  0.0639]
 [-6.44   -6.44    1.     -3.98   -3.98   -0.3102]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=10 With Scoring method accuracyKNeighborsClassifier(algorithm='auto', leaf_si
ze=30, metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=50, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=10 With Scoring method accuracy is

[[[ 92 212]
  [ 77 377]]

 [[377  77]
  [212  92]]]
None
[[  1.41         1.34         0.2085      -5.0982      -5.0982
    0.0696    ]
 [  2.97         2.93         0.5623       2.604        2.604
    0.0534    ]
 [  0.45         0.44         0.0246       0.609        0.609
    0.1002    ]
 ...
 [  0.61         0.61         0.9975     -12.0039     -12.0039
    0.0282    ]
 [  6.8          6.7          0.7796      10.852       10.852
    0.0427    ]
 [  7.4649       7.317        0.2118      -1.11877868  -0.98953043
    0.166428  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    1.4s finished

The best estimator for RUN 3 percentile=10 With Scoring method f1KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
                     metric_params=None, n_jobs=-1, n_neighbors=5, p=2,
                     weights='uniform')
The Confusion matrix for RUN3percentile=10 With Scoring method f1 is

[[[121 165]
  [135 337]]

 [[337 135]
  [165 121]]]
None
```

In [32]:
```
print('The parameters combination that would give best accuracy is : ')
print(KNN_GS.best_params_)
```

```
The parameters combination that would give best accuracy is :
{'n_jobs': -1, 'n_neighbors': 5}
```

In [33]:
```
resultsDF
```

Out[33]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494

Naive Bayes

In [34]:
```
for i in range (0,3):
    for p in percentiles:
        for score in scores:
            X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.2)
            X_train, X_test, y_train, y_test = preprocess_percentile(X_train, X_test, y_train, y_test, p)
            print(X_train)
            param_grid = {'var_smoothing': [1e-9, 2e-9, 3e-9, 1e-10,1]}
            GNB_GS = GridSearchCV(GaussianNB(),scoring = score,param_grid=param_grid , cv = 30,refit=True,verbose=1, n
_jobs=-1)
            GNB_GS.fit(X_train,y_train)
            y_pred = GNB_GS.predict(X_test)
            resultsGNB = list(precision_recall_fscore_support(y_test, y_pred, average='macro'))
            resultsGNB.insert(0,'Gaussian Naive Bayes RUN ' + str(i+1) + " percentile=" + str(p) + " With Scoring meth
od " + score)
            resultsGNB.pop(4)
            resultsGNB.insert(4, GNB_GS.score(X_train, y_train))
            resultsGNB.insert(5, GNB_GS.score(X_test, y_test))


            GNB_dataframe = pd.DataFrame([resultsGNB], columns = ['Classifier','Precision','Recall','Fscore', 'Train s
core', 'Test score']).set_index('Classifier')

            resultsDF = resultsDF.append([GNB_dataframe])
            print("The best estimator for RUN " + str(i+1) + " percentile=" + str(p) + " With Scoring method " + score
+ str(GNB_GS.best_estimator_))
            print("The Confusion matrix for RUN" + str(i+1) + "percentile=" + str(p) + " With Scoring method " + score
+ " is \n")
            print(print(multilabel_confusion_matrix(y_test, y_pred)))
```

```
[[0.5634]
 [0.6881]
 [0.5026]
 ...
 [0.1768]
 [0.4256]
 [0.1731]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=1 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=1 With Scoring method recall is

[[[ 75 224]
  [ 79 380]]

 [[380  79]
  [224  75]]]
None
[[1.    ]
 [0.    ]
 [0.2153]
 ...
 [0.2235]
 [0.5891]
 [0.4659]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=1 With Scoring method precision is

[[[ 75 235]
  [ 76 372]]

 [[372  76]
  [235  75]]]
None
[[0.3915]
 [0.1639]
 [0.6463]
 ...
 [0.1893]
 [0.5585]
 [0.5536]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=1 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=1 With Scoring method accuracy is

[[[ 74 251]
  [ 65 368]]

 [[368  65]
  [251  74]]]
None
[[1.    ]
 [1.    ]
 [0.2895]
 ...
 [0.6098]
 [0.4111]
 [0.8445]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
The best estimator for RUN 1 percentile=1 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=1 With Scoring method f1 is

[[[ 70 248]
  [ 68 372]]

 [[372  68]
  [248  70]]]
None
[[ 4.34        4.24        0.8311    ]
 [ 0.9         0.9         0.7837    ]
 [ 3.99        3.91        0.2263    ]
 ...
 [ 2.68        2.67        0.6705    ]
 [-1.02       -1.02        0.        ]
 [-0.77       -0.77        0.62680842]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=5 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=5 With Scoring method recall is

[[[236  65]
  [310 147]]

 [[147 310]
  [ 65 236]]]
None
[[ -3.9         -3.9          0.29268293]
 [-10.64       -10.64         1.        ]
 [  1.71         1.7          0.4403    ]
 ...
 [  1.33         1.31         0.1657    ]
 [  0.63         0.63         1.        ]
 [-17.596      -17.74         0.        ]]

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=5 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN1percentile=5 With Scoring method precision is

[[[160 145]
  [205 248]]

 [[248 205]
  [145 160]]]
None
[[ 3.83        0.4268      0.0547    ]
 [ 1.01        0.6985      0.0114    ]
 [-3.38260052  1.         -0.16487107]
 ...
 [ 2.66        0.3923      0.0412    ]
 [ 4.27        0.5315      0.0446    ]
 [-1.72        1.         -0.1093    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
C:\Users\shava\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
The best estimator for RUN 1 percentile=5 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN1percentile=5 With Scoring method accuracy is

[[[  0 319]
  [  0 439]]

 [[439   0]
  [319   0]]]
None
[[ 1.2     0.6807  0.0401]
 [-4.69    0.2176 -0.0751]
 [ 1.114   0.7483  0.0305]
 ...
 [ 2.25    0.8597  0.0599]
 [-0.04    1.     -0.0057]
 [ 3.      0.2477  0.0597]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=5 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=5 With Scoring method f1 is

[[[228 100]
  [265 165]]

 [[165 265]
  [100 228]]]
None
[[-2.     -2.      0.     -1.2316 -1.2316 -0.1084]
 [-0.7659 -0.7659  0.676   3.294   3.294  -0.0461]
 [ 3.1     3.08    1.      2.957   2.957   0.069 ]
 ...
 [-1.3438 -1.3438  0.2284  1.7668  1.7668 -0.0791]
 [ 0.6     0.6     0.291  -0.212  -0.212   0.0369]
 [-0.04   -0.04    0.3557 -0.658  -0.658  -0.0086]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10 With Scoring method recallGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN1percentile=10 With Scoring method recall is

[[[269  41]
  [345 103]]

 [[103 345]
  [ 41 269]]]
None
[[ 0.81       0.8        0.3038     0.688      0.688      0.0362    ]
 [ 0.4        0.4        0.5866    -1.674     -1.674      0.0395    ]
 [ 1.5146     1.4974     0.5485     1.768      1.768      0.0552    ]
 ...
 [ 1.31       1.3        1.         1.47151075 1.45692504 0.0624    ]
 [ 1.27       1.26       0.271      0.7        0.7        0.0592    ]
 [ 0.76       0.75       0.6248    -0.101     -0.101      0.0845    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
```

```
The best estimator for RUN 1 percentile=10 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=10 With Scoring method precision is

[[[274  49]
  [328 107]]

 [[107 328]
  [ 49 274]]]
None
[[-0.12     -0.12      0.3912   -0.5533   -0.5533   -0.0307  ]
 [ 0.6053    0.5932    0.3404    4.204     4.204     0.0351  ]
 [ 2.28      2.24      0.5474    3.256     3.256     0.0428  ]
 ...
 [ 1.233     1.225     0.1295    0.768     0.768     0.0323  ]
 [ 1.85      1.85      0.2003    1.968     1.968     0.166428]
 [ 3.71      3.69      1.        4.461     4.461     0.0626  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=10 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN1percentile=10 With Scoring method accuracy is

[[[268  62]
  [317 111]]

 [[111 317]
  [ 62 268]]]
None
[[ 1.6282      1.6234      0.836      -2.93886708  0.0385     -1.32733176]
 [ 2.5         2.45        0.2875      1.0005      0.0525     -3.1176    ]
 [ 1.32        1.32        1.          1.281       0.0483     -1.2222    ]
 ...
 [ 0.8         0.77        0.551       3.0025      0.0185     -1.1501    ]
 [ 0.29        0.29        0.6023      0.192       0.0738     -0.0106    ]
 [ 0.41        0.4         0.686      -0.292       0.0293     -0.1505    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 1 percentile=10 With Scoring method f1GaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN1percentile=10 With Scoring method f1 is

[[[260  45]
  [340 113]]

 [[113 340]
  [ 45 260]]]
None
[[0.3366]
 [0.355 ]
 [1.    ]
 ...
 [0.7456]
 [0.0728]
 [0.7783]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished

The best estimator for RUN 2 percentile=1 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=1 With Scoring method recall is

[[[ 82 211]
  [ 74 391]]

 [[391  74]
  [211  82]]]
None
[[0.214 ]
 [0.1408]
 [1.    ]
 ...
 [0.2814]
 [0.3964]
 [1.    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=1 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=1 With Scoring method precision is

[[[ 86 210]
  [ 64 398]]

 [[398  64]
  [210  86]]]
None
[[0.3151]
 [0.809 ]
 [1.    ]
 ...
 [0.115 ]
 [0.2513]
 [0.247 ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 2 percentile=1 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=1 With Scoring method accuracy is

[[[ 84 234]
  [ 74 366]]

 [[366  74]
  [234  84]]]
None
[[0.2743]
 [0.5546]
 [0.7752]
 ...
 [0.1375]
 [1.    ]
 [1.    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished

The best estimator for RUN 2 percentile=1 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=1 With Scoring method f1 is

[[[ 83 219]
  [ 74 382]]

 [[382  74]
  [219  83]]]
None
[[-2.2522 -2.2522  0.    ]
 [ 3.      2.99    0.3537]
 [ 0.2     0.2     0.1723]
 ...
 [ 2.78    2.7     0.4288]
 [-0.32   -0.32    0.0469]
 [ 1.8     1.8     0.7456]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 2 percentile=5 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=5 With Scoring method recall is

[[[235  57]
  [325 141]]

 [[141 325]
  [ 57 235]]]
None
[[ 1.89    1.88    0.1715]
 [ 0.08    0.08    0.5657]
 [-1.1    -1.1     0.2844]
 ...
 [ 6.06    5.45    0.8367]
 [-3.1    -3.1     0.    ]
 [-1.8    -1.8     0.4599]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=5 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=5 With Scoring method precision is

[[[245  76]
  [285 152]]

 [[152 285]
  [ 76 245]]]
None
[[1.51       0.2671     0.0614    ]
 [0.69407821 1.         0.0605    ]
 [1.43       1.         0.0328    ]
 ...
 [0.64       0.4999     0.0794    ]
 [1.80275849 0.9801     0.0675    ]
 [1.031      0.3486     0.0602    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 2 percentile=5 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN2percentile=5 With Scoring method accuracy is

[[[  0 313]
  [  0 445]]

 [[445   0]
  [313   0]]]
None
[[-0.15       -0.15        0.9209    ]
 [-0.21       -0.21        0.7158    ]
 [-1.53       -1.53        0.1133    ]
 ...
 [ 5.08        5.06        0.1952    ]
 [ 1.67        1.66        0.66097366]
 [-3.6        -3.6         0.0107    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
C:\Users\shava\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetricWarning: Precision
and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
The best estimator for RUN 2 percentile=5 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=5 With Scoring method f1 is

[[[254  67]
  [310 127]]

 [[127 310]
  [ 67 254]]]
None
[[-2.20000000e-01 -2.20000000e-01  2.87200000e-01 -1.15000000e-01
  -1.15000000e-01 -9.48000000e-02]
 [-1.35898413e-01 -1.78184806e-01  4.44444444e-01  4.79317410e-01
   5.04757374e-01 -3.91151515e-02]
 [-2.20000000e-01 -2.20000000e-01  1.76100000e-01  3.32000000e-01
   3.32000000e-01 -1.14000000e-02]
 ...
 [ 1.38000000e+00  1.37000000e+00  6.03400000e-01  1.44800000e+00
   1.44800000e+00  5.16000000e-02]
 [ 8.32200000e-01  8.26200000e-01  1.00000000e+00  1.11347600e+01
   1.15510000e+01  4.06000000e-02]
 [-1.10000000e-01 -1.10000000e-01  2.08700000e-01 -3.23000000e-01
  -3.23000000e-01 -2.53000000e-02]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 2 percentile=10 With Scoring method recallGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN2percentile=10 With Scoring method recall is

[[[255  39]
  [377  87]]

 [[ 87 377]
  [ 39 255]]]
None
[[ 1.04000e+00  1.04000e+00  2.43100e-01 -9.00000e-03 -9.00000e-03
   6.10000e-02]
 [-1.75960e+01 -1.77400e+01  6.24100e-01  1.36900e+00  1.36900e+00
  -7.30328e-01]
 [ 2.50000e-01  2.50000e-01  2.01900e-01 -3.20000e-01 -3.20000e-01
   3.80000e-02]
 ...
 [ 3.52000e+00  3.52000e+00  4.68900e-01  4.53600e+00  4.53600e+00
   6.05000e-02]
 [ 4.36000e+00  4.36000e+00  5.02100e-01 -6.58500e+00 -6.58500e+00
   1.66428e-01]
 [ 1.06000e+00  1.06000e+00  4.17300e-01 -2.50450e+00 -2.50450e+00
   3.16000e-02]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
The best estimator for RUN 2 percentile=10 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN2percentile=10 With Scoring method precision is

[[[256  34]
  [351 117]]

 [[117 351]
  [ 34 256]]]
None
[[ 8.20000e-01  8.20000e-01  8.46500e-01 -2.04250e+01 -2.04250e+01
   1.84000e-02]
 [ 1.42000e+00  1.42000e+00  3.57900e-01  3.17000e+00  3.17000e+00
   2.24000e-02]
 [ 2.07000e+00  2.02000e+00  1.00000e+00  2.11300e+00  2.11300e+00
   6.53000e-02]
 ...
 [-2.34000e+00 -2.34000e+00  2.26100e-01 -1.85340e+00 -1.85340e+00
  -6.25700e-01]
 [ 1.27000e+00  1.27000e+00  1.40800e-01  3.09700e+00  3.09700e+00
   4.20000e-02]
 [ 5.20000e+00  4.16000e+00  9.58300e-01 -8.93000e-01 -8.93000e-01
   1.66428e-01]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 2 percentile=10 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN2percentile=10 With Scoring method accuracy is

[[[247  49]
  [344 118]]

 [[118 344]
  [ 49 247]]]
None
[[ 0.98    0.95    0.3406  0.8572  0.8572  0.0216]
 [ 1.825   1.815   0.1988 -0.26   -0.26    0.0495]
 [ 1.51    1.49    0.2207 -0.23   -0.23    0.0384]
 ...
 [-0.61   -0.61    0.705  -1.165  -1.165  -0.0306]
 [ 1.73    1.725   0.3156  3.123   3.123   0.0466]
 [ 1.55    1.55    0.4114  0.621   0.621   0.0369]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10 With Scoring method f1GaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN2percentile=10 With Scoring method f1 is

[[[287  39]
  [333  99]]

 [[ 99 333]
  [ 39 287]]]
None
[[0.7291]
 [0.5754]
 [0.2146]
 ...
 [0.779 ]
 [0.5252]
 [0.384 ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=1 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=1 With Scoring method recall is

[[[ 80 210]
  [ 70 398]]

 [[398  70]
  [210  80]]]
None
[[0.2141]
 [0.2659]
 [0.3934]
 ...
 [0.9325]
 [0.7338]
 [0.    ]]

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=1 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=1 With Scoring method precision is

[[[ 86 205]
  [ 75 392]]

 [[392  75]
  [205  86]]]
None
[[0.2051]
 [0.1549]
 [0.    ]
 ...
 [0.2927]
 [0.7578]
 [0.6731]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

The best estimator for RUN 3 percentile=1 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=1 With Scoring method accuracy is

[[[ 72 232]
  [ 64 390]]

 [[390  64]
  [232  72]]]
None
[[0.8708]
 [0.3954]
 [0.8818]
 ...
 [0.1172]
 [1.    ]
 [0.5533]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=1 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=1 With Scoring method f1 is

[[[ 84 222]
  [ 66 386]]

 [[386  66]
  [222  84]]]
None
[[-5.62   -5.62    0.    ]
 [ 3.9     3.88    0.3918]
 [ 7.4649  7.317   0.5208]
 ...
 [ 0.4956  0.472   0.6681]
 [ 3.77    3.71    0.383 ]
 [ 4.27    4.24    1.    ]]

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=5 With Scoring method recallGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=5 With Scoring method recall is

[[[207 111]
  [202 238]]

 [[238 202]
  [111 207]]]
None
[[-0.09        0.5413     -0.0075    ]
 [ 0.65        0.5641      0.0725    ]
 [-6.4         0.52071638 -0.16487107]
 ...
 [ 3.59        1.          0.0913    ]
 [ 2.87        0.7927      0.0834    ]
 [ 1.24        1.          0.0315    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
```

```
The best estimator for RUN 3 percentile=5 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=5 With Scoring method precision is

[[[241  71]
  [276 170]]

 [[170 276]
  [ 71 241]]]
None
[[ 0.1205 -0.234  -0.234 ]
 [ 0.3881  1.7727  1.7727]
 [ 0.6714 -1.2971 -1.2971]
 ...
 [ 0.7685  0.532   0.532 ]
 [ 1.     -0.154  -0.154 ]
 [ 0.3847  1.71    1.71  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=5 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN3percentile=5 With Scoring method accuracy is

[[[ 48 252]
  [ 60 398]]

 [[398  60]
  [252  48]]]
None
[[  4.36        4.36        0.5021   ]
 [ -0.02       -0.02        0.3793346]
 [ -0.87       -0.87        0.6263   ]
 ...
 [ -0.21       -0.21        0.7158   ]
 [  6.06        5.45        0.8367   ]
 [-17.596     -17.74        0.9449   ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
```

```
The best estimator for RUN 3 percentile=5 With Scoring method f1GaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=5 With Scoring method f1 is

[[[228  65]
  [295 170]]

 [[170 295]
  [ 65 228]]]
None
[[-0.68       -0.68        1.           3.125        3.125       -0.1097    ]
 [-0.31       -0.31        0.4903      -0.359       -0.359       -0.189     ]
 [ 4.82        4.74        0.1967       6.182        6.182        0.0568    ]
 ...
 [ 0.42        0.41        0.56015038   0.47931741   0.50475737 -0.03911515]
 [ 3.63        3.45        1.           3.467        3.467        0.0587    ]
 [ 2.01        1.98        0.6389       2.2149       2.2149       0.0404    ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=10 With Scoring method recallGaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN3percentile=10 With Scoring method recall is

[[[273  30]
  [362  93]]

 [[ 93 362]
  [ 30 273]]]
None
[[ -1.72       -1.72        1.        -22.461712 -22.6175     -0.1093  ]
 [  2.1945     2.1601       0.7906     -1.104     -1.104       0.0448  ]
 [ -0.12       -0.12        0.3482      0.098      0.098      -0.024   ]
 ...
 [  0.37        0.37        0.5655      0.551      0.551       0.0384  ]
 [  1.475       1.46        0.2791      1.088      1.088       0.0543  ]
 [  5.11        5.1         0.25       -1.153     -1.153       0.0592  ]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=10 With Scoring method precisionGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=10 With Scoring method precision is

[[[250  46]
  [346 116]]

 [[116 346]
  [ 46 250]]]
None
[[ 1.01    1.       0.1201 -0.7822 -0.7822  0.0273]
 [-0.06   -0.06     0.5856  0.166   0.166  -0.0123]
 [ 2.07    2.06     0.6781 -3.666  -3.666   0.01  ]
 ...
 [-0.47   -0.47     0.562   0.9398  0.9398 -0.0352]
 [ 0.92    0.91     1.      2.88    2.88    0.0677]
 [ 1.42    1.39     0.4902  1.171   1.171   0.0439]]

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

Fitting 30 folds for each of 5 candidates, totalling 150 fits
The best estimator for RUN 3 percentile=10 With Scoring method accuracyGaussianNB(priors=None, var_smoothing=1e-09)
The Confusion matrix for RUN3percentile=10 With Scoring method accuracy is

[[[253  34]
  [346 125]]

 [[125 346]
  [ 34 253]]]
None
[[ 1.57    1.56    1.       0.767   0.767   0.072 ]
 [ 2.56    2.54    0.536   -1.858  -1.858   0.0797]
 [ 2.1795  2.1795  0.3791  -0.618  -0.618   0.0899]
 ...
 [ 2.3     2.3     0.754   -2.602  -2.602   0.0535]
 [ 1.77    1.74    1.       1.937   1.937   0.0611]
 [ 1.01    1.      0.5833   1.7953  1.7953  0.0226]]
Fitting 30 folds for each of 5 candidates, totalling 150 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10 With Scoring method f1GaussianNB(priors=None, var_smoothing=1)
The Confusion matrix for RUN3percentile=10 With Scoring method f1 is

[[[261  32]
  [384  81]]

 [[ 81 384]
  [ 32 261]]]
None
```

```
[Parallel(n_jobs=-1)]: Done 150 out of 150 | elapsed:    0.1s finished
```

In [35]:
```
print('The parameters combination that would give best accuracy is : ')
print(GNB_GS.best_params_)
```

```
The parameters combination that would give best accuracy is :
{'var_smoothing': 1}
```

In [36]:
```
resultsDF
```

Out[36]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503

Decision Tree Classifier

In [37]:
```
for i in range (0,3):
    for p in percentiles:
        for score in scores:
            X_train, X_test, y_train, y_test = train_test_split(features, label, test_size=0.2)
            X_train, X_test, y_train, y_test = preprocess_percentile(X_train, X_test, y_train, y_test, p)
            tree_para = {'criterion':['gini','entropy'],'max_leaf_nodes':[4,5,6,7,8,9,10,11,12,15,20,30,40,50,70], 'ma
x_depth':[5,10,15,20,30]}
            DTC_GS = GridSearchCV(DecisionTreeClassifier(), scoring = score, param_grid = tree_para, cv=10, return_tra
in_score = True, verbose = 1, n_jobs = -1)
            DTC_GS.fit(X_train,y_train)

            y_pred = DTC_GS.predict(X_test)
            results = list(precision_recall_fscore_support(y_test, y_pred, average='macro'))
            results.insert(0,'Decision Tree Classifier RUN ' + str(i+1) + " percentile=" + str(p) + " With Scoring met
hod " + score)
            results.pop(4)
            results.insert(4, DTC_GS.score(X_train, y_train))
            results.insert(5, DTC_GS.score(X_test, y_test))
            df11 = pd.DataFrame([results], columns = ['Classifier','Precision','Recall','Fscore', 'Train score', 'Test
score']).set_index('Classifier')
            resultsDF = resultsDF.append([df11])
            print("The best estimator for RUN " + str(i+1) + " percentile=" + str(p) + " With Scoring method " + score
+ str(DTC_GS.best_estimator_))
            print("The Confusion matrix for RUN" + str(i+1) + "percentile=" + str(p) + " With Scoring method " + score
+ " is \n")
            print(print(multilabel_confusion_matrix(y_test, y_pred)))
```

```
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=1 With Scoring method recall is

[[[128 191]
  [140 299]]

 [[299 140]
  [191 128]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1265 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=1 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=20,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=1 With Scoring method precision is

[[[ 69 247]
  [ 66 376]]

 [[376  66]
  [247  69]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1173 tasks      | elapsed:    0.8s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.0s finished

The best estimator for RUN 1 percentile=1 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=1 With Scoring method accuracy is

[[[ 85 232]
  [ 73 368]]

 [[368  73]
  [232  85]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 1173 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 1 percentile=1 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=1 With Scoring method f1 is

[[[ 94 226]
  [101 337]]

 [[337 101]
  [226  94]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.5s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=5 With Scoring method recall is

[[[ 62 212]
  [ 88 396]]

 [[396  88]
  [212  62]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.6s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=6,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=5 With Scoring method precision is

[[[ 50 257]
  [ 23 428]]

 [[428  23]
  [257  50]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=5 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=5,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=5 With Scoring method accuracy is

[[[ 53 268]
  [ 12 425]]

 [[425  12]
  [268  53]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.6s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 1 percentile=5 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=5 With Scoring method f1 is

[[[135 180]
  [126 317]]

 [[317 126]
  [180 135]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1295 tasks      | elapsed:    1.8s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weig
ht=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=10 With Scoring method recall is

[[[ 85 251]
  [ 47 375]]

 [[375  47]
  [251  85]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1260 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_w
eight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=9,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=10 With Scoring method precision is

[[[ 42 269]
  [ 12 435]]

 [[435  12]
  [269  42]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 1 percentile=10 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=6,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=10 With Scoring method accuracy is

[[[ 37 264]
  [ 17 440]]

 [[440  17]
  [264  37]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    1.9s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.2s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 1 percentile=10 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=N
one, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN1percentile=10 With Scoring method f1 is

[[[ 80 228]
  [ 68 382]]

 [[382  68]
  [228  80]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=1 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=30, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=1 With Scoring method recall is

[[[122 181]
  [141 314]]

 [[314 141]
  [181 122]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=1 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=11,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=1 With Scoring method precision is

[[[ 91 224]
  [ 84 359]]

 [[359  84]
  [224  91]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    0.8s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.0s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.

The best estimator for RUN 2 percentile=1 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=5,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=1 With Scoring method accuracy is

[[[ 81 244]
  [ 57 376]]

 [[376  57]
  [244  81]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=1 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=1 With Scoring method f1 is

[[[ 72 218]
  [ 80 388]]

 [[388  80]
  [218  72]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.5s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=5 With Scoring method recall is

[[[ 86 215]
  [ 86 371]]

 [[371  86]
  [215  86]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.3s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.6s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=6,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=5 With Scoring method precision is

[[[ 56 268]
  [ 17 417]]

 [[417  17]
  [268  56]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:    1.0s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=5 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=7,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=5 With Scoring method accuracy is

[[[ 53 261]
  [ 21 423]]

 [[423  21]
  [261  53]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.8s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=5 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=30, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=5 With Scoring method f1 is

[[[ 77 219]
  [ 54 408]]

 [[408  54]
  [219  77]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.5s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weig
ht=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=10 With Scoring method recall is

[[[125 184]
  [129 320]]

 [[320 129]
  [184 125]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1265 tasks      | elapsed:    1.9s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_w
eight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=4,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=10 With Scoring method precision is

[[[ 33 303]
  [ 15 407]]

 [[407  15]
  [303  33]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 2 percentile=10 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=4,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=10 With Scoring method accuracy is

[[[ 51 265]
  [ 14 428]]

 [[428  14]
  [265  51]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1295 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 2 percentile=10 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=N
one, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN2percentile=10 With Scoring method f1 is

[[[131 160]
  [123 344]]

 [[344 123]
  [160 131]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=1 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=30, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=1 With Scoring method recall is

[[[106 216]
  [112 324]]

 [[324 112]
  [216 106]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=1 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=5,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=1 With Scoring method precision is

[[[ 77 243]
  [ 66 372]]

 [[372  66]
  [243  77]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1202 tasks      | elapsed:    0.9s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.1s finished

The best estimator for RUN 3 percentile=1 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=15,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=1 With Scoring method accuracy is

[[[ 89 236]
  [ 58 375]]

 [[375  58]
  [236  89]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
[Parallel(n_jobs=-1)]: Done 1173 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

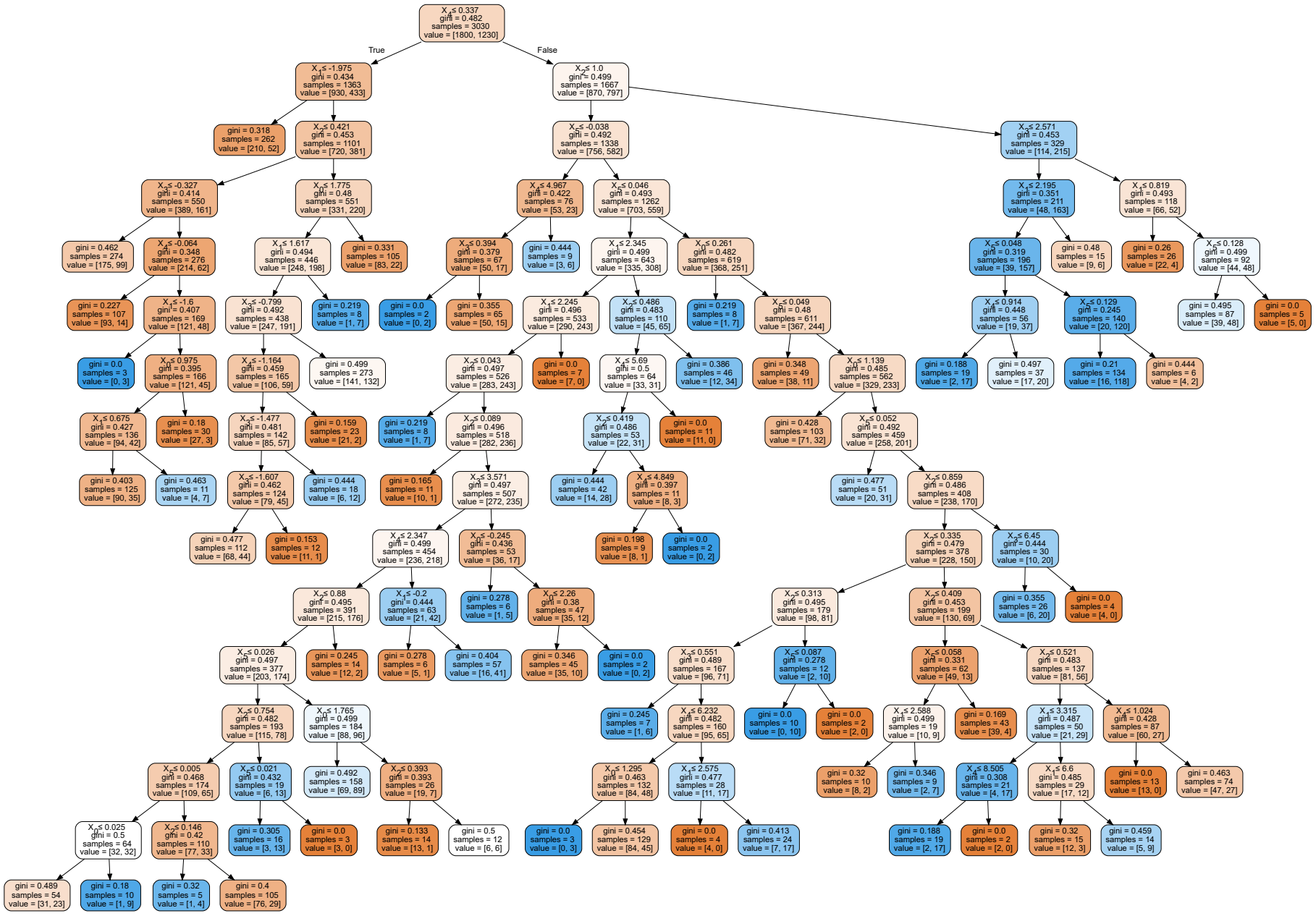
```
The best estimator for RUN 3 percentile=1 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=20, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=1 With Scoring method f1 is

[[[103 219]
  [ 78 358]]

 [[358  78]
  [219 103]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weigh
t=None, criterion='gini',
                       max_depth=10, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=5 With Scoring method recall is

[[[ 75 229]
  [ 70 384]]

 [[384  70]
  [229  75]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=7,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=5 With Scoring method precision is

[[[ 65 244]
  [ 26 423]]

 [[423  26]
  [244  65]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.1s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.4s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=5 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_wei
ght=None, criterion='entropy',
                       max_depth=5, max_features=None, max_leaf_nodes=4,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=5 With Scoring method accuracy is

[[[ 49 255]
  [ 26 428]]

 [[428  26]
  [255  49]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    1.7s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s
```

```
The best estimator for RUN 3 percentile=5 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=No
ne, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=5 With Scoring method f1 is

[[[ 88 231]
  [ 77 362]]

 [[362  77]
  [231  88]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1265 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10 With Scoring method recallDecisionTreeClassifier(ccp_alpha=0.0, class_weig
ht=None, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=10 With Scoring method recall is

[[[129 171]
  [141 317]]

 [[317 141]
  [171 129]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1280 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.3s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10 With Scoring method precisionDecisionTreeClassifier(ccp_alpha=0.0, class_w
eight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=6,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=10 With Scoring method precision is

[[[ 52 252]
  [ 26 428]]

 [[428  26]
  [252  52]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1231 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.1s finished
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  28 tasks      | elapsed:    0.0s

The best estimator for RUN 3 percentile=10 With Scoring method accuracyDecisionTreeClassifier(ccp_alpha=0.0, class_we
ight=None, criterion='entropy',
                       max_depth=15, max_features=None, max_leaf_nodes=10,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=10 With Scoring method accuracy is

[[[ 26 292]
  [ 15 425]]

 [[425  15]
  [292  26]]]
None
Fitting 10 folds for each of 150 candidates, totalling 1500 fits

[Parallel(n_jobs=-1)]: Done 1265 tasks      | elapsed:    2.0s
```

```
The best estimator for RUN 3 percentile=10 With Scoring method f1DecisionTreeClassifier(ccp_alpha=0.0, class_weight=N
one, criterion='gini',
                       max_depth=15, max_features=None, max_leaf_nodes=70,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
The Confusion matrix for RUN3percentile=10 With Scoring method f1 is

[[[128 195]
  [115 320]]

 [[320 115]
  [195 128]]]
None

[Parallel(n_jobs=-1)]: Done 1500 out of 1500 | elapsed:    2.3s finished
```

In [38]:
```
dot_data = tree.export_graphviz(DTC_GS.best_estimator_, out_file=None,
                      filled=True, rounded=True,
                      special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

Out[38]:



In [39]:
```
print('The parameters combination that would give best accuracy is : ')
print(DTC_GS.best_params_)
```

```
The parameters combination that would give best accuracy is :
{'criterion': 'gini', 'max_depth': 15, 'max_leaf_nodes': 70}
```

In [40]:
```
resultsDF = resultsDF.sort_values(by = ["Precision"], ascending = False)
print("Ranked by Precision")
resultsDF
```

Ranked by Precision

Out[40]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286
Decision Tree Classifier RUN 2 percentile=5 With Scoring method accuracy	0.667319	0.560746	0.511598	0.632673	0.627968
Decision Tree Classifier RUN 1 percentile=10 With Scoring method accuracy	0.655093	0.542862	0.483209	0.626733	0.629288
Decision Tree Classifier RUN 1 percentile=5 With Scoring method precision	0.654875	0.555934	0.508340	0.716172	0.684932
Decision Tree Classifier RUN 3 percentile=10 With Scoring method precision	0.648039	0.556892	0.513551	0.704082	0.666667
Decision Tree Classifier RUN 3 percentile=5 With Scoring method accuracy	0.639990	0.551958	0.505717	0.628383	0.629288
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
Decision Tree Classifier RUN 2 percentile=10 With Scoring method precision	0.630370	0.531335	0.445478	0.770563	0.687500
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
Decision Tree Classifier RUN 1 percentile=10 With Scoring method recall	0.621490	0.570801	0.539448	0.341002	0.252976
Decision Tree Classifier RUN 2 percentile=5 With Scoring method f1	0.619252	0.571626	0.554984	0.466520	0.360656
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
Decision Tree Classifier RUN 3 percentile=10 With Scoring method accuracy	0.613447	0.523835	0.439753	0.636964	0.594987
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
Decision Tree Classifier RUN 3 percentile=1 With Scoring method accuracy	0.609595	0.569948	0.547755	0.615512	0.612137
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
Decision Tree Classifier RUN 2 percentile=1 With Scoring method accuracy	0.596704	0.558796	0.532021	0.611221	0.602902
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
Decision Tree Classifier RUN 3 percentile=1 With Scoring method f1	0.594756	0.570488	0.558177	0.458502	0.409543
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
Decision Tree Classifier RUN 1 percentile=10 With Scoring method f1	0.583385	0.554315	0.535816	0.500815	0.350877
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
Decision Tree Classifier RUN 1 percentile=5 With Scoring method f1	0.577534	0.572074	0.571609	0.580159	0.468750
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
Decision Tree Classifier RUN 1 percentile=1 With Scoring method accuracy	0.575654	0.551303	0.532454	0.612871	0.597625
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
Decision Tree Classifier RUN 3 percentile=10 With Scoring method f1	0.574054	0.565959	0.562991	0.577607	0.452297
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
Decision Tree Classifier RUN 3 percentile=5 With Scoring method f1	0.571894	0.550232	0.532593	0.502128	0.363636
Decision Tree Classifier RUN 3 percentile=5 With Scoring method recall	0.571834	0.546263	0.526925	0.353082	0.246711
Decision Tree Classifier RUN 3 percentile=1 With Scoring method precision	0.571670	0.544970	0.519583	0.565637	0.538462
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
Decision Tree Classifier RUN 2 percentile=1 With Scoring method precision	0.567890	0.549636	0.535617	0.559157	0.520000
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
Decision Tree Classifier RUN 2 percentile=5 With Scoring method recall	0.566553	0.548765	0.537523	0.399361	0.285714
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
Decision Tree Classifier RUN 3 percentile=10 With Scoring method recall	0.563684	0.561070	0.561411	0.521947	0.430000
Decision Tree Classifier RUN 2 percentile=10 With Scoring method recall	0.563523	0.558613	0.557807	0.556270	0.404531
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
Decision Tree Classifier RUN 1 percentile=1 With Scoring method precision	0.557321	0.534517	0.506045	0.577143	0.511111
Decision Tree Classifier RUN 2 percentile=1 With Scoring method f1	0.556974	0.538668	0.524162	0.452103	0.325792
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
Decision Tree Classifier RUN 2 percentile=1 With Scoring method recall	0.549111	0.546375	0.546074	0.462400	0.402640
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
Decision Tree Classifier RUN 1 percentile=1 With Scoring method recall	0.543908	0.541174	0.539909	0.474068	0.401254
Decision Tree Classifier RUN 3 percentile=1 With Scoring method recall	0.543119	0.536156	0.528264	0.413485	0.329193
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
Decision Tree Classifier RUN 1 percentile=1 With Scoring method f1	0.540315	0.531578	0.519188	0.470470	0.365049
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
Decision Tree Classifier RUN 1 percentile=5 With Scoring method recall	0.532325	0.522230	0.508864	0.357310	0.226277
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156

In [41]:
```
resultsDF = resultsDF.sort_values(by = ["Recall"], ascending = False)
print("Ranked by Recall")
resultsDF
```

Ranked by Recall

Out[41]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
Decision Tree Classifier RUN 1 percentile=5 With Scoring method f1	0.577534	0.572074	0.571609	0.580159	0.468750
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
Decision Tree Classifier RUN 2 percentile=5 With Scoring method f1	0.619252	0.571626	0.554984	0.466520	0.360656
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
Decision Tree Classifier RUN 1 percentile=10 With Scoring method recall	0.621490	0.570801	0.539448	0.341002	0.252976
Decision Tree Classifier RUN 3 percentile=1 With Scoring method f1	0.594756	0.570488	0.558177	0.458502	0.409543
Decision Tree Classifier RUN 3 percentile=1 With Scoring method accuracy	0.609595	0.569948	0.547755	0.615512	0.612137
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
Decision Tree Classifier RUN 3 percentile=10 With Scoring method f1	0.574054	0.565959	0.562991	0.577607	0.452297
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601

Classifier	Precision	Recall	Fscore	Train score	Test score
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
Decision Tree Classifier RUN 3 percentile=10 With Scoring method recall	0.563684	0.561070	0.561411	0.521947	0.430000
Decision Tree Classifier RUN 2 percentile=5 With Scoring method accuracy	0.667319	0.560746	0.511598	0.632673	0.627968
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
Decision Tree Classifier RUN 2 percentile=1 With Scoring method accuracy	0.596704	0.558796	0.532021	0.611221	0.602902
Decision Tree Classifier RUN 2 percentile=10 With Scoring method recall	0.563523	0.558613	0.557807	0.556270	0.404531
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
Decision Tree Classifier RUN 3 percentile=10 With Scoring method precision	0.648039	0.556892	0.513551	0.704082	0.666667
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
Decision Tree Classifier RUN 1 percentile=5 With Scoring method precision	0.654875	0.555934	0.508340	0.716172	0.684932
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
Decision Tree Classifier RUN 1 percentile=10 With Scoring method f1	0.583385	0.554315	0.535816	0.500815	0.350877
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Decision Tree Classifier RUN 3 percentile=5 With Scoring method accuracy	0.639990	0.551958	0.505717	0.628383	0.629288
Decision Tree Classifier RUN 1 percentile=1 With Scoring method accuracy	0.575654	0.551303	0.532454	0.612871	0.597625
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
Decision Tree Classifier RUN 3 percentile=5 With Scoring method f1	0.571894	0.550232	0.532593	0.502128	0.363636
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
Decision Tree Classifier RUN 2 percentile=1 With Scoring method precision	0.567890	0.549636	0.535617	0.559157	0.520000
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
Decision Tree Classifier RUN 2 percentile=5 With Scoring method recall	0.566553	0.548765	0.537523	0.399361	0.285714
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
Decision Tree Classifier RUN 2 percentile=1 With Scoring method recall	0.549111	0.546375	0.546074	0.462400	0.402640
Decision Tree Classifier RUN 3 percentile=5 With Scoring method recall	0.571834	0.546263	0.526925	0.353082	0.246711
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
Decision Tree Classifier RUN 3 percentile=1 With Scoring method precision	0.571670	0.544970	0.519583	0.565637	0.538462
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
Decision Tree Classifier RUN 1 percentile=10 With Scoring method accuracy	0.655093	0.542862	0.483209	0.626733	0.629288
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
Decision Tree Classifier RUN 1 percentile=1 With Scoring method recall	0.543908	0.541174	0.539909	0.474068	0.401254
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
Decision Tree Classifier RUN 2 percentile=1 With Scoring method f1	0.556974	0.538668	0.524162	0.452103	0.325792
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
Decision Tree Classifier RUN 3 percentile=1 With Scoring method recall	0.543119	0.536156	0.528264	0.413485	0.329193
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
Decision Tree Classifier RUN 1 percentile=1 With Scoring method precision	0.557321	0.534517	0.506045	0.577143	0.511111
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
Decision Tree Classifier RUN 1 percentile=1 With Scoring method f1	0.540315	0.531578	0.519188	0.470470	0.365049
Decision Tree Classifier RUN 2 percentile=10 With Scoring method precision	0.630370	0.531335	0.445478	0.770563	0.687500
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
Decision Tree Classifier RUN 3 percentile=10 With Scoring method accuracy	0.613447	0.523835	0.439753	0.636964	0.594987
Decision Tree Classifier RUN 1 percentile=5 With Scoring method recall	0.532325	0.522230	0.508864	0.357310	0.226277
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559

In [42]:
```
resultsDF = resultsDF.sort_values(by = ["Fscore"], ascending = False)
print("Ranked by F Measure")
resultsDF
```

Ranked by F Measure

Out[42]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
Decision Tree Classifier RUN 1 percentile=5 With Scoring method f1	0.577534	0.572074	0.571609	0.580159	0.468750
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
Decision Tree Classifier RUN 3 percentile=10 With Scoring method f1	0.574054	0.565959	0.562991	0.577607	0.452297
Decision Tree Classifier RUN 3 percentile=10 With Scoring method recall	0.563684	0.561070	0.561411	0.521947	0.430000
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
Decision Tree Classifier RUN 3 percentile=1 With Scoring method f1	0.594756	0.570488	0.558177	0.458502	0.409543
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
Decision Tree Classifier RUN 2 percentile=10 With Scoring method recall	0.563523	0.558613	0.557807	0.556270	0.404531
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
Decision Tree Classifier RUN 2 percentile=5 With Scoring method f1	0.619252	0.571626	0.554984	0.466520	0.360656
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
Decision Tree Classifier RUN 3 percentile=1 With Scoring method accuracy	0.609595	0.569948	0.547755	0.615512	0.612137
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
Decision Tree Classifier RUN 2 percentile=1 With Scoring method recall	0.549111	0.546375	0.546074	0.462400	0.402640
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
Decision Tree Classifier RUN 1 percentile=1 With Scoring method recall	0.543908	0.541174	0.539909	0.474068	0.401254
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
Decision Tree Classifier RUN 1 percentile=10 With Scoring method recall	0.621490	0.570801	0.539448	0.341002	0.252976
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
Decision Tree Classifier RUN 2 percentile=5 With Scoring method recall	0.566553	0.548765	0.537523	0.399361	0.285714
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
Decision Tree Classifier RUN 1 percentile=10 With Scoring method f1	0.583385	0.554315	0.535816	0.500815	0.350877
Decision Tree Classifier RUN 2 percentile=1 With Scoring method precision	0.567890	0.549636	0.535617	0.559157	0.520000
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
Decision Tree Classifier RUN 3 percentile=5 With Scoring method f1	0.571894	0.550232	0.532593	0.502128	0.363636
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
Decision Tree Classifier RUN 1 percentile=1 With Scoring method accuracy	0.575654	0.551303	0.532454	0.612871	0.597625
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
Decision Tree Classifier RUN 2 percentile=1 With Scoring method accuracy	0.596704	0.558796	0.532021	0.611221	0.602902
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
Decision Tree Classifier RUN 3 percentile=1 With Scoring method recall	0.543119	0.536156	0.528264	0.413485	0.329193
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
Decision Tree Classifier RUN 3 percentile=5 With Scoring method recall	0.571834	0.546263	0.526925	0.353082	0.246711
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
Decision Tree Classifier RUN 2 percentile=1 With Scoring method f1	0.556974	0.538668	0.524162	0.452103	0.325792
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
Decision Tree Classifier RUN 3 percentile=1 With Scoring method precision	0.571670	0.544970	0.519583	0.565637	0.538462
Decision Tree Classifier RUN 1 percentile=1 With Scoring method f1	0.540315	0.531578	0.519188	0.470470	0.365049
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
Decision Tree Classifier RUN 3 percentile=10 With Scoring method precision	0.648039	0.556892	0.513551	0.704082	0.666667

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
Decision Tree Classifier RUN 2 percentile=5 With Scoring method accuracy	0.667319	0.560746	0.511598	0.632673	0.627968
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
Decision Tree Classifier RUN 1 percentile=5 With Scoring method recall	0.532325	0.522230	0.508864	0.357310	0.226277
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
Decision Tree Classifier RUN 1 percentile=5 With Scoring method precision	0.654875	0.555934	0.508340	0.716172	0.684932
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
Decision Tree Classifier RUN 1 percentile=1 With Scoring method precision	0.557321	0.534517	0.506045	0.577143	0.511111
Decision Tree Classifier RUN 3 percentile=5 With Scoring method accuracy	0.639990	0.551958	0.505717	0.628383	0.629288
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
Decision Tree Classifier RUN 1 percentile=10 With Scoring method accuracy	0.655093	0.542862	0.483209	0.626733	0.629288
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Decision Tree Classifier RUN 2 percentile=10 With Scoring method precision	0.630370	0.531335	0.445478	0.770563	0.687500
Decision Tree Classifier RUN 3 percentile=10 With Scoring method accuracy	0.613447	0.523835	0.439753	0.636964	0.594987
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156

In [43]:
```
resultsDF = resultsDF.sort_values(by = ["Train score"], ascending = False)
print("Ranked by Train score")
resultsDF
```

Ranked by Train score

Out[43]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
Decision Tree Classifier RUN 2 percentile=10 With Scoring method precision	0.630370	0.531335	0.445478	0.770563	0.687500
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
Decision Tree Classifier RUN 1 percentile=5 With Scoring method precision	0.654875	0.555934	0.508340	0.716172	0.684932
Decision Tree Classifier RUN 3 percentile=10 With Scoring method precision	0.648039	0.556892	0.513551	0.704082	0.666667
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
Decision Tree Classifier RUN 3 percentile=10 With Scoring method accuracy	0.613447	0.523835	0.439753	0.636964	0.594987
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
Decision Tree Classifier RUN 2 percentile=5 With Scoring method accuracy	0.667319	0.560746	0.511598	0.632673	0.627968
Decision Tree Classifier RUN 3 percentile=5 With Scoring method accuracy	0.639990	0.551958	0.505717	0.628383	0.629288
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
Decision Tree Classifier RUN 1 percentile=10 With Scoring method accuracy	0.655093	0.542862	0.483209	0.626733	0.629288
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
Decision Tree Classifier RUN 3 percentile=1 With Scoring method accuracy	0.609595	0.569948	0.547755	0.615512	0.612137
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
Decision Tree Classifier RUN 1 percentile=1 With Scoring method accuracy	0.575654	0.551303	0.532454	0.612871	0.597625
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475
Decision Tree Classifier RUN 2 percentile=1 With Scoring method accuracy	0.596704	0.558796	0.532021	0.611221	0.602902
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264

Classifier	Precision	Recall	Fscore	Train score	Test score
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
Decision Tree Classifier RUN 1 percentile=5 With Scoring method f1	0.577534	0.572074	0.571609	0.580159	0.468750
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
Decision Tree Classifier RUN 3 percentile=10 With Scoring method f1	0.574054	0.565959	0.562991	0.577607	0.452297
Decision Tree Classifier RUN 1 percentile=1 With Scoring method precision	0.557321	0.534517	0.506045	0.577143	0.511111
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
Decision Tree Classifier RUN 3 percentile=1 With Scoring method precision	0.571670	0.544970	0.519583	0.565637	0.538462
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
Decision Tree Classifier RUN 2 percentile=1 With Scoring method precision	0.567890	0.549636	0.535617	0.559157	0.520000
Decision Tree Classifier RUN 2 percentile=10 With Scoring method recall	0.563523	0.558613	0.557807	0.556270	0.404531
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
Decision Tree Classifier RUN 3 percentile=10 With Scoring method recall	0.563684	0.561070	0.561411	0.521947	0.430000
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
Decision Tree Classifier RUN 3 percentile=5 With Scoring method f1	0.571894	0.550232	0.532593	0.502128	0.363636
Decision Tree Classifier RUN 1 percentile=10 With Scoring method f1	0.583385	0.554315	0.535816	0.500815	0.350877
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
Decision Tree Classifier RUN 1 percentile=1 With Scoring method recall	0.543908	0.541174	0.539909	0.474068	0.401254

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 1 percentile=1 With Scoring method f1	0.540315	0.531578	0.519188	0.470470	0.365049
Decision Tree Classifier RUN 2 percentile=5 With Scoring method f1	0.619252	0.571626	0.554984	0.466520	0.360656
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
Decision Tree Classifier RUN 2 percentile=1 With Scoring method recall	0.549111	0.546375	0.546074	0.462400	0.402640
Decision Tree Classifier RUN 3 percentile=1 With Scoring method f1	0.594756	0.570488	0.558177	0.458502	0.409543
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
Decision Tree Classifier RUN 2 percentile=1 With Scoring method f1	0.556974	0.538668	0.524162	0.452103	0.325792
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
Decision Tree Classifier RUN 3 percentile=1 With Scoring method recall	0.543119	0.536156	0.528264	0.413485	0.329193
Decision Tree Classifier RUN 2 percentile=5 With Scoring method recall	0.566553	0.548765	0.537523	0.399361	0.285714
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
Decision Tree Classifier RUN 1 percentile=5 With Scoring method recall	0.532325	0.522230	0.508864	0.357310	0.226277
Decision Tree Classifier RUN 3 percentile=5 With Scoring method recall	0.571834	0.546263	0.526925	0.353082	0.246711
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
Decision Tree Classifier RUN 1 percentile=10 With Scoring method recall	0.621490	0.570801	0.539448	0.341002	0.252976
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000

In [44]:
```
resultsDF = resultsDF.sort_values(by = ["Test score"], ascending = False)
print("Ranked by Test score")
resultsDF
```

Ranked by Test score

Out[44]:

Classifier	Precision	Recall	Fscore	Train score	Test score
SVM NON Linear RUN 1 percentile=5 With Scoring method precision :	0.804491	0.501684	0.381845	0.833333	1.000000
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method recall	0.593009	0.552693	0.451944	0.896800	0.900990
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method recall	0.576694	0.548826	0.465112	0.872888	0.867742
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method recall	0.546979	0.527423	0.422836	0.895949	0.867347
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method recall	0.565882	0.553685	0.488171	0.798573	0.804795
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method recall	0.562815	0.552858	0.498361	0.767572	0.784053
Decision Tree Classifier RUN 1 percentile=10 With Scoring method precision	0.697838	0.554101	0.493001	0.776190	0.777778
Decision Tree Classifier RUN 2 percentile=5 With Scoring method precision	0.687941	0.566834	0.513712	0.694805	0.767123
Decision Tree Classifier RUN 3 percentile=5 With Scoring method precision	0.674234	0.576225	0.541532	0.664773	0.714286
Decision Tree Classifier RUN 2 percentile=10 With Scoring method precision	0.630370	0.531335	0.445478	0.770563	0.687500
Decision Tree Classifier RUN 1 percentile=5 With Scoring method precision	0.654875	0.555934	0.508340	0.716172	0.684932
Decision Tree Classifier RUN 3 percentile=10 With Scoring method precision	0.648039	0.556892	0.513551	0.704082	0.666667
SVMLinear RUN 1 percentile=1 With Scoring method precision :	0.624927	0.567680	0.528912	0.531835	0.661157
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method recall	0.594030	0.595926	0.586379	0.649393	0.650943
KNN RUN 3 percentile=1 With Scoring method accuracy	0.615196	0.590480	0.586880	0.602970	0.638522
SVMLinear RUN 1 percentile=10 With Scoring method accuracy :	0.611524	0.568986	0.553249	0.617822	0.635884
SVMLinear RUN 3 percentile=10 With Scoring method accuracy :	0.613964	0.567715	0.548752	0.619472	0.633245
Decision Tree Classifier RUN 2 percentile=10 With Scoring method accuracy	0.701110	0.564859	0.510951	0.626073	0.631926
Decision Tree Classifier RUN 1 percentile=5 With Scoring method accuracy	0.714330	0.568825	0.513412	0.626733	0.630607
Decision Tree Classifier RUN 3 percentile=5 With Scoring method accuracy	0.639990	0.551958	0.505717	0.628383	0.629288
Decision Tree Classifier RUN 1 percentile=10 With Scoring method accuracy	0.655093	0.542862	0.483209	0.626733	0.629288
Decision Tree Classifier RUN 2 percentile=5 With Scoring method accuracy	0.667319	0.560746	0.511598	0.632673	0.627968
SVMLinear RUN 1 percentile=5 With Scoring method accuracy :	0.612310	0.553374	0.518024	0.616832	0.621372
KNN RUN 2 percentile=10 With Scoring method accuracy	0.608541	0.571292	0.553844	0.634983	0.620053
KNN RUN 3 percentile=10 With Scoring method accuracy	0.592223	0.566514	0.555961	0.643564	0.618734
KNN RUN 3 percentile=5 With Scoring method precision	0.634527	0.577528	0.560689	0.641905	0.612903
SVM NON Linear RUN 1 percentile=10 With Scoring method accuracy :	0.612137	1.000000	0.759411	0.585479	0.612137
Decision Tree Classifier RUN 3 percentile=1 With Scoring method accuracy	0.609595	0.569948	0.547755	0.615512	0.612137
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.568267	0.545003	0.529292	0.598350	0.612137
KNN RUN 2 percentile=5 With Scoring method precision	0.628845	0.582452	0.568062	0.627178	0.611111
SVMLinear RUN 1 percentile=1 With Scoring method accuracy :	0.570850	0.548423	0.534451	0.605281	0.610818
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method accuracy	0.578211	0.547936	0.526090	0.604290	0.609499
SVMLinear RUN 2 percentile=5 With Scoring method accuracy :	0.578807	0.553763	0.538240	0.618152	0.608179
KNN RUN 1 percentile=5 With Scoring method accuracy	0.598836	0.557619	0.530090	0.637294	0.608179
SVMLinear RUN 1 percentile=5 With Scoring method precision :	0.617771	0.563662	0.536391	0.596070	0.606838
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method f1	0.590147	0.554767	0.477067	0.580379	0.606765
KNN RUN 1 percentile=10 With Scoring method accuracy	0.581732	0.560524	0.548638	0.657096	0.604222
Decision Tree Classifier RUN 2 percentile=1 With Scoring method accuracy	0.596704	0.558796	0.532021	0.611221	0.602902
KNN RUN 2 percentile=5 With Scoring method accuracy	0.613155	0.565152	0.532156	0.641584	0.601583
KNN RUN 3 percentile=5 With Scoring method accuracy	0.584981	0.558012	0.539664	0.641914	0.601583
SVMLinear RUN 3 percentile=1 With Scoring method accuracy :	0.572660	0.543189	0.516821	0.609571	0.600264
Decision Tree Classifier RUN 1 percentile=1 With Scoring method accuracy	0.575654	0.551303	0.532454	0.612871	0.597625
SVMLinear RUN 2 percentile=10 With Scoring method accuracy :	0.597800	0.546502	0.502820	0.627723	0.597625
SVM NON Linear RUN 1 percentile=1 With Scoring method accuracy :	0.566859	0.544649	0.524976	0.603300	0.594987
SVMLinear RUN 3 percentile=5 With Scoring method accuracy :	0.562238	0.538126	0.513007	0.622442	0.594987
Decision Tree Classifier RUN 3 percentile=10 With Scoring method accuracy	0.613447	0.523835	0.439753	0.636964	0.594987
KNN RUN 2 percentile=1 With Scoring method precision	0.606959	0.567170	0.545463	0.546341	0.594406
SVM NON Linear RUN 1 percentile=5 With Scoring method accuracy :	0.593668	1.000000	0.745033	0.589109	0.593668
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method accuracy	0.570823	0.547985	0.528394	0.608581	0.593668
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method accuracy	0.528376	0.514498	0.476853	0.579538	0.588391
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method accuracy	0.293536	0.500000	0.369909	0.590759	0.587071
KNN RUN 2 percentile=10 With Scoring method precision	0.627008	0.594375	0.589870	0.617406	0.586207
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method accuracy	0.563441	0.538788	0.509293	0.613531	0.583113
SVMLinear RUN 2 percentile=1 With Scoring method accuracy :	0.550806	0.531123	0.503206	0.612541	0.583113
KNN RUN 1 percentile=1 With Scoring method accuracy	0.538105	0.526838	0.509694	0.611881	0.580475

Classifier	Precision	Recall	Fscore	Train score	Test score
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method accuracy	0.289578	0.500000	0.366750	0.592739	0.579156
KNN RUN 2 percentile=1 With Scoring method accuracy	0.504652	0.500515	0.390185	0.597030	0.576517
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method f1	0.574262	0.550954	0.472236	0.588235	0.574586
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method f1	0.552497	0.540948	0.488273	0.568513	0.574011
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method precision	0.613969	0.576006	0.564788	0.528053	0.573333
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.579980	0.551216	0.523488	0.516535	0.570470
KNN RUN 3 percentile=1 With Scoring method precision	0.572743	0.547304	0.517560	0.546900	0.568627
SVMLinear RUN 3 percentile=1 With Scoring method precision :	0.607300	0.565608	0.548752	0.546087	0.567164
SVMLinear RUN 3 percentile=10 With Scoring method precision :	0.596408	0.559973	0.540360	0.594041	0.561151
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method f1	0.579675	0.571874	0.522269	0.567417	0.558824
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method f1	0.560733	0.532489	0.418390	0.595263	0.556503
SVMLinear RUN 2 percentile=10 With Scoring method precision :	0.584595	0.550347	0.523761	0.587242	0.556391
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method f1	0.542558	0.539421	0.515120	0.561086	0.555420
KNN RUN 1 percentile=5 With Scoring method precision	0.599507	0.574723	0.567021	0.611111	0.553073
KNN RUN 1 percentile=10 With Scoring method precision	0.597780	0.571992	0.563072	0.622611	0.551724
SVMLinear RUN 3 percentile=5 With Scoring method precision :	0.574425	0.537948	0.499560	0.602740	0.545455
SVMLinear RUN 1 percentile=10 With Scoring method precision :	0.590120	0.557373	0.540130	0.593466	0.542254
KNN RUN 3 percentile=10 With Scoring method precision	0.592021	0.567351	0.558007	0.604956	0.540698
Decision Tree Classifier RUN 3 percentile=1 With Scoring method precision	0.571670	0.544970	0.519583	0.565637	0.538462
SVMLinear RUN 2 percentile=1 With Scoring method precision :	0.583727	0.554989	0.537793	0.543441	0.536913
SVMLinear RUN 2 percentile=5 With Scoring method precision :	0.576338	0.549686	0.528912	0.575175	0.536913
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method precision	0.595389	0.567467	0.558687	0.530081	0.534161
SVM NON Linear RUN 1 percentile=1 With Scoring method precision :	0.568240	0.547046	0.529388	0.527157	0.521739
Decision Tree Classifier RUN 2 percentile=1 With Scoring method precision	0.567890	0.549636	0.535617	0.559157	0.520000
Decision Tree Classifier RUN 1 percentile=1 With Scoring method precision	0.557321	0.534517	0.506045	0.577143	0.511111
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method accuracy	0.549869	0.535734	0.477588	0.499010	0.500000
SVM NON Linear RUN 1 percentile=10 With Scoring method precision :	0.545635	0.500497	0.374468	1.000000	0.500000
KNN RUN 3 percentile=1 With Scoring method f1	0.554490	0.556185	0.553066	0.606928	0.500000
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method accuracy	0.604267	0.573463	0.483966	0.496700	0.498681
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method precision	0.554769	0.536146	0.515296	0.548440	0.496689
KNN RUN 1 percentile=1 With Scoring method recall	0.565390	0.566115	0.565648	0.588845	0.486577
KNN RUN 1 percentile=1 With Scoring method precision	0.532079	0.505471	0.406644	0.548872	0.484848
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method accuracy	0.562261	0.544935	0.466066	0.502310	0.481530
Decision Tree Classifier RUN 2 percentile=10 With Scoring method f1	0.599144	0.593394	0.594641	0.590628	0.480734
KNN RUN 3 percentile=1 With Scoring method recall	0.549337	0.549668	0.549454	0.668006	0.475728
KNN RUN 3 percentile=5 With Scoring method f1	0.579719	0.578103	0.578691	0.619190	0.469751
Decision Tree Classifier RUN 1 percentile=5 With Scoring method f1	0.577534	0.572074	0.571609	0.580159	0.468750
Gaussian Naive Bayes RUN 3 percentile=5 With Scoring method precision	0.585773	0.576801	0.538164	0.451399	0.466151
Gaussian Naive Bayes RUN 2 percentile=5 With Scoring method precision	0.564465	0.555533	0.516468	0.444341	0.462264
Gaussian Naive Bayes RUN 1 percentile=10 With Scoring method precision	0.570523	0.547137	0.477265	0.438079	0.455150
KNN RUN 1 percentile=5 With Scoring method f1	0.559816	0.554690	0.551568	0.609319	0.454545
Decision Tree Classifier RUN 3 percentile=10 With Scoring method f1	0.574054	0.565959	0.562991	0.577607	0.452297
KNN RUN 1 percentile=10 With Scoring method recall	0.554663	0.554141	0.554336	0.671725	0.451827
KNN RUN 3 percentile=10 With Scoring method f1	0.571985	0.568530	0.569243	0.640137	0.446494
KNN RUN 2 percentile=10 With Scoring method recall	0.558128	0.555559	0.554955	0.657143	0.439024
Gaussian Naive Bayes RUN 1 percentile=5 With Scoring method precision	0.534700	0.536026	0.531950	0.455154	0.438356
KNN RUN 2 percentile=5 With Scoring method recall	0.572549	0.568761	0.568839	0.669082	0.437299
KNN RUN 3 percentile=10 With Scoring method recall	0.533377	0.533088	0.533170	0.684337	0.435065
KNN RUN 1 percentile=10 With Scoring method f1	0.550714	0.547824	0.547588	0.646421	0.430605
Decision Tree Classifier RUN 3 percentile=10 With Scoring method recall	0.563684	0.561070	0.561411	0.521947	0.430000
KNN RUN 3 percentile=5 With Scoring method recall	0.568826	0.564568	0.564315	0.665056	0.424437
Gaussian Naive Bayes RUN 2 percentile=10 With Scoring method precision	0.598290	0.566379	0.474410	0.446705	0.421746
Gaussian Naive Bayes RUN 3 percentile=10 With Scoring method precision	0.567756	0.547838	0.466166	0.447268	0.419463
KNN RUN 2 percentile=10 With Scoring method f1	0.537620	0.534876	0.533296	0.637643	0.417544
KNN RUN 2 percentile=5 With Scoring method f1	0.544992	0.541312	0.539883	0.642018	0.417266
Decision Tree Classifier RUN 3 percentile=1 With Scoring method f1	0.594756	0.570488	0.558177	0.458502	0.409543

Classifier	Precision	Recall	Fscore	Train score	Test score
Decision Tree Classifier RUN 2 percentile=10 With Scoring method recall	0.563523	0.558613	0.557807	0.556270	0.404531
Decision Tree Classifier RUN 2 percentile=1 With Scoring method recall	0.549111	0.546375	0.546074	0.462400	0.402640
Decision Tree Classifier RUN 1 percentile=1 With Scoring method recall	0.543908	0.541174	0.539909	0.474068	0.401254
KNN RUN 1 percentile=5 With Scoring method recall	0.549016	0.544502	0.541423	0.568404	0.384615
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593148	0.562947	0.542698	0.364021	0.378601
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method f1	0.597434	0.564246	0.548361	0.342982	0.368421
Decision Tree Classifier RUN 1 percentile=1 With Scoring method f1	0.540315	0.531578	0.519188	0.470470	0.365049
SVMLinear RUN 3 percentile=5 With Scoring method f1 :	0.592759	0.562324	0.549903	0.375200	0.364045
Decision Tree Classifier RUN 3 percentile=5 With Scoring method f1	0.571894	0.550232	0.532593	0.502128	0.363636
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method f1	0.582135	0.556277	0.542228	0.351293	0.361656
SVMLinear RUN 3 percentile=10 With Scoring method f1 :	0.598307	0.565096	0.554984	0.378914	0.360656
Decision Tree Classifier RUN 2 percentile=5 With Scoring method f1	0.619252	0.571626	0.554984	0.466520	0.360656
SVMLinear RUN 2 percentile=1 With Scoring method f1 :	0.613160	0.568865	0.551258	0.328889	0.359447
Decision Tree Classifier RUN 1 percentile=10 With Scoring method f1	0.583385	0.554315	0.535816	0.500815	0.350877
SVMLinear RUN 1 percentile=5 With Scoring method f1 :	0.576137	0.550060	0.535416	0.388889	0.338673
SVMLinear RUN 2 percentile=5 With Scoring method f1 :	0.612221	0.561973	0.527383	0.331563	0.338462
SVMLinear RUN 1 percentile=10 With Scoring method f1 :	0.622596	0.566448	0.542815	0.338076	0.336538
SVMLinear RUN 2 percentile=10 With Scoring method f1 :	0.603777	0.559276	0.532494	0.341714	0.336364
Decision Tree Classifier RUN 3 percentile=1 With Scoring method recall	0.543119	0.536156	0.528264	0.413485	0.329193
SVMLinear RUN 3 percentile=1 With Scoring method f1 :	0.571077	0.543596	0.516263	0.350993	0.326087
Decision Tree Classifier RUN 2 percentile=1 With Scoring method f1	0.556974	0.538668	0.524162	0.452103	0.325792
SVMLinear RUN 1 percentile=1 With Scoring method f1 :	0.573101	0.542314	0.508812	0.351831	0.314410
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method f1	0.553623	0.532790	0.504452	0.364420	0.307018
KNN RUN 2 percentile=1 With Scoring method f1	0.493173	0.494587	0.481789	0.466327	0.306773
KNN RUN 1 percentile=1 With Scoring method f1	0.494261	0.495601	0.478885	0.552553	0.304950
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.608133	0.572789	0.553839	0.259984	0.291411
Decision Tree Classifier RUN 2 percentile=5 With Scoring method recall	0.566553	0.548765	0.537523	0.399361	0.285714
Gaussian Naive Bayes RUN 2 percentile=1 With Scoring method recall	0.587571	0.560362	0.549076	0.261111	0.279863
Gaussian Naive Bayes RUN 3 percentile=1 With Scoring method recall	0.593969	0.563145	0.551707	0.260491	0.275862
KNN RUN 2 percentile=1 With Scoring method recall	0.436039	0.439879	0.436640	0.510367	0.267559
SVMLinear RUN 3 percentile=1 With Scoring method recall :	0.566148	0.545736	0.533743	0.264613	0.261324
Decision Tree Classifier RUN 1 percentile=10 With Scoring method recall	0.621490	0.570801	0.539448	0.341002	0.252976
Gaussian Naive Bayes RUN 1 percentile=1 With Scoring method recall	0.558076	0.539361	0.523042	0.270335	0.250836
SVMLinear RUN 2 percentile=10 With Scoring method recall :	0.634508	0.574577	0.550012	0.225443	0.247588
Decision Tree Classifier RUN 3 percentile=5 With Scoring method recall	0.571834	0.546263	0.526925	0.353082	0.246711
SVM NON Linear RUN 1 percentile=1 With Scoring method recall :	0.548152	0.532305	0.504588	0.273251	0.245562
SVMLinear RUN 3 percentile=10 With Scoring method recall :	0.546520	0.531795	0.515514	0.292761	0.243243
SVMLinear RUN 1 percentile=1 With Scoring method recall :	0.544852	0.529543	0.507145	0.261080	0.233974
SVMLinear RUN 1 percentile=5 With Scoring method recall :	0.590087	0.552749	0.524012	0.242695	0.233645
SVMLinear RUN 1 percentile=10 With Scoring method recall :	0.622045	0.565906	0.538683	0.238134	0.232258
SVMLinear RUN 2 percentile=1 With Scoring method recall :	0.553361	0.533552	0.505448	0.260374	0.228395
Decision Tree Classifier RUN 1 percentile=5 With Scoring method recall	0.532325	0.522230	0.508864	0.357310	0.226277
SVMLinear RUN 2 percentile=5 With Scoring method recall :	0.585106	0.545145	0.508755	0.247766	0.204969
SVMLinear RUN 3 percentile=5 With Scoring method recall :	0.579277	0.542686	0.509795	0.244355	0.204473
SVM NON Linear RUN 1 percentile=5 With Scoring method f1 :	0.547894	0.507770	0.425618	0.095103	0.089172
SVM NON Linear RUN 1 percentile=10 With Scoring method f1 :	0.572559	1.000000	0.728188	0.003249	0.000000
SVM NON Linear RUN 1 percentile=10 With Scoring method recall :	0.598945	1.000000	0.749175	0.003203	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.597625	1.000000	0.748142	0.002404	0.000000
SVM NON Linear RUN 1 percentile=1 With Scoring method f1 :	0.593668	1.000000	0.745033	0.000000	0.000000
SVM NON Linear RUN 1 percentile=5 With Scoring method recall :	0.558047	1.000000	0.716342	0.000000	0.000000

In [ ]:

In [ ]: