pyCLAMs (CLassifiability Analysis Metrics)

An integrated toolkit for classifiability analysis

API demostration of pyCLAMs.py

import the library

In [1]:
```
from pyCLAMs.pyCLAMs import *
```

A list of all supported metrics

```
['classification.ACC',
 'classification.Kappa',
 'classification.F1_Score',
 'classification.Jaccard',
 'classification.Precision',
 'classification.Recall',
 'classification.CrossEntropy',
 'classification.AP',
 'classification.Brier',
 'classification.ROC_AUC',
 'classification.PR_AUC',
 'classification.BER',
 'correlation.IG',
 'correlation.IG.max',
 'correlation.r',
 'correlation.r.p',
 'correlation.r.max',
 'correlation.r.p.min',
 'correlation.rho',
 'correlation.rho.p',
 'correlation.rho.max',
```

In [2]:

```
# Total metric number
len(metrics_keys())
```

Out[2]:

```
68
```

The following demostrate main API functions

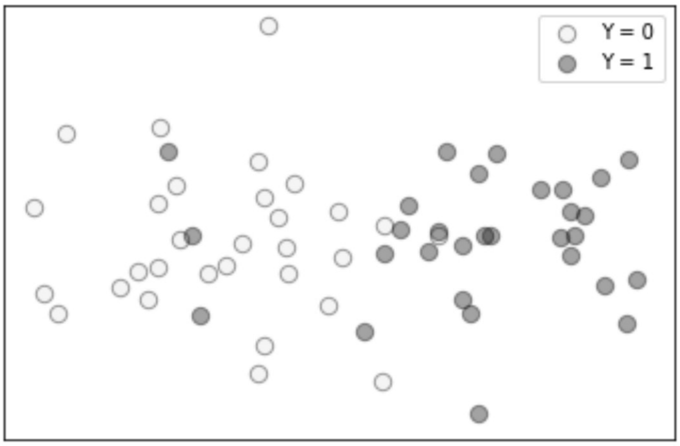
mvg

generates a 2D dataset with a specified between-class distance

In [18]:

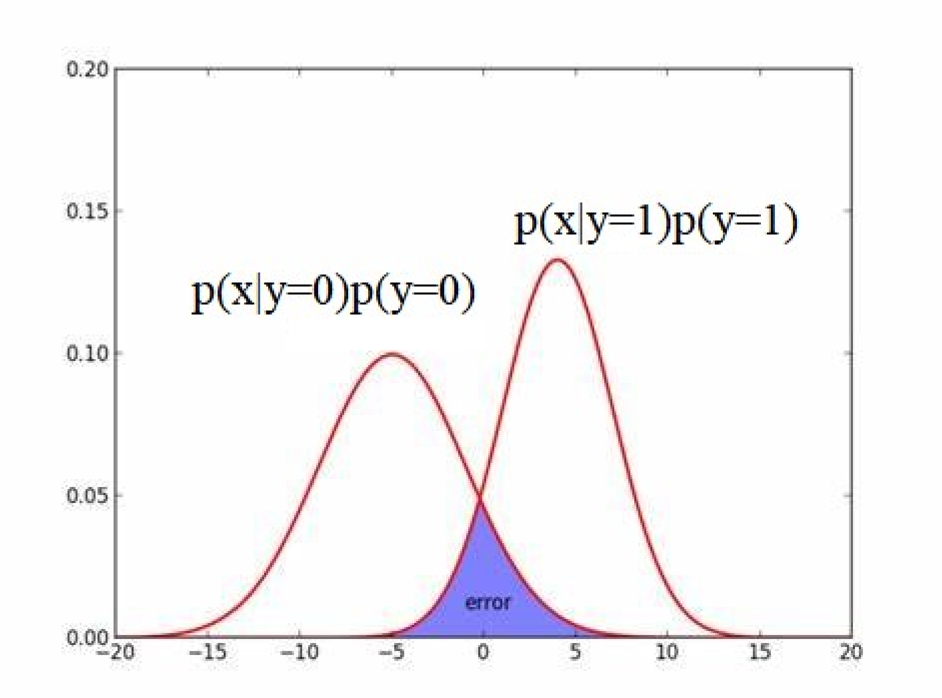
```
df = pd.read_csv('sample.csv')
X = np.array(df.iloc[:,:-1]) # skip first and last cols
y = np.array(df.iloc[:,-1])
```

In [19]:
```
# or generate a toy dataset by X,y = mvg(md = 2)
X.shape, y.shape
plotComponents2D(X,y,labels = set(y))
```



BER

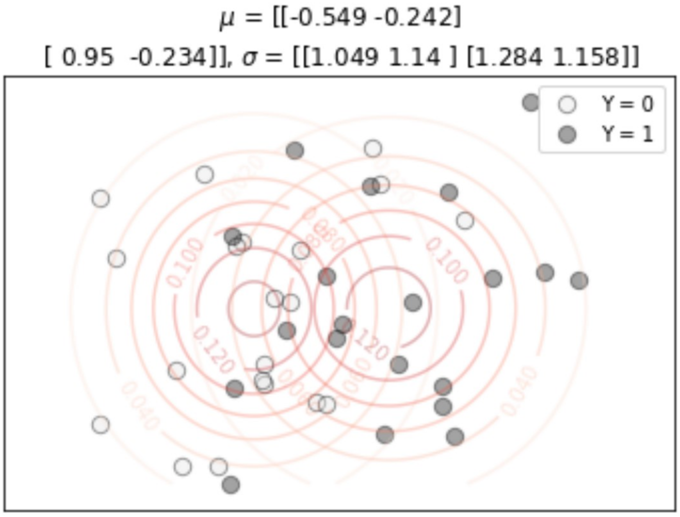
Bayes Error Rate.



$$BER = 1 - E(\max_j P(Y = j|X)$$

It is the lowest possible test error rate in classification which is produced by the Bayes classifier. It is analogous to the irreducible error rate.
Because of noise (inherently stochastic), the error incurred by the oracle prediction model from the true distribution p(x,y) is the Bayes error.
The Bayes optimal decision boundary will correspond to the point where two densities are equal.

In [4]:
```
ber, _ = BER(X,y, show = True)
```

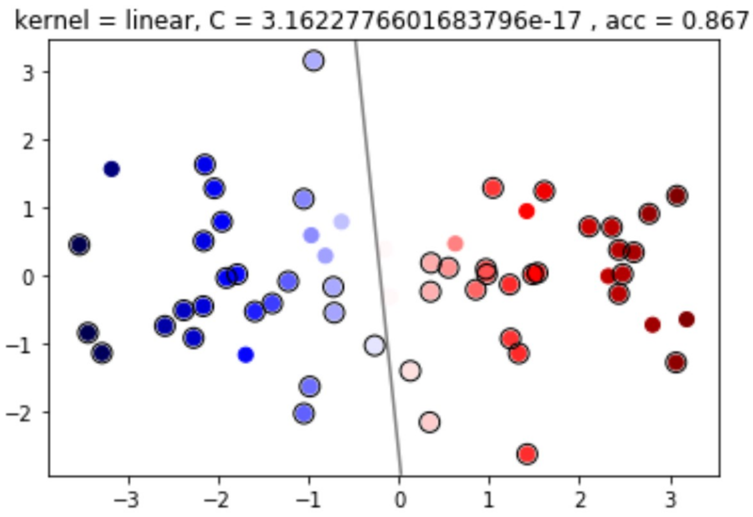


CLF

Classification Accuracy by a n-fold CV SVM

In [20]:
```
_ = CLF(X, y, verbose = True, show = True)
```

```
Best parameters set found by GridSearchCV:
{'C': 3.1622776601683796e-17, 'kernel': 'linear'}
Grid scores on cv set:
0.875 (+/- 0.17678) for {'C': 1e-20, 'kernel': 'linear'}
0.89583 (+/- 0.21246) for {'C': 3.1622776601683796e-17, 'kernel':
'linear'}
0.875 (+/- 0.17678) for {'C': 1e-13, 'kernel': 'linear'}
0.875 (+/- 0.17678) for {'C': 3.1622776601683795e-10, 'kernel': 'l
inear'}
0.875 (+/- 0.17678) for {'C': 1e-06, 'kernel': 'linear'}
0.875 (+/- 0.17678) for {'C': 0.0031622776601683794, 'kernel': 'li
near'}
0.85417 (+/- 0.1559) for {'C': 10.0, 'kernel': 'linear'}

Detailed classification report:

#### Training Set ####
              precision    recall  f1-score   support

           0       0.91      0.88      0.89        24
           1       0.88      0.92      0.90        24

    accuracy                           0.90        48
   macro avg       0.90      0.90      0.90        48
weighted avg       0.90      0.90      0.90        48


#### Test Set ####
              precision    recall  f1-score   support

           0       0.80      0.67      0.73         6
           1       0.71      0.83      0.77         6

    accuracy                           0.75        12
   macro avg       0.76      0.75      0.75        12
weighted avg       0.76      0.75      0.75        12


#### All Set ####
              precision    recall  f1-score   support

           0       0.89      0.83      0.86        30
           1       0.84      0.90      0.87        30

    accuracy                           0.87        60
   macro avg       0.87      0.87      0.87        60
weighted avg       0.87      0.87      0.87        60
```



IG

Information Gain. Output the IG between each feature Xi and y.

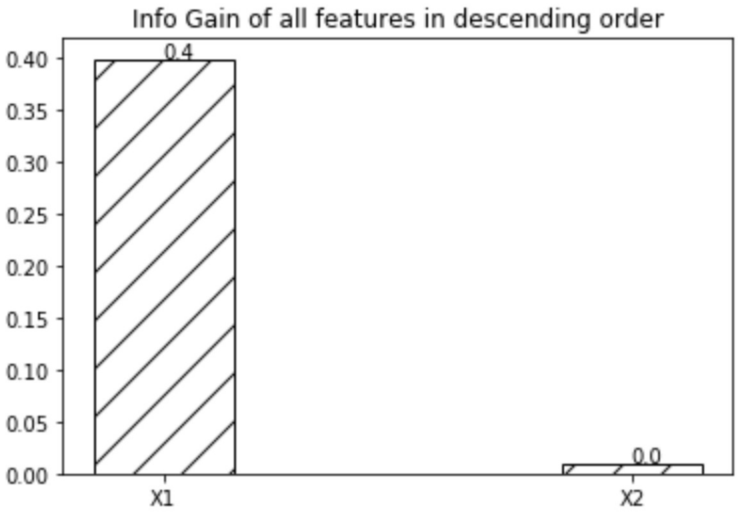
Information gain has been used in decision tree. For a specific feature, Information gain (IG) measures how much “information” a feature gives us about the class.

$$IG(Y|X) = H(Y) - H(Y|X)$$

In information theory, IG answers "if we transmit Y, how many bits can be saved if both sender and receiver know X?" Or "how much information of Y is implied in X?"

Attribute/feature X with a high IG is a good split on Y.

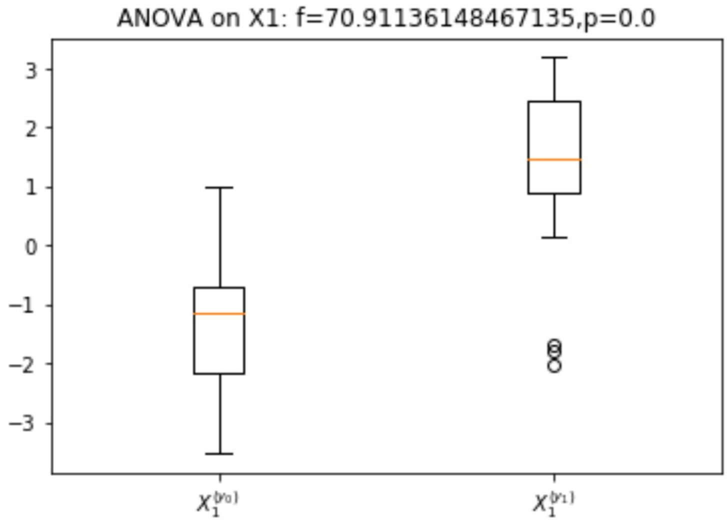
In [21]:
```
ig, _ = IG(X,y,show = True)
```



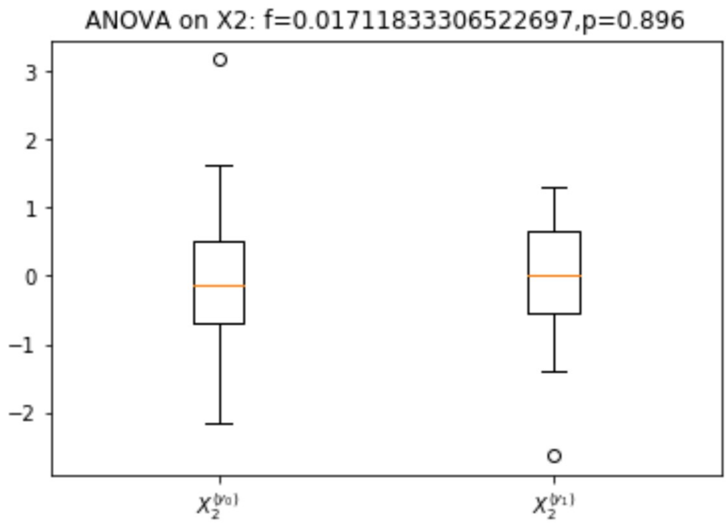
ANOVA

Perform ANOVA test on each feature Xi.

```
In [22]: p, F, _ = ANOVA(X,y, verbose = True, show = True)
```




```
ANOVA on X1: f=70.91136148467135,p=0.0
```




```
ANOVA on X2: f=0.01711833306522697,p=0.896
```

MANOVA

Perform MANOVA test on the first N features.

```
In [23]: p, F, _ = MANOVA(X,y, verbose = True)
```

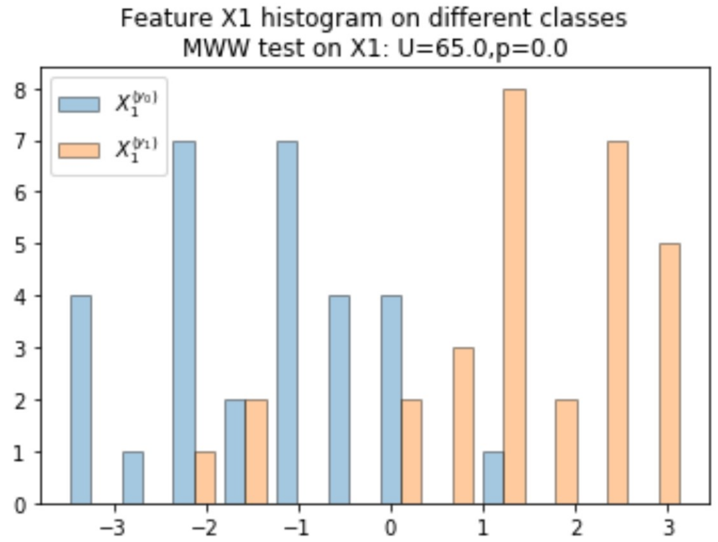
```
endog: ['X1', 'X2']
exog: ['Intercept', 'y']

                  Multivariate linear model
============================================================

------------------------------------------------------------
       Intercept        Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
          Wilks' lambda 0.6201 2.0000 57.0000 17.4576 0.0000
         Pillai's trace 0.3799 2.0000 57.0000 17.4576 0.0000
 Hotelling-Lawley trace 0.6125 2.0000 57.0000 17.4576 0.0000
    Roy's greatest root 0.6125 2.0000 57.0000 17.4576 0.0000
------------------------------------------------------------

------------------------------------------------------------
           y            Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
          Wilks' lambda 0.4497 2.0000 57.0000 34.8823 0.0000
         Pillai's trace 0.5503 2.0000 57.0000 34.8823 0.0000
 Hotelling-Lawley trace 1.2239 2.0000 57.0000 34.8823 0.0000
    Roy's greatest root 1.2239 2.0000 57.0000 34.8823 0.0000
============================================================
```

MWW

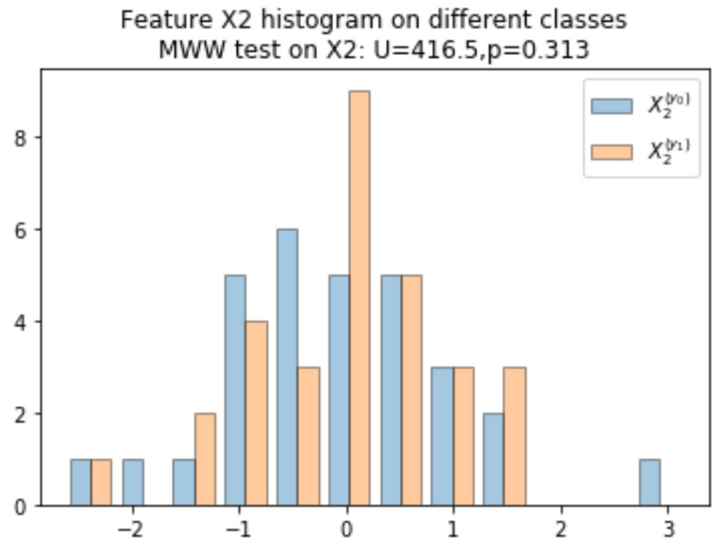
Mann–Whitney U test (also called the Mann–Whitney–Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon–Mann–Whitney test) is a nonparametric test. This test can be used to determine whether two independent （独立、非配对） samples were selected from populations having the same distribution.

Other non-parametric tests are not suitable for classifiability analysis. For example, chi-square test is for nominal data. Signed rank sum test (符号秩和检验) is for paired (相关\配对样本) samples.

```
In [24]: p, U, _ = MWW(X,y, verbose = True, show = True)
```




```
MWW test on X1: U=65.0,p=0.0
```




```
MWW test on X2: U=416.5,p=0.313
```

cohen_d

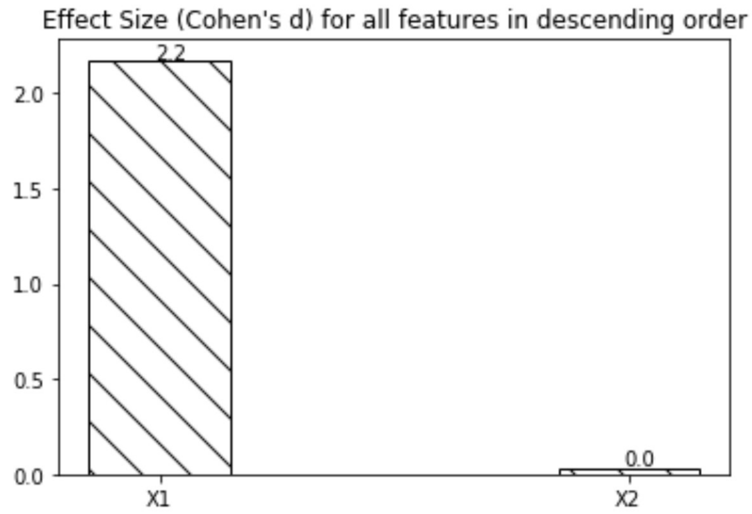
Cohen's d effect size. Use the pooled standard deviation internally.

$$Cohen_d = \frac{\mu_2 - \mu_1}{\sigma} = \frac{\mu_2 - \mu_1}{SD_{pooled}}$$

The Pooled Standard Deviation is

$$SD_{pooled} = \sqrt{\frac{(n_1 - 1)SD_1^2 + (n_2 - 1)SD_2^2}{n_1 + n_2 - 2}}$$

In [25]:
```
es, _ = cohen_d(X, y, show = True)
```



Pearson r, Spearman rho, Kendall tau

Besides these three correlation coefficients, IG/MI can also be seen as a measure of correlation.

In [26]:
```
correlate(X,y, verbose = True)
```

```
#### Correlation between X1 and y ####

Pearson r: 0.742, p-value: 0.0
Spearman rho: 0.741, p-value: 0.0
Kendall's tau: 0.61, p-value: 0.0

#### Correlation between X2 and y ####

Pearson r: 0.017, p-value: 0.896
Spearman rho: 0.064, p-value: 0.625
Kendall's tau: 0.053, p-value: 0.62
```

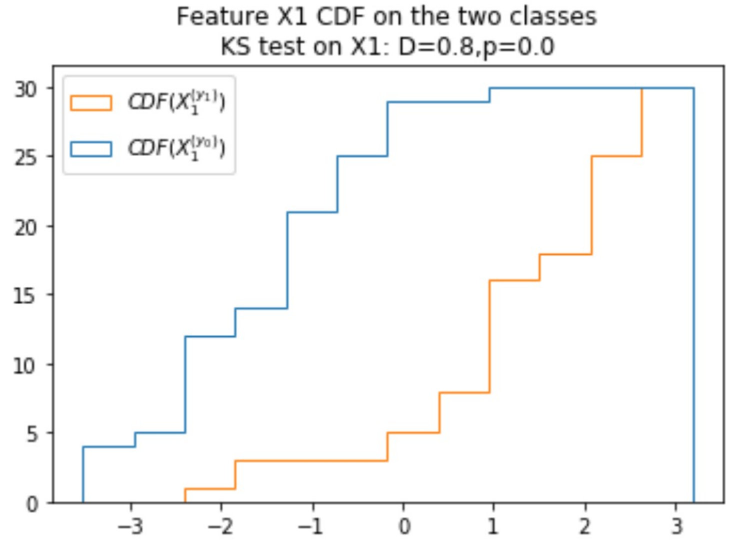
Out[26]:
```
({'correlation.r': [0.7416727355635289, 0.01717721134097977],
  'correlation.r.p': [1.2116341137234975e-11, 0.8963569761285588],
  'correlation.r.max': 0.7416727355635289,
  'correlation.r.p.min': 1.2116341137234975e-11,
  'correlation.rho': [0.7410357742436601, 0.0644806320198157],
  'correlation.rho.p': [1.2886434965392313e-11, 0.624510425255551
6],
  'correlation.rho.max': 0.7410357742436601,
  'correlation.rho.p.min': 1.2886434965392313e-11,
  'correlation.tau': [0.6100744502949613, 0.053099402170811265],
  'correlation.tau.p': [1.2555706904068783e-08, 0.620398864171839
6],
  'correlation.tau.max': 0.6100744502949613,
  'correlation.tau.p.min': 1.2555706904068783e-08},
 "\n\n#### Correlation between X1 and y ####\n\nPearson r: 0.742,
p-value: 0.0\nSpearman rho: 0.741, p-value: 0.0\nKendall's tau: 0.
61, p-value: 0.0\n\n#### Correlation between X2 and y ####\n\nPear
son r: 0.017, p-value: 0.896\nSpearman rho: 0.064, p-value: 0.625\
nKendall's tau: 0.053, p-value: 0.62")
```

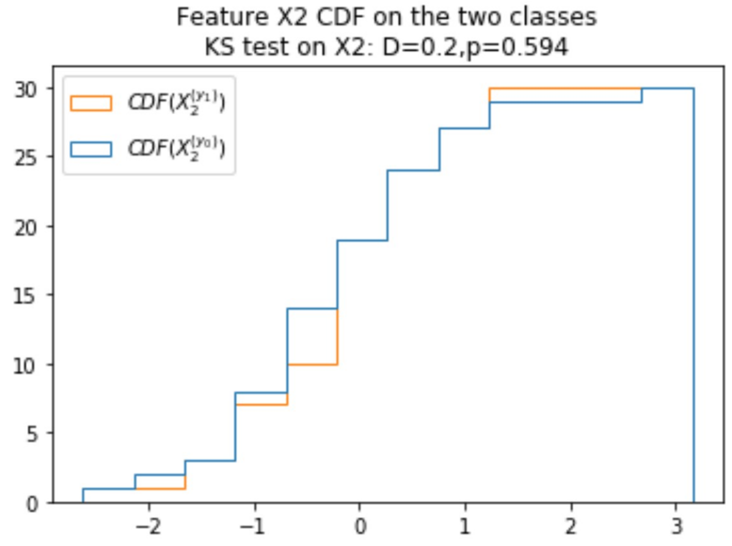
KS

Two-sample Kolmogorov–Smirnov test.

The two-sample KS test is one of the most useful and general nonparametric methods for comparing two samples, as it is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples. The Kolmogorov–Smirnov test can serve as a goodness of fit test. KS test is suitable for continous numeric values while chi-square test is for nominal values.

In [27]:
```
_ = KS(X,y, show = True)
```





ECoL

In [28]:
```
setup_ECoL()
ECoL_metrics(X,y)
```

```
R[write to console]: Installing packages into 'C:/Users/eleve/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
```

Out[28]:
```
({'overlapping.F1.mean': 0.4027582031317379,
  'overlapping.F1.sd': 0.47839672891310947,
  'overlapping.F1v.mean': 0.725,
  'overlapping.F1v.sd': 0.03535533905932741,
  'overlapping.F2.mean': 0.7166666666666667,
  'overlapping.F2.sd': 0.2616330237780185,
  'overlapping.F3.mean': 0.2114385777354317,
  'overlapping.F3.sd': 0.11241300675009834,
  'overlapping.F4.mean': nan,
  'overlapping.F4.sd': 0.07724417539130421,
  'neighborhood.N1': 0.09634745359014747,
  'neighborhood.N2.mean': 0.2542372881355932,
  'neighborhood.N2.sd': 0.4372884724873967,
  'neighborhood.N3.mean': 0.2175486755928619,
  'neighborhood.N3.sd': 0.09985393810178872,
  'neighborhood.N4.mean': 0.26666666666666666,
  'neighborhood.N4.sd': 0.4459484908564835,
  'neighborhood.T1.mean': 0.1356667751336051,
  'neighborhood.T1.sd': 0.3452659113826446,
  'neighborhood.LSC': 0.03333333333333333,
  'linearity.L1.mean': 0.03333333333333333,
  'linearity.L1.sd': 1.0},
 'overlapping.F1.mean\t0.4027582031317379\noverlapping.F1.sd\t0.47839672891310947\noverlapping.F1v.mean\t0.725\noverlapping.F1v.sd\t0.03535533905932741\noverlapping.F2.mean\t0.7166666666666667\noverlapping.F2.sd\t0.2616330237780185\noverlapping.F3.mean\t0.2114385777354317\noverlapping.F3.sd\t0.11241300675009834\noverlapping.F4.mean\tnan\noverlapping.F4.sd\t0.07724417539130421\nneighborhood.N1\t0.09634745359014747\nneighborhood.N2.mean\t0.2542372881355932\nneighborhood.N2.sd\t0.4372884724873967\nneighborhood.N3.mean\t0.2175486755928619\nneighborhood.N3.sd\t0.09985393810178872\nneighborhood.N4.mean\t0.26666666666666666\nneighborhood.N4.sd\t0.4459484908564835\nneighborhood.T1.mean\t0.1356667751336051\nneighborhood.T1.sd\t0.3452659113826446\nneighborhood.LSC\t0.03333333333333333\nlinearity.L1.mean\t0.03333333333333333\nlinearity.L1.sd\t1.0\n')
```

get_metrics()

Return a dictionary of all metrics

In [29]:

```
get_metrics(X,y)
```

Out[29]:
```
({'classification.ACC': 0.8833333333333333,
  'classification.Kappa': 0.7666666666666666,
  'classification.F1_Score': 0.8813559322033899,
  'classification.Jaccard': 0.7878787878787878,
  'classification.Precision': 0.896551724137931,
  'classification.Recall': 0.8666666666666667,
  'classification.CrossEntropy': 0.6718170066747992,
  'classification.AP': 0.9496893963781106,
  'classification.Brier': 0.239339882375674,
  'classification.ROC_AUC': 0.9277777777777778,
  'classification.PR_AUC': 0.9490515409214031,
  'classification.BER': 0.020255040296037308,
  'correlation.IG': [0.39806530243419913, 0.010054453452518208],
  'correlation.IG.max': 0.39806530243419913,
  'correlation.r': [0.7416727355635289, 0.01717721134097977],
  'correlation.r.p': [1.2116341137234975e-11, 0.8963569761285588],
  'correlation.r.max': 0.7416727355635289,
  'correlation.r.p.min': 1.2116341137234975e-11,
  'correlation.rho': [0.7410357742436601, 0.0644806320198157],
  'correlation.rho.p': [1.2886434965392313e-11, 0.624510425255551
6],
  'correlation.rho.max': 0.7410357742436601,
  'correlation.rho.p.min': 1.2886434965392313e-11,
  'correlation.tau': [0.6100744502949613, 0.053099402170811265],
  'correlation.tau.p': [1.2555706904068783e-08, 0.620398864171839
6],
  'correlation.tau.max': 0.6100744502949613,
  'correlation.tau.p.min': 1.2555706904068783e-08,
  'test.ES': [2.174264036169O415, 0.03378198046812041],
  'test.ES.max': 2.1742640361690415,
  'test.ANOVA': [1.2116341137234846e-11, 0.8963569761285523],
  'test.ANOVA.min': 1.2116341137234846e-11,
  'test.ANOVA.min.log10': -10.91662850754676,
  'test.ANOVA.F': [70.91136148467135, 0.01711833306522697],
  'test.ANOVA.F.max': 70.91136148467135,
  'test.MANOVA': 1.27933e-10,
  'test.MANOVA.log10': -9.893017415886254,
  'test.MANOVA.F': 34.8823,
  'test.MWW': [6.555515210378031e-09, 0.3128128812888492],
  'test.MWW.min': 6.555515210378031e-09,
  'test.MWW.min.log10': -8.183393170605266,
  'test.MWW.U': [65.0, 416.5],
  'test.MWW.U.min': 65.0,
  'test.KS': [8.466416460035895e-10, 0.5940706297759378],
  'test.KS.min': 8.466416460035895e-10,
  'test.KS.min.log10': -9.072300372544376,
  'test.KS.D': [0.8, 0.2],
  'test.KS.D.max': 0.8,
  'overlapping.F1.mean': 0.4027582031317379,
  'overlapping.F1.sd': 0.47839672891310947,
  'overlapping.F1v.mean': 0.725,
  'overlapping.F1v.sd': 0.0353553390593274l,
  'overlapping.F2.mean': 0.7166666666666667,
  'overlapping.F2.sd': 0.2616330237780185,
  'overlapping.F3.mean': 0.2114385777354317,
  'overlapping.F3.sd': 0.11241300675009834,
  'overlapping.F4.mean': nan,
  'overlapping.F4.sd': 0.08911035186165892,
  'neighborhood.N1': 0.12581465854687468,
  'neighborhood.N2.mean': 0.2542372881355932,
```

```
 'neighborhood.N2.sd': 0.4372884724873967,
 'neighborhood.N3.mean': 0.2175486755928619,
 'neighborhood.N3.sd': 0.09985393810178872,
 'neighborhood.N4.mean': 0.26666666666666666,
 'neighborhood.N4.sd': 0.4459484908564835,
 'neighborhood.T1.mean': 0.07029806369059628,
 'neighborhood.T1.sd': 0.25360212892507084,
 'neighborhood.LSC': 0.03333333333333333,
 'linearity.L1.mean': 0.03333333333333333,
 'linearity.L1.sd': 1.0},
{'classification.ACC': 0.8833333333333333,
 'classification.Kappa': 0.7666666666666666,
 'classification.F1_Score': 0.8813559322033899,
 'classification.Jaccard': 0.7878787878787878,
 'classification.Precision': 0.896551724137931,
 'classification.Recall': 0.8666666666666667,
 'classification.CrossEntropy': 0.6718170066747992,
 'classification.AP': 0.9496893963781106,
 'classification.Brier': 0.239339882375674,
 'classification.ROC_AUC': 0.9277777777777778,
 'classification.PR_AUC': 0.9490515409214031,
 'classification.BER': 0.020255040296037308,
 'correlation.IG.max': 0.39806530243419913,
 'correlation.r.max': 0.7416727355635289,
 'correlation.r.p.min': 1.2116341137234975e-11,
 'correlation.rho.max': 0.7410357742436601,
 'correlation.rho.p.min': 1.2886434965392313e-11,
 'correlation.tau.max': 0.6100744502949613,
 'correlation.tau.p.min': 1.2555706904068783e-08,
 'test.ES.max': 2.1742640361690415,
 'test.ANOVA.min': 1.2116341137234846e-11,
 'test.ANOVA.min.log10': -10.91662850754676,
 'test.ANOVA.F.max': 70.91136148467135,
 'test.MANOVA': 1.27933e-10,
 'test.MANOVA.log10': -9.893017415886254,
 'test.MANOVA.F': 34.8823,
 'test.MWW.min': 6.555515210378031e-09,
 'test.MWW.min.log10': -8.183393170605266,
 'test.MWW.U.min': 65.0,
 'test.KS.min': 8.466416460035895e-10,
 'test.KS.min.log10': -9.072300372544376,
 'test.KS.D.max': 0.8,
 'overlapping.F1.mean': 0.4027582031317379,
 'overlapping.F1.sd': 0.47839672891310947,
 'overlapping.F1v.mean': 0.725,
 'overlapping.F1v.sd': 0.03535533905932741,
 'overlapping.F2.mean': 0.7166666666666667,
 'overlapping.F2.sd': 0.2616330237780185,
 'overlapping.F3.mean': 0.2114385777354317,
 'overlapping.F3.sd': 0.11241300675009834,
 'overlapping.F4.mean': nan,
 'overlapping.F4.sd': 0.08911035186165892,
 'neighborhood.N1': 0.12581465854687468,
 'neighborhood.N2.mean': 0.2542372881355932,
 'neighborhood.N2.sd': 0.4372884724873967,
 'neighborhood.N3.mean': 0.2175486755928619,
 'neighborhood.N3.sd': 0.09985393810178872,
 'neighborhood.N4.mean': 0.26666666666666666,
 'neighborhood.N4.sd': 0.4459484908564835,
 'neighborhood.T1.mean': 0.07029806369059628,
```

```
'neighborhood.T1.sd': 0.25360212892507084,
'neighborhood.LSC': 0.03333333333333333,
'linearity.L1.mean': 0.03333333333333333,
'linearity.L1.sd': 1.0})
```

get_json

In [30]:

```
get_json(X,y)
```

Out[30]:
'[{"classification.ACC": 0.9, "classification.Kappa": 0.8, "classi
fication.F1_Score": 0.896551724137931, "classification.Jaccard":
0.8125, "classification.Precision": 0.9285714285714286, "classific
ation.Recall": 0.8666666666666667, "classification.CrossEntropy":
0.6718170066747992, "classification.AP": 0.9496893963781106, "clas
sification.Brier": 0.239339882375674, "classification.ROC_AUC": 0.
9277777777777778, "classification.PR_AUC": 0.9490515409214031, "cl
assification.BER": 0.020939517976839683, "correlation.IG": [0.3980
6530243419913, 0.010054453452518208], "correlation.IG.max": 0.3980
6530243419913, "correlation.r": [0.7416727355635289, 0.01717721134
097977], "correlation.r.p": [1.2116341137234975e-11, 0.89635697612
85588], "correlation.r.max": 0.7416727355635289, "correlation.r.p.
min": 1.2116341137234975e-11, "correlation.rho": [0.74103577424366
01, 0.0644806320198157], "correlation.rho.p": [1.2886434965392313e
-11, 0.6245104252555516], "correlation.rho.max": 0.741035774243660
1, "correlation.rho.p.min": 1.2886434965392313e-11, "correlation.t
au": [0.6100744502949613, 0.053099402170811265], "correlation.tau.
p": [1.2555706904068783e-08, 0.6203988641718396], "correlation.ta
u.max": 0.6100744502949613, "correlation.tau.p.min": 1.25557069040
68783e-08, "test.ES": [2.1742640361690415, 0.03378198046812041], "
test.ES.max": 2.1742640361690415, "test.ANOVA": [1.211634113723484
6e-11, 0.8963569761285523], "test.ANOVA.min": 1.2116341137234846e-
11, "test.ANOVA.min.log10": -10.91662850754676, "test.ANOVA.F": [7
0.91136148467135, 0.017118333065226 97], "test.ANOVA.F.max": 70.911
36148467135, "test.MANOVA": 1.27933e-10, "test.MANOVA.log10": -9.8
93017415886254, "test.MANOVA.F": 34.8823, "test.MWW": [6.555515210
378031e-09, 0.3128128812888492], "test.MWW.min": 6.555515210378031
e-09, "test.MWW.min.log10": -8.183393170605266, "test.MWW.U": [65.
0, 416.5], "test.MWW.U.min": 65.0, "test.KS": [8.466416460035895e-
10, 0.5940706297759378], "test.KS.min": 8.466416460035895e-10, "te
st.KS.min.log10": -9.072300372544376, "test.KS.D": [0.8, 0.2], "te
st.KS.D.max": 0.8, "overlapping.F1.mean": 0.4027582031317379, "ove
rlapping.F1.sd": 0.47839672891310947, "overlapping.F1v.mean": 0.72
5, "overlapping.F1v.sd": 0.03535533905932741, "overlapping.F2.mea
n": 0.7166666666666667, "overlapping.F2.sd": 0.2616330237780185, "
overlapping.F3.mean": 0.2114385777354317, "overlapping.F3.sd": 0.1
1241300675009834, "overlapping.F4.mean": NaN, "overlapping.F4.sd":
0.10929627410078556, "neighborhood.N1": 0.16008558916117246, "neig
hborhood.N2.mean": 0.2542372881355932, "neighborhood.N2.sd": 0.437
2884724873967, "neighborhood.N3.mean": 0.2175486755928619, "neighb
orhood.N3.sd": 0.09985393810178872, "neighborhood.N4.mean": 0.2666
666666666666, "neighborhood.N4.sd": 0.4459484908564835, "neighbor
hood.T1.mean": 0.07579123013033624, "neighborhood.T1.sd": 0.258762
64356184453, "neighborhood.LSC": 0.03333333333333333, "linearity.L
1.mean": 0.03333333333333333, "linearity.L1.sd": 1.0}, {"classific
ation.ACC": 0.9, "classification.Kappa": 0.8, "classification.F1_S
core": 0.896551724137931, "classification.Jaccard": 0.8125, "class
ification.Precision": 0.9285714285714286, "classification.Recall":
0.8666666666666667, "classification.CrossEntropy": 0.6718170066747
992, "classification.AP": 0.9496893963781106, "classification.Brie
r": 0.239339882375674, "classification.ROC_AUC": 0.927777777777777
8, "classification.PR_AUC": 0.9490515409214031, "classification.BE
R": 0.020939517976839683, "correlation.IG.max": 0.3980653024341991
3, "correlation.r.max": 0.7416727355635289, "correlation.r.p.min":
1.2116341137234975e-11, "correlation.rho.max": 0.7410357742436601,
"correlation.rho.p.min": 1.2886434965392313e-11, "correlation.tau.
max": 0.6100744502949613, "correlation.tau.p.min": 1.2555706904068
783e-08, "test.ES.max": 2.174264036169415, "test.ANOVA.min": 1.21
16341137234846e-11, "test.ANOVA.min.log10": -10.91662850754676, "t
est.ANOVA.F.max": 70.91136148467135, "test.MANOVA": 1.27933e-10, "

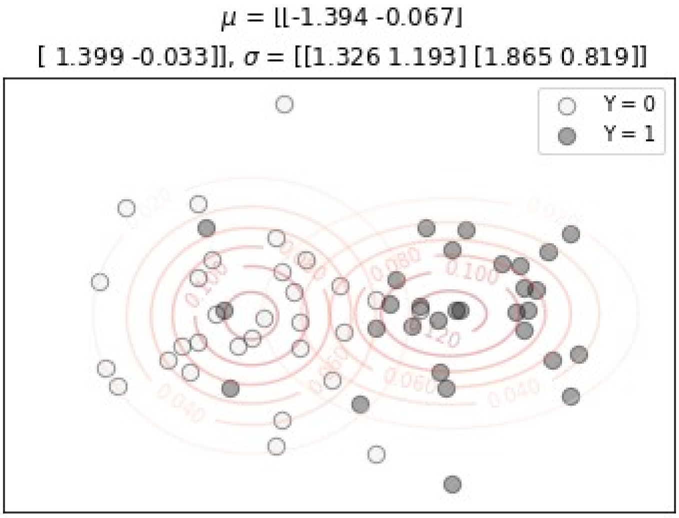
test.MANOVA.log10": -9.893017415886254, "test.MANOVA.F": 34.8823, "test.MWW.min": 6.555515210378031e-09, "test.MWW.min.log10": -8.183393170605266, "test.MWW.U.min": 65.0, "test.KS.min": 8.466416460035895e-10, "test.KS.min.log10": -9.072300372544376, "test.KS.D.max": 0.8, "overlapping.F1.mean": 0.4027582031317379, "overlapping.F1.sd": 0.47839672891310947, "overlapping.F1v.mean": 0.725, "overlapping.F1v.sd": 0.03535533905932741, "overlapping.F2.mean": 0.7166666666666667, "overlapping.F2.sd": 0.2616330237780185, "overlapping.F3.mean": 0.2114385777354317, "overlapping.F3.sd": 0.11241300675009834, "overlapping.F4.mean": NaN, "overlapping.F4.sd": 0.10929627410078556, "neighborhood.N1": 0.16008558916117246, "neighborhood.N2.mean": 0.2542372881355932, "neighborhood.N2.sd": 0.4372884724873967, "neighborhood.N3.mean": 0.2175486755928619, "neighborhood.N3.sd": 0.09985393810178872, "neighborhood.N4.mean": 0.26666666666666666, "neighborhood.N4.sd": 0.4459484908564835, "neighborhood.T1.mean": 0.07579123013033624, "neighborhood.T1.sd": 0.25876264356184453, "neighborhood.LSC": 0.03333333333333333, "linearity.L1.mean": 0.03333333333333333, "linearity.L1.sd": 1.0}]'

get_html

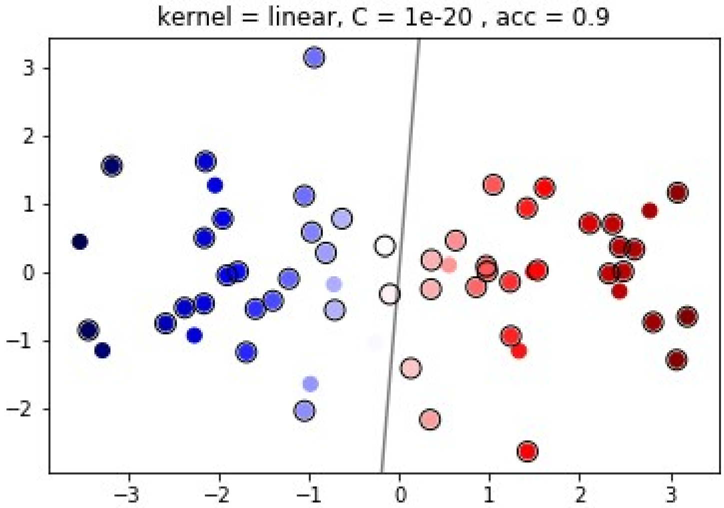
Return a piece of HTML to be embedded in web applications.

In [31]:
```
from IPython.display import display, HTML
display(HTML(get_html(X,y)))
```

Metric/Statistic
BER = 0.020675239294194347

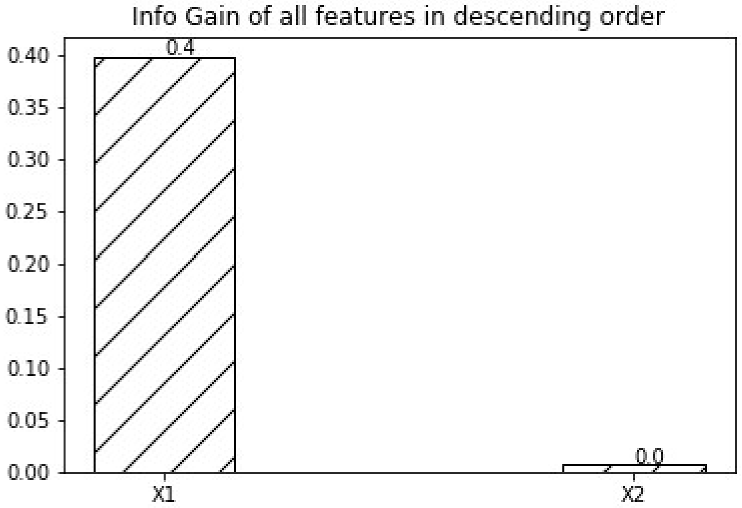


{'classification.ACC': 0.9, 'classification.Kappa': 0.8, 'classification.F1_Score': 0.9, 'classification.Jaccard': 0.8181818181818182, 'classification.Precision': 0.9, 'classification.Recall': 0.9, 'classification.CrossEntropy': 0.6718170066747992, 'classification.AP': 0.9496893963781106, 'classification.Brier': 0.239339882375674, 'classification.ROC_AUC': 0.9277777777777778, 'classification.PR_AUC': 0.9490515409214031}



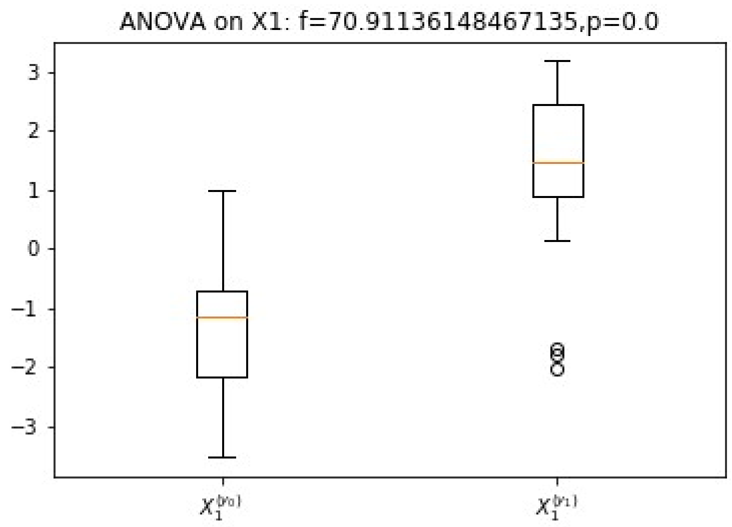
```
Best parameters set found by GridSearchCV:
{'C': 1e-20, 'kernel': 'linear'}
Grid scores on cv set:
0.89583 (+/- 0.05893) for {'C': 1e-20, 'kernel': 'linear'}
0.89583 (+/- 0.05893) for {'C': 3.1622776601683796e-17, 'kernel': 'linear'}
0.89583 (+/- 0.05893) for {'C': 1e-13, 'kernel': 'linear'}
0.89583 (+/- 0.05893) for {'C': 3.1622776601683795e-10, 'kernel': 'linear'}
0.89583 (+/- 0.05893) for {'C': 1e-06, 'kernel': 'linear'}
0.89583 (+/- 0.05893) for {'C': 0.0031622776601683794, 'kernel': 'linear'}
0.875 (+/- 0.10206) for {'C': 10.0, 'kernel': 'linear'}

Detailed classification report:

#### Training Set ####
              precision    recall  f1-score   support

           0       0.91      0.88      0.89        24
           1       0.88      0.92      0.90        24

    accuracy                           0.90        48
   macro avg       0.90      0.90      0.90        48
weighted avg       0.90      0.90      0.90        48


#### Test Set ####
              precision    recall  f1-score   support

           0       0.86      1.00      0.92         6
           1       1.00      0.83      0.91         6

    accuracy                           0.92        12
   macro avg       0.93      0.92      0.92        12
weighted avg       0.93      0.92      0.92        12


#### All Set ####
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        30
           1       0.90      0.90      0.90        30

    accuracy                           0.90        60
   macro avg       0.90      0.90      0.90        60
weighted avg       0.90      0.90      0.90        60


classification.ACC      0.9
classification.Kappa    0.8
classification.F1_Score 0.9
classification.Jaccard  0.8181818181818182
classification.Precision       0.9
classification.Recall   0.9
classification.CrossEntropy     0.6718170066747992
classification.AP       0.9496893963781106
classification.Brier    0.239339882375674
classification.ROC_AUC  0.9277777777777778
classification.PR_AUC   0.9490515409214031
```

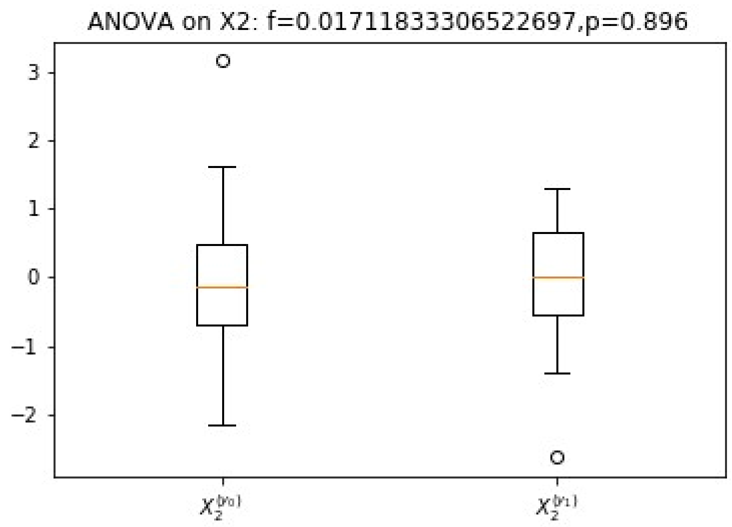
IG = [0.3980653 0.00727668]




```
#### Correlation between X1 and y ####

Pearson r: 0.742, p-value: 0.0
Spearman rho: 0.741, p-value: 0.0
Kendall's tau: 0.61, p-value: 0.0

#### Correlation between X2 and y ####

Pearson r: 0.017, p-value: 0.896
Spearman rho: 0.064, p-value: 0.625
Kendall's tau: 0.053, p-value: 0.62
```

ANOVA p[1.2116341137234846e-11, 0.8963569761285523]





MANOVA p = 1.27933e-10

```
                                 endog: ['X1', 'X2']
                          exog: ['Intercept', 'y']

                                Multivariate linear model
============================================================

------------------------------------------------------------
       Intercept        Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
          Wilks' lambda 0.6201 2.0000 57.0000 17.4576 0.0000
         Pillai's trace 0.3799 2.0000 57.0000 17.4576 0.0000
 Hotelling-Lawley trace 0.6125 2.0000 57.0000 17.4576 0.0000
    Roy's greatest root 0.6125 2.0000 57.0000 17.4576 0.0000
------------------------------------------------------------

------------------------------------------------------------
           y            Value  Num DF  Den DF F Value Pr > F
------------------------------------------------------------
          Wilks' lambda 0.4497 2.0000 57.0000 34.8823 0.0000
         Pillai's trace 0.5503 2.0000 57.0000 34.8823 0.0000
 Hotelling-Lawley trace 1.2239 2.0000 57.0000 34.8823 0.0000
    Roy's greatest root 1.2239 2.0000 57.0000 34.8823 0.0000
============================================================
```

MWW p = [6.555515210378031e-09, 0.3128128812888492]

Feature X1 histogram on different classes
MWW test on X1: U=65.0,p=0.0

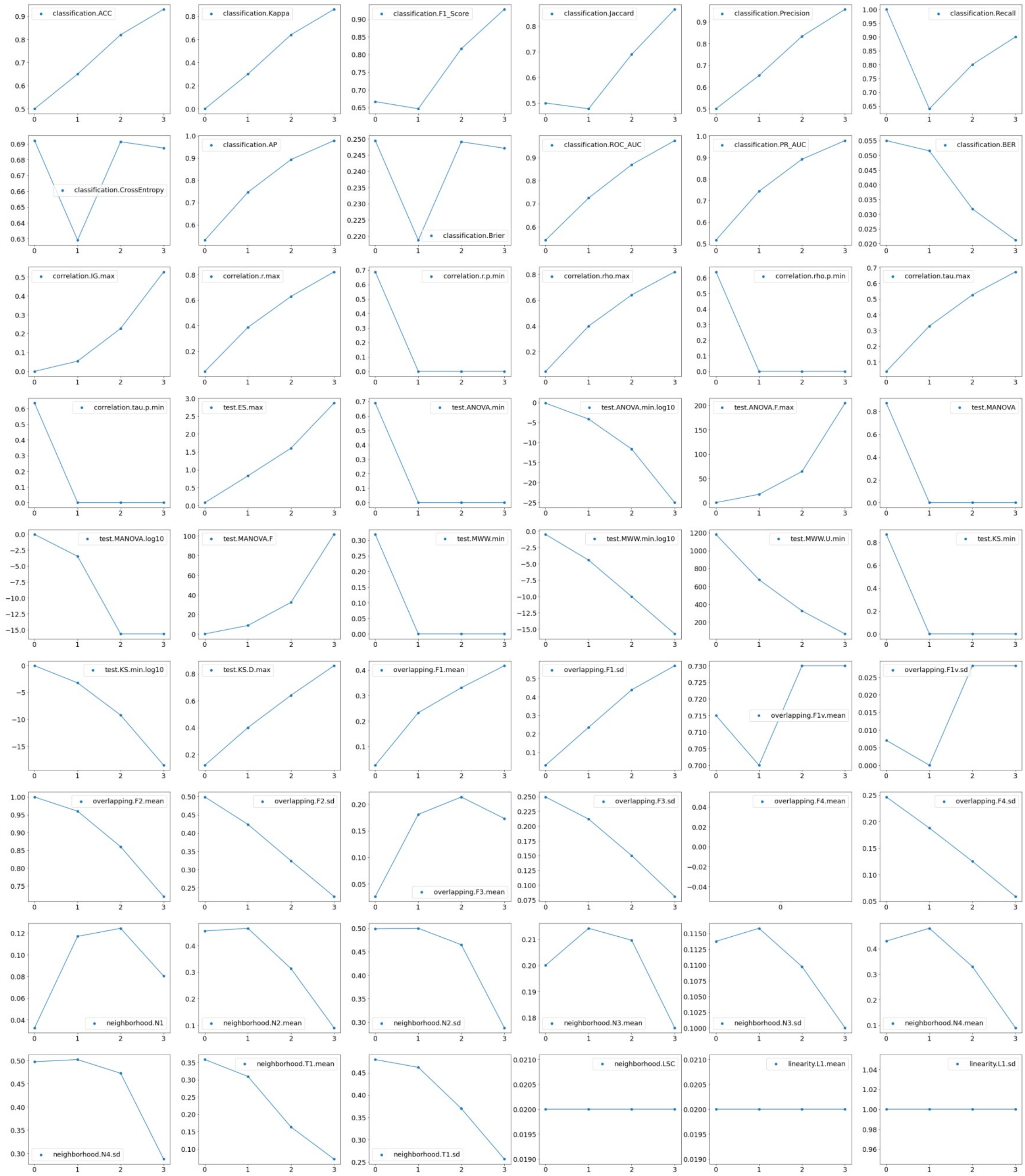
Metric Consistency Test

1. Analyze how the metrics change with datasets / their consistency
2. Extract a common component.

Generate a series of sample datasets with different class distances, and calculate metrics on these datasets.

In [46]:
```
dcts = simulate(np.linspace(0, 3, 4), repeat = 1, nobs = 50)
visualize_dcts(dcts)
```

```
100%|| 4/4 [00:09<00:00,  2.31s/it]
```

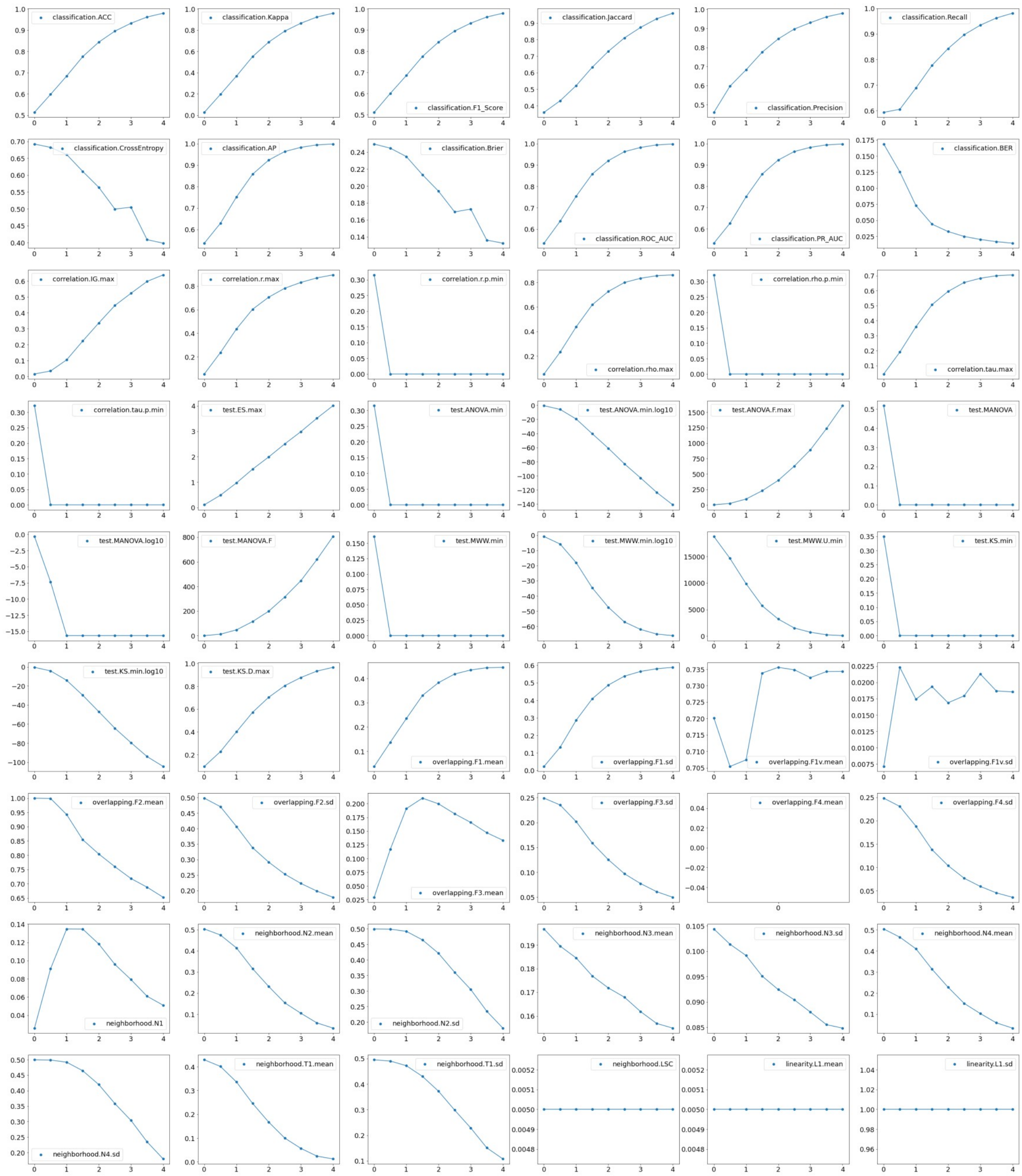


Repeat N times to get the averaged curves against different md values.

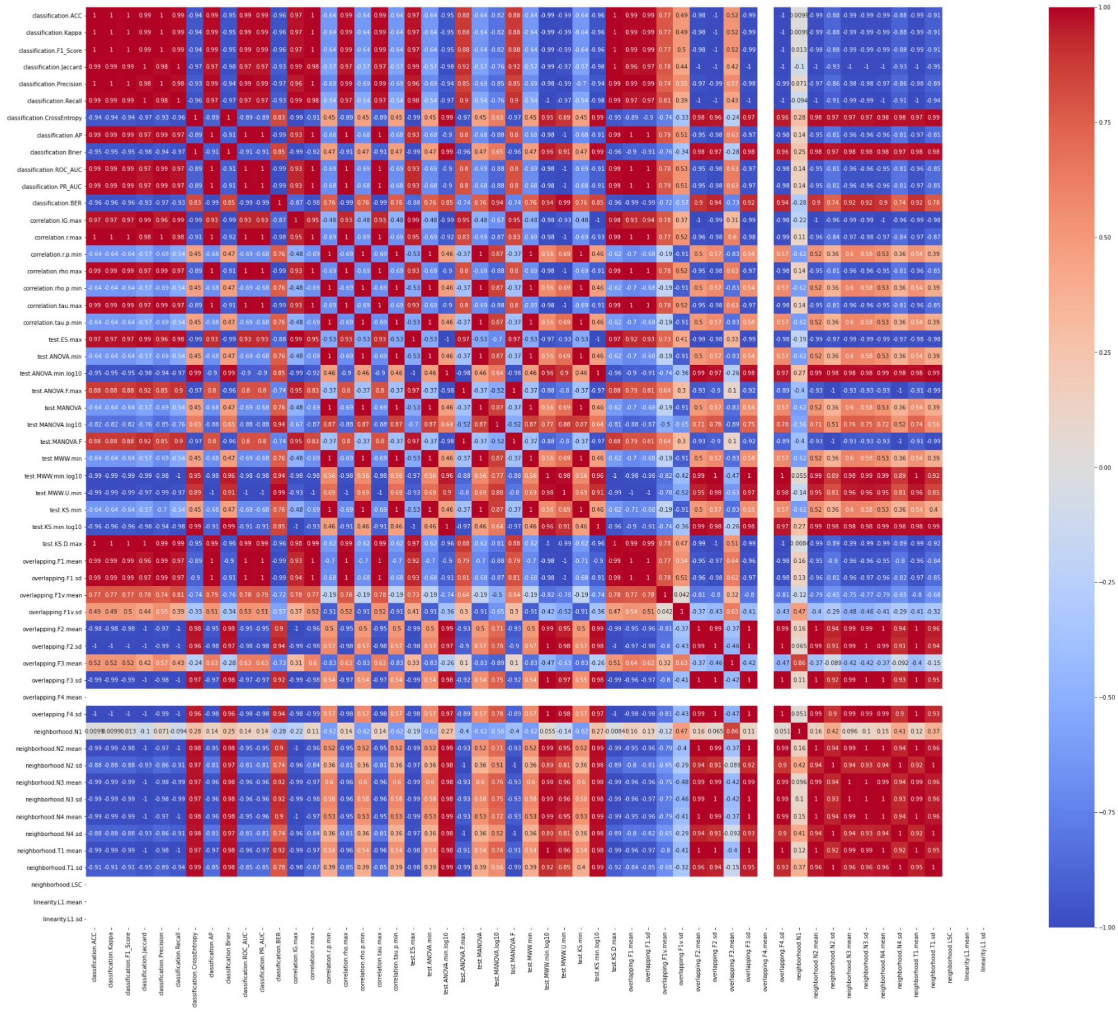
In [47]:

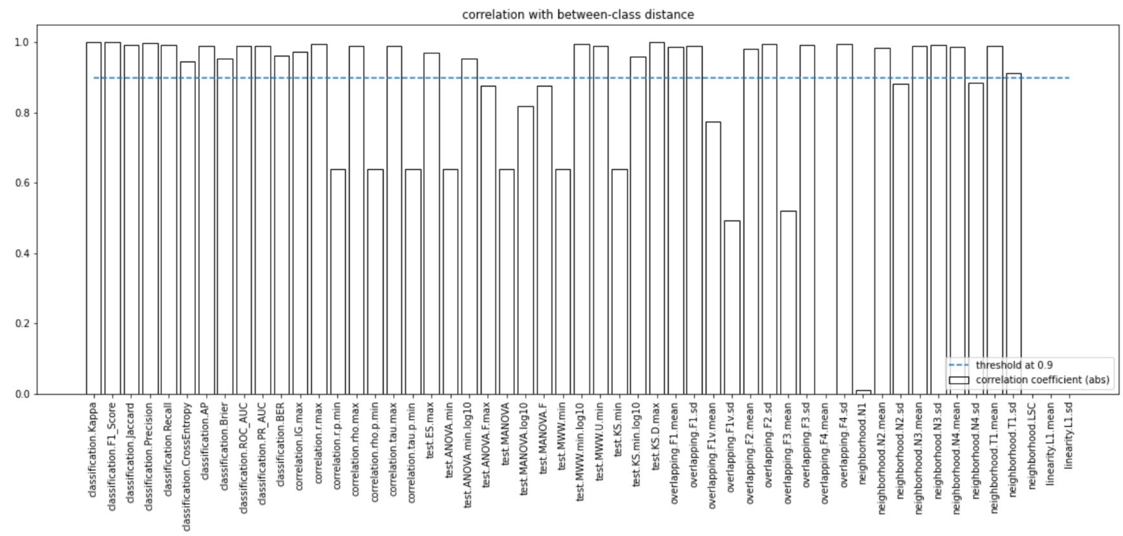
```
dcts = simulate(np.linspace(0, 4, 9), repeat = 50, nobs = 200)
visualize_dcts(dcts)
```

```
100%|██████████████████████████████████████████████| 450/450 [22:52<00:00,  3.05s/it]
```



In [58]:
```
visualize_corr_matrix(dcts, 'coolwarm')
```


classification.ACC
classification.Kappa
classification.F1_Score
classification.Jaccard
classification.Precision
classification.Recall
classification.CrossEntropy
classification.AP
classification.Brier
classification.ROC_AUC
classification.PR_AUC
classification.BER
correlation.IG.max
correlation.r.max
correlation.r.p.min
correlation.rho.max
correlation.rho.p.min
correlation.tau.max
correlation.tau.p.min
test.ES.max
test.ANOVA.min
test.ANOVA.min.log10
test.ANOVA.F.max
test.MANOVA
test.MANOVA.log10
test.MANOVA.F
test.MWW.min
test.MWW.min.log10
test.MWW.U.min
test.KS.min
test.KS.min.log10
test.KS.D.max
overlapping.F1.mean
overlapping.F1.sd
overlapping.F1v.mean
overlapping.F1v.sd
overlapping.F2.mean
overlapping.F2.sd
overlapping.F3.mean
overlapping.F3.sd
overlapping.F4.mean
overlapping.F4.sd
neighborhood.N1
neighborhood.N2.mean
neighborhood.N2.sd
neighborhood.N3.mean
neighborhood.N3.sd
neighborhood.N4.mean
neighborhood.N4.sd
neighborhood.T1.mean
neighborhood.T1.sd
neighborhood.LSC
linearity.L1.mean
linearity.L1.sd
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00


correlation with between-class distance
1.0
0.8
0.6
0.4
0.2
0.0
threshold at 0.9
correlation coefficient (abs)

```
Metrics above the threshold:  ['classification.Kappa' 'classificat
ion.F1_Score' 'classification.Jaccard'
 'classification.Precision' 'classification.Recall'
 'classification.CrossEntropy' 'classification.AP' 'classificatio
```

```
pca = extract_PC(dcts)
pca.explained_variance_ratio_, pca.components_[0]
```

Appendix: Libraries and versions

In [53]:

```
import sklearn
import scipy
import statsmodels
import numpy
import pandas
import rpy2
import seaborn
import matplotlib

print("sklearn", sklearn.__version__, "\n",
      "scipy", scipy.__version__, "\n",
      "statsmodels", statsmodels.__version__,  "\n",
      "numpy", numpy.__version__, "\n",
      "pandas", pandas.__version__, "\n",
      "matplotlib", matplotlib.__version__, "\n",
      "seaborn", seaborn.__version__, "\n",
      "rpy2", rpy2.__version__)

print("ECoL (R package) 0.4.2")
```

```
sklearn 0.24.1
 scipy 1.6.2
 statsmodels 0.12.2
 numpy 1.19.2
 pandas 1.2.3
 matplotlib 3.3.4
 seaborn 0.11.1
 rpy2 3.4.4
ECoL (R package) 0.4.2
```

Appendix: py script used as a call interface for upper applications

```
from pyCLAMs.pyCLAMs import *
import sys
import json
import uuid
import os

def generate(d, n):

    X,y = mvg(nobs = n, md = d)

    # get the local file path
    fn = os.path.dirname(os.path.realpath(__file__)) + "/" + str(uuid.u
uid4()) + ".csv"

    # save to csv file
    save_file(X,y, fn)

    return fn

def analyze(csv):

    # X,y = load_file(csv)
    # s = get_html(X,y)

    # store html result into a local html file
    fn = os.path.dirname(os.path.realpath(__file__)) + "/" + str(uuid.u
uid4()) + ".html"
    with open(fn, 'w') as f:
```

Trouble Shooting

1. module 'rpy2.robjects.conversion' has no attribute 'py2rpy'

> pip insall rpy2==3.4.4 # use the correct version

2. R_HOME must be set in the environment or Registry

```
Install R
Create R_HOME system variable
Add R_HOME\bin to the PATH, in order to execute R from python
Add R_HOME\bin\x64 to the PATH, in order to load R.dll
Install package tzlocal
May also need to reinstall rpy2
```

3. unable to initialize the JIT

```
Happens only on Windows Server OS. Remains to research
```

Appendix: wCLAMs - A web GUI tool based on pyCLAMs

Appendix: Code Ocean Capsule (latest code)

TODO in next vers.

Cochran's Q Test pred vs actual
chi-square test, require feature to be boolean