Assignment 2: Evaluation and Decision Trees

In this assignment, we'll look at different ways of splitting data for training and testing, assessing the predictions made by data mining tools, and then explore classification using decision trees. Like Assignment 1, you will be expected to complete the assignment by answering the questions in Markdown cells and supporting your answer with corresponding code cells.

Please submit your solution as two iPython notebooks (for Part 1 and Part 2) and two PDF files (for Part 1 and Part 2) generated from your iPython notebook that show all code execution results. The due date for this assignment is officially 1/27/16 at 11:59pm, but this may be extended based on the feedback I receive on the eCommons survey about projects and course pace, so keep an eye out for that survey.

Part 1: Cross-validation and Evaluation Metrics (40 points)

Let's start by checking out the dataset evaluation tools availble in scikit-learn. The cross-validation (http://scikit-learn.org/stable/modules/cross_validation.html#cross-validation) documentation will be helpful to review before completing this part of the assignment. We'll work with two datasets, one generated and one real dataset, and then look at different ways of splitting that data into train and test sets, as well as computing evaluation metrics on each of the datasets.

In [45]:
```
## Preliminaries

#Show plots in the notebook
%matplotlib inline

from sklearn import datasets, preprocessing, cross_validation, metrics,
from scipy import stats
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import urllib2
```

In [46]:
```
# Let's get our first dataset, the Iris data that's the default choise f
iris = datasets.load_iris()

# Let's also generate some data. See http://scikit-learn.org/stable/modu
(gendata_d, gendata_t) = datasets.make_classification(n_samples=500, ran
gendata_data = pd.DataFrame(gendata_d)
gendata_targets = pd.DataFrame(gendata_t)
```

In [47]:
```
# We'll start with the simplest holdout procedure: a single train-test s
[iris_train_data, iris_test_data,  iris_train_labels, iris_test_labels]

# Some notes on how this works:
## There are four outputs. The first is the dataframe containing the tra
## The second output is the dataframe containing the testing data (attri
## are the training labels (used to train your classifier) and the testi

## The first two arguments to the split are all the data (attributes onl
## The test size is the fraction of data included in the test set, 25% i
## Since we're randomly sampling data, there's a chance every student wo
##  To avoid this, we pass a random_state (20160121) to make sure the "r

#Let's see how big the train and test sets are:
print iris_train_data.shape, iris_test_data.shape, iris_train_labels.sha
```

```
(112, 4) (38, 4) (112,) (38,)
```

Question 1: Creating a simple train-test split (5 points)

Use the `train_test_split` function (see the [documentation (http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html#sklearn.cross_v](http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html#sklearn.cross_v) to generate training and testing sets for the Iris and generated data.

1. Generate splits with 10%, 33%, and 50% test data for both datasets (using `random_state` parameter)
2. Compute the percentage of the training labels and testing labels that belong to each class fo the distribution of labels appear to be well-matched between the train and test set?
3. Compute the descriptive statistics for each feature in the train and test set for the 10% split. the biggest difference in mean value across datasets.

Question 1 Answers

1. see code below

2. For the IRIS dataset the distribution of the labels appears to be well-matched for the 10% and 50% split, but unbalanced for the 33% split
For the RAND Dataset the distribution of the labels does not appear to be as closely matched as the IRIS dataset. While the values are relatively close to each other, the test labels distribution does not match the train set label distribution

3. For the IRIS data set the biggest difference in mean value is the 3rd feature petal width: 1.2074 -> 1.12
For the RAND dataset the biggest difference in mean value is in attribue 8 : 0.018226 -> -0.30529
See cell below for better formatting

```
2.

--------- IRIS DATASET ----------
Train data split 10%:
0: 45 / 135 = 33.33%
1: 45 / 135 = 33.33%
2: 45 / 135 = 33.33%
Test data split 10%:
0: 5 / 15 = 33.33%
1: 5 / 15 = 33.33%
2: 5 / 15 = 33.33%

Train data split 33%:
0: 32 / 100 = 32.0%
1: 34 / 100 = 34.0%
2: 34 / 100 = 34.0%
Test data split 33%:
0: 18 / 50 = 36.0%
1: 16 / 50 = 32.0%
2: 16 / 50 = 32.0%

Train data split 50%:
0: 25 / 75 = 33.33%
1: 25 / 75 = 33.33%
2: 25 / 75 = 33.33%
Test data split 50%:
0: 25 / 75 = 33.33%
1: 25 / 75 = 33.33%
2: 25 / 75 = 33.33%


--------- RAND DATASET ----------
Train data split 10%:
0: 226 / 450 = 50.22%
1: 224 / 450 = 49.78%
Test data split 10%:
0: 23 / 50   = 46.0%
1: 27 / 50   = 54.0%

Train data split 33%:
0: 171 / 335 = 51.04%
1: 164 / 335 = 48.96%
Test data split 33%:
0: 87 / 165  = 52.73%
1: 78 / 165  = 47.27%

Train data split 50%:
0: 129 / 250 = 51.6%
1: 121 / 250 = 48.4%
Test data split 50%:
0: 120 / 250 = 49.2%
1: 130 / 250 = 50.8%

3. Compute the descriptive statistics for each feature in the train
```

```
and test set for the 10% split. Which feature has the biggest
difference in mean value across datasets.
For the IRIS data set the biggest difference in mean value is the 3rd
feature petal width: 1.2074 -> 1.12
For the RAND dataset the biggest difference in mean value is in
attribue 8 : 0.018226 -> -0.30529
```

In [48]:

```python
# Iris CV
[iris_train_data_10, iris_test_data_10,  iris_train_labels_10, iris_test
[iris_train_data_33, iris_test_data_33,  iris_train_labels_33, iris_test
[iris_train_data_50, iris_test_data_50,  iris_train_labels_50, iris_test

# Gendata CV
[gendata_train_data_10, gendata_test_data_10,  gendata_train_labels_10,
[gendata_train_data_33, gendata_test_data_33,  gendata_train_labels_33,
[gendata_train_data_50, gendata_test_data_50,  gendata_train_labels_50,
```

In [49]:

```
all_datasets = {'gendata_50': { 'train': gendata_train_labels_50, 'test'
                'gendata_33': { 'train': gendata_train_labels_33, 'test'
                'gendata_10': { 'train': gendata_train_labels_10, 'test'
                'iris_50': { 'train': iris_train_labels_50, 'test': iris
                'iris_33': { 'train': iris_train_labels_33, 'test': iris
                'iris_10': { 'train': iris_train_labels_10, 'test': iris
               }
for dataset in all_datasets.keys():
    print dataset
    print all_datasets[dataset]['train'][0].value_counts(sort=False)
    print all_datasets[dataset]['test'][0].value_counts(sort=False)
    print "\n"
```

```
gendata_50
0    129
1    121
Name: 0, dtype: int64
0    120
1    130
Name: 0, dtype: int64


iris_33
0    32
1    34
2    34
Name: 0, dtype: int64
0    18
1    16
2    16
Name: 0, dtype: int64


gendata_10
0    226
1    224
Name: 0, dtype: int64
0    23
1    27
Name: 0, dtype: int64


iris_10
0    45
1    45
2    45
Name: 0, dtype: int64
0    5
1    5
2    5
Name: 0, dtype: int64


iris_50
0    25
1    25
2    25
Name: 0, dtype: int64
0    25
1    25
2    25
Name: 0, dtype: int64


gendata_33
0    171
1    164
```

```
Name: 0, dtype: int64
0    78
1    87
Name: 0, dtype: int64
```

In [50]:
```
print iris_train_data_10.describe()
print iris_test_data_10.describe()

print gendata_train_data_10.describe()
print gendata_test_data_10.describe()
```

```
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.842963    3.062222    3.766667    1.207407
std      0.837867    0.422124    1.776295    0.760953
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.846667   2.980000   3.686667   1.120000
std     0.760514   0.537454   1.710834   0.805517
min     4.800000   2.200000   1.400000   0.100000
25%     5.150000   2.600000   1.550000   0.300000
50%     5.800000   3.100000   4.100000   1.000000
75%     6.400000   3.150000   5.000000   1.750000
max     7.100000   4.100000   5.900000   2.300000
               0           1           2           3           4
5   \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.015595   -0.007781    0.013634    0.016649    0.093196
0.045582
std      1.014951    1.001806    0.962817    1.008849    1.039005
0.997071
min     -2.515058   -2.433603   -2.647789   -2.518441   -3.223148
-2.725530
25%     -0.661821   -0.739336   -0.605543   -0.666526   -0.635592
-0.627144
50%     -0.010015   -0.072732    0.020328    0.024608    0.113445
0.105096
75%      0.719686    0.548756    0.641902    0.702521    0.748359
0.723925
max      3.594499    3.039572    3.003953    2.619359    2.712419
3.432993

               6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.001562   -0.012272    0.018226   -0.007054    0.010869
-0.076953
std      1.182731    0.997360    1.012774    1.010068    1.164712
0.955194
min     -3.326552   -2.736625   -2.807253   -2.690036   -3.315302
-3.102408
25%     -1.024900   -0.691724   -0.649506   -0.781671   -0.836271
-0.712939
50%     -0.060348    0.015647   -0.005171    0.009211    0.078181
-0.084789
75%      0.911349    0.633550    0.706345    0.701114    1.038988
0.629994
max      3.023143    2.654571    3.080764    3.135951    2.544278
```

```
2.664877

                12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.014286   -0.013744    0.111635   -0.001662   -0.010041
-0.050198
std      0.989100    1.230692    0.949534    0.948191    1.016657
0.991363
min     -3.109823   -3.210675   -3.239524   -2.675548   -3.068310
-2.883941
25%     -0.638233   -0.998816   -0.512089   -0.664533   -0.771602
-0.746687
50%     -0.032929   -0.127085    0.134519    0.051188   -0.012593
-0.007451
75%      0.740198    1.013260    0.798428    0.672380    0.679608
0.617835
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

                18          19
count  450.000000  450.000000
mean     0.022037    0.053671
std      1.045171    1.017259
min     -3.389571   -3.110583
25%     -0.647093   -0.602597
50%      0.056958    0.051361
75%      0.726783    0.670130
max      2.811399    3.498259
               0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.132358   0.219547  -0.181742   0.144902  -0.108895   0.005
421
std     1.084205   1.043357   0.781276   1.108929   1.050024   1.036
938
min    -1.815645  -1.615432  -1.849257  -2.530270  -2.137125  -2.966
130
25%    -0.598300  -0.469860  -0.738886  -0.589564  -0.787458  -0.524
189
50%     0.035678   0.111592  -0.088987   0.311619  -0.221786  -0.038
448
75%     1.017673   0.753362   0.423989   0.893433   0.562336   0.811
786
max     2.247138   3.012955   1.041469   2.479612   2.307088   1.835
230

               6          7          8          9          10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.291133  -0.143091  -0.305290   0.092221  -0.213771  -0.003
```

```
974
std      1.128013   1.092923   1.038049   1.048631   1.188927   0.979
900
min     -2.798720  -2.295046  -3.060349  -3.021813  -3.265268  -2.235
046
25%     -0.959467  -0.811699  -1.045868  -0.512540  -0.802693  -0.696
424
50%     -0.551110  -0.280232  -0.382899   0.011575  -0.105014  -0.127
539
75%      0.556449   0.413665   0.467908   0.580116   0.673023   0.626
365
max      2.639439   2.731177   1.666533   2.726222   1.730126   2.144
330

                12         13         14         15         16
17  \
count   50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    -0.299245   0.070258   0.259905  -0.131053   0.083768  -0.135
486
std      0.941931   1.125245   1.096413   1.001909   1.087815   0.930
853
min     -2.504971  -2.123482  -2.477589  -3.404818  -2.500514  -2.049
138
25%     -0.890362  -0.762860  -0.374569  -0.743837  -0.428832  -0.838
399
50%     -0.216580  -0.243333   0.136180  -0.093642   0.003066  -0.091
279
75%      0.156539   1.020259   0.900320   0.613982   0.963794   0.371
342
max      1.774276   2.097715   2.760497   1.719203   2.891136   1.863
623

                18         19
count   50.000000  50.000000
mean     0.029602  -0.150480
std      0.772789   1.055486
min     -1.341417  -3.335489
25%     -0.555508  -0.659712
50%      0.122657  -0.230268
75%      0.599628   0.493884
max      1.768622   2.561847
```

The next set of cross-validation split generators deal with folds - so instead of giving you a single set of outputs, they'll return an *iterator* that produces a series of train and test splits.

Question 2: Cross-validation with k folds (10 points)

Use the `KFold` cross-validation function to generate multiple train-test splits from each dataset

1. Generate 3, 5, and 10 folds for each dataset (using `random_state=20160121` as a parameter)
2. Compute the percentage of the training labels and testing labels that belong to each class for each set of folds. Does the distribution of labels appear to be well-matched between the train and test set?
3. Compute the mean value for each feature in the train and test set for each fold in the 10-fold split. Which feature has the largest difference between train and test for each of the ten folds? What is the largest difference of the averaged means for the ten folds?

Question 2 Answers

1. See bellow

2. The percentagaes seem to appear well matched between train and test for all the different folds

3. For IRIS dataset the largest difference between train and test mean is **Fold 6** and feature 3 (petal length) with difference of 0.69037
Largest difference in the averaged means for the ten folds is in feature 2, sepal width

```
Question 2 Answers

The percentagaes of the training labels and testing labels seem to
appear well matched between train and test for all the different
folds
1,2
Dataset: iris - nfold: 1/3
Train data
0    32 / 100 = 32%
1    34 / 100 = 34%
2    34 / 100 = 34%
Name: 0, dtype: int64
Test data
0    18 / 50 = 36%
1    16 / 50 = 32%
2    16 / 50 = 32%
Name: 0, dtype: int64


Dataset: iris - nfold: 2/3
Train data
0    35 / 100 = 35%
1    35 / 100 = 35%
2    30 / 100 = 30%
Name: 0, dtype: int64
Test data
0    15 / 50 = 30%
1    15 / 50 = 30%
2    20 / 50 = 40%
Name: 0, dtype: int64


Dataset: iris - nfold: 3/3
Train data
0    33 / 100 = 33%
1    31 / 100 = 31%
2    36 / 100 = 36%
Name: 0, dtype: int64
Test data
0    17 / 50 = 34%
1    19 / 50 = 38%
2    14 / 50 = 28%
Name: 0, dtype: int64


Dataset: iris - nfold: 1/5
Train data
0    40 / 120 = 33.33%
1    39 / 120 = 32.5%
2    41 / 120 = 34.17
Name: 0, dtype: int64
Test data
0    10 / 30 = 33.33%
1    11 / 30 = 36.67%
```

```
2     9 / 30 = 30%
Name: 0, dtype: int64


Dataset: iris - nfold: 2/5
Train data
0    40 / 120 = 33.33%
1    39 / 120 = 32.5%
2    41 / 120 = 34.17%
Name: 0, dtype: int64
Test data
0    10 / 30 = 33.33%
1    11 / 30 = 36.67%
2     9 / 30 = 30%
Name: 0, dtype: int64


Dataset: iris - nfold: 3/5
Train data
0    42 / 120 = 35%
1    42 / 120 = 35%
2    36 / 120 = 30%
Name: 0, dtype: int64
Test data
0     8 / 30 = 26.67%
1     8 / 30 = 26.67%
2    14 / 30 = 46.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 4/5
Train data
0    39 / 120 = 32.5%
1    42 / 120 = 35%
2    39 / 120 = 32.5%
Name: 0, dtype: int64
Test data
0    11 / 30 = 36.67%
1     8 / 30 = 26.67%
2    11 / 30 = 36.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 5/5
Train data
0    39 / 120 = 32.5%
1    38 / 120 = 31.67%
2    43 / 120 = 35.83%
Name: 0, dtype: int64
Test data
0    11 / 30 = 36.67%
1    12 / 30 = 40%
2     7 / 30 = 23.33%
Name: 0, dtype: int64
```

```
Dataset: iris - nfold: 1/10
Train data
0    45 / 135 = 33.33%
1    45 / 135 = 33.33%
2    45 / 135 = 33.33%
Name: 0, dtype: int64
Test data
0    5 / 15 = 33.33%
1    5 / 15 = 33.33%
2    5 / 15 = 33.33%
Name: 0, dtype: int64


Dataset: iris - nfold: 2/10
Train data
0    45 / 135 = 33.33%
1    44 / 135 = 32.59%
2    46 / 135 = 34.08%
Name: 0, dtype: int64
Test data
0    5 / 15 = 33.33%
1    6 / 15 = 40%
2    4 / 15 = 26.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 3/10
Train data
0    43 / 135 = 31.85%
1    47 / 135 = 34.81%
2    45 / 135 = 33.33%
Name: 0, dtype: int64
Test data
0    7 / 15 = 46.67%
1    3 / 15 = 20%
2    5 / 15 = 33.33%
Name: 0, dtype: int64


Dataset: iris - nfold: 4/10
Train data
0    47 / 135 = 34.81%
1    42 / 135 = 31.11%
2    46 / 135 = 34.08%
Name: 0, dtype: int64
Test data
0    3 / 15 = 20%
1    8 / 15 = 53.33%
2    4 / 15 = 26.67%
Name: 0, dtype: int64
```

```
Dataset: iris - nfold: 5/10
Train data
0    45 / 135 = 33.33%
1    47 / 135 = 34.81%
2    43 / 135 = 31.86%
Name: 0, dtype: int64
Test data
0    5 / 15 = 33.33%
1    3 / 15 = 20%
2    7 / 15 = 46.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 6/10
Train data
0    47 / 135 = 34.81%
1    45 / 135 = 33.33%
2    43 / 135 = 31.86%
Name: 0, dtype: int64
Test data
0    3 / 15 = 20%
1    5 / 15 = 33.33%
2    7 / 15 = 46.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 7/10
Train data
0    44 / 135 = 32.59%
1    48 / 135 = 35.55%
2    43 / 135 = 31.85%
Name: 0, dtype: int64
Test data
0    6 / 15 = 40%
1    2 / 15 = 13.33%
2    7 / 15 = 46.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 8/10
Train data
0    45 / 135 = 33.33%
1    44 / 135 = 32.59%
2    46 / 135 = 34.08%
Name: 0, dtype: int64
Test data
0    5 / 15 = 33.33%
1    6 / 15 = 40%
2    4 / 15 = 26.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 9/10
Train data
```

```
0    43 / 135 = 31.85%
1    43 / 135 = 31.85%
2    49 / 135 = 36.30%
Name: 0, dtype: int64
Test data
0    7 / 15 = 46.67%
1    7 / 15 = 46.67%
2    1 / 15 = 6.67%
Name: 0, dtype: int64


Dataset: iris - nfold: 10/10
Train data
0    46 / 135 = 34.07%
1    45 / 135 = 33.33%
2    44 / 135 = 32.59%
Name: 0, dtype: int64
Test data
0    4 / 15 = 26.67%
1    5 / 15 = 33.33%
2    6 / 15 = 40%
Name: 0, dtype: int64



Dataset: gen - nfold: 1/3
Train data
0    170 / 333 = 51.05%
1    163 / 333 = 48.95%
Name: 0, dtype: int64
Test data
0    79 / 167 = 47.31%
1    88 / 167 = 52.69%
Name: 0, dtype: int64


Dataset: gen - nfold: 2/3
Train data
0    168 / 333 = 50.45%
1    165 / 333 = 49.55%
Name: 0, dtype: int64
Test data
0    81 / 167 = 48.50%
1    86 / 167 = 51.50%
Name: 0, dtype: int64


Dataset: gen - nfold: 3/3
Train data
0    160 / 333 = 48.05%
1    174 / 333 = 51.95%
Name: 0, dtype: int64
Test data
0    89 / 167 = 53.29%
```

```
1    77 / 167 = 46.61%
Name: 0, dtype: int64


Dataset: gen - nfold: 1/5
Train data
0    202 / 400 = 50.5%
1    198 / 400 = 49.5%
Name: 0, dtype: int64
Test data
0    47 / 100 = 47%
1    53 / 100 = 53%
Name: 0, dtype: int64


Dataset: gen - nfold: 2/5
Train data
0    204 / 400 = 51%
1    196 / 400 = 49%
Name: 0, dtype: int64
Test data
0    45 / 100 = 45%
1    55 / 100 = 55%
Name: 0, dtype: int64


Dataset: gen - nfold: 3/5
Train data
0    198 / 400 = 49.5%
1    202 / 400 = 50.5%
Name: 0, dtype: int64
Test data
0    51 / 100 = 51%
1    49 / 100 = 49%
Name: 0, dtype: int64


Dataset: gen - nfold: 4/5
Train data
0    195 / 400 = 48.75%
1    205 / 400 = 51.25%
Name: 0, dtype: int64
Test data
0    54 / 100 = 54%
1    46 / 100 = 46%
Name: 0, dtype: int64


Dataset: gen - nfold: 5/5
Train data
0    197 / 400 = 49.25%
1    203 / 400 = 51.65%
Name: 0, dtype: int64
Test data
```

```
0    52 / 100 = 52%
1    48 / 100 = 48%
Name: 0, dtype: int64


Dataset: gen - nfold: 1/10
Train data
0    226 / 450 = 50.22%
1    224 / 450 = 49.78%
Name: 0, dtype: int64
Test data
0    23 / 50 = 46%
1    27 / 50 = 54%
Name: 0, dtype: int64


Dataset: gen - nfold: 2/10
Train data
0    225 / 450 = 50%
1    225 / 450 = 50%
Name: 0, dtype: int64
Test data
0    24 / 50 = 48%
1    26 / 50 = 52%
Name: 0, dtype: int64


Dataset: gen - nfold: 3/10
Train data
0    226 / 450 = 50.22%
1    224 / 450 = 49.78%
Name: 0, dtype: int64
Test data
0    23 / 50 = 46%
1    27 / 50 = 54%
Name: 0, dtype: int64


Dataset: gen - nfold: 4/10
Train data
0    227 / 450 = 50.44%
1    223 / 450 = 49.56%
Name: 0, dtype: int64
Test data
0    22 / 50 = 44%
1    28 / 50 = 56%
Name: 0, dtype: int64


Dataset: gen - nfold: 5/10
Train data
0    221 / 450 = 49.11%
1    229 / 450 = 50.89%
Name: 0, dtype: int64
Test data
```

```
Test data
0    28 / 50 = 56%
1    22 / 50 = 44%
Name: 0, dtype: int64


Dataset: gen - nfold: 6/10
Train data
0    226 / 450 = 50.22%
1    224 / 450 = 49.78%
Name: 0, dtype: int64
Test data
0    23 / 50 = 46%
1    27 / 50 = 54%
Name: 0, dtype: int64


Dataset: gen - nfold: 7/10
Train data
0    224 / 450 = 49.78%
1    226 / 450 = 50.22%
Name: 0, dtype: int64
Test data
0    25 / 50 = 50%
1    25 / 50 = 50%
Name: 0, dtype: int64


Dataset: gen - nfold: 8/10
Train data
0    220 / 450 = 48.89%
1    230 / 450 = 51.11%
Name: 0, dtype: int64
Test data
0    29 / 50 = 58%
1    21 / 50 = 42%
Name: 0, dtype: int64


Dataset: gen - nfold: 9/10
Train data
0    224 / 450 = 49.78%
1    226 / 450 = 50.22%
Name: 0, dtype: int64
Test data
0    25 / 50 = 50%
1    25 / 50 = 50%
Name: 0, dtype: int64


Dataset: gen - nfold: 10/10
Train data
0    222 / 450 = 49.33%
1    228 / 450 = 50.67%
Name: 0, dtype: int64
```

```
Name: 0, dtype: int64
Test data
0    27 / 50 = 54%
1    23 / 50 = 46%
Name: 0, dtype: int64


3. For IRIS dataset the largest difference between train and test
mean is Fold 6 and featre 3 (petal length) with difference of 0.69037
Largest difference in the averaged means for the ten folds is in
feature 2, sepal width
```

In [51]:
```
from sklearn.cross_validation import KFold

all_datasets = {'iris': {'data': pd.DataFrame(iris.data), 'labels': pd.D
            'gen': { 'data': gendata_data, 'labels': gendata_targets} }

for dataset in all_datasets.keys():
    for nfolds in [3, 5, 10]:
        kfolds = cross_validation.KFold(all_datasets[dataset]['labels'].
        fold = 0
        for train, test in kfolds:
                fold += 1
#                 print ("%s %s" % (train,test))
                if (dataset == 'iris'):
                    train_fold = pd.DataFrame(iris.target[train])
                    test_fold = pd.DataFrame(iris.target[test])
                else:
                    train_fold = pd.DataFrame(gendata_t[train])
                    test_fold = pd.DataFrame(gendata_t[test])

                print "Dataset: " + dataset + " - nfold: " + str(fold) +
                print "Train data"
                print train_fold[0].value_counts(sort=False)
                print "Test data"
                print test_fold[0].value_counts(sort=False)
                print "\n"
```

```
Dataset: iris - nfold: 1/3
Train data
0    32
1    34
2    34
Name: 0, dtype: int64
Test data
0    18
1    16
2    16
Name: 0, dtype: int64


Dataset: iris - nfold: 2/3
Train data
0    35
1    35
2    30
Name: 0, dtype: int64
Test data
0    15
1    15
2    20
Name: 0, dtype: int64


Dataset: iris - nfold: 3/3
Train data
0    33
1    31
2    36
Name: 0, dtype: int64
Test data
0    17
1    19
2    14
Name: 0, dtype: int64


Dataset: iris - nfold: 1/5
Train data
0    40
1    39
2    41
Name: 0, dtype: int64
Test data
0    10
1    11
2     9
Name: 0, dtype: int64


Dataset: iris - nfold: 2/5
Train data
```

```
0    40
1    39
2    41
Name: 0, dtype: int64
Test data
0    10
1    11
2     9
Name: 0, dtype: int64


Dataset: iris - nfold: 3/5
Train data
0    42
1    42
2    36
Name: 0, dtype: int64
Test data
0     8
1     8
2    14
Name: 0, dtype: int64


Dataset: iris - nfold: 4/5
Train data
0    39
1    42
2    39
Name: 0, dtype: int64
Test data
0    11
1     8
2    11
Name: 0, dtype: int64


Dataset: iris - nfold: 5/5
Train data
0    39
1    38
2    43
Name: 0, dtype: int64
Test data
0    11
1    12
2     7
Name: 0, dtype: int64


Dataset: iris - nfold: 1/10
Train data
0    45
1    45
```

```
2    45
Name: 0, dtype: int64
Test data
0    5
1    5
2    5
Name: 0, dtype: int64


Dataset: iris - nfold: 2/10
Train data
0    45
1    44
2    46
Name: 0, dtype: int64
Test data
0    5
1    6
2    4
Name: 0, dtype: int64


Dataset: iris - nfold: 3/10
Train data
0    43
1    47
2    45
Name: 0, dtype: int64
Test data
0    7
1    3
2    5
Name: 0, dtype: int64


Dataset: iris - nfold: 4/10
Train data
0    47
1    42
2    46
Name: 0, dtype: int64
Test data
0    3
1    8
2    4
Name: 0, dtype: int64


Dataset: iris - nfold: 5/10
Train data
0    45
1    47
2    43
Name: 0, dtype: int64
```

```
Test data
0    5
1    3
2    7
Name: 0, dtype: int64


Dataset: iris - nfold: 6/10
Train data
0    47
1    45
2    43
Name: 0, dtype: int64
Test data
0    3
1    5
2    7
Name: 0, dtype: int64


Dataset: iris - nfold: 7/10
Train data
0    44
1    48
2    43
Name: 0, dtype: int64
Test data
0    6
1    2
2    7
Name: 0, dtype: int64


Dataset: iris - nfold: 8/10
Train data
0    45
1    44
2    46
Name: 0, dtype: int64
Test data
0    5
1    6
2    4
Name: 0, dtype: int64


Dataset: iris - nfold: 9/10
Train data
0    43
1    43
2    49
Name: 0, dtype: int64
Test data
0    7
```

```
1    7
2    1
Name: 0, dtype: int64


Dataset: iris - nfold: 10/10
Train data
0    46
1    45
2    44
Name: 0, dtype: int64
Test data
0    4
1    5
2    6
Name: 0, dtype: int64


Dataset: gen - nfold: 1/3
Train data
0    170
1    163
Name: 0, dtype: int64
Test data
0    79
1    88
Name: 0, dtype: int64


Dataset: gen - nfold: 2/3
Train data
0    168
1    165
Name: 0, dtype: int64
Test data
0    81
1    86
Name: 0, dtype: int64


Dataset: gen - nfold: 3/3
Train data
0    160
1    174
Name: 0, dtype: int64
Test data
0    89
1    77
Name: 0, dtype: int64


Dataset: gen - nfold: 1/5
Train data
0    202
```

```
1    198
Name: 0, dtype: int64
Test data
0    47
1    53
Name: 0, dtype: int64


Dataset: gen - nfold: 2/5
Train data
0    204
1    196
Name: 0, dtype: int64
Test data
0    45
1    55
Name: 0, dtype: int64


Dataset: gen - nfold: 3/5
Train data
0    198
1    202
Name: 0, dtype: int64
Test data
0    51
1    49
Name: 0, dtype: int64


Dataset: gen - nfold: 4/5
Train data
0    195
1    205
Name: 0, dtype: int64
Test data
0    54
1    46
Name: 0, dtype: int64


Dataset: gen - nfold: 5/5
Train data
0    197
1    203
Name: 0, dtype: int64
Test data
0    52
1    48
Name: 0, dtype: int64


Dataset: gen - nfold: 1/10
Train data
```

```
0    226
1    224
Name: 0, dtype: int64
Test data
0    23
1    27
Name: 0, dtype: int64


Dataset: gen - nfold: 2/10
Train data
0    225
1    225
Name: 0, dtype: int64
Test data
0    24
1    26
Name: 0, dtype: int64


Dataset: gen - nfold: 3/10
Train data
0    226
1    224
Name: 0, dtype: int64
Test data
0    23
1    27
Name: 0, dtype: int64


Dataset: gen - nfold: 4/10
Train data
0    227
1    223
Name: 0, dtype: int64
Test data
0    22
1    28
Name: 0, dtype: int64


Dataset: gen - nfold: 5/10
Train data
0    221
1    229
Name: 0, dtype: int64
Test data
0    28
1    22
Name: 0, dtype: int64


Dataset: gen - nfold: 6/10
```

```
Train data
0    226
1    224
Name: 0, dtype: int64
Test data
0    23
1    27
Name: 0, dtype: int64


Dataset: gen - nfold: 7/10
Train data
0    224
1    226
Name: 0, dtype: int64
Test data
0    25
1    25
Name: 0, dtype: int64


Dataset: gen - nfold: 8/10
Train data
0    220
1    230
Name: 0, dtype: int64
Test data
0    29
1    21
Name: 0, dtype: int64


Dataset: gen - nfold: 9/10
Train data
0    224
1    226
Name: 0, dtype: int64
Test data
0    25
1    25
Name: 0, dtype: int64


Dataset: gen - nfold: 10/10
Train data
0    222
1    228
Name: 0, dtype: int64
Test data
0    27
1    23
Name: 0, dtype: int64
```

In [52]:
```
from sklearn.cross_validation import KFold

all_datasets = {'iris': {'data': pd.DataFrame(iris.data), 'labels': pd.D
            'gen': { 'data': gendata_data, 'labels': gendata_targets} }

for dataset in all_datasets.keys():
    for nfolds in [10]:
        kfolds = cross_validation.KFold(all_datasets[dataset]['labels'].
        fold = 0
        for train, test in kfolds:
                fold += 1
#                 print ("%s %s" % (train,test))
                if (dataset == 'iris'):
                    train_fold = pd.DataFrame(iris.data[train])
                    test_fold = pd.DataFrame(iris.data[test])
                else:
                    train_fold = pd.DataFrame(gendata_d[train])
                    test_fold = pd.DataFrame(gendata_d[test])

                print "Dataset: " + dataset + " - nfold: " + str(fold) +
                print "Train data"
                print train_fold.describe()
                print "Test data"
                print test_fold.describe()
                print "\n"
```

```
Dataset: iris - nfold: 1/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.842963    3.062222    3.766667    1.207407
std      0.837867    0.422124    1.776295    0.760953
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.846667   2.980000   3.686667   1.120000
std     0.760514   0.537454   1.710834   0.805517
min     4.800000   2.200000   1.400000   0.100000
25%     5.150000   2.600000   1.550000   0.300000
50%     5.800000   3.100000   4.100000   1.000000
75%     6.400000   3.150000   5.000000   1.750000
max     7.100000   4.100000   5.900000   2.300000


Dataset: iris - nfold: 2/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.871852    3.054815    3.796296    1.205926
std      0.846296    0.438062    1.789622    0.769063
min      4.400000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.450000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.586667   3.046667   3.420000   1.133333
std     0.603403   0.405087   1.530266   0.729644
min     4.300000   2.500000   1.100000   0.100000
25%     5.250000   2.800000   1.550000   0.300000
50%     5.600000   3.000000   3.900000   1.300000
75%     6.050000   3.200000   4.650000   1.650000
max     6.700000   3.800000   5.200000   2.300000


Dataset: iris - nfold: 3/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.866667    3.039259    3.811852    1.209630
std      0.827602    0.422165    1.749800    0.754773
min      4.300000    2.000000    1.000000    0.100000
25%      5.150000    2.800000    1.600000    0.300000
```

```
50%       5.800000    3.000000    4.400000    1.300000
75%       6.450000    3.250000    5.100000    1.800000
max       7.900000    4.200000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.633333   3.186667   3.280000   1.100000
std     0.830376   0.523541   1.885357   0.856905
min     4.800000   2.300000   1.300000   0.200000
25%     4.950000   3.000000   1.450000   0.300000
50%     5.500000   3.300000   3.300000   1.000000
75%     6.300000   3.450000   5.000000   1.750000
max     7.700000   4.400000   6.100000   2.400000


Dataset: iris - nfold: 4/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.837778    3.054074    3.726667    1.195556
std      0.826146    0.440279    1.785062    0.781776
min      4.300000    2.000000    1.100000    0.100000
25%      5.100000    2.800000    1.550000    0.300000
50%      5.800000    3.000000    4.200000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.700000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.893333   3.053333   4.046667   1.226667
std     0.872981   0.381476   1.592333   0.589754
min     4.600000   2.300000   1.000000   0.200000
25%     5.250000   2.850000   3.750000   1.100000
50%     6.000000   3.000000   4.500000   1.400000
75%     6.350000   3.250000   4.850000   1.550000
max     7.900000   3.800000   6.400000   2.000000


Dataset: iris - nfold: 5/10
Train data
                0           1           2          3
count  135.000000  135.000000  135.000000  135.00000
mean     5.823704    3.051111    3.732593    1.19037
std      0.810400    0.428964    1.738623    0.76274
min      4.300000    2.000000    1.000000    0.10000
25%      5.100000    2.800000    1.600000    0.30000
50%      5.700000    3.000000    4.400000    1.30000
75%      6.400000    3.300000    5.100000    1.80000
max      7.900000    4.400000    6.900000    2.50000
Test data
               0         1          2          3
count  15.000000  15.00000  15.000000  15.000000
mean    6.020000   3.08000   3.993333   1.273333
std     0.987204   0.48873   2.033458   0.789635
```

```
min     4.500000   2.30000   1.200000   0.200000
25%     5.500000   2.75000   1.550000   0.350000
50%     5.900000   3.00000   4.300000   1.500000
75%     6.550000   3.25000   5.650000   1.900000
max     7.700000   4.00000   6.700000   2.200000


Dataset: iris - nfold: 6/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.797778    3.061481    3.689630    1.176296
std      0.808026    0.443188    1.758743    0.770269
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.500000    0.300000
50%      5.700000    3.000000    4.200000    1.300000
75%      6.350000    3.350000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    6.253333   2.986667   4.380000   1.400000
std     0.921076   0.339888   1.751408   0.686607
min     4.400000   2.500000   1.300000   0.200000
25%     5.600000   2.750000   3.450000   1.150000
50%     6.600000   3.000000   4.900000   1.700000
75%     6.750000   3.150000   5.700000   1.800000
max     7.600000   3.800000   6.600000   2.400000


Dataset: iris - nfold: 7/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.853333    3.051111    3.760000    1.196296
std      0.814221    0.445854    1.739411    0.753540
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2         3
count  15.000000  15.000000  15.000000  15.00000
mean    5.753333   3.080000   3.746667   1.22000
std     0.970910   0.312136   2.043060   0.87358
min     4.400000   2.600000   1.300000   0.10000
25%     5.000000   2.800000   1.500000   0.25000
50%     5.700000   3.000000   4.200000   1.30000
75%     6.450000   3.400000   5.400000   2.00000
max     7.700000   3.600000   6.700000   2.50000


Dataset: iris - nfold: 8/10
```

```
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.839259    3.051111    3.760000    1.201481
std      0.843951    0.427046    1.765896    0.764315
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.550000    0.300000
50%      5.800000    3.000000    4.400000    1.300000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.700000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.880000   3.080000   3.746667   1.173333
std     0.691995   0.504551   1.812601   0.778705
min     4.900000   2.600000   1.200000   0.100000
25%     5.550000   2.700000   1.700000   0.350000
50%     5.800000   3.000000   4.100000   1.300000
75%     6.050000   3.150000   5.100000   1.700000
max     7.700000   4.200000   6.900000   2.400000


Dataset: iris - nfold: 9/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.880741    3.062222    3.826667    1.229630
std      0.829538    0.431738    1.773234    0.765592
min      4.300000    2.200000    1.000000    0.100000
25%      5.150000    2.800000    1.600000    0.300000
50%      5.800000    3.000000    4.400000    1.400000
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    5.506667   2.980000   3.146667   0.920000
std     0.759198   0.458569   1.609732   0.704273
min     4.400000   2.000000   1.400000   0.100000
25%     5.000000   2.850000   1.550000   0.250000
50%     5.200000   3.000000   3.500000   1.000000
75%     5.950000   3.200000   4.500000   1.400000
max     6.900000   3.800000   5.700000   2.300000


Dataset: iris - nfold: 10/10
Train data
                0           1           2           3
count  135.000000  135.000000  135.000000  135.000000
mean     5.819259    3.052593    3.716296    1.174074
std      0.835097    0.436887    1.768388    0.749324
min      4.300000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.550000    0.300000
50%      5.700000    3.000000    4.200000    1.300000
```

```
75%      6.400000    3.300000    5.100000    1.800000
max      7.900000    4.400000    6.900000    2.500000
Test data
               0          1          2          3
count  15.000000  15.000000  15.000000  15.000000
mean    6.060000   3.066667   4.140000   1.420000
std     0.752899   0.416905   1.740197   0.875214
min     4.700000   2.200000   1.300000   0.200000
25%     5.450000   2.950000   2.850000   0.650000
50%     6.300000   3.200000   4.700000   1.500000
75%     6.600000   3.350000   5.450000   2.150000
max     7.000000   3.700000   6.000000   2.500000


Dataset: gen - nfold: 1/10
Train data
               0           1           2           3           4
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.015595   -0.007781    0.013634    0.016649    0.093196
0.045582
std      1.014951    1.001806    0.962817    1.008849    1.039005
0.997071
min     -2.515058   -2.433603   -2.647789   -2.518441   -3.223148
-2.725530
25%     -0.661821   -0.739336   -0.605543   -0.666526   -0.635592
-0.627144
50%     -0.010015   -0.072732    0.020328    0.024608    0.113445
0.105096
75%      0.719686    0.548756    0.641902    0.702521    0.748359
0.723925
max      3.594499    3.039572    3.003953    2.619359    2.712419
3.432993

               6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.001562   -0.012272    0.018226   -0.007054    0.010869
-0.076953
std      1.182731    0.997360    1.012774    1.010068    1.164712
0.955194
min     -3.326552   -2.736625   -2.807253   -2.690036   -3.315302
-3.102408
25%     -1.024900   -0.691724   -0.649506   -0.781671   -0.836271
-0.712939
50%     -0.060348    0.015647   -0.005171    0.009211    0.078181
-0.084789
75%      0.911349    0.633550    0.706345    0.701114    1.038988
0.629994
max      3.023143    2.654571    3.080764    3.135951    2.544278
2.664877
```

```
               12          13          14          15          16  
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.014286   -0.013744    0.111635   -0.001662   -0.010041  
-0.050198
std      0.989100    1.230692    0.949534    0.948191    1.016657  
0.991363
min     -3.109823   -3.210675   -3.239524   -2.675548   -3.068310  
-2.883941
25%     -0.638233   -0.998816   -0.512089   -0.664533   -0.771602  
-0.746687
50%     -0.032929   -0.127085    0.134519    0.051188   -0.012593  
-0.007451
75%      0.740198    1.013260    0.798428    0.672380    0.679608  
0.617835
max      2.706916    3.303948    3.446098    2.630985    3.094925  
3.264062

               18          19
count  450.000000  450.000000
mean     0.022037    0.053671
std      1.045171    1.017259
min     -3.389571   -3.110583
25%     -0.647093   -0.602597
50%      0.056958    0.051361
75%      0.726783    0.670130
max      2.811399    3.498259
Test data
              0          1          2          3          4  
5  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.132358   0.219547  -0.181742   0.144902  -0.108895   0.005
421
std     1.084205   1.043357   0.781276   1.108929   1.050024   1.036
938
min    -1.815645  -1.615432  -1.849257  -2.530270  -2.137125  -2.966
130
25%    -0.598300  -0.469860  -0.738886  -0.589564  -0.787458  -0.524
189
50%     0.035678   0.111592  -0.088987   0.311619  -0.221786  -0.038
448
75%     1.017673   0.753362   0.423989   0.893433   0.562336   0.811
786
max     2.247138   3.012955   1.041469   2.479612   2.307088   1.835
230

              6          7          8          9         10  
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.291133  -0.143091  -0.305290   0.092221  -0.213771  -0.003
974
```

```
std     1.128013   1.092923   1.038049   1.048631   1.188927   0.979
900
min    -2.798720  -2.295046  -3.060349  -3.021813  -3.265268  -2.235
046
25%    -0.959467  -0.811699  -1.045868  -0.512540  -0.802693  -0.696
424
50%    -0.551110  -0.280232  -0.382899   0.011575  -0.105014  -0.127
539
75%     0.556449   0.413665   0.467908   0.580116   0.673023   0.626
365
max     2.639439   2.731177   1.666533   2.726222   1.730126   2.144
330

              12         13         14         15         16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.299245   0.070258   0.259905  -0.131053   0.083768  -0.135
486
std     0.941931   1.125245   1.096413   1.001909   1.087815   0.930
853
min    -2.504971  -2.123482  -2.477589  -3.404818  -2.500514  -2.049
138
25%    -0.890362  -0.762860  -0.374569  -0.743837  -0.428832  -0.838
399
50%    -0.216580  -0.243333   0.136180  -0.093642   0.003066  -0.091
279
75%     0.156539   1.020259   0.900320   0.613982   0.963794   0.371
342
max     1.774276   2.097715   2.760497   1.719203   2.891136   1.863
623

              18         19
count  50.000000  50.000000
mean    0.029602  -0.150480
std     0.772789   1.055486
min    -1.341417  -3.335489
25%    -0.555508  -0.659712
50%     0.122657  -0.230268
75%     0.599628   0.493884
max     1.768622   2.561847


Dataset: gen - nfold: 2/10
Train data
               0           1           2           3           4
5   \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.026050    0.010612   -0.028223    0.021825    0.064013
0.032559
std      1.018835    1.002774    0.943754    1.016988    1.044935
0.987550
min     -2.515058   -2.409782   -2.538242   -2.530270   -3.223148
```

```
-2.966130
25%     -0.656625   -0.731285   -0.621644   -0.680306   -0.650527
-0.600580
50%     -0.004579   -0.065710   -0.001878    0.021160    0.072590
0.036412
75%      0.737007    0.611232    0.567467    0.724557    0.703074
0.730848
max      3.594499    3.039572    3.003953    2.553777    2.701731
3.432993

               6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.035755   -0.009109   -0.020325   -0.001763   -0.004324
-0.069980
std      1.171501    1.004397    1.018922    1.000889    1.163385
0.964629
min     -3.326552   -2.588259   -3.060349   -3.021813   -3.315302
-3.102408
25%     -1.009258   -0.702219   -0.669560   -0.748013   -0.829435
-0.727375
50%     -0.149009    0.013188   -0.061167    0.016847    0.084635
-0.085270
75%      0.889160    0.651285    0.693884    0.640449    0.986144
0.640427
max      3.023143    2.731177    2.949241    3.135951    2.394142
2.664877

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.030508   -0.017143    0.116693   -0.016262   -0.002223
-0.038372
std      0.991519    1.215496    0.946422    0.955138    1.012392
0.985784
min     -3.109823   -3.210675   -2.477589   -3.404818   -2.500514
-2.883941
25%     -0.687928   -0.998630   -0.501167   -0.699596   -0.730476
-0.707763
50%     -0.061614   -0.169169    0.126455    0.020755   -0.014262
0.008914
75%      0.672173    0.986469    0.793648    0.661546    0.677235
0.600035
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
mean     0.033888    0.041430
std      1.020286    1.009555
min     -3.389571   -3.335489
25%     -0.624639   -0.602597
```

```
50%       0.070552    0.019848
75%       0.726783    0.658348
max       2.811399    3.498259
Test data
              0          1          2          3          4          
5  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.038262   0.054007   0.194965   0.098316   0.153759   0.122
628
std     1.056174   1.056671   0.965787   1.043247   1.009659   1.114
637
min    -2.312866  -2.433603  -2.647789  -2.331489  -1.389525  -2.725
530
25%    -0.687586  -0.607539  -0.307969  -0.519517  -0.634776  -0.620
738
50%     0.053764   0.135871   0.252712   0.090076  -0.090751   0.377
410
75%     0.696368   0.600728   0.850966   0.734815   0.941969   0.803
461
max     2.370750   2.234953   2.391449   2.619359   2.712419   2.129
713

              6          7          8          9         10          
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.016604  -0.171557   0.041677   0.044603  -0.077031  -0.066
733
std     1.260623   1.028054   1.027462   1.129533   1.217925   0.893
927
min    -2.272284  -2.736625  -2.156165  -2.690036  -2.552006  -2.396
023
25%    -1.115137  -0.839423  -0.633624  -0.901056  -0.872884  -0.536
695
50%     0.133255  -0.033975   0.007951  -0.109452  -0.206088  -0.118
690
75%     1.126697   0.394442   0.567698   1.002914   0.974135   0.553
447
max     2.596622   2.237697   3.080764   2.399263   2.544278   1.860
893

             12         13         14         15         16          
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.103904   0.100850   0.214376   0.000352   0.013407  -0.241
916
std     0.957611   1.265105   1.125886   0.947518   1.127166   0.967
570
min    -1.857877  -2.853214  -3.239524  -2.067902  -3.068310  -2.383
044
25%    -0.554286  -0.916190  -0.559397  -0.641827  -0.907653  -0.985
634
```

```
50%      0.100113   0.365227   0.242098   0.104780   0.196711  -0.219
574
75%      0.864881   1.137323   0.879004   0.592967   0.866703   0.424
260
max      1.748159   2.301200   2.877559   2.116475   2.781388   1.647
596

              18         19
count  50.000000  50.000000
mean   -0.077060  -0.040308
std     1.028069   1.135462
min    -1.931508  -2.911766
25%    -0.904866  -0.708436
50%     0.126267   0.122914
75%     0.561591   0.672803
max     2.477594   1.928412


Dataset: gen - nfold: 3/10
Train data
               0           1           2           3           4
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.047933    0.007291   -0.006521    0.033717    0.059505
0.036812
std      1.021801    0.999899    0.934966    1.003862    1.047233
1.006639
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148
-2.966130
25%     -0.650934   -0.731285   -0.611001   -0.603799   -0.658458
-0.643422
50%      0.020745   -0.054029    0.008816    0.023268    0.036682
0.091458
75%      0.748852    0.558387    0.630666    0.731642    0.707796
0.758392
max      3.594499    3.039572    2.786393    2.619359    2.712419
3.432993

               6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.023675   -0.037418    0.010098    0.031182   -0.003217
-0.062153
std      1.191962    1.019248    1.028934    1.013198    1.159833
0.947067
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302
-3.102408
25%     -1.004924   -0.715119   -0.661007   -0.717004   -0.817548
-0.690305
50%     -0.135069   -0.000029    0.003053    0.032018    0.052684
-0.078091
75%      0.897405    0.615801    0.712259    0.716696    0.991318
```

```
0.629994
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.664877

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.031181   -0.010936    0.105002   -0.002544   -0.007395
-0.067799
std      0.998220    1.224767    0.963941    0.941169    1.023449
1.010611
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310
-2.883941
25%     -0.693880   -0.996848   -0.512093   -0.657638   -0.742380
-0.791377
50%     -0.064618   -0.173535    0.118933    0.040907   -0.016392
-0.064097
75%      0.695672    1.013260    0.793648    0.654095    0.679456
0.633308
max      2.706916    3.303948    3.136039    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
mean     0.018088    0.019022
std      1.005422    0.992194
min     -3.389571   -3.335489
25%     -0.647093   -0.606930
50%      0.069620    0.031042
75%      0.700627    0.643042
max      2.811399    2.700357
Test data
              0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.158680   0.083899  -0.000351  -0.008713   0.194324   0.084
351
std     1.010437   1.079654   1.062898   1.155486   0.982882   0.948
133
min    -2.474989  -1.761486  -2.538242  -2.387491  -2.027319  -2.502
706
25%    -0.743145  -0.664030  -0.576362  -0.853300  -0.307026  -0.333
269
50%    -0.222097  -0.065181   0.055827   0.056824   0.147435   0.074
835
75%     0.645991   0.851087   0.524876   0.729377   0.804464   0.513
033
max     1.902482   2.555292   3.003953   2.422573   2.457268   2.421
206

              6          7          8          9         10
11  \
```

```
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.092112   0.083229  -0.232132  -0.251901  -0.086996  -0.137
172
std     1.069701   0.890149   0.904127   0.988457   1.247944   1.049
569
min    -2.028585  -1.989720  -2.024629  -2.248639  -2.900659  -2.250
563
25%    -1.092935  -0.409998  -0.789847  -0.835001  -0.984440  -0.853
941
50%    -0.052825   0.055972  -0.195684  -0.466825   0.122797  -0.274
917
75%     0.837576   0.879671   0.186830   0.509353   0.854002   0.602
307
max     1.849474   1.617238   2.406138   2.110171   1.887018   2.231
575

              12         13         14         15         16        
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.109964   0.044983   0.319598  -0.123110   0.059953   0.022
928
std     0.890580   1.184522   0.963030   1.061800   1.029706   0.715
892
min    -1.582977  -1.989643  -2.407276  -2.446034  -2.107179  -1.467
864
25%    -0.566124  -0.998781  -0.347892  -0.937595  -0.835969  -0.265
400
50%     0.072052  -0.012036   0.508317  -0.085224   0.287962   0.076
997
75%     0.802721   1.083914   0.932677   0.654644   0.934273   0.369
975
max     1.799546   2.330057   3.446098   2.040498   1.771825   1.966
577

              18         19
count  50.000000  50.000000
mean    0.065139   0.161366
std     1.158490   1.263033
min    -2.479270  -3.110583
25%    -0.569150  -0.742918
50%     0.178099   0.019221
75%     0.974235   0.881872
max     2.117619   3.498259


Dataset: gen - nfold: 4/10
Train data
                0           1           2           3           4    
5   \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.021381    0.019990   -0.044326    0.057702    0.046661
```

```
0.024812
std      1.007187    1.014515    0.946655    1.025704    1.046111
1.007880
min     -2.502781   -2.433603   -2.647789   -2.530270   -3.223148
-2.966130
25%     -0.667635   -0.723217   -0.623707   -0.613074   -0.662161
-0.643422
50%     -0.004579   -0.042863   -0.040664    0.046302    0.014821
0.050078
75%      0.713168    0.621871    0.560004    0.752412    0.721192
0.717283
max      3.594499    3.039572    3.003953    2.619359    2.712419
3.432993

                6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.042327   -0.068944   -0.022161   -0.010804   -0.015081
-0.071848
std      1.184920    0.997026    1.031587    1.028591    1.175870
0.945954
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302
-3.102408
25%     -1.024900   -0.745929   -0.669560   -0.779440   -0.829435
-0.720462
50%     -0.149009   -0.048821   -0.055719   -0.018445    0.051598
-0.090765
75%      0.882965    0.597393    0.681881    0.684015    0.972415
0.618305
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.664877

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.055900   -0.008587    0.165719   -0.026275    0.017580
-0.071305
std      0.981919    1.225184    0.962852    0.945221    1.032658
0.979377
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310
-2.883941
25%     -0.702542   -0.993650   -0.487920   -0.709153   -0.750497
-0.780782
50%     -0.091031   -0.138393    0.160812    0.031218    0.021411
-0.052805
75%      0.626182    0.986469    0.850638    0.646993    0.710499
0.599745
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
```

```
mean     0.048677    0.015358
std      1.025005    1.031670
min     -3.389571   -3.335489
25%     -0.610119   -0.652167
50%      0.111300    0.019056
75%      0.713964    0.640797
max      2.811399    3.498259
Test data
               0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.080283  -0.030398   0.339893  -0.224577   0.309926   0.192
350
std     1.152693   0.948115   0.890316   0.925964   0.969857   0.923
235
min    -2.515058  -1.666063  -2.403479  -2.391383  -2.285826  -2.406
626
25%    -0.607547  -0.780748  -0.056329  -0.784568  -0.185102  -0.402
089
50%     0.066205  -0.117236   0.374814  -0.253876   0.312054   0.338
791
75%     0.936858   0.507468   0.827671   0.295357   0.696889   0.821
258
max     2.548451   2.358609   2.384275   2.553777   2.392053   2.003
789

               6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.075749   0.366960   0.058196   0.125967   0.019778  -0.049
918
std     1.135399   1.020691   0.902940   0.863403   1.104288   1.060
952
min    -2.114891  -2.063264  -1.607720  -1.802546  -2.613164  -2.428
047
25%    -0.872145  -0.214425  -0.549470  -0.415672  -0.894302  -0.627
919
50%     0.182944   0.409253   0.064049   0.103725   0.123945   0.164
708
75%     0.927504   0.920932   0.659906   0.740404   1.051406   0.768
406
max     2.402138   2.376203   1.901959   2.366867   1.766627   1.873
812

              12         13         14         15         16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.332435   0.023842  -0.226851   0.090465  -0.164825   0.054
481
std     0.984233   1.181370   0.920543   1.028696   0.927578   1.036
861
```

```
min    -2.387439  -2.150834  -2.029740  -1.759327  -2.330537  -2.852
202
25%    -0.236465  -1.022930  -1.041336  -0.609488  -0.692739  -0.623
611
50%     0.235711   0.036670   0.041565  -0.034799  -0.082744   0.079
070
75%     1.053227   1.069809   0.506140   0.850593   0.472276   0.588
899
max     2.373076   2.077982   1.671952   2.253107   1.636746   2.598
636

              18         19
count  50.000000  50.000000
mean   -0.210156   0.194339
std     0.958259   0.923226
min    -2.322067  -1.694281
25%    -0.778059  -0.354186
50%    -0.269645   0.193363
75%     0.522646   0.962214
max     1.575827   1.880036


Dataset: gen - nfold: 5/10
Train data
                0           1           2           3           4
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.046363    0.030152    0.006538    0.033181    0.099820
0.012035
std      1.031098    0.994598    0.947866    1.019143    1.026529
0.994721
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148
-2.966130
25%     -0.646985   -0.661953   -0.572908   -0.660300   -0.624044
-0.642141
50%      0.024569   -0.035679    0.049062    0.024608    0.081943
0.056967
75%      0.746789    0.621871    0.641902    0.734815    0.770385
0.699026
max      3.594499    3.039572    3.003953    2.619359    2.712419
3.432993

                6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.042617   -0.047969   -0.014090   -0.002329   -0.028629
-0.063051
std      1.173131    1.010163    0.998365    1.000765    1.154321
0.970082
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.265268
-3.102408
25%     -1.024900   -0.712943   -0.661007   -0.748013   -0.865448
```

```
-0.713709
50%     -0.135069   -0.002986   -0.055719    0.009211    0.048201
-0.084789
75%      0.882965    0.603222    0.612204    0.641698    0.956005
0.633333
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.664877

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.006098    0.007154    0.116388   -0.018399   -0.014021
-0.047301
std      0.981139    1.213557    0.966622    0.966508    1.022400
0.953709
min     -2.773482   -3.210675   -3.239524   -3.404818   -3.068310
-2.852202
25%     -0.647631   -0.998337   -0.502874   -0.709153   -0.749780
-0.740812
50%     -0.052692   -0.063395    0.133001    0.020755   -0.005925
-0.031089
75%      0.710747    1.043694    0.798428    0.661546    0.679608
0.591535
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
mean    -0.004349    0.014115
std      1.007673    1.037517
min     -3.389571   -3.335489
25%     -0.637652   -0.656001
50%      0.063249    0.002156
75%      0.685464    0.658348
max      2.810356    3.498259
Test data
              0          1          2          3          4
5  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.144554  -0.121854  -0.117880  -0.003884  -0.168508   0.307
341
std     0.922941   1.116559   0.944985   1.025803   1.144582   1.019
563
min    -1.830076  -1.888726  -1.915641  -2.288779  -2.476065  -2.212
777
25%    -0.906128  -1.069087  -0.754008  -0.583098  -1.045061  -0.363
919
50%    -0.221809  -0.208768  -0.320661   0.062377  -0.131580   0.318
812
75%     0.521024   0.438991   0.387035   0.714186   0.573436   0.999
719
max     2.092328   2.958475   2.786393   2.415195   2.259766   2.633
```

```
551

               6          7          8          9         10    
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.078364   0.178188  -0.014437   0.049689   0.141708  -0.129
092
std     1.242265   0.963176   1.199663   1.130308   1.286097   0.835
731
min    -2.354476  -1.803324  -2.332982  -2.222834  -3.315302  -1.973
770
25%    -0.921488  -0.595136  -0.729337  -0.936270  -0.463698  -0.688
107
50%     0.185016   0.235087   0.157994   0.037797   0.257667  -0.232
141
75%     1.284925   0.790579   1.020022   0.849992   1.369733   0.519
539
max     2.292472   2.389154   2.211789   2.521555   1.985425   1.518
230

              12         13         14         15         16    
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.225546  -0.117827   0.217126   0.019584   0.119587  -0.161
553
std     1.035350   1.281443   0.955494   0.834571   1.033305   1.237
858
min    -3.109823  -3.183078  -1.748208  -1.690709  -1.999276  -2.883
941
25%    -0.874243  -1.009867  -0.446718  -0.457395  -0.742380  -0.828
197
50%    -0.032883  -0.301077   0.137673   0.102965  -0.026089  -0.074
459
75%     0.290427   0.624355   0.820334   0.616294   0.778329   0.644
963
max     1.757267   3.233625   3.136039   1.423433   2.961835   2.209
712

              18         19  
count  50.000000  50.000000  
mean    0.267077   0.205524  
std     1.111285   0.858645  
min    -2.432191  -1.830377  
25%    -0.475018  -0.201549  
50%     0.244329   0.210627  
75%     0.992336   0.660021  
max     2.811399   2.497299  


Dataset: gen - nfold: 6/10
Train data
                 0          1          2          3          4
```

```
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.034948    0.022931    0.001797    0.055040    0.080748
0.062969
std      1.024961    1.030424    0.961383    1.027323    1.049393
1.020453
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148
-2.966130
25%     -0.656625   -0.757168   -0.612178   -0.628294   -0.650527
-0.565832
50%     -0.010015   -0.054029    0.017096    0.049403    0.068448
0.096372
75%      0.739349    0.633139    0.641902    0.741273    0.767130
0.806209
max      3.594499    3.039572    3.003953    2.619359    2.712419
3.432993

               6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.038416   -0.017512   -0.009348    0.007595   -0.020109
-0.070187
std      1.183650    1.016225    1.033119    1.001166    1.193580
0.947522
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302
-2.428047
25%     -1.024900   -0.706993   -0.661007   -0.720322   -0.880605
-0.727375
50%     -0.113862    0.001030   -0.055719    0.024779    0.078181
-0.104708
75%      0.889160    0.612958    0.698088    0.666772    1.041532
0.618305
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.664877

              12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.002853   -0.000247    0.110314   -0.007301    0.022894
-0.073099
std      0.970968    1.230291    0.976518    0.952818    1.032903
0.990868
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310
-2.883941
25%     -0.647631   -0.998630   -0.537947   -0.699596   -0.740451
-0.768824
50%     -0.026798   -0.172144    0.129713    0.031218    0.015291
-0.057160
75%      0.718462    1.050397    0.776360    0.672380    0.721685
0.577599
max      2.706916    3.303948    3.446098    2.630985    3.094925
```

```
3.264062

              18          19
count  450.000000  450.000000
mean     0.037066    0.054942
std      0.996717    1.039004
min     -3.142108   -3.335489
25%     -0.621608   -0.577526
50%      0.088467    0.035258
75%      0.706239    0.701204
max      2.811399    3.498259
Test data
             0          1          2          3          4
5  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.041817  -0.056865  -0.075211  -0.200620   0.003143  -0.151
063
std     0.997755   0.772851   0.815657   0.916578   0.967332   0.776
135
min    -1.947067  -1.650312  -1.658852  -2.518441  -2.538309  -1.813
616
25%    -0.836478  -0.508724  -0.600076  -0.818219  -0.638669  -0.711
585
50%     0.204223  -0.118773  -0.015056  -0.091489  -0.000904  -0.029
113
75%     0.719686   0.390336   0.423136   0.327996   0.470702   0.360
674
max     1.619961   1.706666   1.783282   1.906339   2.701731   1.500
832

             6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.040551  -0.095925  -0.057118  -0.039618   0.065027  -0.064
873
std     1.150558   0.925459   0.888901   1.127247   0.910720   1.048
254
min    -1.760252  -2.129410  -1.799736  -2.405565  -1.990374  -3.102
408
25%    -0.931524  -0.737415  -0.674054  -0.943659  -0.444629  -0.571
006
50%    -0.124982   0.014238   0.053458  -0.174143   0.027254   0.083
625
75%     0.900319   0.699310   0.443421   0.889941   0.665386   0.651
903
max     2.608173   2.297256   1.702336   2.301998   1.772622   2.104
481

            12         13         14         15         16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
```

```
mean   -0.196347  -0.051215   0.271786  -0.080304  -0.212643   0.070
624
std     1.125488   1.131066   0.849783   0.966283   0.913980   0.929
074
min    -2.773482  -2.315388  -1.377981  -2.675548  -2.260204  -2.045
862
25%    -0.897507  -0.945838  -0.276254  -0.641910  -0.809100  -0.514
692
50%    -0.234161   0.012838   0.152490  -0.007452  -0.142092   0.079
495
75%     0.498659   0.745972   1.005421   0.500476   0.295900   0.853
167
max     2.373443   2.349200   2.104381   1.887152   1.831406   1.890
737

              18         19
count  50.000000  50.000000
mean   -0.105659  -0.161918
std     1.218663   0.836319
min    -3.389571  -1.861160
25%    -0.821180  -0.768410
50%    -0.132029  -0.142474
75%     0.833180   0.479297
max     2.810356   1.858929


Dataset: gen - nfold: 7/10
Train data
                0           1           2           3           4
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.036740    0.014916   -0.006072    0.007583    0.069091
0.051612
std      1.017976    1.004621    0.944245    1.010129    1.036955
1.001793
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148
-2.966130
25%     -0.647959   -0.724340   -0.611001   -0.672724   -0.648860
-0.612146
50%     -0.004579   -0.042863    0.014514    0.023268    0.068448
0.142059
75%      0.748852    0.621871    0.598635    0.717413    0.703074
0.746735
max      3.594499    3.012955    3.003953    2.619359    2.712419
2.633551

                6           7           8           9          10
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.046243   -0.022292   -0.023175    0.008486   -0.007191
-0.080209
std      1.161287    1.002691    0.994802    1.018478    1.160975
```

```
0.964937
min     -2.798720   -2.736625   -3.060349   -3.021813   -3.315302
-3.102408
25%     -1.018254   -0.706430   -0.661485   -0.748013   -0.836248
-0.727375
50%     -0.175696    0.013668   -0.061167    0.009211    0.054986
-0.119987
75%      0.882965    0.626561    0.658825    0.691254    0.986144
0.639564
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.231575

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.014978   -0.020293    0.110723   -0.017047   -0.010679
-0.045648
std      0.979527    1.193027    0.966864    0.967518    1.006980
0.992364
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310
-2.883941
25%     -0.687928   -0.993650   -0.574768   -0.710041   -0.742380
-0.721050
50%     -0.042304   -0.131162    0.122016    0.028767   -0.014262
0.007654
75%      0.695672    0.973173    0.791351    0.670164    0.691821
0.617835
max      2.706916    3.233625    3.446098    2.630985    2.961835
3.264062

               18          19
count  450.000000  450.000000
mean     0.000724    0.039868
std      1.019778    1.005891
min     -3.389571   -3.335489
25%     -0.658369   -0.597633
50%      0.061362    0.035258
75%      0.706239    0.663925
max      2.811399    3.498259
Test data
              0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.057950   0.015266  -0.004389   0.226494   0.108052  -0.048
849
std     1.059930   1.041302   0.984888   1.084955   1.085196   0.990
432
min    -2.502781  -1.424634  -2.227829  -2.388802  -2.906092  -2.060
752
25%    -0.742419  -0.708514  -0.547763  -0.311879  -0.644718  -0.565
832
50%    -0.059863  -0.154268   0.142939   0.129766   0.037278  -0.206
```

```
659
75%     0.536716   0.363766   0.641943   0.949866   0.926968   0.277
525
max     2.175765   3.039572   1.894596   2.500452   2.183237   3.432
993

               6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.110992  -0.052904   0.067326  -0.047643  -0.051230   0.025
327
std     1.336908   1.054339   1.223417   0.974394   1.240060   0.885
123
min    -3.326552  -1.660284  -2.310975  -2.174052  -3.154704  -2.317
728
25%    -0.984599  -0.712110  -0.713976  -0.801967  -0.824754  -0.556
569
50%     0.236644  -0.141543   0.141037  -0.019932   0.103387  -0.027
968
75%     1.131140   0.470471   1.048520   0.619165   1.015582   0.552
066
max     2.626074   2.654571   2.658719   1.732563   1.575914   2.664
877

              12         13         14         15         16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.035863   0.129199   0.268108   0.007416   0.089507  -0.176
432
std     1.072285   1.445216   0.946171   0.824246   1.166897   0.916
108
min    -2.635559  -2.648808  -2.261836  -2.450310  -1.812295  -2.186
798
25%    -0.668375  -1.045445  -0.417215  -0.549693  -0.774563  -0.851
092
50%    -0.085373  -0.151277   0.440333   0.067219   0.069451  -0.213
254
75%     0.801315   1.475864   0.983746   0.638759   0.674098   0.519
258
max     2.374274   3.303948   1.989304   1.458285   3.094925   1.911
468

              18         19
count  50.000000  50.000000
mean    0.221421  -0.026251
std     1.016364   1.165788
min    -1.813266  -2.402426
25%    -0.549969  -0.677957
50%     0.331074  -0.029731
75%     0.773438   0.600520
max     2.222720   2.289408
```

```
Dataset: gen - nfold: 8/10
Train data
                0           1           2           3           4  
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.006700    0.023016   -0.006120    0.021239    0.054742  
0.050440
std      1.008438    0.994951    0.952849    1.028801    1.047196  
1.003928
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148  
-2.966130
25%     -0.691143   -0.688381   -0.608366   -0.680306   -0.662161  
-0.559330
50%     -0.015535   -0.042863    0.014514    0.024608    0.048451  
0.107385
75%      0.684741    0.590625    0.620679    0.707981    0.711960  
0.741408
max      2.548451    3.039572    3.003953    2.619359    2.712419  
3.432993

                6           7           8           9          10  
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.021810   -0.023670   -0.032432   -0.004865   -0.024825  
-0.067294
std      1.167293    1.008217    1.019983    1.009422    1.156159  
0.962153
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302  
-3.102408
25%     -0.976854   -0.706993   -0.673817   -0.757163   -0.829435  
-0.713709
50%     -0.083149    0.013188   -0.074047   -0.028831    0.048201  
-0.086521
75%      0.887912    0.615801    0.680169    0.691254    0.940381  
0.633333
max      2.639439    2.731177    3.080764    2.726222    2.544278  
2.664877

               12          13          14          15          16  
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.016083    0.015617    0.145993   -0.013583   -0.006326  
-0.056327
std      1.005443    1.215894    0.980495    0.944013    1.034905  
0.970464
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310  
-2.883941
25%     -0.693880   -0.993035   -0.492080   -0.653201   -0.775738  
-0.746687
50%     -0.032929   -0.063395    0.168298    0.033666   -0.012593
```

```
-0.007451
75%      0.718462    1.027576    0.809160    0.638759    0.715479
0.633029
max      2.588905    3.303948    3.446098    2.253107    3.094925
2.598636

               18          19
count  450.000000  450.000000
mean     0.026970    0.063260
std      1.029827    1.031697
min     -3.389571   -3.335489
25%     -0.633296   -0.579277
50%      0.069620    0.044369
75%      0.713305    0.687292
max      2.811399    3.498259
Test data
              0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.333016  -0.057634  -0.003960   0.103597   0.237191  -0.038
302
std     1.096725   1.120496   0.905778   0.930627   0.976166   0.971
473
min    -1.942028  -2.409782  -1.906511  -1.799940  -1.927679  -2.226
213
25%    -0.389660  -1.068046  -0.601492  -0.538157  -0.430618  -0.730
115
50%     0.191112  -0.133728   0.019083   0.188662   0.150188  -0.090
879
75%     1.141918   0.669761   0.601928   0.893631   0.792581   0.701
651
max     3.594499   2.184190   2.287830   1.864179   2.454954   1.795
237

              6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.108898  -0.040509   0.150634   0.072516   0.107470  -0.090
904
std     1.294182   1.005197   1.004293   1.055995   1.275035   0.917
779
min    -2.070612  -2.588259  -2.150621  -2.139863  -2.356504  -2.031
530
25%    -1.071386  -0.764361  -0.480396  -0.803103  -1.043205  -0.633
569
50%    -0.422645  -0.004554   0.062777   0.082652   0.288597  -0.104
311
75%     0.907700   0.752983   0.681906   0.654627   1.298378   0.438
982
max     3.023143   1.639151   2.770888   3.135951   2.394142   1.721
826
```

```
               12         13         14         15         16        
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.025923  -0.193992  -0.049324  -0.023762   0.050334  -0.080
323
std     0.823844   1.250621   0.799833   1.044777   0.919469   1.117
288
min    -2.355433  -2.826211  -1.591517  -2.300765  -1.936547  -2.836
018
25%    -0.526708  -1.041973  -0.632451  -0.751112  -0.565006  -0.838
208
50%    -0.199222  -0.675497  -0.126280  -0.047126   0.051282  -0.141
586
75%     0.549039   0.972225   0.640799   0.826880   0.648137   0.423
191
max     2.706916   2.064077   1.652798   2.630985   2.638585   3.264
062

               18         19  
count  50.000000  50.000000  
mean   -0.014796  -0.236778  
std     0.942014   0.893632  
min    -2.677941  -2.444144  
25%    -0.684343  -0.778016  
50%     0.078252  -0.225821  
75%     0.646968   0.483833  
max     2.149012   1.410672  


Dataset: gen - nfold: 9/10
Train data
                0           1           2           3           4  
5   \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.023723    0.009562   -0.013205    0.036225    0.095560    
0.025289
std      1.028938    1.011822    0.938545    1.025710    1.026697    
0.999585
min     -2.515058   -2.433603   -2.647789   -2.530270   -2.906092   
-2.966130
25%     -0.661821   -0.723217   -0.611001   -0.660300   -0.624044   
-0.642141
50%     -0.015535   -0.070331    0.019083    0.041148    0.088074    
0.067131
75%      0.719686    0.558387    0.602079    0.737938    0.748359    
0.714348
max      3.594499    3.039572    3.003953    2.619359    2.712419    
3.432993

                6           7           8           9          10  
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
```

```
50.000000
mean    -0.031073    0.001353   -0.042307    0.007403   -0.004214
-0.074398
std      1.185311    1.004318    1.022313    1.027091    1.173251
0.964401
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302
-3.102408
25%     -1.031734   -0.693011   -0.684833   -0.775204   -0.836271
-0.713709
50%     -0.123878    0.013668   -0.073023    0.010985    0.083188
-0.090765
75%      0.897405    0.646937    0.625374    0.715077    0.986144
0.618305
max      3.023143    2.731177    3.080764    3.135951    2.544278
2.664877

               12          13          14          15          16
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.031769   -0.014363    0.131445   -0.011874    0.000624
-0.074372
std      0.985334    1.225877    0.964733    0.959553    1.025045
0.987903
min     -3.109823   -3.183078   -3.239524   -3.404818   -3.068310
-2.883941
25%     -0.692961   -0.998954   -0.501167   -0.699596   -0.740451
-0.764249
50%     -0.053167   -0.155192    0.115239    0.027387   -0.012593
-0.064097
75%      0.690560    1.013260    0.815575    0.694127    0.691821
0.583738
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
mean     0.008841    0.011713
std      1.023278    1.022393
min     -3.389571   -3.335489
25%     -0.647093   -0.656001
50%      0.063249   -0.002340
75%      0.706239    0.647097
max      2.811399    3.498259
Test data
              0          1          2          3          4
5   \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.059211   0.063457   0.059806  -0.031280  -0.130166   0.188
059
std     0.961720   0.973953   1.031235   0.962440   1.151620   1.003
184
min    -2.339159  -1.939914  -2.134426  -1.679796  -3.223148  -2.394
```

```
232
25%    -0.563672  -0.814068  -0.479487  -0.691738  -0.859833  -0.352
811
50%     0.110441   0.053726  -0.003525  -0.168635  -0.277699   0.244
251
75%     0.785789   0.699058   0.625490   0.464423   0.618434   0.933
102
max     2.227447   2.692643   2.368665   2.205885   2.222516   2.271
239

               6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.025537  -0.265714   0.239510  -0.037899  -0.078028  -0.026
967
std     1.137284   1.008504   0.960640   0.888268   1.127806   0.895
029
min    -2.793221  -2.298243  -2.155828  -2.145933  -2.837208  -1.555
600
25%    -0.864647  -0.995150  -0.339074  -0.514536  -0.829435  -0.635
187
50%    -0.200039  -0.155741   0.094301  -0.044399  -0.054906  -0.037
465
75%     0.734890   0.463924   0.846494   0.470714   1.018460   0.657
946
max     2.273996   1.691462   2.949241   2.526021   1.971572   1.871
874

              12         13         14         15         16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.115253   0.075827   0.081607  -0.039147  -0.012213   0.082
081
std     1.012818   1.171993   0.976438   0.905487   1.017027   0.955
734
min    -2.179764  -3.210675  -1.986878  -2.055123  -2.450816  -2.020
142
25%    -0.570858  -0.896245  -0.480644  -0.664533  -0.794118  -0.640
479
50%     0.086052   0.118357   0.334026   0.086195   0.111097   0.266
736
75%     0.701044   1.020224   0.711403   0.504329   0.713471   1.000
791
max     2.588905   2.252081   1.760996   1.749939   1.841037   1.630
532

              18         19
count  50.000000  50.000000
mean    0.148367   0.227144
std     0.997158   1.006852
min    -2.420103  -2.607841
25%    -0.409629  -0.285652
```

```
50%     0.247262   0.303599
75%     0.835712   0.774977
max     2.126456   2.700357


Dataset: gen - nfold: 10/10
Train data
               0           1           2           3           4  
5  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean     0.026681    0.018825    0.023459    0.011582    0.066536  
0.073548
std      1.040869    1.017423    0.939421    1.021392    1.043558  
0.980666
min     -2.515058   -2.433603   -2.647789   -2.530270   -3.223148  
-2.966130
25%     -0.661821   -0.739336   -0.567800   -0.683631   -0.648860  
-0.526205
50%     -0.010015   -0.064273    0.060238    0.021160    0.063600  
0.107385
75%      0.748852    0.616571    0.621777    0.710096    0.703074  
0.739775
max      3.594499    3.039572    3.003953    2.619359    2.712419  
3.432993

               6           7           8           9          10  
11  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.021713   -0.015703   -0.005737    0.000882   -0.019230  
-0.060478
std      1.193286    1.008536    1.027191    1.023808    1.177062  
0.947967
min     -3.326552   -2.736625   -3.060349   -3.021813   -3.315302  
-3.102408
25%     -1.031734   -0.693011   -0.661485   -0.775162   -0.833440  
-0.641066
50%     -0.103775    0.022120   -0.044877   -0.009602    0.052684  
-0.069446
75%      0.903723    0.626561    0.685562    0.666772    0.995299  
0.610594
max      3.023143    2.731177    3.080764    3.135951    2.544278  
2.664877

              12          13          14          15          16  
17  \
count  450.000000  450.000000  450.000000  450.000000  450.000000  4
50.000000
mean    -0.013485    0.009103    0.150704   -0.031062    0.002986  
-0.062844
std      0.996088    1.223123    0.970925    0.954590    1.025116  
0.986052
min     -3.109823   -3.210675   -3.239524   -3.404818   -3.068310  
```

```
-2.883941
25%     -0.687928   -0.998630   -0.484163   -0.710041   -0.742380
-0.707763
50%     -0.038385   -0.127085    0.168298    0.020631   -0.012593
-0.067336
75%      0.710027    1.040225    0.814356    0.638759    0.721685
0.577599
max      2.706916    3.303948    3.446098    2.630985    3.094925
3.264062

               18          19
count  450.000000  450.000000
mean     0.035993    0.019182
std      1.031553    1.030245
min     -3.389571   -3.335489
25%     -0.663105   -0.666518
50%      0.082181    0.038882
75%      0.734925    0.658348
max      2.811399    3.498259
Test data
              0          1          2          3          4
5  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean    0.032585  -0.019912  -0.270172   0.190509   0.131047  -0.246
276
std     0.836508   0.919633   0.986959   0.990999   1.024370   1.131
403
min    -1.919142  -2.032010  -2.396087  -2.376917  -2.017500  -2.536
563
25%    -0.632911  -0.642941  -0.879084  -0.336533  -0.629139  -0.899
032
50%     0.065036  -0.013714  -0.419932   0.130153   0.048048  -0.239
968
75%     0.532069   0.554325   0.560949   0.993971   0.832003   0.716
057
max     2.047575   1.852219   1.836540   1.973954   2.343747   1.716
865

              6          7          8          9         10
11  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.109770  -0.112213  -0.089617   0.020797   0.057122  -0.152
247
std     1.054753   0.998111   0.947469   0.923170   1.090656   1.040
866
min    -1.534965  -2.464415  -2.807253  -1.904268  -2.226001  -1.981
393
25%    -0.992201  -0.807175  -0.669386  -0.654550  -0.829126  -1.041
054
50%    -0.225398  -0.194554  -0.111076   0.083974   0.078181  -0.217
631
75%     0.726639   0.338091   0.616144   0.706976   0.952572   0.731
```

```
496
max      2.261461   2.493480   1.710843   1.915901   2.098559   1.984
996

              12          13          14          15          16
17  \
count  50.000000  50.000000  50.000000  50.000000  50.000000  50.000
000
mean   -0.049299  -0.135364  -0.091717   0.133550  -0.033476  -0.021
671
std     0.921525   1.193181   0.889897   0.939547   1.015845   0.983
784
min    -1.864570  -2.642507  -1.958835  -2.288876  -2.182213  -1.960
882
25%    -0.625902  -0.993650  -0.734582  -0.474516  -0.821377  -0.859
561
50%    -0.089135  -0.350251  -0.054190   0.043948   0.070567   0.138
554
75%     0.595613   0.686333   0.666393   0.883080   0.612214   0.754
718
max     1.774074   2.235496   1.685332   1.737713   2.074714   2.161
604

              18          19
count  50.000000  50.000000
mean   -0.096001   0.159925
std     0.916736   0.943669
min    -3.142108  -1.451373
25%    -0.513807  -0.561864
50%    -0.128926  -0.069843
75%     0.542121   0.772346
max     1.735289   2.601407
```

Sometimes it's important to ensure you get equal samples of each class, particularly when the classes are imbalanced. Let's generate some data with a label skew and see what happens in cross-validation.

In [53]:

```
#Let's make some skewed data where the instances with label 1 are very r
(skew_gendata_d, skew_gendata_t) = datasets.make_classification(n_sample
skew_gendata_data = pd.DataFrame(skew_gendata_d)
skew_gendata_targets = pd.DataFrame(skew_gendata_t)
```

Question 3: Stratified folds (5 points)

1. Use the `KFold` cross-validation method and generate 10 folds (using `random_state=20160121`). What is the range of percentages for label 1 across the folds?
2. Use the `StratifiedKFold` cross-validation method and generate 10 folds (using

`random_state=20160121`). What is the range of percentages for label 1 across the folds?

Question 3 Answer

1. The range of percentages for label 1 is from 10.22% to 11.11% for the train data and from 8% to 16% for the test data

2. The range of percentages for label 1 is from 10.69% to 10.86% for the train data from 10% to 11.76% for the test data

```
1.
Dataset: skew - nfold: 1/10
Train data
0    401 / 450 = 89.1%
1     49 / 450 = 10.9%
Name: 0, dtype: int64
Test data
0    45 / 50 = 90%
1     5 / 50 = 10%
Name: 0, dtype: int64


Dataset: skew - nfold: 2/10
Train data
0    403 / 450 = 89.56%
1     47 / 450 = 10.44%
Name: 0, dtype: int64
Test data
0    43 / 50 = 86%
1     7 / 50 = 14%
Name: 0, dtype: int64


Dataset: skew - nfold: 3/10
Train data
0    403 / 450 = 89.56%
1     47 / 450 = 10.44%
Name: 0, dtype: int64
Test data
0    43 / 50 = 86%
1     7 / 50 = 14%
Name: 0, dtype: int64


Dataset: skew - nfold: 4/10
Train data
0    400 / 450 = 88.89%
1     50 / 450 = 11.11%
Name: 0, dtype: int64
Test data
0    46 / 50 = 92%
1     4 / 50 = 8%
Name: 0, dtype: int64


Dataset: skew - nfold: 5/10
Train data
0    400 / 450 = 88.89%
1     50 / 450 = 11.11%
Name: 0, dtype: int64
Test data
0    46 / 50 = 92%
1     4 / 50 = 8%
Name: 0, dtype: int64
```

```
Name: 0, dtype: int64


Dataset: skew - nfold: 6/10
Train data
0    404 / 450 = 89.78%
1     46 / 450 = 10.22%
Name: 0, dtype: int64
Test data
0    42 / 50 = 84%
1     8 / 50 = 16%
Name: 0, dtype: int64


Dataset: skew - nfold: 7/10
Train data
0    403 / 450 = 89.56%
1     47 / 450 = 10.44%
Name: 0, dtype: int64
Test data
0    43 / 50 = 86%
1     7 / 50 = 14%
Name: 0, dtype: int64


Dataset: skew - nfold: 8/10
Train data
0    400 / 450 = 88.89%
1     50 / 450 = 11.11$
Name: 0, dtype: int64
Test data
0    46 / 50 = 92%
1     4 / 50 = 8%
Name: 0, dtype: int64


Dataset: skew - nfold: 9/10
Train data
0    400 / 450 = 88.89%
1     50 / 450 = 11.11%
Name: 0, dtype: int64
Test data
0    46 / 50 = 92%
1     4 / 50 = 8%
Name: 0, dtype: int64


Dataset: skew - nfold: 10/10
Train data
0    400 / 450 = 88.89%
1     50 / 450 = 11.11%
Name: 0, dtype: int64
Test data
0    46 / 50 = 92%
1     4 / 50 = 8%
```

```
Name: 0, dtype: int64


What is the range of percentages for label 1 across the folds?
The range of percentages for label 1 is
from 10.22% to 11.11% for the train data and
from 8% to 16% for the test data


2.
ataset: skew - nfold: 1/10
Train data
0    401 / 449 = 89.31%
1     48 / 449 = 10.69%
Name: 0, dtype: int64
Test data
0    45 / 51 = 88.24%
1     6 / 51 = 11.76%
Name: 0, dtype: int64


Dataset: skew - nfold: 2/10
Train data
0    401 / 449 = 89.31%
1     48 / 449 = 10.69%
Name: 0, dtype: int64
Test data
0    45 / 51 = 88.24%
1     6 / 51 = 11.76%
Name: 0, dtype: int64


Dataset: skew - nfold: 3/10
Train data
0    401 / 449 = 89.31%
1     48 / 449 = 10.69%
Name: 0, dtype: int64
Test data
0    45 / 51 = 88.24%
1     6 / 51 = 11.76%
Name: 0, dtype: int64


Dataset: skew - nfold: 4/10
Train data
0    401 / 449 = 89.31%
1     48 / 449 = 10.69%
Name: 0, dtype: int64
Test data
0    45 / 51 = 88.24%
1     6 / 51 = 11.76%
Name: 0, dtype: int64
```

```
Dataset: skew - nfold: 5/10
Train data
0    401 / 449 = 89.31%
1     48 / 449 = 10.69%
Name: 0, dtype: int64
Test data
0    45 / 51 = 88.24%
1     6 / 51 = 11.76%
Name: 0, dtype: int64


Dataset: skew - nfold: 6/10
Train data
0    401 / 450 = 89.11%
1     49 / 450 = 10.89%
Name: 0, dtype: int64
Test data
0    45 / 50 = 90%
1     5 / 50 = 10%
Name: 0, dtype: int64


Dataset: skew - nfold: 7/10
Train data
0    402 / 451 = 89.14%
1     49 / 451 = 10.86%
Name: 0, dtype: int64
Test data
0    44 / 49 = 89.80%
1     5 / 49 = 10.20%
Name: 0, dtype: int64


Dataset: skew - nfold: 8/10
Train data
0    402 / 451 = 89.14%
1     49 / 451 = 10.86%
Name: 0, dtype: int64
Test data
0    44 / 49 = 89.8%
1     5 / 49 = 10.2%
Name: 0, dtype: int64


Dataset: skew - nfold: 9/10
Train data
0    402 / 451 = 89.14%
1     49 / 451 = 10.86%
Name: 0, dtype: int64
Test data
0    44 / 49 = 89.8%
1     5 / 49 = 10.2%
Name: 0, dtype: int64
```

```
Dataset: skew - nfold: 10/10
Train data
0    402 / 451 = 89.14%
1     49 / 451 = 10.86%
Name: 0, dtype: int64
Test data
0    44 / 49 = 89.8%
1     5 / 49 = 10.2%
Name: 0, dtype: int64
```

```
What is the range of percentages for label 1 across the folds?
The range of percentages for label 1 is
from 10.69% to 10.86% for the train data
from 10% to 11.76% for the test data
```

In [54]:
```
all_datasets = {'skew': { 'data': skew_gendata_data, 'labels': skew_gend

for dataset in all_datasets.keys():
    for nfolds in [10]:
        kfolds = cross_validation.KFold(all_datasets[dataset]['labels'].
        fold = 0
        for train, test in kfolds:
            fold += 1
            train_fold = pd.DataFrame(skew_gendata_t[train])
            test_fold = pd.DataFrame(skew_gendata_t[test])

            print "Dataset: " + dataset + " - nfold: " + str(fold) + "/"
            print "Train data"
            print train_fold[0].value_counts(sort=False)
            print "Test data"
            print test_fold[0].value_counts(sort=False)
            print "\n"
```

```
Dataset: skew - nfold: 1/10
Train data
0    401
1     49
Name: 0, dtype: int64
Test data
0    45
1     5
Name: 0, dtype: int64


Dataset: skew - nfold: 2/10
Train data
0    403
1     47
Name: 0, dtype: int64
Test data
0    43
1     7
Name: 0, dtype: int64


Dataset: skew - nfold: 3/10
Train data
0    403
1     47
Name: 0, dtype: int64
Test data
0    43
1     7
Name: 0, dtype: int64


Dataset: skew - nfold: 4/10
Train data
0    400
1     50
Name: 0, dtype: int64
Test data
0    46
1     4
Name: 0, dtype: int64


Dataset: skew - nfold: 5/10
Train data
0    400
1     50
Name: 0, dtype: int64
Test data
0    46
1     4
Name: 0, dtype: int64
```

```
Dataset: skew - nfold: 6/10
Train data
0    404
1     46
Name: 0, dtype: int64
Test data
0    42
1     8
Name: 0, dtype: int64


Dataset: skew - nfold: 7/10
Train data
0    403
1     47
Name: 0, dtype: int64
Test data
0    43
1     7
Name: 0, dtype: int64


Dataset: skew - nfold: 8/10
Train data
0    400
1     50
Name: 0, dtype: int64
Test data
0    46
1     4
Name: 0, dtype: int64


Dataset: skew - nfold: 9/10
Train data
0    400
1     50
Name: 0, dtype: int64
Test data
0    46
1     4
Name: 0, dtype: int64


Dataset: skew - nfold: 10/10
Train data
0    400
1     50
Name: 0, dtype: int64
Test data
0    46
1     4
Name: 0, dtype: int64
```

In [55]:
```
# print skew_gendata_targets.shape
print np.reshape(skew_gendata_targets.values,[500,]).shape
```

```
(500,)
```

In [56]:
```
# Use the StratifiedKFold cross-validation method and generate 10 folds
# What is the range of percentages for label 1 across the folds?
from sklearn.cross_validation import StratifiedKFold

kfolds = cross_validation.StratifiedKFold(np.reshape(skew_gendata_target
fold = 0
for train, test in kfolds:
    fold += 1
    train_fold = pd.DataFrame(skew_gendata_t[train])
    test_fold = pd.DataFrame(skew_gendata_t[test])

    print "Dataset: " + dataset + " - nfold: " + str(fold) + "/" + str(n
    print "Train data"
    print train_fold[0].value_counts(sort=False)
    print "Test data"
    print test_fold[0].value_counts(sort=False)
    print "\n"
```

```
Dataset: skew - nfold: 1/10
Train data
0    401
1     48
Name: 0, dtype: int64
Test data
0    45
1     6
Name: 0, dtype: int64


Dataset: skew - nfold: 2/10
Train data
0    401
1     48
Name: 0, dtype: int64
Test data
0    45
1     6
Name: 0, dtype: int64


Dataset: skew - nfold: 3/10
Train data
0    401
1     48
Name: 0, dtype: int64
Test data
0    45
1     6
Name: 0, dtype: int64


Dataset: skew - nfold: 4/10
Train data
0    401
1     48
Name: 0, dtype: int64
Test data
0    45
1     6
Name: 0, dtype: int64


Dataset: skew - nfold: 5/10
Train data
0    401
1     49
Name: 0, dtype: int64
Test data
0    45
1     5
Name: 0, dtype: int64
```

```
Dataset: skew - nfold: 6/10
Train data
0    401
1     49
Name: 0, dtype: int64
Test data
0    45
1     5
Name: 0, dtype: int64


Dataset: skew - nfold: 7/10
Train data
0    402
1     49
Name: 0, dtype: int64
Test data
0    44
1     5
Name: 0, dtype: int64


Dataset: skew - nfold: 8/10
Train data
0    402
1     49
Name: 0, dtype: int64
Test data
0    44
1     5
Name: 0, dtype: int64


Dataset: skew - nfold: 9/10
Train data
0    402
1     49
Name: 0, dtype: int64
Test data
0    44
1     5
Name: 0, dtype: int64


Dataset: skew - nfold: 10/10
Train data
0    402
1     49
Name: 0, dtype: int64
Test data
0    44
1     5
Name: 0, dtype: int64
```

One of the techniques we discussed was the bootstrap: a method to get a training set the size of our dataset while still getting useful training data. You can use the resample (http://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html) method from sklearn's util module to implement the bootstrap. Resample will generate a training set for you and, by default, make it the same size as the dataset. The only remaining challenge is figuring out what to put in the test set. You can use the `index` attribute of a pandas DataFrame to help you there...

Question 4: The Bootstrap (10 points)

1. Generate a bootstrap based train-test split for the three datasets (gendata, skew_gendata, and iris) using the `resample` method. Remember to set `random_state=20160121`
2. Compute the label proportions in the train and test sets. How do they compare?
3. Compute the descriptive statistics for each of the attributes. Which attribute has the largest difference in mean across the train and test sets?

Question 4 Answers

1. See code below

2. Gendata label proportions are not great, but they are close enough to be considered okay for our experiments.
See cell bellow for formatting

3. 15th attribute has the largest difference in mean across the train and test sets

```
Train data
0    259 / 500 = 52%
1    241 / 500 = 48%
Name: 0, dtype: int64
Test data
0    80 / 175 = 46%
1    95 / 175 = 54%
Name: 0, dtype: int64

GENDATA

                0           1           2           3           4
      5  \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
mean    -0.003139   -0.013427    0.021906    0.035423    0.012670
0.096852
std      0.991457    1.020868    1.018835    0.981776    1.041781
1.003704
min     -2.421082   -2.433603   -2.538242   -2.518441   -3.223148
-2.536563
25%     -0.720644   -0.757001   -0.615010   -0.571672   -0.740107
-0.535868
50%     -0.013046   -0.070751   -0.000330    0.008836    0.047413
0.171539
75%      0.661775    0.620804    0.679261    0.723654    0.644627
0.862462
max      3.594499    3.039572    3.003953    2.619359    2.454954
3.432993

                6           7           8           9          10
     11  \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
mean    -0.023746   -0.068885    0.042949    0.004360    0.024588
-0.075621
std      1.180548    1.008090    1.026896    0.973219    1.174396
0.979879
min     -3.326552   -2.588259   -2.807253   -2.690036   -3.315302
-3.102408
25%     -1.023971   -0.783844   -0.652606   -0.718984   -0.843480
-0.738850
50%     -0.103775   -0.064137    0.099790    0.030052    0.078181
-0.085270
75%      0.929908    0.643163    0.803920    0.658938    1.040590
0.649183
max      3.023143    2.731177    2.949241    3.135951    2.544278
2.664877

               12          13          14          15          16
     17  \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
```

```
mean    -0.024864   -0.043657    0.075582    0.058304   -0.033319
-0.027178
std      0.961753    1.191530    0.933707    0.965128    0.975602
1.013068
min     -3.109823   -3.210675   -2.261836   -3.404818   -2.450816
-2.883941
25%     -0.624986   -0.998630   -0.612217   -0.643052   -0.759561
-0.800636
50%     -0.091031   -0.229074    0.085333    0.101073   -0.032146
0.063717
75%      0.695669    0.955272    0.796331    0.798477    0.619367
0.642004
max      2.706916    3.233625    2.760497    2.630985    3.094925
3.264062

               18          19
count  500.000000  500.000000
mean     0.057141   -0.014074
std      1.032833    1.002647
min     -3.389571   -3.335489
25%     -0.593457   -0.728227
50%      0.118198   -0.007640
75%      0.708951    0.590290
max      2.811399    3.498259
                0           1           2           3           4
      5  \
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean     0.005619    0.093439    0.032655    0.026498    0.168349
0.051515
std      1.050662    0.974157    0.855438    1.087417    1.041658
1.011249
min     -2.515058   -1.888726   -2.647789   -2.530270   -2.538309
-2.966130
25%     -0.647823   -0.514815   -0.510086   -0.699466   -0.490877
-0.511488
50%     -0.029770    0.019143    0.069402    0.075448    0.123224
0.088291
75%      0.725104    0.621529    0.613381    0.740656    0.841647
0.691024
max      2.370750    3.012955    2.391449    2.479612    2.712419
2.421206

                6           7           8           9          10
     11  \
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean    -0.061467    0.029711   -0.075015    0.022355   -0.108282
-0.041691
std      1.202664    0.999409    0.995259    1.080562    1.138179
0.892856
min     -2.798720   -2.736625   -3.060349   -3.021813   -3.265268
-2.250563
25%     -1.048574   -0.599275   -0.683936   -0.800349   -0.823582
```

```
-0.600243
50%     -0.260886    0.060532   -0.100929   -0.012445   -0.106142
-0.130859
75%      0.859118    0.601171    0.469951    0.770578    0.702921
0.552187
max      2.639439    2.366253    3.080764    2.726222    1.985425
1.873812

               12          13          14          15          16
     17  \
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean     0.006369    0.089047    0.150264   -0.100891    0.078559
-0.122936
std      1.015259    1.247470    1.010591    0.898343    1.096469
0.907329
min     -2.451936   -3.183078   -3.239524   -2.446034   -3.068310
-2.852202
25%     -0.731692   -0.912062   -0.470415   -0.725092   -0.759722
-0.683651
50%      0.010469    0.106732    0.208500   -0.067307    0.081378
-0.116021
75%      0.754626    1.068886    0.775779    0.522074    0.869717
0.439826
max      2.373247    3.303948    3.446098    2.253107    2.961835
2.598636

               18          19
count  175.000000  175.000000
mean    -0.087160    0.027925
std      0.969802    1.060347
min     -2.479270   -3.110583
25%     -0.762449   -0.576496
50%      0.043538    0.031946
75%      0.585937    0.723740
max      2.477594    2.865515

-----------------------------
SKEW

Skew data label proportions are not very good, especially the under
representation of label 0 in the test set. It's under represented
which means that our model might not make the right predictions
accordingly
15th attribute has the largest difference in mean across the train
and test sets

Train data Skew
0    458 / 500 = 92%
1     42 / 500 = 8%
Name: 0, dtype: int64
Test data Skew
0    149 / 175 = 85%
1     26 / 175 = 15%
```

```
Name: 0, dtype: int64

               0          1          2          3          4
     5   \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
mean    -0.003139   -0.536252    0.021906    0.035423    0.012670
0.096852
std      0.991457    0.767778    1.018835    0.981776    1.041781
1.003704
min     -2.421082   -2.433603   -2.538242   -2.518441   -3.223148
-2.536563
25%     -0.720644   -1.192733   -0.615010   -0.571672   -0.740107
-0.535868
50%     -0.013046   -0.456621   -0.000330    0.008836    0.047413
0.171539
75%      0.661775    0.031560    0.679261    0.723654    0.644627
0.862462
max      3.594499    1.763571    3.003953    2.619359    2.454954
3.432993

               6          7          8          9         10
    11   \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
mean     0.026556   -0.068885    0.042949    0.004360    0.711829
-0.075621
std      1.088736    1.008090    1.026896    0.973219    0.806941
0.979879
min     -1.911546   -2.588259   -2.807253   -2.690036   -1.990374
-3.102408
25%     -0.928071   -0.783844   -0.652606   -0.718984    0.184751
-0.738850
50%     -0.060348   -0.064137    0.099790    0.030052    0.844790
-0.085270
75%      0.949863    0.643163    0.803920    0.658938    1.347421
0.649183
max      3.023143    2.731177    2.949241    3.135951    2.544278
2.664877

              12         13         14         15         16
    17   \
count  500.000000  500.000000  500.000000  500.000000  500.000000
500.000000
mean    -0.024864   -0.820048    0.075582    0.058304   -0.033319
-0.027178
std      0.961753    0.720215    0.933707    0.965128    0.975602
1.013068
min     -3.109823   -3.210675   -2.261836   -3.404818   -2.450816
-2.883941
25%     -0.624986   -1.133534   -0.612217   -0.643052   -0.759561
-0.800636
50%     -0.091031   -0.955740    0.085333    0.101073   -0.032146
0.063717
```

```
75%       0.695669   -0.698445    0.796331    0.798477    0.619367
0.642004
max       2.706916    1.877995    2.760497    2.630985    3.094925
3.264062

               18          19
count  500.000000  500.000000
mean     0.057141   -0.014074
std      1.032833    1.002647
min     -3.389571   -3.335489
25%     -0.593457   -0.728227
50%      0.118198   -0.007640
75%      0.708951    0.590290
max      2.811399    3.498259
                0           1           2           3           4
      5  \
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean     0.005619   -0.398621    0.032655    0.026498    0.168349
0.051515
std      1.050662    0.812624    0.855438    1.087417    1.041658
1.011249
min     -2.515058   -2.046301   -2.647789   -2.530270   -2.538309
-2.966130
25%     -0.647823   -1.060373   -0.510086   -0.699466   -0.490877
-0.511488
50%     -0.029770   -0.362135    0.069402    0.075448    0.123224
0.088291
75%      0.725104    0.147743    0.613381    0.740656    0.841647
0.691024
max      2.370750    2.245763    2.391449    2.479612    2.712419
2.421206

                6           7           8           9          10
     11  \
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean    -0.094514    0.029711   -0.075015    0.022355    0.560796
-0.041691
std      1.108151    0.999409    0.995259    1.080562    0.919052
0.892856
min     -2.272284   -2.736625   -3.060349   -3.021813   -2.552006
-2.250563
25%     -1.027302   -0.599275   -0.683936   -0.800349   -0.074823
-0.600243
50%     -0.336453    0.060532   -0.100929   -0.012445    0.721716
-0.130859
75%      0.829838    0.601171    0.469951    0.770578    1.240600
0.552187
max      2.626074    2.366253    3.080764    2.726222    2.017827
1.873812

               12          13          14          15          16
     17  \
```

```
count  175.000000  175.000000  175.000000  175.000000  175.000000
175.000000
mean     0.006369   -0.717808    0.150264   -0.100891    0.078559
-0.122936
std      1.015259    0.992545    1.010591    0.898343    1.096469
0.907329
min     -2.451936   -3.183078   -3.239524   -2.446034   -3.068310
-2.852202
25%     -0.731692   -1.193541   -0.470415   -0.725092   -0.759722
-0.683651
50%      0.010469   -0.946576    0.208500   -0.067307    0.081378
-0.116021
75%      0.754626   -0.325370    0.775779    0.522074    0.869717
0.439826
max      2.373247    3.303948    3.446098    2.253107    2.961835
2.598636

               18          19
count  175.000000  175.000000
mean    -0.087160    0.027925
std      0.969802    1.060347
min     -2.479270   -3.110583
25%     -0.762449   -0.576496
50%      0.043538    0.031946
75%      0.585937    0.723740
max      2.477594    2.865515


-----------------------------
IRIS

Iris data label proportions seem good. Better than the other two
datasets
2nd attribute has the largest difference in mean across the train and
test sets


Train data Iris
0    54 / 150 = 36%
1    50 / 150 = 33%
2    46 / 150 = 31%
Name: 0, dtype: int64
Test data Iris
0    20 / 54 = 37%
1    16 / 54 = 30%
2    18 / 54 = 33.%
Name: 0, dtype: int64


Train data Iris
                0           1           2           3
count  150.000000  150.000000  150.000000  150.000000
mean     5.815333    3.018000    3.649333    1.128667
std      0.782590    0.398162    1.731498    0.748810
```

```
min       4.400000    2.000000    1.000000    0.100000
25%       5.100000    2.800000    1.500000    0.200000
50%       5.800000    3.000000    4.350000    1.300000
75%       6.375000    3.300000    5.000000    1.800000
max       7.700000    4.400000    6.900000    2.500000

Test data Iris
               0          1          2          3
count  54.000000  54.000000  54.000000  54.000000
mean    5.783333   3.114815   3.618519   1.157407
std     0.855559   0.459864   1.845042   0.802261
min     4.300000   2.200000   1.100000   0.100000
25%     5.100000   2.800000   1.500000   0.225000
50%     5.600000   3.000000   4.050000   1.300000
75%     6.300000   3.475000   5.100000   1.800000
max     7.900000   4.200000   6.700000   2.500000
```

In [57]:
```
from sklearn.utils import resample

boot_gendata_train = resample(gendata_targets, random_state=20160121)
boot_gendata_test = gendata_targets.loc[~gendata_targets.index.isin(list

# print boot_gendata_train
# print boot_gendata_test

print "Train data"
print boot_gendata_train[0].value_counts(sort=False)
print "Test data"
print boot_gendata_test[0].value_counts(sort=False)
print "\n"

boot_gendata_train = resample(gendata_data, random_state=20160121)
boot_gendata_test = gendata_data.loc[~gendata_data.index.isin(list(boot_

print boot_gendata_train.describe()
print boot_gendata_test.describe()
```

```
Train data
0    259
1    241
Name: 0, dtype: int64
Test data
0    80
1    95
Name: 0, dtype: int64


               0           1           2           3           4
5  \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean    -0.003139   -0.013427    0.021906    0.035423    0.012670
0.096852
std      0.991457    1.020868    1.018835    0.981776    1.041781
1.003704
min     -2.421082   -2.433603   -2.538242   -2.518441   -3.223148
-2.536563
25%     -0.720644   -0.757001   -0.615010   -0.571672   -0.740107
-0.535868
50%     -0.013046   -0.070751   -0.000330    0.008836    0.047413
0.171539
75%      0.661775    0.620804    0.679261    0.723654    0.644627
0.862462
max      3.594499    3.039572    3.003953    2.619359    2.454954
3.432993

               6           7           8           9          10
11  \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean    -0.023746   -0.068885    0.042949    0.004360    0.024588
-0.075621
std      1.180548    1.008090    1.026896    0.973219    1.174396
0.979879
min     -3.326552   -2.588259   -2.807253   -2.690036   -3.315302
-3.102408
25%     -1.023971   -0.783844   -0.652606   -0.718984   -0.843480
-0.738850
50%     -0.103775   -0.064137    0.099790    0.030052    0.078181
-0.085270
75%      0.929908    0.643163    0.803920    0.658938    1.040590
0.649183
max      3.023143    2.731177    2.949241    3.135951    2.544278
2.664877

              12          13          14          15          16
17  \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean    -0.024864   -0.043657    0.075582    0.058304   -0.033319
-0.027178
```

```
std       0.961753    1.191530    0.933707    0.965128    0.975602
1.013068
min      -3.109823   -3.210675   -2.261836   -3.404818   -2.450816
-2.883941
25%      -0.624986   -0.998630   -0.612217   -0.643052   -0.759561
-0.800636
50%      -0.091031   -0.229074    0.085333    0.101073   -0.032146
0.063717
75%       0.695669    0.955272    0.796331    0.798477    0.619367
0.642004
max       2.706916    3.233625    2.760497    2.630985    3.094925
3.264062

               18          19
count  500.000000  500.000000
mean     0.057141   -0.014074
std      1.032833    1.002647
min     -3.389571   -3.335489
25%     -0.593457   -0.728227
50%      0.118198   -0.007640
75%      0.708951    0.590290
max      2.811399    3.498259
               0           1           2           3           4
5  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean     0.005619    0.093439    0.032655    0.026498    0.168349
0.051515
std      1.050662    0.974157    0.855438    1.087417    1.041658
1.011249
min     -2.515058   -1.888726   -2.647789   -2.530270   -2.538309
-2.966130
25%     -0.647823   -0.514815   -0.510086   -0.699466   -0.490877
-0.511488
50%     -0.029770    0.019143    0.069402    0.075448    0.123224
0.088291
75%      0.725104    0.621529    0.613381    0.740656    0.841647
0.691024
max      2.370750    3.012955    2.391449    2.479612    2.712419
2.421206

               6           7           8           9          10
11  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean    -0.061467    0.029711   -0.075015    0.022355   -0.108282
-0.041691
std      1.202664    0.999409    0.995259    1.080562    1.138179
0.892856
min     -2.798720   -2.736625   -3.060349   -3.021813   -3.265268
-2.250563
25%     -1.048574   -0.599275   -0.683936   -0.800349   -0.823582
-0.600243
50%     -0.260886    0.060532   -0.100929   -0.012445   -0.106142
```

```
-0.130859
75%       0.859118    0.601171    0.469951    0.770578    0.702921
0.552187
max       2.639439    2.366253    3.080764    2.726222    1.985425
1.873812

                12          13          14          15          16
17  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean     0.006369    0.089047    0.150264   -0.100891    0.078559
-0.122936
std      1.015259    1.247470    1.010591    0.898343    1.096469
0.907329
min     -2.451936   -3.183078   -3.239524   -2.446034   -3.068310
-2.852202
25%     -0.731692   -0.912062   -0.470415   -0.725092   -0.759722
-0.683651
50%      0.010469    0.106732    0.208500   -0.067307    0.081378
-0.116021
75%      0.754626    1.068886    0.775779    0.522074    0.869717
0.439826
max      2.373247    3.303948    3.446098    2.253107    2.961835
2.598636

                18          19
count  175.000000  175.000000
mean    -0.087160    0.027925
std      0.969802    1.060347
min     -2.479270   -3.110583
25%     -0.762449   -0.576496
50%      0.043538    0.031946
75%      0.585937    0.723740
max      2.477594    2.865515
```

In [58]:
```
from sklearn.utils import resample

boot_skew_gendata_train = resample(skew_gendata_targets, random_state=20
boot_skew_gendata_test = skew_gendata_targets.loc[~skew_gendata_targets.

# print boot_skew_gendata_train.shape
# print boot_skew_gendata_test.shape

print "Train data Skew"
print boot_skew_gendata_train[0].value_counts(sort=False)
print "Test data Skew"
print boot_skew_gendata_test[0].value_counts(sort=False)
print "\n"

boot_skew_gendata_train = resample(skew_gendata_data, random_state=20160
boot_skew_gendata_test = skew_gendata_data.loc[~skew_gendata_data.index.

print boot_skew_gendata_train.describe()
print boot_skew_gendata_test.describe()
```

```
Train data Skew
0    458
1     42
Name: 0, dtype: int64
Test data Skew
0    149
1     26
Name: 0, dtype: int64


               0           1           2           3           4  
5   \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean    -0.003139   -0.536252    0.021906    0.035423    0.012670  
0.096852
std      0.991457    0.767778    1.018835    0.981776    1.041781  
1.003704
min     -2.421082   -2.433603   -2.538242   -2.518441   -3.223148  
-2.536563
25%     -0.720644   -1.192733   -0.615010   -0.571672   -0.740107  
-0.535868
50%     -0.013046   -0.456621   -0.000330    0.008836    0.047413  
0.171539
75%      0.661775    0.031560    0.679261    0.723654    0.644627  
0.862462
max      3.594499    1.763571    3.003953    2.619359    2.454954  
3.432993

               6           7           8           9          10  
11  \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean     0.026556   -0.068885    0.042949    0.004360    0.711829  
-0.075621
std      1.088736    1.008090    1.026896    0.973219    0.806941  
0.979879
min     -1.911546   -2.588259   -2.807253   -2.690036   -1.990374  
-3.102408
25%     -0.928071   -0.783844   -0.652606   -0.718984    0.184751  
-0.738850
50%     -0.060348   -0.064137    0.099790    0.030052    0.844790  
-0.085270
75%      0.949863    0.643163    0.803920    0.658938    1.347421  
0.649183
max      3.023143    2.731177    2.949241    3.135951    2.544278  
2.664877

              12          13          14          15          16  
17  \
count  500.000000  500.000000  500.000000  500.000000  500.000000  5
00.000000
mean    -0.024864   -0.820048    0.075582    0.058304   -0.033319  
-0.027178
```

```
std       0.961753    0.720215    0.933707    0.965128    0.975602
1.013068
min      -3.109823   -3.210675   -2.261836   -3.404818   -2.450816
-2.883941
25%      -0.624986   -1.133534   -0.612217   -0.643052   -0.759561
-0.800636
50%      -0.091031   -0.955740    0.085333    0.101073   -0.032146
0.063717
75%       0.695669   -0.698445    0.796331    0.798477    0.619367
0.642004
max       2.706916    1.877995    2.760497    2.630985    3.094925
3.264062

               18          19
count  500.000000  500.000000
mean     0.057141   -0.014074
std      1.032833    1.002647
min     -3.389571   -3.335489
25%     -0.593457   -0.728227
50%      0.118198   -0.007640
75%      0.708951    0.590290
max      2.811399    3.498259
                0           1           2           3           4
5  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean     0.005619   -0.398621    0.032655    0.026498    0.168349
0.051515
std      1.050662    0.812624    0.855438    1.087417    1.041658
1.011249
min     -2.515058   -2.046301   -2.647789   -2.530270   -2.538309
-2.966130
25%     -0.647823   -1.060373   -0.510086   -0.699466   -0.490877
-0.511488
50%     -0.029770   -0.362135    0.069402    0.075448    0.123224
0.088291
75%      0.725104    0.147743    0.613381    0.740656    0.841647
0.691024
max      2.370750    2.245763    2.391449    2.479612    2.712419
2.421206

                6           7           8           9          10
11  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean    -0.094514    0.029711   -0.075015    0.022355    0.560796
-0.041691
std      1.108151    0.999409    0.995259    1.080562    0.919052
0.892856
min     -2.272284   -2.736625   -3.060349   -3.021813   -2.552006
-2.250563
25%     -1.027302   -0.599275   -0.683936   -0.800349   -0.074823
-0.600243
50%     -0.336453    0.060532   -0.100929   -0.012445    0.721716
```

```
-0.130859
75%       0.829838    0.601171    0.469951    0.770578    1.240600
0.552187
max       2.626074    2.366253    3.080764    2.726222    2.017827
1.873812

                12          13          14          15          16
17  \
count  175.000000  175.000000  175.000000  175.000000  175.000000  1
75.000000
mean     0.006369   -0.717808    0.150264   -0.100891    0.078559
-0.122936
std      1.015259    0.992545    1.010591    0.898343    1.096469
0.907329
min     -2.451936   -3.183078   -3.239524   -2.446034   -3.068310
-2.852202
25%     -0.731692   -1.193541   -0.470415   -0.725092   -0.759722
-0.683651
50%      0.010469   -0.946576    0.208500   -0.067307    0.081378
-0.116021
75%      0.754626   -0.325370    0.775779    0.522074    0.869717
0.439826
max      2.373247    3.303948    3.446098    2.253107    2.961835
2.598636

                18          19
count  175.000000  175.000000
mean    -0.087160    0.027925
std      0.969802    1.060347
min     -2.479270   -3.110583
25%     -0.762449   -0.576496
50%      0.043538    0.031946
75%      0.585937    0.723740
max      2.477594    2.865515
```

In [59]:
```
from sklearn.utils import resample

iris_data = pd.DataFrame(iris.target)
boot_iris_train = resample(iris_data, random_state=20160121)

boot_iris_test = iris_data.loc[~iris_data.index.isin(list(boot_iris_trai

print "Train data Iris "
print boot_iris_train[0].value_counts(sort=False)
print "Test data Iris"
print boot_iris_test[0].value_counts(sort=False)
print "\n"

iris_data = pd.DataFrame(iris.data)
boot_iris_train = resample(iris_data, random_state=20160121)

boot_iris_test = iris_data.loc[~iris_data.index.isin(list(boot_iris_trai

print "Train data Iris"
print boot_iris_train.describe()
print "\nTest data Iris"
print boot_iris_test.describe()
print "\n"
```

```
Train data Iris
0    54
1    50
2    46
Name: 0, dtype: int64
Test data Iris
0    20
1    16
2    18
Name: 0, dtype: int64


Train data Iris
                0           1           2           3
count  150.000000  150.000000  150.000000  150.000000
mean     5.815333    3.018000    3.649333    1.128667
std      0.782590    0.398162    1.731498    0.748810
min      4.400000    2.000000    1.000000    0.100000
25%      5.100000    2.800000    1.500000    0.200000
50%      5.800000    3.000000    4.350000    1.300000
75%      6.375000    3.300000    5.000000    1.800000
max      7.700000    4.400000    6.900000    2.500000

Test data Iris
               0          1          2          3
count  54.000000  54.000000  54.000000  54.000000
mean    5.783333   3.114815   3.618519   1.157407
std     0.855559   0.459864   1.845042   0.802261
min     4.300000   2.200000   1.100000   0.100000
25%     5.100000   2.800000   1.500000   0.225000
50%     5.600000   3.000000   4.050000   1.300000
75%     6.300000   3.475000   5.100000   1.800000
max     7.900000   4.200000   6.700000   2.500000
```

Evaluating Models

Now that we've looked at various means of splitting our data, we can explore the performance metrics used to evaluate our results. You can find a full list of these metrics documented in the metrics documentation (http://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics). Since it's not very interesting to look at these metrics in isolation, we can generate some classification output for our data and see how the different methods perform.

One of the other topics we discussed was comparing schemes or methods to decide if the differences between them were statistically meaningful. We evaluate the statistical significance of these results using t-tests. The t-test isn't in scikit-learn, but in a related module, `scipy`. You can read about all of the statistical tests in the scipy stats documentation (http://docs.scipy.org/doc/scipy/reference/stats.html)

In [60]:
```
# Train the dummy classifier on the Iris data:
dummy_classifier_iris = dummy.DummyClassifier();
dummy_classifier_iris.fit(iris_train_data, iris_train_labels);
dummy_iris_predictions = dummy_classifier_iris.predict(iris_test_data);
dummy_iris_predictions_proba = dummy_classifier_iris.predict_proba(iris_

#Train the decision tree classifier on the Iris data
dtree_iris = tree.DecisionTreeClassifier();
dtree_iris.fit(iris_train_data, iris_train_labels);
dtree_iris_predictions = dtree_iris.predict(iris_test_data);
dtree_iris_predictions_proba = dtree_iris.predict_proba(iris_test_data);

#The classification report is a handy tool to see many metrics from one
print "Dummy classifier report"
print metrics.classification_report(iris_test_labels, dummy_iris_predict

print "Iris classifier report"
print metrics.classification_report(iris_test_labels, dtree_iris_predict

#We can compare the two predictions using the t-test (from scipy)
stats.ttest_rel(dummy_iris_predictions, dtree_iris_predictions)
```

```
Dummy classifier report
             precision    recall  f1-score   support

          0       0.18      0.15      0.17        13
          1       0.36      0.31      0.33        13
          2       0.31      0.42      0.36        12

avg / total       0.29      0.29      0.28        38

Iris classifier report
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        13
          1       0.80      0.92      0.86        13
          2       0.90      0.75      0.82        12

avg / total       0.90      0.89      0.89        38
```

Out[60]:
```
Ttest_relResult(statistic=1.0519230276378431, pvalue=0.2996567025096
6109)
```

Question 5: Evaluation Metrics: Discrete Predictions (10 points)

1. Compute the accuracy for each classifier
2. Make a simple train-test split of gendata (using `randomstate=20160121`) and perform the same analysis: classification report and t-test
3. One of the issues in realistic evaluation settings is that the test outputs are not

always generated in the same testing regime. Perform 5-fold cross-validation using the DummyClassifier and 10-fold cross-validation using the DecisionTree classifier. Read the documentation to determine the correct t-test function to use to compare these two sets of results, and perform the t-test across folds

Question 5 Answers

1. Dummy Classifier accuracy: 0.421052631579
Decision Tree Classifier accuracy: 0.894736842105

2. Ttest_relResult(statistic=-0.74098388608089094, pvalue=0.45976285530296646) see cell below for better formattting

3. Ttest_indResult(statistic=14.289053827191161, pvalue=2.509241547322248e-09)

```
Part 2
Dummy classifier report
             precision    recall  f1-score   support

          0       0.45      0.49      0.47        78
          1       0.50      0.46      0.48        87

avg / total       0.47      0.47      0.47       165

Iris classifier report
             precision    recall  f1-score   support

          0       0.86      0.86      0.86        78
          1       0.87      0.87      0.87        87

avg / total       0.87      0.87      0.87       165

Ttest_relResult(statistic=-0.74098388608089094,
pvalue=0.45976285530296646)
```

In [61]:

```
# part 1
from sklearn.metrics import accuracy_score

print "Dummy Classifier accuracy: " + str(accuracy_score(iris_test_label
print "Decision Tree Classifier accuracy: " + str(accuracy_score(iris_te
```

```
Dummy Classifier accuracy: 0.289473684211
Decision Tree Classifier accuracy: 0.894736842105
```

In [62]:
```
# part 2
[gendata_train_data_33, gendata_test_data_33,  gendata_train_labels_33,

dummy_classifier_gendata = dummy.DummyClassifier();
dummy_classifier_gendata.fit(gendata_train_data_33, gendata_train_labels
dummy_gendata_predictions = dummy_classifier_gendata.predict(gendata_tes
dummy_gendata_predictions_proba = dummy_classifier_gendata.predict_proba

dtree_gendata = tree.DecisionTreeClassifier();
dtree_gendata.fit(gendata_train_data_33, gendata_train_labels_33)
dtree_gendata_predictions = dtree_gendata.predict(gendata_test_data_33)
dtree_gendata_predictions_prob = dtree_gendata.predict_proba(gendata_tes

#The classification report is a handy tool to see many metrics from one
print "Dummy classifier report"
print metrics.classification_report(gendata_test_labels_33, dummy_gendat

print "Iris classifier report"
print metrics.classification_report(gendata_test_labels_33, dtree_gendat

#We can compare the two predictions using the t-test (from scipy)
print stats.ttest_rel(dummy_gendata_predictions, dtree_gendata_predictio
```

```
Dummy classifier report
             precision    recall  f1-score   support

          0       0.48      0.54      0.51        78
          1       0.53      0.47      0.50        87

avg / total       0.51      0.50      0.50       165

Iris classifier report
             precision    recall  f1-score   support

          0       0.81      0.82      0.82        78
          1       0.84      0.83      0.83        87

avg / total       0.82      0.82      0.82       165

Ttest_relResult(statistic=-0.9431404248511186, pvalue=0.346996382419
67168)
```

In [63]:
```
# part 3
from sklearn.metrics import accuracy_score
kfolds = cross_validation.KFold(500, n_folds=5, shuffle=True, random_sta
fold = 0

dummy_accuracy = []
for train, test in kfolds:
    fold += 1
    train_fold = gendata_d[train]
    train_labels = gendata_t[train]

    test_fold = gendata_d[test]
    test_labels = gendata_t[test]

    dummy_classifier_gendata = dummy.DummyClassifier();
    dummy_classifier_gendata.fit(train_fold, train_labels)
    dummy_gendata_predictions = dummy_classifier_gendata.predict(test_fo
    dummy_gendata_predictions_proba = dummy_classifier_gendata.predict_p

    score = 1 - accuracy_score(test_labels, dummy_gendata_predictions)
#     score = accuracy_score(test_labels, dummy_gendata_predictions)
    dummy_accuracy.append(score)

dtree_accuracy = []
kfolds = cross_validation.KFold(500, n_folds=10, shuffle=True, random_st
for train, test in kfolds:
    fold += 1
    train_fold = gendata_d[train]
    train_labels = gendata_t[train]

    test_fold = gendata_d[test]
    test_labels = gendata_t[test]

    dtree_gendata = tree.DecisionTreeClassifier();
    dtree_gendata.fit(train_fold, train_labels)
    dtree_gendata_predictions = dtree_gendata.predict(test_fold)
    dtree_gendata_predictions_prob = dtree_gendata.predict_proba(test_fo

    score = 1 - accuracy_score(test_labels, dtree_gendata_predictions)
#     score = accuracy_score(test_labels, dtree_gendata_predictions)
    dtree_accuracy.append(score)

#We can compare the two predictions using the t-test (from scipy)
print dummy_accuracy
print dtree_accuracy
stats.ttest_ind(dummy_accuracy, dtree_accuracy)
```

```
[0.48999999999999999, 0.48999999999999999, 0.40000000000000002, 0.53
000000000000003, 0.60999999999999999]
[0.19999999999999996, 0.07999999999999996, 0.099999999999999978, 0.0
7999999999999996, 0.099999999999999978, 0.16000000000000003, 0.02000
000000000018, 0.060000000000000053, 0.14000000000000001, 0.04000000
0000000036]
```

Out[63]:
```
Ttest_indResult(statistic=11.871860348739082, pvalue=2.3767018632247
007e-08)
```

In [ ]:
```
```