# **Import Python Packages**

In [2]:

**import** os

**import** sys

**import** numpy **as** np

**import** pandas **as** pd

​

**from** sklearn.model\_selection **import** train\_test\_split

**from** scipy.io **import** loadmat

# **Import Pyod Packages & the methods**

In [3]:

**from** pyod.models.pca **import** PCA

**from** pyod.models.mcd **import** MCD

**from** pyod.models.ocsvm **import** OCSVM

**from** pyod.models.lof **import** LOF

**from** pyod.models.cblof **import** CBLOF

**from** pyod.models.knn **import** KNN

**from** pyod.models.hbos **import** HBOS

**from** pyod.models.abod **import** ABOD

**from** pyod.models.iforest **import** IForest

**from** pyod.models.feature\_bagging **import** FeatureBagging

# **Import Metrics Packages**

In [4]:

**from** pyod.utils.utility **import** standardizer

**from** pyod.utils.utility **import** precision\_n\_scores

**from** sklearn.metrics **import** roc\_auc\_score

# **Define data file and read X and y**

In [6]:

mat\_file\_list **=** ['arrhythmia.mat',

'cardio.mat',

'glass.mat',

'ionosphere.mat',

'letter.mat',

'lympho.mat',

'mnist.mat',

'musk.mat',

'optdigits.mat',

'pendigits.mat',

'pima.mat',

'satellite.mat',

'satimage-2.mat',

'shuttle.mat',

'vertebral.mat',

'vowels.mat',

'wbc.mat']

# **How to load Mat File**

In [8]:

data**=**loadmat('data/cardio.mat')

In [9]:

data

Out[9]:

{'\_\_header\_\_': b'MATLAB 5.0 MAT-file, written by Octave 3.8.0, 2014-12-18 10:48:09 UTC',

'\_\_version\_\_': '1.0',

'\_\_globals\_\_': [],

'X': array([[ 0.00491231, 0.69319077, -0.20364049, ..., 0.23149795,

-0.28978574, -0.49329397],

[ 0.11072935, -0.07990259, -0.20364049, ..., 0.09356344,

-0.25638541, -0.49329397],

[ 0.21654639, -0.27244466, -0.20364049, ..., 0.02459619,

-0.25638541, 1.14001753],

...,

[-0.41835583, -0.91998844, -0.16463485, ..., -1.49268341,

0.24461959, -0.49329397],

[-0.41835583, -0.91998844, -0.15093411, ..., -1.42371616,

0.14441859, -0.49329397],

[-0.41835583, -0.91998844, -0.20364049, ..., -1.28578165,

3.58465295, -0.49329397]]),

'y': array([[0.],

[0.],

[0.],

...,

[1.],

[1.],

[1.]])}

In [10]:

len(data)

Out[10]:

5

In [11]:

data.keys()

Out[11]:

dict\_keys(['\_\_header\_\_', '\_\_version\_\_', '\_\_globals\_\_', 'X', 'y'])

In [12]:

data.values()

Out[12]:

dict\_values([b'MATLAB 5.0 MAT-file, written by Octave 3.8.0, 2014-12-18 10:48:09 UTC', '1.0', [], array([[ 0.00491231, 0.69319077, -0.20364049, ..., 0.23149795,

-0.28978574, -0.49329397],

[ 0.11072935, -0.07990259, -0.20364049, ..., 0.09356344,

-0.25638541, -0.49329397],

[ 0.21654639, -0.27244466, -0.20364049, ..., 0.02459619,

-0.25638541, 1.14001753],

...,

[-0.41835583, -0.91998844, -0.16463485, ..., -1.49268341,

0.24461959, -0.49329397],

[-0.41835583, -0.91998844, -0.15093411, ..., -1.42371616,

0.14441859, -0.49329397],

[-0.41835583, -0.91998844, -0.20364049, ..., -1.28578165,

3.58465295, -0.49329397]]), array([[0.],

[0.],

[0.],

...,

[1.],

[1.],

[1.]])])

# **Input(Independent) Feature Shape in Mat file format**

In [13]:

type(data['X']),data['X'].shape

Out[13]:

(numpy.ndarray, (1831, 21))

# **Dependent/ Target /Output Feature shape**

In [14]:

type(data['y']),data['y'].shape

Out[14]:

(numpy.ndarray, (1831, 1))

# **Exploring All Mat files**

In [\*]:

**from** time **import** time

random\_state **=** np.random.RandomState(42)

​

**for** mat\_file **in** mat\_file\_list:

print("\n... Processing", mat\_file, '...')

mat **=** loadmat(os.path.join('data', mat\_file))

​

X **=** mat['X']

y **=** mat['y'].ravel()

outliers\_fraction **=** np.count\_nonzero(y) **/** len(y)

outliers\_percentage **=** round(outliers\_fraction **\*** 100, ndigits**=**4)

​

*# construct containers for saving results*

roc\_list **=** [mat\_file[:**-**4], X.shape[0], X.shape[1], outliers\_percentage]

prn\_list **=** [mat\_file[:**-**4], X.shape[0], X.shape[1], outliers\_percentage]

time\_list **=** [mat\_file[:**-**4], X.shape[0], X.shape[1], outliers\_percentage]

​

*# 60% data for training and 40% for testing*

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.4,

random\_state**=**random\_state)

​

*# standardizing data for processing*

X\_train\_norm, X\_test\_norm **=** standardizer(X\_train, X\_test)

​

classifiers **=** {'Angle-based Outlier Detector (ABOD)': ABOD(

contamination**=**outliers\_fraction),

'Cluster-based Local Outlier Factor': CBLOF(

contamination**=**outliers\_fraction, check\_estimator**=False**,

random\_state**=**random\_state),

'Feature Bagging': FeatureBagging(contamination**=**outliers\_fraction,

random\_state**=**random\_state),

'Histogram-base Outlier Detection (HBOS)': HBOS(

contamination**=**outliers\_fraction),

'Isolation Forest': IForest(contamination**=**outliers\_fraction,

random\_state**=**random\_state),

'K Nearest Neighbors (KNN)': KNN(contamination**=**outliers\_fraction),

'Local Outlier Factor (LOF)': LOF(

contamination**=**outliers\_fraction),

'Minimum Covariance Determinant (MCD)': MCD(

contamination**=**outliers\_fraction, random\_state**=**random\_state),

'One-class SVM (OCSVM)': OCSVM(contamination**=**outliers\_fraction),

'Principal Component Analysis (PCA)': PCA(

contamination**=**outliers\_fraction, random\_state**=**random\_state),

}

​

**for** clf\_name, clf **in** classifiers.items():

t0 **=** time()

clf.fit(X\_train\_norm)

test\_scores **=** clf.decision\_function(X\_test\_norm)

t1 **=** time()

duration **=** round(t1 **-** t0, ndigits**=**4)

time\_list.append(duration)

​

roc **=** round(roc\_auc\_score(y\_test, test\_scores), ndigits**=**4)

prn **=** round(precision\_n\_scores(y\_test, test\_scores), ndigits**=**4)

​

print('{clf\_name} ROC:{roc}, precision @ rank n:{prn}, '

'execution time: {duration}s'.format(

clf\_name**=**clf\_name, roc**=**roc, prn**=**prn, duration**=**duration))

​

roc\_list.append(roc)

prn\_list.append(prn)

​

temp\_df **=** pd.DataFrame(time\_list).transpose()

temp\_df.columns **=** df\_columns

time\_df **=** pd.concat([time\_df, temp\_df], axis**=**0)

​

temp\_df **=** pd.DataFrame(roc\_list).transpose()

temp\_df.columns **=** df\_columns

roc\_df **=** pd.concat([roc\_df, temp\_df], axis**=**0)

​

temp\_df **=** pd.DataFrame(prn\_list).transpose()

temp\_df.columns **=** df\_columns

prn\_df **=** pd.concat([prn\_df, temp\_df], axis**=**0)