COMP5212 Machine Learning Project 1

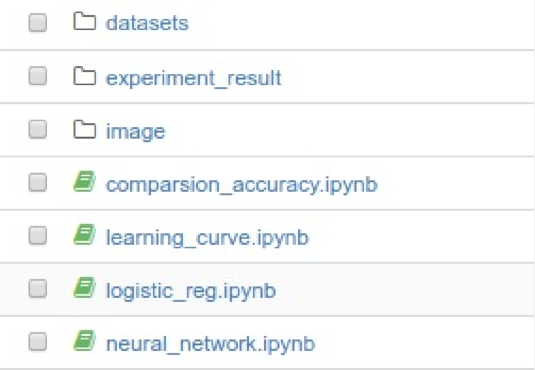
Requirement

1. Anaconda
2. jupyter notebook

> http://jupyter.readthedocs.io/en/latest/install.html

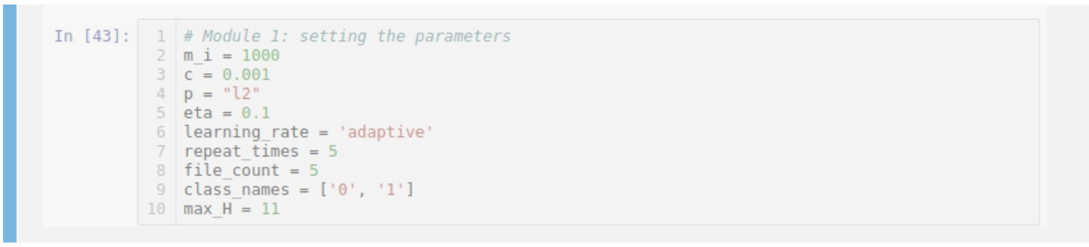
How to use

1. File structure



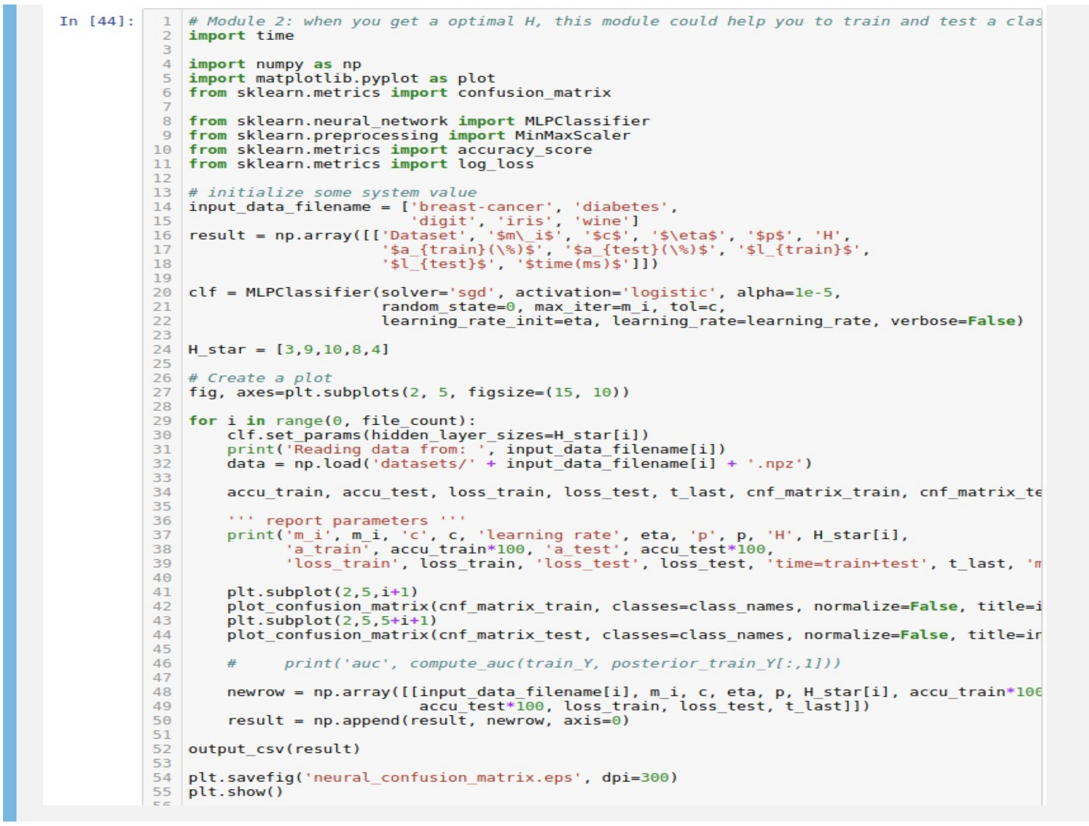
□ datasets
□ experiment_result
□ image
□ comparsion_accuracy.ipynb
□ learning_curve.ipynb
□ logistic_reg.ipynb
□ neural_network.ipynb

2. There are three main modules for this projects: logistic_reg.ipynb, neural network.ipynb, svm.ipynb, while comparsion_accuracy.ipynb is for drawing comparsion curves between the four models. All operations like building classificer, training and testing are wrote in the three main files with same code structure.
    - neural_network.ipynb
        1. Module 1: Setting the parameters



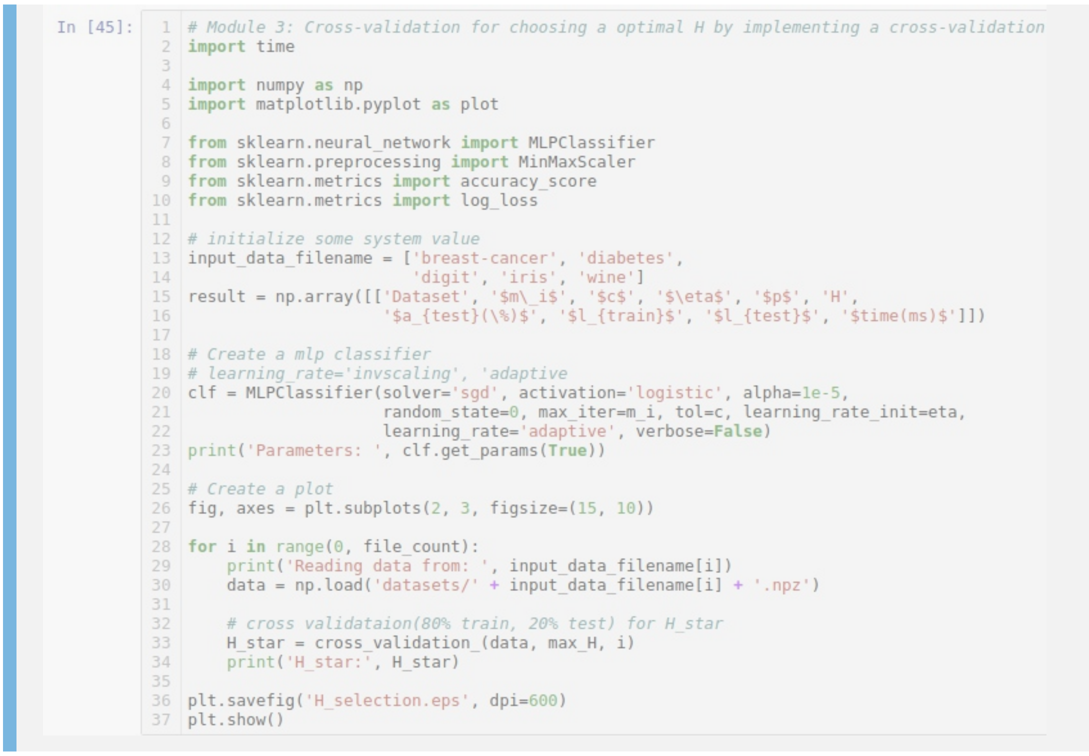
```
In [43]:
# Module 1: setting the parameters
m_i = 1000
c = 0.001
p = "l2"
eta = 0.1
learning_rate = 'adaptive'
repeat_times = 5
file_count = 5
class_names = ['0', '1']
max_H = 11
```

2. Module 2: When you get a optimal H, this module could help you to **train and test a classifier**.



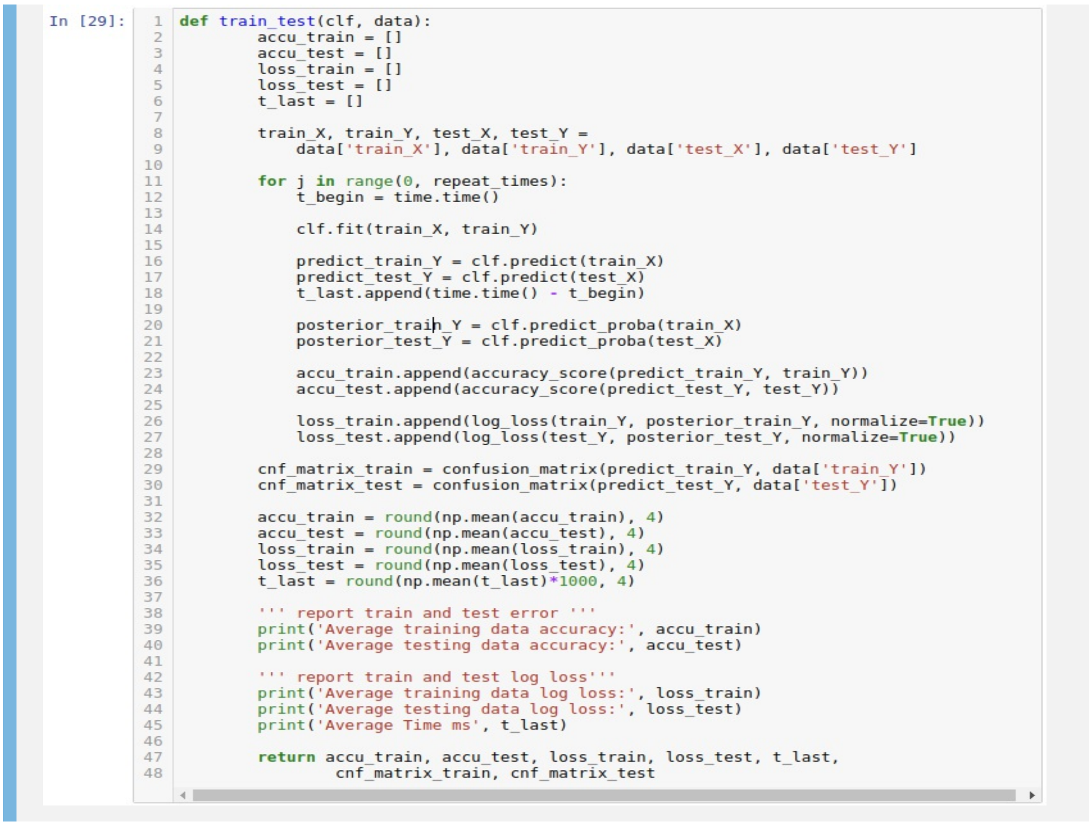
```
In [44]:
# Module 2: when you get a optimal H, this module could help you to train and test a clas
import time

import numpy as np
import matplotlib.pyplot as plot
from sklearn.metrics import confusion_matrix

from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss

# initialize some system value
input_data_filename = ['breast-cancer', 'diabetes',
                       'digit', 'iris', 'wine']
result = np.array([['Dataset', '$m\_i$', '$c$', '$\eta$', '$p$', 'H',
                    '$a_{train}(\%)$', '$a_{test}(\%)$', '$l_{train}$',
                    '$l_{test}$', '$time(ms)$']])

clf = MLPClassifier(solver='sgd', activation='logistic', alpha=1e-5,
                    random_state=0, max_iter=m_i, tol=c,
                    learning_rate_init=eta, learning_rate=learning_rate, verbose=False)

H_star = [3,9,10,8,4]

# Create a plot
fig, axes=plt.subplots(2, 5, figsize=(15, 10))

for i in range(0, file_count):
    clf.set_params(hidden_layer_sizes=H_star[i])
    print('Reading data from: ', input_data_filename[i])
    data = np.load('datasets/' + input_data_filename[i] + '.npz')

    accu_train, accu_test, loss_train, loss_test, t_last, cnf_matrix_train, cnf_matrix_te

    ''' report parameters '''
    print('m_i', m_i, 'c', c, 'learning rate', eta, 'p', p, 'H', H_star[i],
          'a_train', accu_train*100, 'a_test', accu_test*100,
          'loss_train', loss_train, 'loss_test', loss_test, 'time=train+test', t_last, 'm

    plt.subplot(2,5,i+1)
    plot_confusion_matrix(cnf_matrix_train, classes=class_names, normalize=False, title=i
    plt.subplot(2,5,5+i+1)
    plot_confusion_matrix(cnf_matrix_test, classes=class_names, normalize=False, title=ir

    #     print('auc', compute_auc(train_Y, posterior_train_Y[:,1]))

    newrow = np.array([[input_data_filename[i], m_i, c, eta, p, H_star[i], accu_train*100
                        accu_test*100, loss_train, loss_test, t_last]])
    result = np.append(result, newrow, axis=0)

output_csv(result)

plt.savefig('neural_confusion_matrix.eps', dpi=300)
plt.show()
```

3. Module 3: **Cross-validation for choosing a optimal H** by implementing a cross-validation



```
# Module 3: Cross-validation for choosing a optimal H by implementing a cross-validation
import time

import numpy as np
import matplotlib.pyplot as plot

from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss

# initialize some system value
input_data_filename = ['breast-cancer', 'diabetes',
                       'digit', 'iris', 'wine']
result = np.array([['Dataset', '$m\_i$', '$c$', '$\eta$', '$p$', 'H',
                    '$a_{test}(\%)$', '$l_{train}$', '$l_{test}$', '$time(ms)$']])

# Create a mlp classifier
# learning_rate='invscaling', 'adaptive
clf = MLPClassifier(solver='sgd', activation='logistic', alpha=1e-5,
                    random_state=0, max_iter=m_i, tol=c, learning_rate_init=eta,
                    learning_rate='adaptive', verbose=False)
print('Parameters: ', clf.get_params(True))

# Create a plot
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

for i in range(0, file_count):
    print('Reading data from: ', input_data_filename[i])
    data = np.load('datasets/' + input_data_filename[i] + '.npz')

    # cross validataion(80% train, 20% test) for H_star
    H_star = cross_validation_(data, max_H, i)
    print('H_star:', H_star)

plt.savefig('H_selection.eps', dpi=600)
plt.show()
```

4. Other modules: Defining some functions



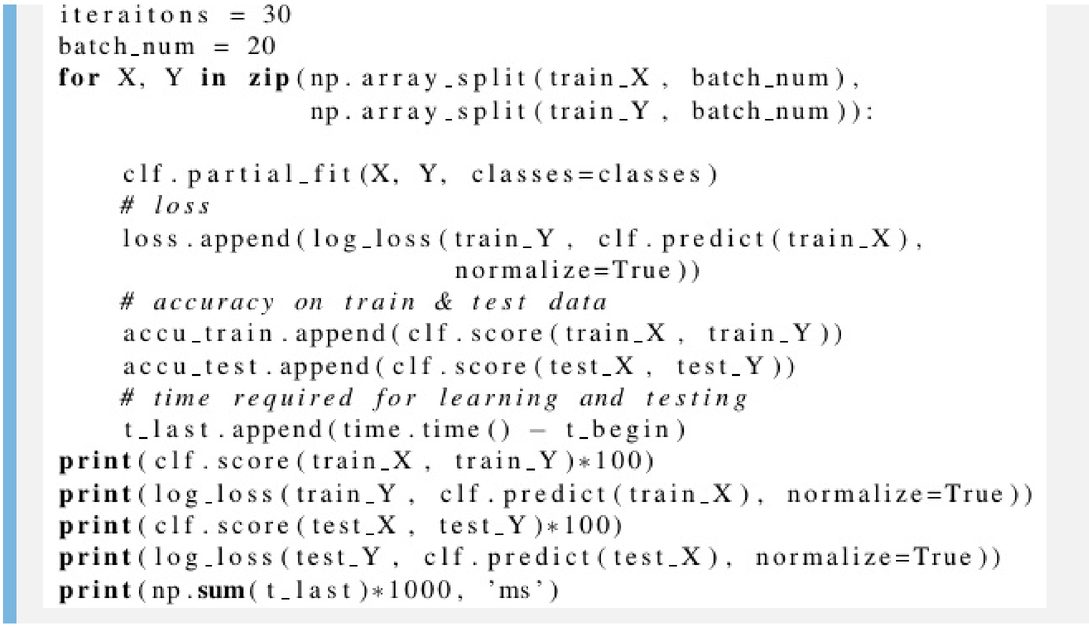
```
def train_test(clf, data):
        accu_train = []
        accu_test = []
        loss_train = []
        loss_test = []
        t_last = []

        train_X, train_Y, test_X, test_Y =
            data['train_X'], data['train_Y'], data['test_X'], data['test_Y']

        for j in range(0, repeat_times):
            t_begin = time.time()

            clf.fit(train_X, train_Y)

            predict_train_Y = clf.predict(train_X)
            predict_test_Y = clf.predict(test_X)
            t_last.append(time.time() - t_begin)

            posterior_train_Y = clf.predict_proba(train_X)
            posterior_test_Y = clf.predict_proba(test_X)

            accu_train.append(accuracy_score(predict_train_Y, train_Y))
            accu_test.append(accuracy_score(predict_test_Y, test_Y))

            loss_train.append(log_loss(train_Y, posterior_train_Y, normalize=True))
            loss_test.append(log_loss(test_Y, posterior_test_Y, normalize=True))

        cnf_matrix_train = confusion_matrix(predict_train_Y, data['train_Y'])
        cnf_matrix_test = confusion_matrix(predict_test_Y, data['test_Y'])

        accu_train = round(np.mean(accu_train), 4)
        accu_test = round(np.mean(accu_test), 4)
        loss_train = round(np.mean(loss_train), 4)
        loss_test = round(np.mean(loss_test), 4)
        t_last = round(np.mean(t_last)*1000, 4)

        ''' report train and test error '''
        print('Average training data accuracy:', accu_train)
        print('Average testing data accuracy:', accu_test)

        ''' report train and test log loss'''
        print('Average training data log loss:', loss_train)
        print('Average testing data log loss:', loss_test)
        print('Average Time ms', t_last)

        return accu_train, accu_test, loss_train, loss_test, t_last,
                cnf_matrix_train, cnf_matrix_test
```

- svm.ipynb
  1. Module 1: Setting the parameters
  2. Module 2: When you get a optimal gamma, this module could help you to **train and test a classifier**.
  3. Module 3: **Cross-validation for choosing a**

optimal gamma by implementing a cross-validation

    4. Other modules: Defining some functions.

  - logistic_reg.ipynb
    1. Module 1: Setting the parameters
    2. Module 2: This module could help you to **train and test a classifier**
    3. Module 3: measuring model's performance overtimes



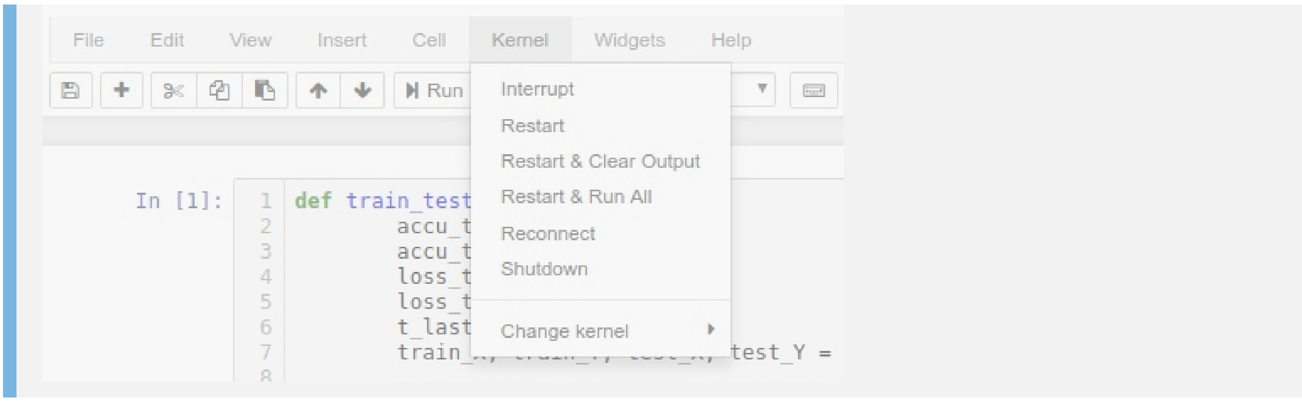
```
iteraitons = 30
batch_num = 20
for X, Y in zip(np.array_split(train_X , batch_num),
                np.array_split(train_Y , batch_num)):

    clf.partial_fit(X, Y, classes=classes)
    # loss
    loss.append(log_loss(train_Y , clf.predict(train_X),
                         normalize=True))
    # accuracy on train & test data
    accu_train.append(clf.score(train_X , train_Y))
    accu_test.append(clf.score(test_X , test_Y))
    # time required for learning and testing
    t_last.append(time.time() − t_begin)
print(clf.score(train_X , train_Y)*100)
print(log_loss(train_Y , clf.predict(train_X), normalize=True))
print(clf.score(test_X , test_Y)*100)
print(log_loss(test_Y , clf.predict(test_X), normalize=True))
print(np.sum(t_last)*1000, 'ms')
```

    4. Other modules: Defining some functions

  - Notes
    1. Different from neural_network.ipynb and svm.ipynb, logistic_reg.ipynb does not have the **cross-validation module**, but have a **performance measurement module**.

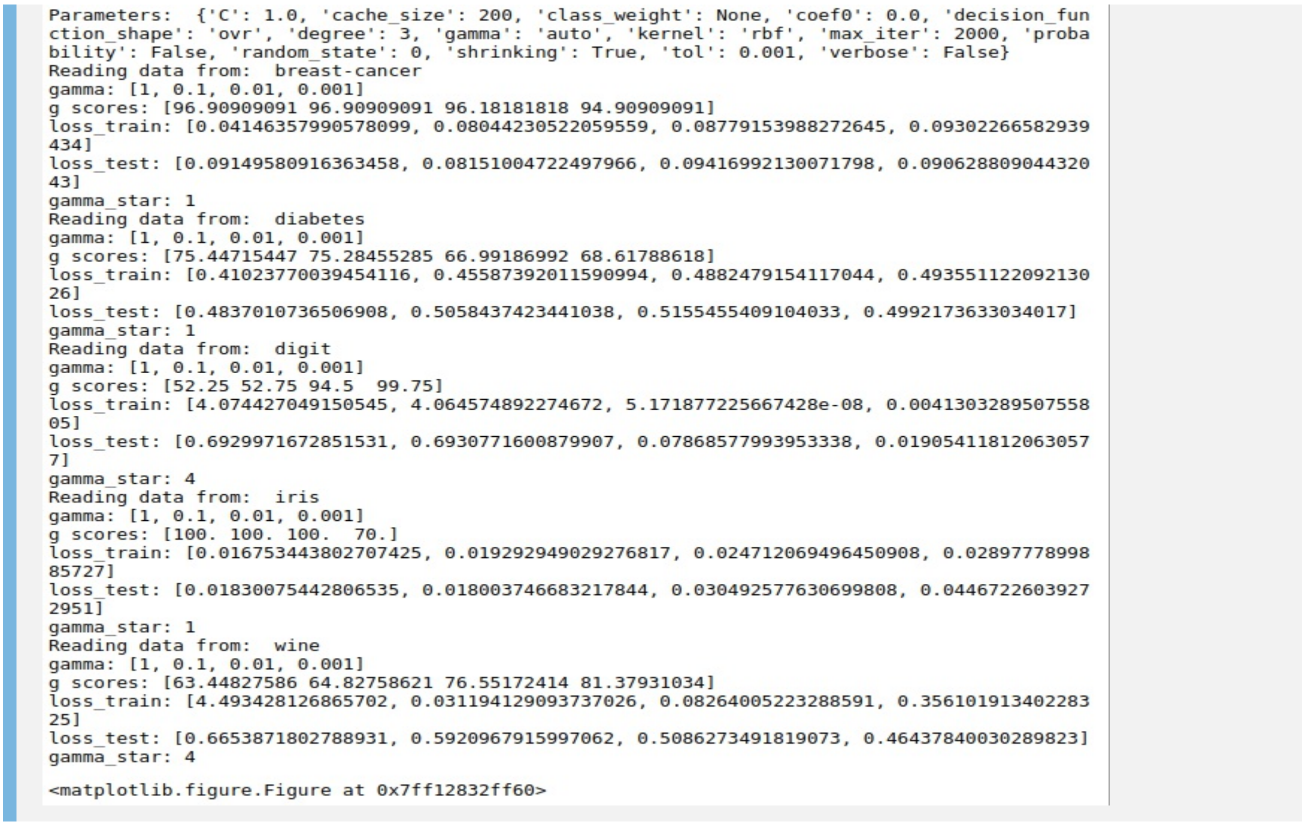
Resulting

1. If you need to verify my result, **please choose any ipynb file, and select Kernel -> Restart & Run All, all the modules in this file will run automatically in order.** You can check my output of the module 2 and module 3.



```
In [1]:  def train_test
                 accu_t
                 accu_t
                 loss_t
                 loss_t
                 t_last
                 train_x, train_Y, test_X, test_Y =
```

2. When you run the code successfully, you could see the result directly like this:



```
Parameters:  {'C': 1.0, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_fun
ction_shape': 'ovr', 'degree': 3, 'gamma': 'auto', 'kernel': 'rbf', 'max_iter': 2000, 'proba
bility': False, 'random_state': 0, 'shrinking': True, 'tol': 0.001, 'verbose': False}
Reading data from:  breast-cancer
gamma: [1, 0.1, 0.01, 0.001]
g scores: [96.90909091 96.90909091 96.18181818 94.90909091]
loss_train: [0.04146357990578099, 0.08044230522059559, 0.08779153988272645, 0.09302266582939
434]
loss_test: [0.09149580916363458, 0.08151004722497966, 0.09416992130071798, 0.090628809044320
43]
gamma_star: 1
Reading data from:  diabetes
gamma: [1, 0.1, 0.01, 0.001]
g scores: [75.44715447 75.28455285 66.99186992 68.61788618]
loss_train: [0.41023770039454116, 0.45587392011590994, 0.4882479154117044, 0.49355112209213
026]
loss_test: [0.4837010736506908, 0.5058437423441038, 0.5155455409104033, 0.4992173633034017]
gamma_star: 1
Reading data from:  digit
gamma: [1, 0.1, 0.01, 0.001]
g scores: [52.25 52.75 94.5  99.75]
loss_train: [4.074427049150545, 4.064574892274672, 5.171877225667428e-08, 0.004130328950755
805]
loss_test: [0.6929971672851531, 0.6930771600879907, 0.07868577993953338, 0.01905411812063057
7]
gamma_star: 4
Reading data from:  iris
gamma: [1, 0.1, 0.01, 0.001]
g scores: [100. 100. 100.  70.]
loss_train: [0.016753443802707425, 0.019292949029276817, 0.024712069496450908, 0.02897778998
85727]
loss_test: [0.01830075442806535, 0.018003746683217844, 0.030492577630699808, 0.0446722603927
2951]
gamma_star: 1
Reading data from:  wine
gamma: [1, 0.1, 0.01, 0.001]
g scores: [63.44827586 64.82758621 76.55172414 81.37931034]
loss_train: [4.493428126865702, 0.031194129093737026, 0.08264005223288591, 0.356101913402283
25]
loss_test: [0.6653871802788931, 0.5920967915997062, 0.5086273491819073, 0.4643784003028982 3]
gamma_star: 4

<matplotlib.figure.Figure at 0x7ff12832ff60>
```

Contact

1. If you do not about jupyter, or meet some problems about running it, please feel free to contact me: jjiao@ust.hk.