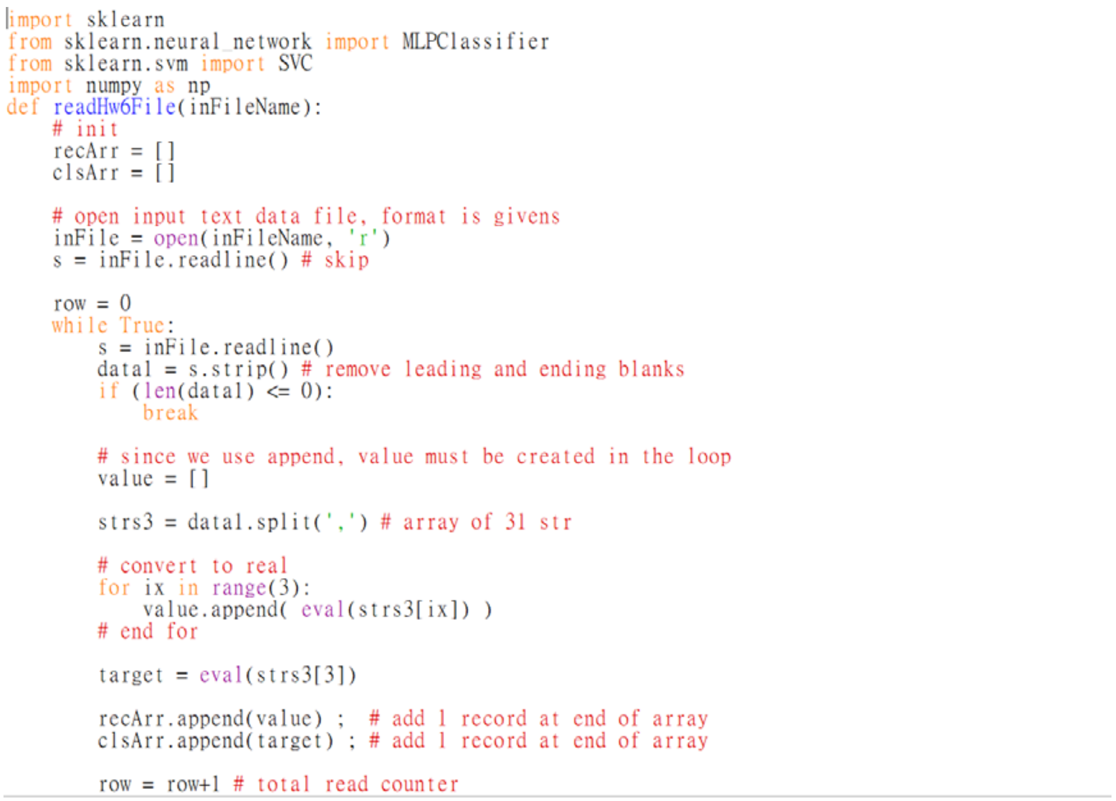
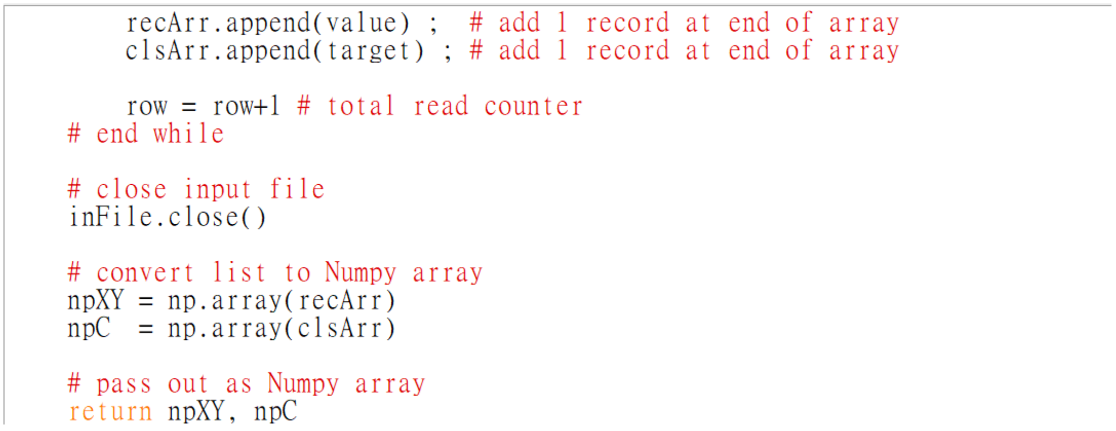
Code screenshots

(1)Use code provided by professor to read in data

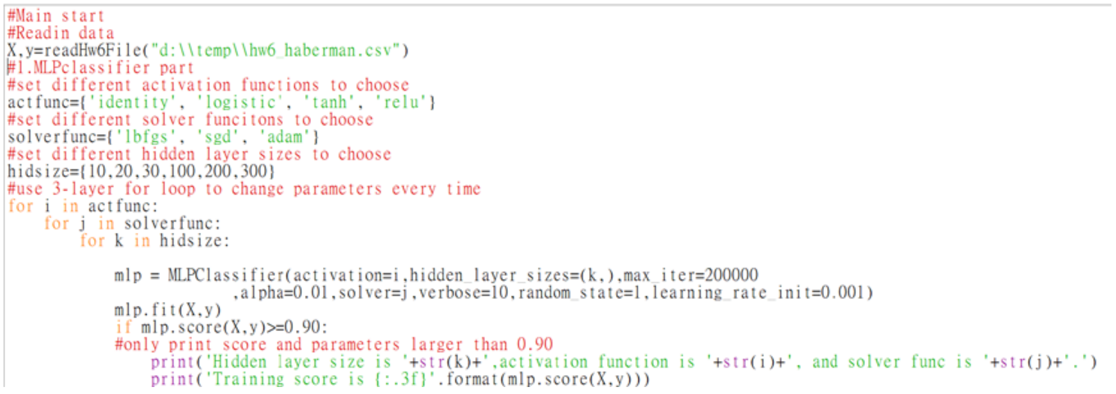


```
import sklearn
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
import numpy as np
def readHw6File(inFileName):
    # init
    recArr = []
    clsArr = []

    # open input text data file, format is givens
    inFile = open(inFileName, 'r')
    s = inFile.readline() # skip

    row = 0
    while True:
        s = inFile.readline()
        data1 = s.strip() # remove leading and ending blanks
        if (len(data1) <= 0):
            break

        # since we use append, value must be created in the loop
        value = []

        strs3 = data1.split(',') # array of 31 str

        # convert to real
        for ix in range(3):
            value.append( eval(strs3[ix]) )
        # end for

        target = eval(strs3[3])

        recArr.append(value) ;  # add 1 record at end of array
        clsArr.append(target) ; # add 1 record at end of array

        row = row+1 # total read counter
```



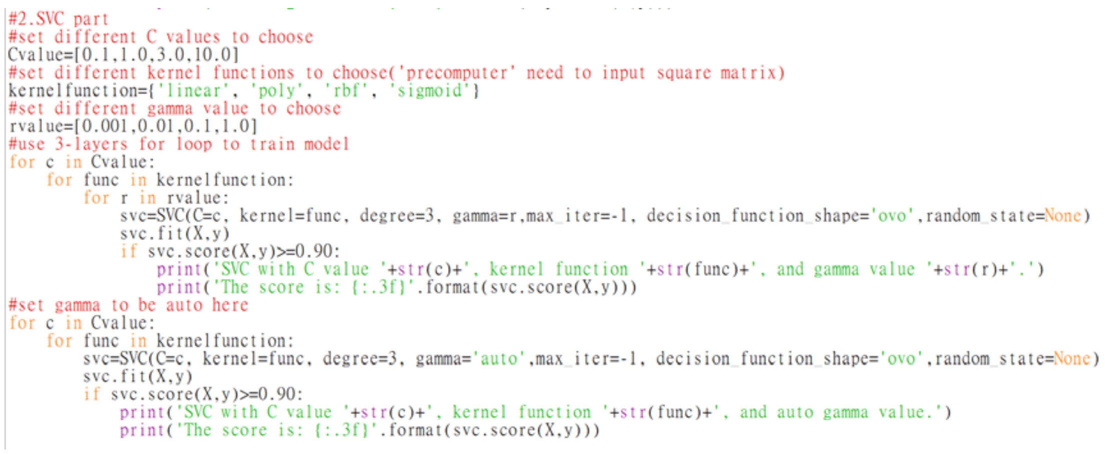
```
        recArr.append(value) ;  # add 1 record at end of array
        clsArr.append(target) ; # add 1 record at end of array

        row = row+1 # total read counter
    # end while

    # close input file
    inFile.close()

    # convert list to Numpy array
    npXY = np.array(recArr)
    npC  = np.array(clsArr)

    # pass out as Numpy array
    return npXY, npC
```

(2)MLP part



```
#Main start
#Readin data
X,y=readHw6File("d:\\temp\\hw6_haberman.csv")
#1.MLPclassifier part
#set different activation functions to choose
actfunc={'identity', 'logistic', 'tanh', 'relu'}
#set different solver funcitons to choose
solverfunc={'lbfgs', 'sgd', 'adam'}
#set different hidden layer sizes to choose
hidsize={10,20,30,100,200,300}
#use 3-layer for loop to change parameters every time
for i in actfunc:
    for j in solverfunc:
        for k in hidsize:

            mlp = MLPClassifier(activation=i,hidden_layer_sizes=(k,),max_iter=200000
                        ,alpha=0.01,solver=j,verbose=10,random_state=1,learning_rate_init=0.001)
            mlp.fit(X,y)
            if mlp.score(X,y)>=0.90:
            #only print score and parameters larger than 0.90
                print('Hidden layer size is '+str(k)+',activation function is '+str(i)+', and solver func is '+str(j)+'.')
                print('Training score is {:.3f}'.format(mlp.score(X,y)))
```

(2)SVC part

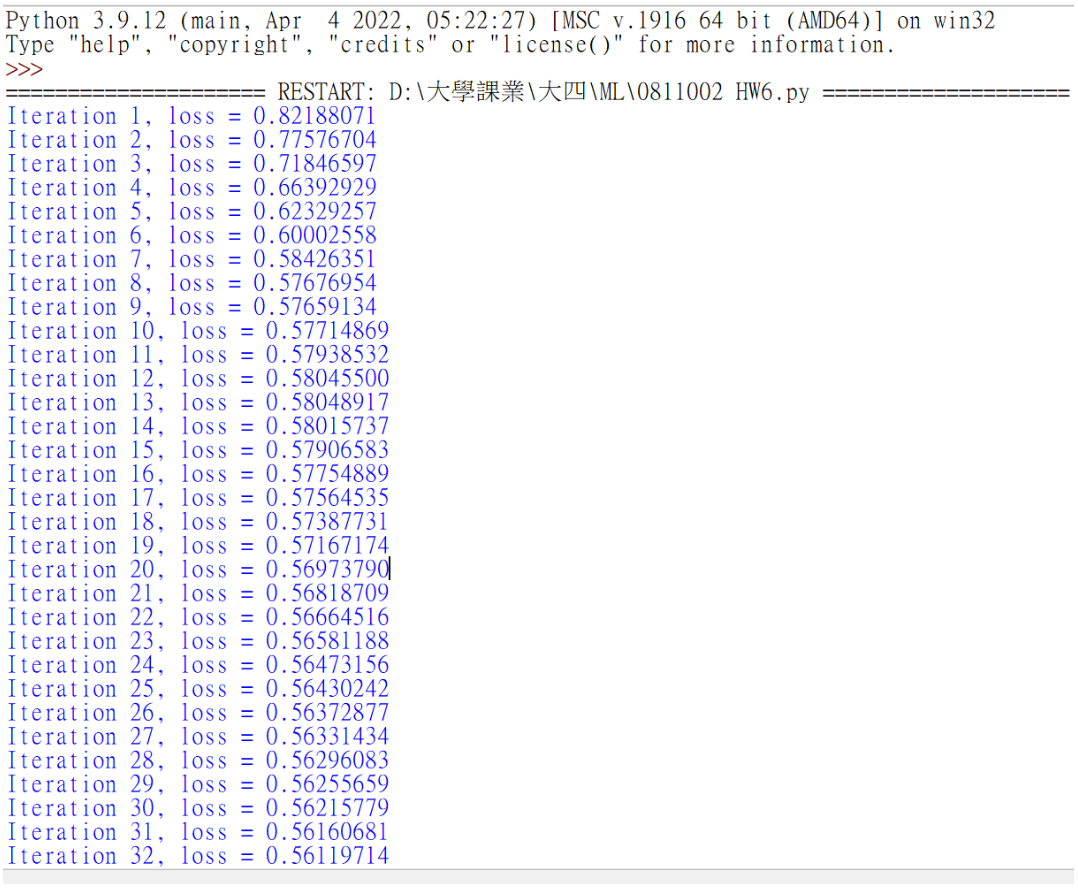


```
#2.SVC part
#set different C values to choose
Cvalue=[0.1,1.0,3.0,10.0]
#set different kernel functions to choose('precomputer' need to input square matrix)
kernelfunction={'linear', 'poly', 'rbf', 'sigmoid'}
#set different gamma value to choose
rvalue=[0.001,0.01,0.1,1.0]
#use 3-layers for loop to train model
for c in Cvalue:
    for func in kernelfunction:
        for r in rvalue:
            svc=SVC(C=c, kernel=func, degree=3, gamma=r,max_iter=-1, decision_function_shape='ovo',random_state=None)
            svc.fit(X,y)
            if svc.score(X,y)>=0.90:
                print('SVC with C value '+str(c)+', kernel function '+str(func)+', and gamma value '+str(r)+'.')
                print('The score is: {:.3f}'.format(svc.score(X,y)))
#set gamma to be auto here
for c in Cvalue:
    for func in kernelfunction:
        svc=SVC(C=c, kernel=func, degree=3, gamma='auto',max_iter=-1, decision_function_shape='ovo',random_state=None)
        svc.fit(X,y)
        if svc.score(X,y)>=0.90:
            print('SVC with C value '+str(c)+', kernel function '+str(func)+', and auto gamma value.')
            print('The score is: {:.3f}'.format(svc.score(X,y)))
```

Compile Result:

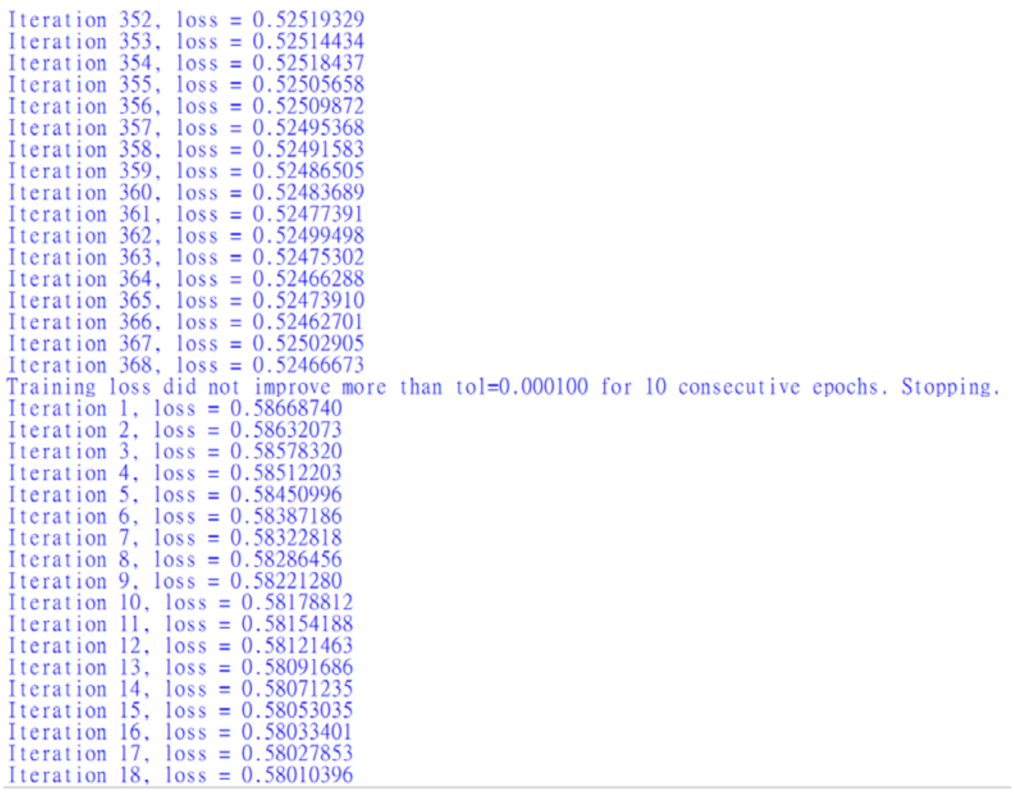
Part 1:MLP classifier

(1)MLP classifier doing iteration



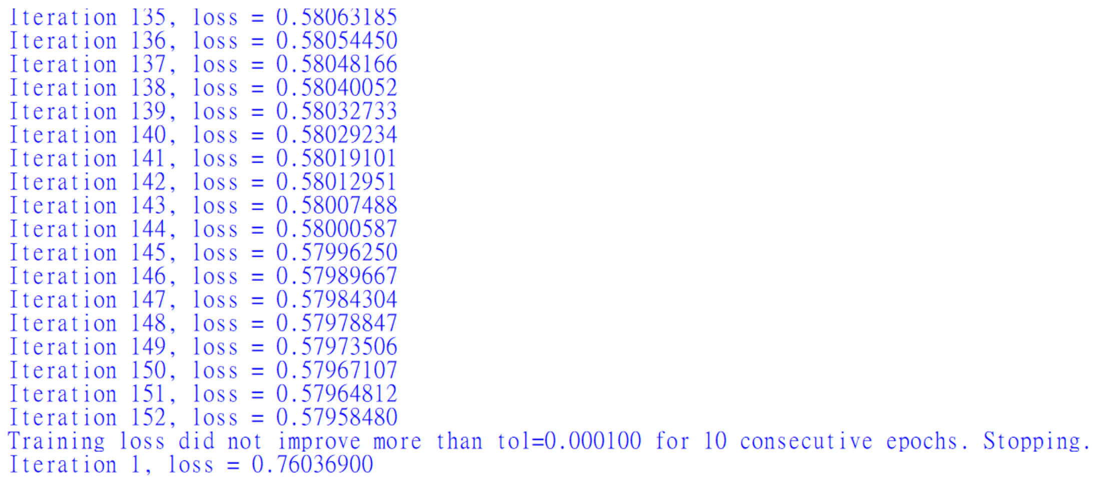
```
Python 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==================== RESTART: D:\大學課業\大四\ML\0811002 HW6.py ====================
Iteration 1, loss = 0.82188071
Iteration 2, loss = 0.77576704
Iteration 3, loss = 0.71846597
Iteration 4, loss = 0.66392929
Iteration 5, loss = 0.62329257
Iteration 6, loss = 0.60002558
Iteration 7, loss = 0.58426351
Iteration 8, loss = 0.57676954
Iteration 9, loss = 0.57659134
Iteration 10, loss = 0.57714869
Iteration 11, loss = 0.57938532
Iteration 12, loss = 0.58045500
Iteration 13, loss = 0.58048917
Iteration 14, loss = 0.58015737
Iteration 15, loss = 0.57906583
Iteration 16, loss = 0.57754889
Iteration 17, loss = 0.57564535
Iteration 18, loss = 0.57387731
Iteration 19, loss = 0.57167174
Iteration 20, loss = 0.56973790
Iteration 21, loss = 0.56818709
Iteration 22, loss = 0.56664516
Iteration 23, loss = 0.56581188
Iteration 24, loss = 0.56473156
Iteration 25, loss = 0.56430242
Iteration 26, loss = 0.56372877
Iteration 27, loss = 0.56331434
Iteration 28, loss = 0.56296083
Iteration 29, loss = 0.56255659
Iteration 30, loss = 0.56215779
Iteration 31, loss = 0.56160681
Iteration 32, loss = 0.56119714
```

(2) Iteration result converged, but training score less than 0.90, so nothing printed.



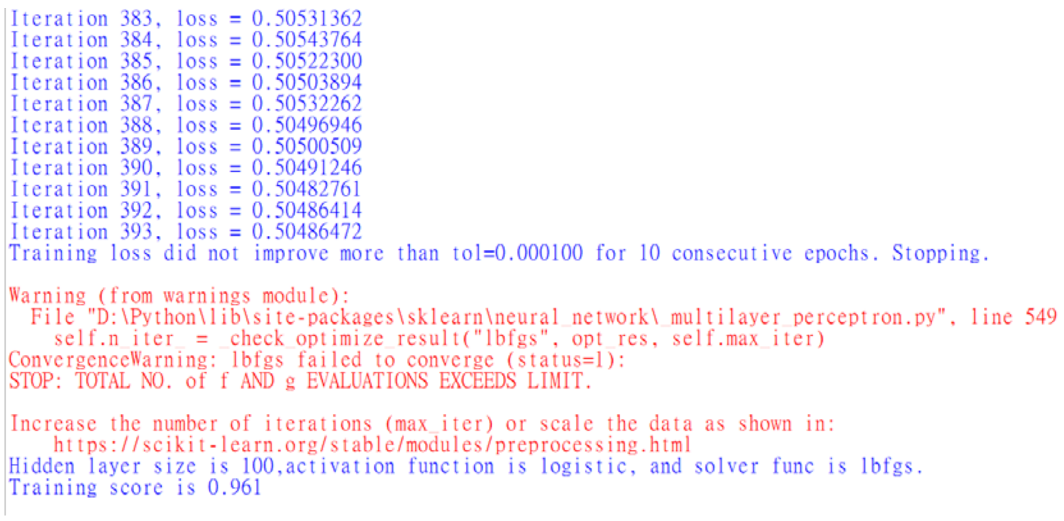
```
Iteration 352, loss = 0.52519329
Iteration 353, loss = 0.52514434
Iteration 354, loss = 0.52518437
Iteration 355, loss = 0.52505658
Iteration 356, loss = 0.52509872
Iteration 357, loss = 0.52495368
Iteration 358, loss = 0.52491583
Iteration 359, loss = 0.52486505
Iteration 360, loss = 0.52483689
Iteration 361, loss = 0.52477391
Iteration 362, loss = 0.52499498
Iteration 363, loss = 0.52475302
Iteration 364, loss = 0.52466288
Iteration 365, loss = 0.52473910
Iteration 366, loss = 0.52462701
Iteration 367, loss = 0.52502905
Iteration 368, loss = 0.52466673
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
Iteration 1, loss = 0.58668740
Iteration 2, loss = 0.58632073
Iteration 3, loss = 0.58578320
Iteration 4, loss = 0.58512203
Iteration 5, loss = 0.58450996
Iteration 6, loss = 0.58387186
Iteration 7, loss = 0.58322818
Iteration 8, loss = 0.58286456
Iteration 9, loss = 0.58221280
Iteration 10, loss = 0.58178812
Iteration 11, loss = 0.58154188
Iteration 12, loss = 0.58121463
Iteration 13, loss = 0.58091686
Iteration 14, loss = 0.58071235
Iteration 15, loss = 0.58053035
Iteration 16, loss = 0.58033401
Iteration 17, loss = 0.58027853
Iteration 18, loss = 0.58010396
```

(3)Trying few models



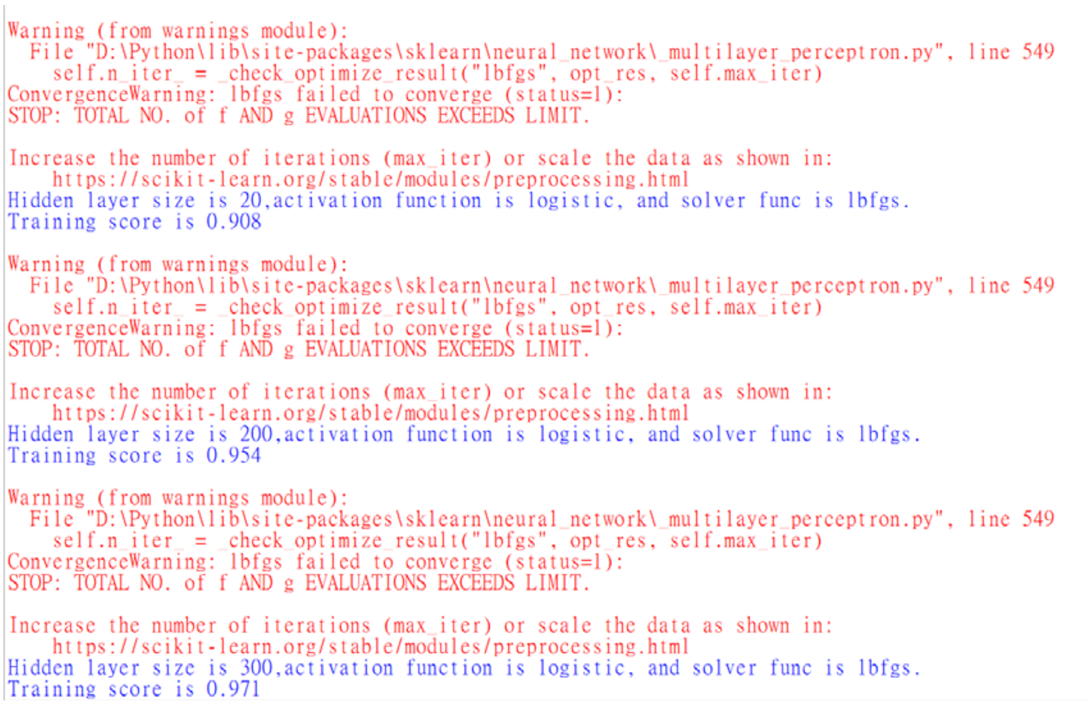
```
Iteration 135, loss = 0.58063185
Iteration 136, loss = 0.58054450
Iteration 137, loss = 0.58048166
Iteration 138, loss = 0.58040052
Iteration 139, loss = 0.58032733
Iteration 140, loss = 0.58029234
Iteration 141, loss = 0.58019101
Iteration 142, loss = 0.58012951
Iteration 143, loss = 0.58007488
Iteration 144, loss = 0.58000587
Iteration 145, loss = 0.57996250
Iteration 146, loss = 0.57989667
Iteration 147, loss = 0.57984304
Iteration 148, loss = 0.57978847
Iteration 149, loss = 0.57973506
Iteration 150, loss = 0.57967107
Iteration 151, loss = 0.57964812
Iteration 152, loss = 0.57958480
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
Iteration 1, loss = 0.76036900
```

(4)Set max iteration number up to 200,000, but still not converge. Even though, the model I have now is good enough (training score larger than 0.90).(這部分有問過老師，使用 lbfgs 不容易收斂，而 MLP 這時會使用迭代到當下的模型來做 training score，沒迭代完仍有生成一個模型，而且分數夠高)

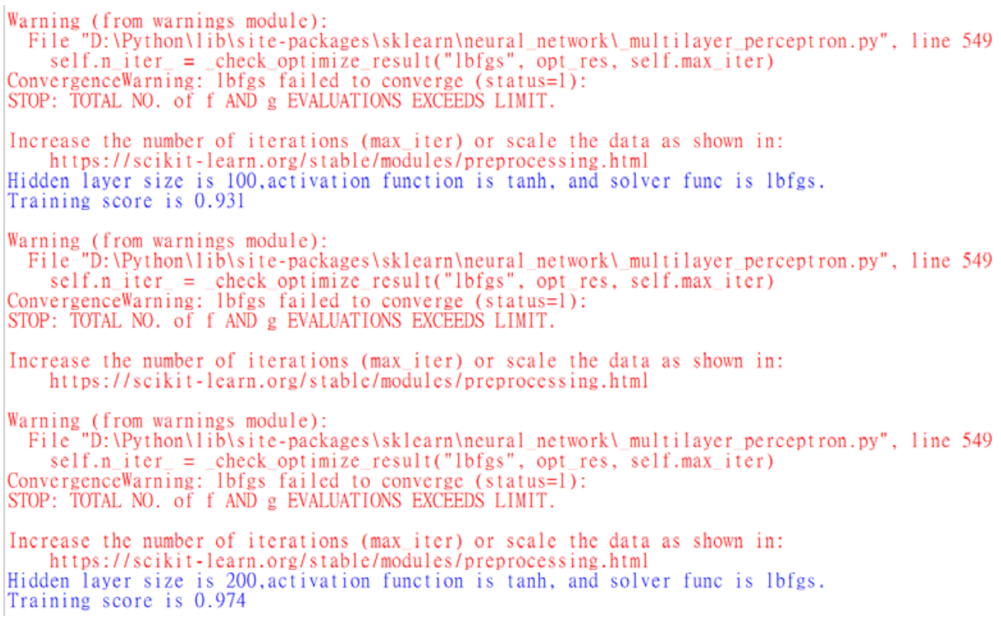


```
Iteration 383, loss = 0.50531362
Iteration 384, loss = 0.50543764
Iteration 385, loss = 0.50522300
Iteration 386, loss = 0.50503894
Iteration 387, loss = 0.50532262
Iteration 388, loss = 0.50496946
Iteration 389, loss = 0.50500509
Iteration 390, loss = 0.50491246
Iteration 391, loss = 0.50482761
Iteration 392, loss = 0.50486414
Iteration 393, loss = 0.50486472
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 100,activation function is logistic, and solver func is lbfgs.
Training score is 0.961
```

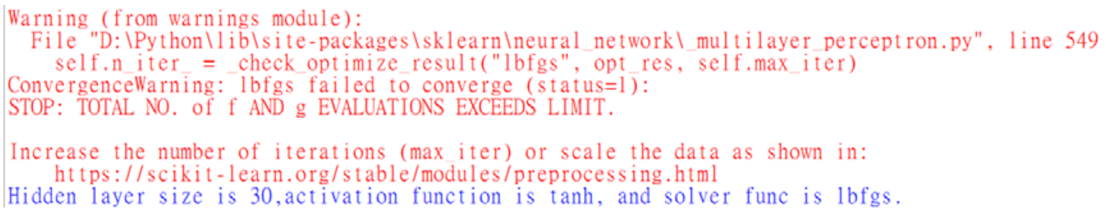
(5)更換參數(hidden layer size, activation)，以 lbfgs 表現最好，但會出現不收斂的問題



```
Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 20,activation function is logistic, and solver func is lbfgs.
Training score is 0.908

Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 200,activation function is logistic, and solver func is lbfgs.
Training score is 0.954

Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 300,activation function is logistic, and solver func is lbfgs.
Training score is 0.971
```

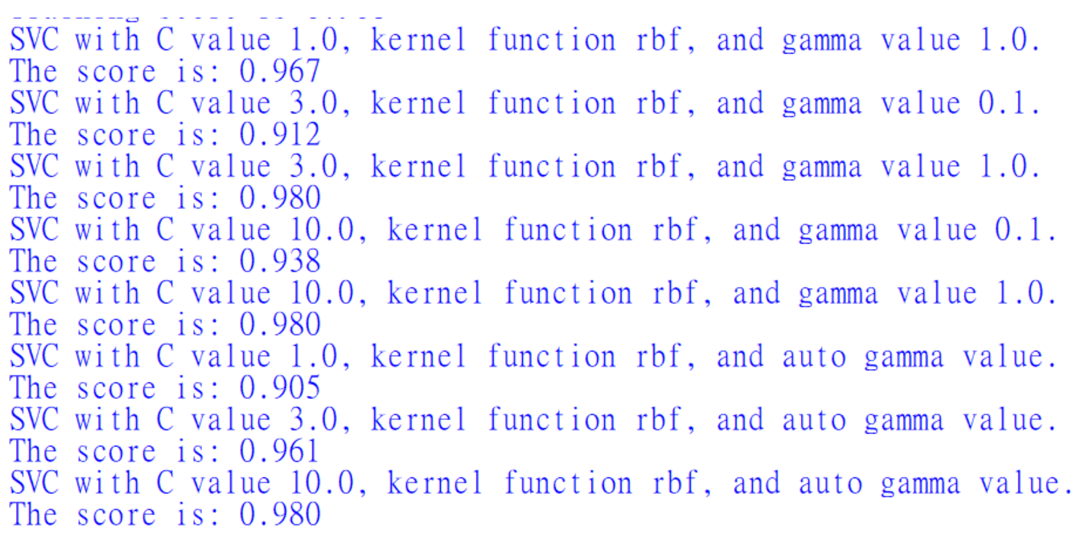


```
Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 100,activation function is tanh, and solver func is lbfgs.
Training score is 0.931

Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html

Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 200,activation function is tanh, and solver func is lbfgs.
Training score is 0.974
```



```
Warning (from warnings module):
  File "D:\Python\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py", line 549
    self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of f AND g EVALUATIONS EXCEEDS LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Hidden layer size is 30,activation function is tanh, and solver func is lbfgs.
```

Part 2: SVC



```
SVC with C value 1.0, kernel function rbf, and gamma value 1.0.
The score is: 0.967
SVC with C value 3.0, kernel function rbf, and gamma value 0.1.
The score is: 0.912
SVC with C value 3.0, kernel function rbf, and gamma value 1.0.
The score is: 0.980
SVC with C value 10.0, kernel function rbf, and gamma value 0.1.
The score is: 0.938
SVC with C value 10.0, kernel function rbf, and gamma value 1.0.
The score is: 0.980
SVC with C value 1.0, kernel function rbf, and auto gamma value.
The score is: 0.905
SVC with C value 3.0, kernel function rbf, and auto gamma value.
The score is: 0.961
SVC with C value 10.0, kernel function rbf, and auto gamma value.
The score is: 0.980
```