

### 1. C-OWA(MATLAB)

```
1. clc
2. clear
3. close all
4. x=[9 8 8 8 9
5.      9 7 8 7 8
6.      8 7 7 7 7
7.      8 8 8 7 8
8.      8 6 7 6 8
9.      9 7 8 6 8
10.     8 7 7 6 8
11.     8 6 7 6 7
12.     ]; %cardiovascular capacity
13.%row heading is the expert number, and the column heading is the
   indicator name
14.x=sort(x,'descend'); %descending order
15.[m,n]=size(x);
16.for i=1:m
17.    c(i)=nchoosek(m-1,i-1); %calculate number of combinations
18.end
19.a=c' ./ (2^(m-1)) ; %weight vector
20.
21.for j=1:n
22.    wj(:,j)=a.*x(:,j); %absolute weight
23.end
24.wj=sum(wj);
25.w=wj./sum(wj) %relative weight
26.%result
27.
28.% x=[8 8 7 8
29.%      8 7 6 7
30.%      9 9 8 9
31.%      8 8 7 8
32.%      8 7 7 8
33.%      9 8 7 8
34.%      8 8 7 8
35.%      8 6 5 7
36.%      ]; %Respiratory exchange capacity
37.% x=[8 9 6 7 8
38.%      9 9 6 6 8
39.%      8 9 6 7 8
40.%      8 8 7 7 7
41.%      9 9 6 8 8
```

```

42.%      8 9 7 7 8
43.%      7 9 6 7 7

%      8 9 7 7 7] %metabolic ability

w =

0.2213    0.1883    0.2017    0.1750    0.2137

fx >>

```

## 2. THE ENTROPY WEIGHT METHOD (MATLAB)

```

1. data=xlsread('D:\Desktop\shangquan.xlsx')
2. data=data(:,3:end) %Only the indicator data is taken
3. data1=data;
4. index=[3,4];
5. for i=1:length(index)
6.     data1(:,index(i))=max(data(:,index(i)))-data(:,index(i));
7. end
8. index=[5];
9. a=90;
10. for i=1:length(index)
11.     data1(:,index(i))=1-abs(data(:,index(i))-a)/max(abs(data(:,inde
        x(i))-a))
12. end
13. data2=mapminmax(data1',0.001,1)
14. data2=data2';
15. [m,n]=size(data2);
16. p=zeros(m,n);
17. for j=1:n
18.     p(:,j)=data2(:,j)/sum(data2(:,j))
19. end
20. for j=1:n
21.     E(j)=-1/log(m)*sum(p(:,j).*log(p(:,j)))
22. end
23.%calculate weight
24. w=(1-E)/sum(1-E)

```

```
w1 =
0.0730 0.0620 0.0660 0.0580 0.0700 0.0820 0.0790 0.0710 0.0810 0.0760 0.0850 0.0590 0.0660 0.0730

w2 =
0.0840 0.0500 0.0820 0.0350 0.0820 0.1140 0.0440 0.0800 0.0940 0.0490 0.1460 0.0440 0.0380 0.0560

w =
0.0780 0.0580 0.0730 0.0480 0.0750 0.0960 0.0640 0.0750 0.0870 0.0640 0.1110 0.0530 0.0540 0.0660
```

### 3. CPETASSESSMENT(PYTHON)

```
1. def CPETAssessment(Data):
2.     import pandas as pd
3.     import numpy as np
4.     import math
5.     from scipy import stats
6.     Parameters = Data.values.tolist()
7.     Parameters = np.array(Parameters)
8.     # data format conversion
9.     # Power=Parameters[:,1].astype('float32')
10.    Power = Parameters[:, 2].astype('float32')
11.    HR = Parameters[:, 3].astype('float32')
12.    SV = Parameters[:, 4].astype('float32')
13.    CO = Parameters[:, 5].astype('float32')
14.    V02 = Parameters[:, 6].astype('float32')
15.    VC02 = Parameters[:, 7].astype('float32')
16.    VE = Parameters[:, 8].astype('float32')
17.    BOA = Parameters[:, 9].astype('float32')
18.
19.    # ventilation equivalent
20.    EQ02 = VE * 1000. / V02
21.    EQ02 = np.array(EQ02) # print(EQ02.shape)
22.    EQC02 = VE * 1000. / VC02
23.    EQC02 = np.array(EQC02)
24.    # oxygen pulse
25.    OP = V02 / HR
26.    # RQ
27.    RQ = VC02 / V02
28.    # Ca_V02
29.    Ca_v02 = V02 / (HR * SV)
30.
31.    # new array
32.    parameters = np.vstack((Parameters[:, 1], Power, HR, SV, CO,
V02, VC02, VE, EQ02, EQC02 , OP, RQ, Ca_v02, BOA))
```

```
33.     parameters = np.transpose(parameters)
34.     np.savetxt("parameters.csv", parameters, delimiter=',', fmt='
    %s')
35.
36.     ##拐点检测 AT 和力竭时间点
37.     # anaerobic threshold
38.     rows = len(Parameters)
39.
40.     for i in range(0, rows - 1):
41.         if (VO2[i] < VCO2[i] and VO2[i + 1] < VCO2[i + 1]):
42.             time_AT = Parameters[i, 1]
43.             print('AT 的时间为%s' % time_AT)
44.             AT = i
45.             break
46.     # exhaustion time point
47.     for i in range(0, rows - 1):
48.         if (Power[i + 1] < Power[i]):
49.             time_MAX = Parameters[i, 1]
50.             print('力竭的时间为%s' % time_MAX)
51.             MAX = i
52.             break
53.
54.     # Physiological characteristic parameters
55.
56.     x1 = HR[AT]
57.     HR_AT=HR[AT]
58.     x1_max = 173
59.     x1_min = 71
60.     # x1=(HR[AT]-x1_min)/(x1_max-x1_min)
61.     x2 = HR[MAX]
62.     x2_max = 198
63.     x2_min = 115
64.     # x2=(HR[MAX]-x2_min)/(x2_max-x2_min)
65.     x3 = SV[AT]
66.     x3_max = 132
67.     x3_min = 8.8
68.     # x3=(SV[AT]-x3_min)/(x3_max-x3_min)
69.     x4 = SV[MAX]
70.     x4_max = 192.1
71.     x4_min = 12.1
72.     # x4=(SV[MAX]-x4_min)/(x4_max-x4_min)
73.     x6 = OP[MAX]
74.     x6_max = 22.6
```

```

75. x6_min = 6.8
76. # x6=(OP[MAX]-x6_min)/(x6_max-x6_min)
77. # Gas exchange function
78. x7 = EQ02[AT]
79. x7_max = 33.1
80. x7_min = 20.2
81. # x7=(x7_max-EQ02[AT])/ (x7_max-x7_min)
82. x8 = EQ02[MAX]
83. x8_max = 72.3
84. x8_min = 22.5
85. # x8=(x8_max-EQ02[MAX])/ (x8_max-x8_min)
86. x9 = VE[AT]
87. x9_max = 82.8
88. x9_min = 18.3
89. # x9=(VE[AT]-x9_min)/ (x9_max-x9_min)
90. x10 = VE[MAX]
91. x10_max = 168.6
92. x10_min = 28.4
93. # x10=(VE[MAX]-x10_min)/ (x10_max-x10_min)
94. # Metabolic functions
95. x11 = V02[AT]
96. x11_max = 39.4
97. x11_min = 14.06
98. # x11=(V02[AT]-x11_min)/ (x11_max-x11_min)
99. x12 = V02[MAX]
100. x12_max = 61.02
101. x12_min = 21.68
102. # x12=(V02[MAX]-x12_min)/ (x12_max-x12_min)
103. x14 = RQ[MAX]
104. x14_max = 1.47
105. x14_min = 0.97
106. # x14=(RQ[MAX]-x14_min)/ (x14_max-x14_min)
107. x15 = Ca_vO2[AT]
108. x15_max = 0.977
109. x15_min = 0.105
110. # x15=(Ca_vO2[AT]-x15_min)/ (x15_max-x15_min)
111. x16 = Ca_vO2[MAX]
112. x16_max = 0.880
113. x16_min = 0.0916
114. # x16=(Ca_vO2[MAX]-x16_min)/ (x16_max-x16_min)
115.
116. # MroScore
117. # Define positive and negative ideal solutions

```

```

118. # Define the initial evaluation matrix
119. matrix = np.zeros((3, 14))
120. matrix[0, :] = [x1, x2, x3, x4, x6, x7, x8, x9, x10, x11, x1
   2, x14, x15, x16]
121. matrix[1, :] = [x1_max, x2_max, x3_max, x4_max, x6_max, x7_m
   ax, x8_max, x9_max, x10_max, x11_max, x12_max, x14_max,
122.           x15_max, x16_max]
123. matrix[2, :] = [x1_min, x2_min, x3_min, x4_min, x6_min, x7_m
   in, x8_min, x9_min, x10_min, x11_min, x12_min, x14_min,
124.           x15_min, x16_min]
125. # integer ideal solution
126. D = np.zeros((2, 14))
127. s1 = 0
128. s2 = 0
129. Weight = [0.078, 0.058, 0.073, 0.048, 0.075, 0.096, 0.064, 0
   .075, 0.087, 0.064, 0.111, 0.053, 0.054, 0.066]
130. for i in range(0, 13):
131.     if (i != 7 and i != 8):
132.         D[0, i] = Weight[i] * ((matrix[0, i] - matrix[1, i])
   / (matrix[1, i] - matrix[2, i]))
133.         s1 = s1 + pow(D[0, i], 2)
134.     else:
135.         D[0, i] = Weight[i] * ((matrix[2, i] - matrix[0, i])
   / (matrix[1, i] - matrix[2, i]))
136.         s1 = s1 + pow(D[0, i], 2)
137. s1 = math.sqrt(s1)
138. for i in range(0, 13):
139.     if (i != 7 and i != 8):
140.         D[1, i] = Weight[i] * ((matrix[0, i] - matrix[2, i])
   / (matrix[1, i] - matrix[2, i]))
141.         s2 = s2 + pow(D[1, i], 2)
142.     else:
143.         D[1, i] = Weight[i] * ((matrix[1, i] - matrix[0, i])
   / (matrix[1, i] - matrix[2, i]))
144.         s2 = s2 + pow(D[1, i], 2)
145. s2 = math.sqrt(s2)
146. MroScore = 70.855 * s2 / (s1 + s2) + 46.615
147. MroScore = round(MroScore, 1)
148. print('心肺耐力综合得分为%10.1f' % MroScore)
149.
150.
151. #cardiovascular capacity Score1
152. Weight1 = [0.235, 0.175, 0.220, 0.144, 0.226]

```

```
153.     D = np.zeros((2, 5))
154.     s1 = 0
155.     s2 = 0
156.     for i in range(0, 4):
157.         D[0, i] = Weight[i] * ((matrix[0, i] - matrix[1, i]) / (
158.             matrix[1, i] - matrix[2, i]))
159.         s1 = s1 + pow(D[0, i], 2)
160.     s1 = math.sqrt(s1)
161.     for i in range(0, 5):
162.         D[1, i] = Weight[i] * ((matrix[0, i] - matrix[2, i]) / (
163.             matrix[1, i] - matrix[2, i]))
164.         s2 = s2 + pow(D[1, i], 2)
165.     s2 = math.sqrt(s2)
166.     Score1 = 70.855 * s2 / (s1 + s2) + 40.615
167.     print(s2 / (s1 + s2) )
168.     Score1 = round(Score1, 1)
169.     print('心血管能力得分为%10.1f' % Score1)
170. 
171. # 呼吸能力 Score2
172. Weight1 = [0.298, 0.198, 0.223, 0.270]
173. D = np.zeros((2, 4))
174. s1 = 0
175. s2 = 0
176. for i in range(0, 3):
177.     D[0, i] = Weight[i] * ((matrix[0, i + 5] - matrix[1, i +
178.         5]) / (matrix[1, i + 5] - matrix[2, i + 5]))
179.     s1 = s1 + pow(D[0, i], 2)
180.     s1 = math.sqrt(s1)
181.     for i in range(0, 4):
182.         D[1, i] = Weight[i] * ((matrix[0, i + 5] - matrix[2, i +
183.             5]) / (matrix[1, i + 5] - matrix[2, i + 5]))
184.         s2 = s2 + pow(D[1, i], 2)
185.     s2 = math.sqrt(s2)
186.     Score2 = 69.855 * s2 / (s1 + s2) + 48.615
187.     Score2 = round(Score2, 1)
188.     print('呼吸能力得分为%10.1f' % Score2)
189. 
190. # metabolic capacity Score3
191. Weight1 = [0.184, 0.319, 0.152, 0.155, 0.190]
192. D = np.zeros((2, 5))
```

```

9]) / (matrix[1, i + 9] - matrix[2, i + 9]))
193.         s1 = s1 + pow(D[0, i], 2)
194.         s1 = math.sqrt(s1)
195.         for i in range(0, 5):
196.             D[1, i] = Weight[i] * ((matrix[0, i + 9] - matrix[2, i + 9])
9]) / (matrix[1, i + 9] - matrix[2, i + 9]))
197.             s2 = s2 + pow(D[1, i], 2)
198.             s2 = math.sqrt(s2)
199.             Score3 = 70.855 * s2 / (s1 + s2) + 44.615
200.             Score3 = round(Score3, 1)
201.             print('代谢能力得分为%10.1f' % Score3)
202.             result=[time_AT,time_MAX,MroScore,Score1,Score2,Score3,HR_AT
]
203.         return result

1.
    import pandas as pd
2. import CPETAssessment
3. Data = pd.read_csv(r'D:\Users\LiQi\Desktop\data_lq.csv')
4. result=CPETAssessment.CPETAssessment(Data)
5. print(result)

```

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and a specific tab for 'pythonProject\_bylw - CPETAssessment.py'. The left sidebar shows the 'Project' view with a tree structure of files and folders. The main code editor window contains the Python script 'CPETAssessment.py'. The 'Run' tab at the bottom shows the command line output of the script's execution.

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject_bylw - CPETAssessment.py
pythonProject_bylw venv
Project Project
pythonProject_bylw C:\Users\LiQi\PycharmProjects\pythonProject_bylw
  CPET-SDK
    __init__.py
    README.md
    subfunction
      CPETAssessment.py
      hhhh.py
      naoyang.py
      parameters.csv
      test.py
      xgPET.py
    venv
    main.py
    text.py
External Libraries
Scratches and Consoles

import pandas as pd
import numpy as np
import math
from scipy import stats
Parameters = Data.values.tolist()
Parameters = np.array(Parameters)
# 数据格式转换
Power = Parameters[:, 1].astype('float32')
HR = Parameters[:, 3].astype('float32')
SV = Parameters[:, 4].astype('float32')
CO = Parameters[:, 5].astype('float32')
VO2 = Parameters[:, 6].astype('float32')
VCO2 = Parameters[:, 7].astype('float32')
VE = Parameters[:, 8].astype('float32')
BOA = Parameters[:, 9].astype('float32')

AT: 10:21
时间: 10:21
力竭时间: 15:06
心肺耐力综合得分为: 82.2
心血管能力得分为: 93.7
呼吸能力得分为: 75.6
代谢能力得分为: 88.2
['10:21', '15:06', 82.2, 93.7, 75.6, 80.2, 124.0]

Process finished with exit code 0

```

#### 4. GA\_XGBOOST(PYTHON)

```
1. # coding=utf-8
```

```

2. from __future__ import division
3. import numpy as np
4. import pandas as pd
5. import random
6. import math
7. from sklearn import metrics
8. from sklearn.model_selection import train_test_split
9. import xgboost as xgb
10. from random import randint
11.
12. # from xgboost.sklearn import XGBClassifiers
13.
14. # generations = 400    # Generations 100
15. pop_size = 500    # Population quantity 500
16. # max_value = 10      # The maximum value allowed in a gene (to
# prevent the number of discrete variables from reaching a power of two,
# limit the maximum value, not used here)
17. chrom_length = 15   # Length of chromosome
18. pc = 0.6    # Crossover probability
19. pm = 0.01    # Mutation probability
20. results = []    # Optimal solution (auc 最高
    值, n_estimators, max_depth)
21. fit_value = []    # 个体适应度
22. fit_mean = []    # 平均适应度
23. # pop = [[0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0] for i in r
# ange(pop_size)] # Initializes the initial sequence of genes of all
# individuals in the population
24.
25. random_seed = 20
26. cons_value = 0.19 / 31  # (0.20-0.01) / (32 - 1)
27.
28. '''The parameters to debug are: (Reference:
http://xgboost.readthedocs.io/en/latest/parameter.html)
29. tree_num: Number of base trees  ----- (parameters to
debug)
30. eta: learning_rate, default value 0.3, range of
[0,1]  ----- (parameters to debug)
31. max_depth: default value 6  ----- (parameters to
debug)
32.
33. Three of the above parameters should be adjusted, and the rest can
be adjusted by ideal values
34.'''

```

```
35.  
36.  
37. def xgboostModel(tree_num, eta, max_depth, random_seed):  
38.     train_xy = loadFile("../Data/train-gao.csv")  
39.     train_xy = train_xy.drop('ID', axis=1) # Delete the ID of the  
        training set  
40.     # Divide the training set into a ratio of 8:2 (training set to  
        validation set ratio)  
41.     train, val = train_test_split(  
42.         train_xy, test_size=0.2, random_state=80)  
43.  
44.     train_y = train.Kind  
45.     train_x = train.drop('Kind', axis=1)  
46.     dtrain = xgb.DMatrix(train_x, label=train_y)  
47.  
48.     val_y = val.Kind  
49.     val_x = val.drop('Kind', axis=1)  
50.     dval = xgb.DMatrix(val_x)  
51.  
52.     params = {  
53.         'booster': 'gbtree', # gbtree used  
54.         'objective': 'binary:logistic',  
55.         'early_stopping_rounds': 100,  
56.         # 'scale_pos_weight': 0.13, # Positive sample weight  
57.         'eval_metric': 'auc',  
58.         'eta': eta, # 0.02  
59.         'max_depth': max_depth, # 8  
60.         'gamma': 0.1,  
61.         'subsample': 0.8,  
62.         'colsample_bytree': 0.8,  
63.         'lambda': 550,  
64.         'alpha': 19,  
65.         'seed': randint(1, 10000),  
66.         'nthread': 3,  
67.         'silent': 1  
68.     }  
69.     model = xgb.train(params, dtrain, num_boost_round=tree_num)  
70.     predict_y = model.predict(dval, ntree_limit=model.best_ntree_  
        limit)  
71.     roc_auc = metrics.roc_auc_score(val_y, predict_y)  
72.     return roc_auc  
73.  
74.
```

```

75.def loadFile(filePath):
76.    fileData = pd.read_csv(filePath)
77.    return fileData
78.
79.
80.# Step 1 : Encode the parameters (used to initialize the gene
sequence, can be selected to initialize the gene sequence, omitted
in this function)
81.def geneEncoding(pop_size, chrom_length):
82.    pop = []
83.    for i in range(pop_size):
84.        temp = []
85.        for j in range(chrom_length):
86.            temp.append(random.randint(0, 1))
87.        pop.append(temp)
88.    return pop[1:]
89.
90.
91.# Step 2 : Calculate the individual objective function value
92.def cal_obj_value(pop):
93.    objvalue = []
94.    variable = decodechrom(pop)
95.    for i in range(len(variable)):
96.        tempVar = variable[i]
97.
98.        tree_num_value = (tempVar[0] + 1) * 10
99.        eta_value = 0.01 + tempVar[1] * cons_value
100.       max_depth_value = 3 + tempVar[2]
101.       min_child_weight_value = 1 + tempVar[3]
102.
103.       aucValue = xgboostModel(tree_num_value, eta_value, max_d
   epth_value, random_seed)
104.       objvalue.append(aucValue)
105.    return objvalue # Objective function value
   objvalue[m] corresponds to Individual genes pop[m]
106.
107.
108.# Decode each individual and split into individual variables,
return tree_num (4) , eta (5) , max_depth (3)
109.def decodechrom(pop):
110.    variable = []
111.    for i in range(len(pop)):
112.        res = []

```

```

113.
114.        # Calculate the first variable value, 0101->10(reverse)
115.        temp1 = pop[i][0:4]
116.        v1 = 0
117.        for i1 in range(4):
118.            v1 += temp1[i1] * (math.pow(2, i1))
119.        res.append(int(v1))
120.
121.        # Calculate the second variable value
122.        temp2 = pop[i][4:9]
123.        v2 = 0
124.        for i2 in range(5):
125.            v2 += temp2[i2] * (math.pow(2, i2))
126.        res.append(int(v2))
127.
128.        # Calculate the third variable value
129.        temp3 = pop[i][9:12]
130.        v3 = 0
131.        for i3 in range(3):
132.            v3 += temp3[i3] * (math.pow(2, i3))
133.        res.append(int(v3))
134.
135.        variable.append(res)
136.    return variable
137.
138.
139.# Step 3: Calculate individual fitness value (calculate maximum
   value, eliminate negative value)
140.def calfitvalue(obj_value):
141.    fit_value = []
142.    temp = 0.0
143.    Cmin = 0
144.    for i in range(len(obj_value)):
145.        if (obj_value[i] + Cmin > 0):
146.            temp = Cmin + obj_value[i]
147.        else:
148.            temp = 0.0
149.        fit_value.append(temp)
150.    return fit_value
151.
152.
153.# Step 4: Find the maximum value of the fitness function, and the
   corresponding individual

```

```

154.def best(pop, fit_value):
155.    best_individual = pop[0]
156.    best_fit = fit_value[0]
157.    for i in range(1, len(pop)):
158.        if (fit_value[i] > best_fit):
159.            best_fit = fit_value[i]
160.            best_individual = pop[i]
161.    return [best_individual, best_fit]
162.
163.
164.# Step 5: Each time evolve, record the best results (convert binary
to decimal)
165.def b2d(best_individual):
166.    # Calculate the first variable value
167.    temp1 = best_individual[0:4]
168.    v1 = 0
169.    for i1 in range(4):
170.        v1 += temp1[i1] * (math.pow(2, i1))
171.    v1 = (v1 + 1) * 10
172.
173.    # Calculate the second variable value
174.    temp2 = best_individual[4:9]
175.    v2 = 0
176.    for i2 in range(5):
177.        v2 += temp2[i2] * (math.pow(2, i2))
178.    v2 = 0.01 + v2 * cons_value
179.
180.    # Calculate the third variable value
181.    temp3 = best_individual[9:12]
182.    v3 = 0
183.    for i3 in range(3):
184.        v3 += temp3[i3] * (math.pow(2, i3))
185.    v3 = 3 + v3
186.
187.    return int(v1), float(v2), int(v3)
188.
189.# Step 6: Natural selection (roulette algorithm)
190.def selection(pop, fit_value):
191.    # Calculate the probability of each adaptation value
192.    new_fit_value = []
193.    total_fit = sum(fit_value)
194.    for i in range(len(fit_value)):
195.        new_fit_value.append(fit_value[i] / total_fit)

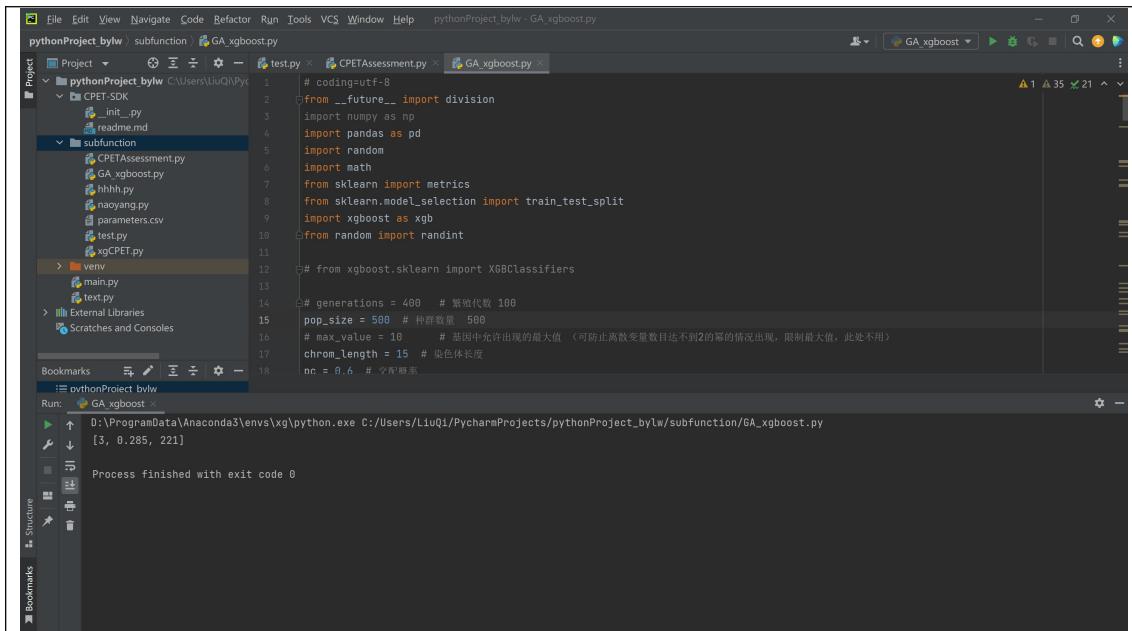
```

```

196. # The cumulative probability of each adaptation
197. cumsum(new_fit_value)
198. # Generate a sequence of random floating point numbers
199. ms = []
200. pop_len = len(pop)
201. for i in range(pop_len):
202.     ms.append(random.random())
203. # Sort the generated random sequence of floating point numbers
204. ms.sort()
205. # Roulette algorithm (the selected individual becomes the next
      round, the unselected individual is eliminated directly, and the
      selected individual is replaced)
206. fitin = 0
207. newin = 0
208. newpop = pop
209. while newin < pop_len:
210.     if (ms[newin] < new_fit_value[fitin]):
211.         newpop[newin] = pop[fitin]
212.         newin = newin + 1
213.     else:
214.         fitin = fitin + 1
215. pop = newpop
216.
217.
218. # Sum up the fitness values
219. def sum(fit_value):
220.     total = 0
221.     for i in range(len(fit_value)):
222.         total += fit_value[i]
223.     return total
224.
225.
226. # Calculate cumulative probability
227. def cumsum(fit_value):
228.     temp = []
229.     for i in range(len(fit_value)):
230.         t = 0
231.         j = 0
232.         while (j <= i):
233.             t += fit_value[j]
234.             j = j + 1
235.         temp.append(t)
236.     for i in range(len(fit_value)):
```

```
237.         fit_value[i] = temp[i]
238.
239.
240.# Step 7: crossover
241.def crossover(pop, pc): # 个体间交叉, 实现基因交换
242.    poplen = len(pop)
243.    for i in range(poplen - 1):
244.        if (random.random() < pc):
245.            cpoint = random.randint(0, len(pop[0]))
246.            temp1 = []
247.            temp2 = []
248.            temp1.extend(pop[i][0: cpoint])
249.            temp1.extend(pop[i + 1][cpoint: len(pop[i])])
250.            temp2.extend(pop[i + 1][0: cpoint])
251.            temp2.extend(pop[i][cpoint: len(pop[i])])
252.            pop[i] = temp1
253.            pop[i + 1] = temp2
254.
255.
256.# Step 8: genetic mutations
257.def mutation(pop, pm):
258.    px = len(pop)
259.    py = len(pop[0])
260.    for i in range(px):
261.        if (random.random() < pm):
262.            mpoint = random.randint(0, py - 1)
263.            if (pop[i][mpoint] == 1):
264.                pop[i][mpoint] = 0
265.            else:
266.                pop[i][mpoint] = 1
267.
268.
269.def writeToFile(var, w_path):
270.    f = file(w_path, "a+")
271.    for item in var:
272.        f.write(str(item) + "\r\n")
273.    f.close()
274.
275.
276.def generAlgo(generations):
277.    pop = geneEncoding(pop_size, chrom_length)
278.    for i in range(generations):
279.        # print("第 " + str(i) + " 代开始繁殖.....")
280.        obj_value = cal_obj_value(pop) # Calculate the objective
```

```
function value
281.      # print(obj_value)
282.      fit_value = calfitvalue(obj_value) # Calculate
    individual fitness value
283.      # print(fit_value)
284.      [best_individual, best_fit] = best(pop, fit_value) # Pi
    ck the best individual and the best value of the function
285.      # print("best_individual: "+ str(best_individual))
286.      v1, v2, v3= b2d(best_individual)
287.      results.append([best_fit, v1, v2, v3]) # At each evolve,
    record the best results
288.      # print(str(best_individual) + " " + str(best_fit))
289.      selection(pop, fit_value) # Natural selection, weeding
    out some of the less fit individuals
290.      crossover(pop, pc) # crossover
291.      mutation(pop, pc) # genetic mutation
292.      # print(results)
293.      results.sort()
294.      # wirte results to file
295.      writeToFile(results, "generation_" + str(generations) + ".tx
    t")
296.      print(results[-1])
297.      # print(xgboostModel(100, 12))
298.
299.
300.if __name__ == '__main__':
301.    gen = [100, 200, 300, 400, 500]
302.    for g in gen:
303.        generAlgo(int(g))
```



```
# coding=utf-8
from __future__ import division
import numpy as np
import pandas as pd
import random
import math
from sklearn import metrics
from sklearn.model_selection import train_test_split
import xgboost as xgb
from random import randint
# from xgboost.sklearn import XGBClassifiers
# generations = 400 # 繁殖代数 100
pop_size = 500 # 种群数量 500
# max_value = 10 # 基因中允许出现的最大值 (可防止离散变量数目达不到2的幂的情况出现, 限制最大值, 此处不用)
chrom_length = 15 # 染色体长度
nc = 0.6 # 交配概率
```

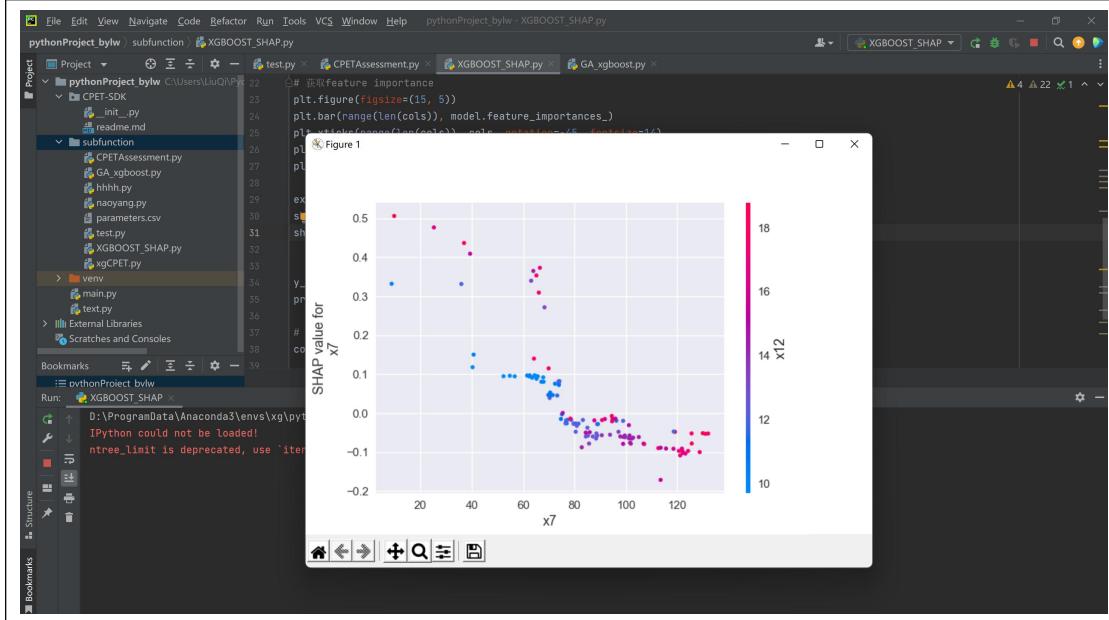
#### 4. XGBOOST\_SHAP(PYTHON)

```
1. import xgboost as xgb
2. import pandas as pd
3. import numpy as np
4. import matplotlib.pyplot as plt
5. import shap
6. from sklearn.model_selection import train_test_split
7. from sklearn.metrics import accuracy_score
8. from sklearn.metrics import confusion_matrix
9. from sklearn.metrics import classification_report
10. from sklearn.metrics import precision_recall_fscore_support
11. from sklearn.metrics import f1_score
12.
13. plt.style.use('seaborn')
14. data=pd.read_excel(r"D:\data.xlsx")
15. cols = ['x1', 'x2', 'x3', 'x4', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11','x12','x14','x15','x16']
16. num=len(data)
17. X_train,X_test,y_train,y_test=train_test_split(data[cols],data['等级'],test_size=0.3,random_state=10)
18. model = xgb.XGBRegressor(max_depth=221, learning_rate=0.285, n_estimators=3)
19. model.fit(X_train, y_train)
20.
21.## Interpretability analysis of the model
22.# get feature importance
23.plt.figure(figsize=(15, 5))
```

```

24.plt.bar(range(len(cols)), model.feature_importances_)
25.plt.xticks(range(len(cols)), cols, rotation=-45, fontsize=14)
26.plt.title('Feature importance', fontsize=14)
27.plt.show()
28.
29.explainer = shap.TreeExplainer(model)
30.shap_values = explainer.shap_values(X_train[cols])
31.shap.dependence_plot('x12',shap_values, X_train[cols]);
32.
33.
34.y_pred=model.predict(X_test)
35.predictions=[round(value) for value in y_pred]
36.
37.# Calculate confusion matrix
38.confusion_matrix(y_test,predictions)
39.
40.accuracy=accuracy_score(y_test,predictions)
41.# print('accu: %.2f%%' % (accuracy *100))
42.print(f1_score(y_test, predictions, average='macro'))

```



The screenshot shows the PyCharm IDE interface with the following details:

- Project Structure:** The project is named "pythonProject\_bylw". It contains a "CPET-SDK" folder with "init\_.py" and "readme.md". A "subfunction" folder contains "CPETAssessment.py", "GA\_xgboost.py", "hhhh.py", "naoyang.py", "parameters.csv", "test.py", "XGBOOST\_SHAP.py", and "xgCPET.py".
- Code Editor:** The "XGBOOST\_SHAP.py" file is open, showing Python code for XGBoost SHAP analysis. The code includes plotting feature importance, creating an explainer, and generating dependence plots. It also includes code for predicting test data and calculating a confusion matrix.
- Run Tab:** The "Run" tab shows the command run: "D:\ProgramData\Anaconda3\envs\xg\python.exe C:/Users/LiuQi/PycharmProjects/pythonProject\_bylw/subfunction/XGBOOST\_SHAP.py". The output shows the model's performance metrics: "IPython could not be loaded!", "ntree\_limit is deprecated, use 'iteration\_range' or model slicing instead.", and a score of "0.9152978354978355". The message "Process finished with exit code 0" is displayed.
- Bottom Navigation:** The navigation bar includes "Version Control", "Run", "TODO", "Problems", "Python Packages", "Python Console", and "Terminal".