19660_Mahmud_Omer_HW5

San Francisco Bay University

CS483 - Fundamentals of Artificial Intelligence
Homework Assignment #5

Due day: 8/6/2022

Instruction:

A. Push the source code to Github
B. Overdue homework submission could not be accepted.
C. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)

1. Create random forest based on the following dataset in **bootstrapping** method taking the recommended number of subset selection (*e.g. sqrt(n)*) on the handouts as reference. And then write Python function to compare with your hand-analysis

ID	Red	Green	Blue	Size (cm)	Fruit (Label)
0	1	0	0	7	Apple
1	0	1	0	20	Water Melon
2	1	0	0	1	Cherry
3	0	1	0	7.5	Apple
4	1	0	0	1	Strawberry
5	1	0	0	0.8	Cherry

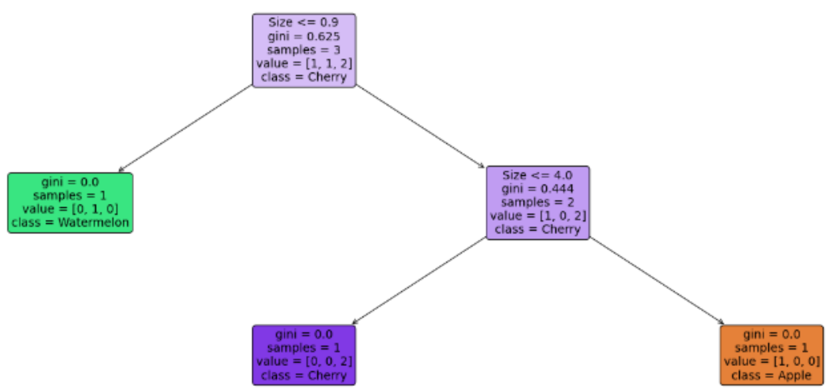
Solution:

```
import pandas as pd

"""
    Loading Data
"""
df = pd.read_csv("fruits.csv")
df.head()

"""
    Feature Selection
"""
#split dataset in features and target variable
feature_cols = ['Red',  'Green',  'Blue','Size']
x = df[feature_cols]  # Features
y = df["FruitType"]      # Target variable

"""
```

```
    Splitting Data
"""
# Split dataset into training set and test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=1)
# 70% training and 30% test
"""
    Building Random Forest Model
"""

from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(bootstrap= True, max_features ='sqrt', max_depth = 4)
model.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)
print(y_pred)

# building random forest
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
estimator = model.estimators_[5]
plt.figure(figsize =(25,10))
a = plot_tree(estimator,
              feature_names = feature_cols,
              class_names = ["Apple","Watermelon", "Cherry","Strawberry"],
              filled = True,
              rounded = True,
              fontsize = 14)
```

Result:

```
['Strawberry' 'Apple']
```



2. Given a function $f(x) = e^{-x^2} + 0.01cos\,(200x)$, find max *f(x)* value if $x \in$ [-2, 2] in Python program by genetic algorithm, considering 1-digit precision of fractional decimal *x*. And then verify your program running result by the function plot curve in Python or Excel

**Notice that in your answer sheet, 1st iteration hand-calculation must be shown including encoding, fitness function, population size determination, Cmin value for parent selection in Roulette Wheel method, crossover rate/mutation rate selections, and the number of evolution generations as termination condition*

Solution:

i. Create fitness function [fit(x)]:

$$f(x) = e^{-x^2} + 0.01cos\,(200x)$$

$$\boldsymbol{fit(x)} = \boldsymbol{f(x)} = e^{-x^2} + 0.01cos\,(200x)\,, x \in [-2, 2]$$

ii. Encoding

f(x), x∈[lower bound, upper bound]

Num. of Bit = log2(upper bound - lower bound) * 10^2

= log2((2 –(- 2)) * 10^2)

= 9

Binary number: ranges from 0_0000_0000 to 1_1111_1111

iii. Decoding binary to decimal

To decode the binary to decimal numbers

where decimal_lower = 0 and decimal_upper = 511

Number of population size = 1.5*9 ≈13 – taking 14

ID	Random #	Conv to bin	decoded value (3-digit precision)
1	9	000001001	-1.929
2	32	000100000	-1.749
3	55	000110111	-1.569
4	103	001100111	-1.194
5	170	010101010	-0.669
6	245	011110101	-0.082
7	320	101000000	0.505
8	380	101111100	0.974
9	410	110011010	1.209
10	480	111100000	1.757
11	510	111111110	1.992
12	280	100011000	0.192
13	150	10010110	-0.825
14	350	101011110	0.739

x = lower_bound + decimal(chromosome)*(upper_bound - lower_bound)/(2^chromosome_size-1)

iv. Selection by Roulette Wheel (or other method)

ID	Random #	Conv to bin	decoded value (3-digit precision)	Fitness Value f(x)	F(x),Cmin*	Probability	Cum. Prob.	Prob. Slots	Rand # in [0,1]	Selected Chro
1	9	000001001	-1.929	0.016	0.016	0.003	0.003	0-0.003	0.659	100011000
2	32	000100000	-1.749	0.042	0.042	0.008	0.011	0.003-0.011	0.442	101000000
3	55	000110111	-1.569	0.095	0.095	0.017	0.028	0.011-0.028	0.229	011110101
4	103	001100111	-1.194	0.25	0.25	0.045	0.073	0.028-0.073	0.781	100011000
5	170	010101010	-0.669	0.636	0.636	0.115	0.188	0.073-0.188	0.305	011110101
6	245	011110101	-0.082	0.986	0.986	0.178	0.366	0.188-0.366	0.312	011110101
7	320	101000000	0.505	0.784	0.784	0.142	0.508	0.366-0.508	0.552	101111100
8	380	101111100	0.974	0.397	0.397	0.072	0.58	0.508-0.58	0.27	011110101
9	410	110011010	1.209	0.222	0.222	0.04	0.62	0.58 - 0.62	0.439	101000000
10	480	111100000	1.757	0.055	0.055	0.01	0.63	0.62 - 0.63	0.427	101000000
11	510	111111110	1.992	0.011	0.011	0.002	0.632	0.63 - 0.632	0.875	010010110
12	280	100011000	0.192	0.971	0.971	0.175	0.807	0.632-0.807	0.453	101000000
13	150	010010110	-0.825	0.506	0.506	0.091	0.898	0.807-0.898	0.306	011110101
14	350	101011110	0.739	0.569	0.569	0.103	1.001	0.898-1.001	0.176	010101010
				sum	5.54	1.001				

C_min = **0**

To get max value:

$$F(x) = \begin{cases} f(x) + Cmin & \text{if } f(x) + Cmin > 0 \\ 0 & \text{if } f(x) + Cmin <= 0 \end{cases}$$

v. Cross over

a. Setup Pc (Crossover Rate) = 0.8; usually in 0.5~0.95
if random number in [0,1] is less then Pc, crossover will be taken

b. Parent selection:
Method 1 : 1-2, 3-4, … … (n-1)-n. E.g. (1-2), (3-4), … … (15-16)
Method 2 : 1-n/2, 2-(n/2+1), … … (n/2-1)-n E.g. (1-9), (2-10), … … (8-16)
Method 1 is taken on this process as follows

Updated ID	Decimal #	Selected Chromosom	Rand # in [0,1]	Pc =0.8	Cross over	Rand Cross Pnt in[1,9]	Generation P(1)	Updated Dec
1	208	100011000	0.957	0.957 > 0.8	No	5	100011000	280
2	320	101000000					101000000	320
3	245	011110101	0.529	0.529<0.8	Yes	3	010011000	152
4	208	100011000					101110101	373
5	245	011110101	0.975	0.975 > 0.8	No	1	011110101	245
6	245	011110101					011110101	245
7	380	101111100	0.058	0.058<0.8	Yes	5	101110101	373
8	245	011110101					011111100	252
9	208	101000000	0.586	0.586<0.8	Yes	6	101000000	320
10	208	101000000					101000000	320
11	150	010010110	0.89	0.89 > 0.8	No	1	010010110	150
12	208	101000000					101000000	320
13	245	011110101	0.744	0.744<0.8	Yes	8	011110110	246
14	170	010101010					010101001	169

vi. Mutation with Mutation Rate

a. Setup Pm(Mutation Rate) = 0.025; usually in 0.1~0.001
b. Calculate how many bits will be mutated in "Generation P(1)"
9(chromosome size) * 14 (population size) * 0.025 (mutation rate) = 3

c. Randomly select 3 bits in all chromosomes and change value from 1 to 0 or 0 to 1

Rand. Sel. Chromosome in [1,14]	Rand. Sel. Mutation Pnt in [1,9]
7	5
12	7
5	6

Updated ID	Generation P(1)	Mutated P(1)	Decimal	Mutated P(1)
1	100011000	100011000	280	280
2	101000000	101000000	320	320
3	010011000	010011000	152	152
4	101110101	101110101	373	373
5	011110101	011111101	245	253
6	011110101	011110101	245	245
7	101110101	101100101	373	357
8	011111100	011111100	252	252
9	101000000	101000000	320	320
10	101000000	101000000	320	320
11	010010110	010010110	150	150
12	101000000	101000100	320	324
13	011110110	011110110	246	246
14	010101001	010101001	169	169

vii. Repeat step iv

a. Selection by Roulette Wheel (or other method)

ID	Generation P(1)	Decimal	decoded value (3-digit precision)	Fitness Value f(x)	F(x),Cmin*	Probability	Cum. Prob.	Prob. Slots	Rand # in [0,1]	Selected Chro
1	100011000	280	0.192	0.971	0.971	0.091	0.091	0 – 0.091	0.732	101000000
2	101000000	320	0.505	0.784	0.784	0.074	0.165	0.091–0.165	0.889	011110110
3	010011000	152	-0.81	0.521	0.521	0.049	0.214	0.165–0.214	0.782	101000100
4	101110101	373	0.92	0.427	0.427	0.04	0.254	0.214–0.254	0.962	010101001
5	011111101	253	-0.02	0.993	0.993	0.093	0.347	0.254–0.347	0.852	011110110
6	011110101	245	-0.082	0.986	0.986	0.093	0.44	0.347–0.44	0.449	101100101
7	101100101	357	0.795	0.528	0.528	0.05	0.49	0.44 – 0.49	0.095	101000000
8	011111100	252	-0.027	1.006	1.006	0.094	0.584	0.49 – 0.584	0.316	011111101
9	101000000	320	0.505	0.784	0.784	0.074	0.658	0.584–0.658	0.619	101000000
10	101000000	320	0.505	0.784	0.784	0.074	0.732	0.658–0.732	0.424	011110101

11	010010110	150	-0.826	0.503	0.503	0.047	0.779	0.732-0.779	0.861	011110110
12	101000100	324	0.536	0.76	0.76	0.071	0.85	0.779-0.85	0.775	010010110
13	011110110	246	-0.074	0.988	0.988	0.093	0.943	0.85 - 0.943	0.477	101100101
14	010101001	169	-0.677	0.623	0.623	0.058	1.001	0.943-1.001	0.754	010010110
				sum	10.657	1.001				

C_min = **0**

To get max value:

$$F(x) = \begin{cases} f(x) + Cmin & \text{if } f(x) + Cmin > 0 \\ 0 & \text{if } f(x) + Cmin <= 0 \end{cases}$$

b. Cross over

a. Setup Pc (Crossover Rate) = 0.8; usually in 0.5~0.95

if random number in [0,1] is less then Pc, crossover will be taken

b. Parent selection:

Method 1 : 1-2, 3-4, … … (n-1)-n. E.g. (1-2), (3-4), … … (15-16)

Method 2 : 1-n/2, 2-(n/2+1), … … (n/2-1)-n E.g. (1-9), (2-10), … … (8-16)

Method 1 is taken on this process as follows

Updated ID	Decimal #	Selected Chromosom	Rand # in [0,1]	Pc =0.8	Cross over	Rand Cross Pnt in[1,9]	Generation P(1)	Updated Dec
1	208	101000000	0.839	0.839> 0.8	No	8	101000000	208
2	320	011110110					011110110	320
3	245	101000100	0.813	0.813>0.8	No	4	101000100	245
4	208	010101001					010101001	208
5	245	011110110	0.266	0.266<0.8	Yes	4	011100101	229
6	245	101100101					101110110	374
7	380	101000000	0.705	0.705<0.8	Yes	1	111111101	509
8	245	011111101					001000000	64
9	208	101000000	0.502	0.502<0.8	Yes	1	111110101	501
10	208	011110101					001000000	64
11	150	011110110	0.559	0.559< 0.8	yes	5	011110110	246
12	208	010010110					010010110	150
13	245	101100101	0.857	0.857>0.8	No	3	101100101	245
14	170	010010110					010010110	170

c. Mutation with Mutation Rate

a. Setup Pm(Mutation Rate) = 0.025; usually in 0.1~0.001

b. Calculate how many bits will be mutated in "Generation P(1)"

9(chromosome size) * 14 (population size) * 0.025 (mutation rate) = 3

c. Randomly select 3 bits in all chromosomes and change value from 1 to 0 or 0 to 1

Rand. Sel. Chromosome in [1,14]	Rand. Sel. Mutation Pnt in [1,9]
11	2
6	4
1	6

Updated ID	Mutated P(1)	Mutated P(1)	decoded value (3-digit precision)	Fitness Value f(x)
1	101001000	328	0.568	0.733
2	011110110	320	0.505	0.784
3	101000100	245	-0.082	0.986
4	010101001	208	-0.372	0.876
5	011100101	229	-0.207	0.95
6	101010110	342	0.677	0.623
7	111111101	509	1.984	0.025
8	001000000	64	-1.499	0.104
9	111110101	501	1.922	0.029
10	001000000	64	-1.499	0.104
11	001110110	118	-1.076	0.314
12	010010110	150	-0.826	0.503
13	101100101	245	-0.082	0.986
14	010010110	170	-0.669	0.636

Conclusion: the maximum value of the function $f(x) = 0.986$ and it occurs at $x = -0.082$

```
import math
import random

decimal = [280,320,152,373,253,245,357,252,320,320,150,324,246,169]
lower_bound = -2
upper_bound = 2
chromosome_size = 9
x = [] # decoded values
for i in decimal:
  x.append(round(lower_bound + i*(upper_bound -
```

```
 lower_bound)/(2**chromosome_size-1),3))

print(x)
######################################################################
#####################
y = [] # Fx -> fitness function
sum = 0 # sum of F(x)
for i in x:
    y.append(round(math.exp(-i**2) + 0.01*(math.cos(200*i)),3))
    sum +=(math.exp(-i**2) + 0.01*(math.cos(200*i)))
print(y)
print(round(sum,3))

######################################################################
####################
# probability
p = []      # probability
s = 0       # sum of probabilities
cp=[]       # commulative probability
for i in range(len(y)):
  p.append(round(y[i]/sum,3))
  s+=(round(y[i]/sum,3))
  cp.append(round(s,3))

print(p)
print(s)
print(cp)

######################################################################
#######################
#probability slots

ps = []  # probability slots
for i in range(len(cp)-1):
  if i == 0:
    Str = str(0) + " - "+ str(cp[i])
    ps.append(Str)
  Str = str(cp[i]) + " - "+ str(cp[i+1])
  ps.append(Str)

print(ps)

######################################################################
#####################
# generate random number between [0-1]
rnd = []  # list of random numbers
for i in range(len(x)):
```

```
  rnd.append(round(random.uniform(0, 1),3))
#print(ps)
print(rnd)

# generate n/2 random numbers between [0,1]
rnd = []  # list of random numbers
for i in range(round(len(x)/2)):
  rnd.append(round(random.uniform(0, 1),3))
#print(ps)
print(rnd)

# generate n/2 random numbers between [1-9]
rnd = []  # list of random numbers
for i in range(round(len(x)/2)):
  rnd.append(random.randint(1, 9))
#print(ps)
print(rnd)

# Mutation
# select 3 random numbers between [1-14] and [1,9]
rnd = []  # list of random numbers
for i in range(3):
  rnd.append(random.randint(1, 9))
#print(ps)
print(rnd)
```