**Logo

Description automatically generated**

**San Francisco Bay University**

**CS483 - Fundamentals of Artificial Intelligence**

**2022 Summer Final Exam**

**Student Name: Mahmud Omer Student ID:19660**

**Instruction**

1. **Put your answer right after each question in the answer sheet**
2. **Make a copy & paste of your program in text mode, NOT image onto the answer sheet.**
3. **Excel is preferred for hand calculation**
4. The following decision tree is built up from one of animal classification datasets. Please calculate information gains based on gini impurity from top to **each** leaf

Diagram, timeline

Description automatically generated

Solution

|  |  |  |  |
| --- | --- | --- | --- |
| **Initial stage: Impurity of root** | | | |
|  | **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | | |
|  | = 2/7\*(1-2/7) +1/7\*(1-1/7) + 2/7\*(1-2/7) + 2/7\*(1-2/7) | | | |
|  | = 0.735 | |  |
|  |  |  |  |
|  | **Ave. Imp** = 7/7 \* 0.735 = **0.735** | | |

**Level 1; Impurity of “is color yellow”**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **True** | **E** | **M** | **T** | **G** |
| **4** | **0** | **0** | **2** | **2** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 0/4\*(1-0/4) +0/4\*(1-0/4) + 2/4\*(1-2/4) + 2/4\*(1-2/4) | | |
| = 0.5 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **False** | **E** | **M** | **T** | **G** |
| **3** | **2** | **2** | **0** | **0** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 2/3\*(1-2/3) +1/3\*(1-1/3) + 0/3\*(1-0/3) + 0/3\*(1-0/3) | | |
| = 0.22 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tot. Ave. Imp.** = 0.5 + 0.22= **0.72** | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Info. Gain** = 0.735 (from Ave.Imp of level init) - 0.72(from Tot. Ave. Imp) = **0.015** | | | | | | | |

**Level 2: impurity of is height >=30**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **True** | **E** | **M** | **T** | **G** |
| **2** | **0** | **0** | **0** | **2** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 0/2\*(1-0/2) +0/2\*(1-0/2) + 0/2\*(1-0/2)+ 2/2\*(1-2/2) | | |
| = 0 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **False** | **E** | **M** | **T** | **G** |
| **2** | **0** | **0** | **2** | **0** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 0/2\*(1-0/2) +0/2\*(1-0/2) + 2/2\*(1-2/2) + 0/2\*(1-0/2) | | |
| = 0 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tot. Ave. Imp.** = 0.0 + 0.0= **0.0** | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Info. Gain** = 0.015 (from Ave.Imp of level 1) - 0.0(from Tot. Ave. Imp) = **0.015** | | | | | | | |

**Level 2: impurity of is height < 30**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **True** | **E** | **M** | **T** | **G** |
| **1** | **0** | **1** | **0** | **0** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 0/1\*(1-0/1) +1/1\*(1-1/1) + 0/1\*(1-0/1)+ 1/1\*(1-1/21 | | |
| = 0 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **False** | **E** | **M** | **T** | **G** |
| **2** | **2** | **0** | **0** | **0** |

|  |  |  |
| --- | --- | --- |
| **imp =** P(E)\*(1-P(E)) + P(M)\*(1-P(M)) + P(T)\*(1-P(T)) + P(G)\*(1-P(G)) | | |
| = 2/2\*(1-2/2) +0/2\*(1-0/2) + 0/2\*(1-0/2) + 0/2\*(1-0/2) | | |
| = 0 |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Tot. Ave. Imp.** = 0.0 + 0.0= **0.0** | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
| **Info. Gain** = 0.015 (from Ave.Imp of level 1) - 0.0(from Tot. Ave. Imp) = **0.015** | | | | | | | |

1. Chromosome crossover operation is one of the processes for generation evolution in Genetic Algorithm. Write the Python function(s) with "Crossover Rate" parameter to implement it, taking the following parent chromosomes as test case in your program.

|  |  |
| --- | --- |
| Parents | New Gen. for example |
| 1101111001 | **1101111011** |
| 1100001011 | **1100001001** |
| 1100001011 | **1100001011** |
| 1110010000 | **1110010000** |
| 1101111001 | **1110010000** |
| 1110010000 | **1101111001** |
| 1110100001 | **1110100001** |
| 0011010000 | **0011010000** |
| 1101111001 | **1101111000** |
| 0100110010 | **0100110011** |
| 1001011010 | **1101111001** |
| 1101111001 | **1001011010** |
| 0111011111 | **0111010101** |
| 1001110101 | **1001111111** |
| 1101111001 | **1101111001** |
| 1110100001 | **1110100001** |

**Solution:**

|  |
| --- |
| import pandas as pd  import numpy as np  df = pd.read\_csv("cross\_over.csv")  df.head()  parents = df["Parents"]      # parent  offspring = df["off\_spring"]  # new off spring frim file  # cross over function  def crossover(parents, offspring\_size,Cross\_Over\_rate):      offspring = np.empty(offspring\_size)      crossover\_point = np.uint8(offspring\_size[1]/2)      for k in range(offspring\_size[0]):          r = random.random()          if r < Cross\_Over\_rate:            parent1\_idx = k%parents.shape[0]            parent2\_idx = (k+1)%parents.shape[0]            offspring[k, 0:crossover\_point] = parents[parent1\_idx, 0:crossover\_point]            offspring[k, crossover\_point:] = parents[parent2\_idx, crossover\_point:]      return offspring  Cross\_Over\_rate = 0.8  off\_spring\_size =10  print(crossover(parents,off\_spring\_size, Cross\_Over\_rate )) |

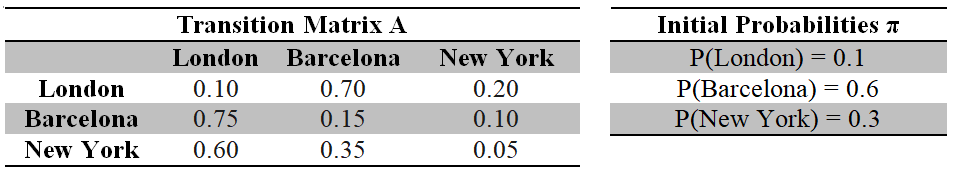
1. Being similar as above, write the Python function(s) with "Mutation Rate" parameter to implement mutation operation for the given chromosome as test case.

|  |
| --- |
| New Gen. |
| 1001011010 |
| 1110010000 |
| 0111010100 |
| 1110010101 |
| 0111010000 |
| 1110010101 |
| 1110100011 |
| 1110100001 |
| 0100111000 |
| 0110010011 |
| 1110001011 |
| 1000010010 |
| 0001011010 |
| 1011010000 |
| 1101101001 |
| 1110010000 |

**Solution:**

|  |
| --- |
| import pandas as pd  import numpy as np  import random  #df = pd.read\_csv("mutation.csv")  #df.head()  # off\_spring = df["off\_spring"]  off\_spring = ['1001011010',  '1110010000',  '111010100',  '1110010101',  '111010000',  '1110010101',  '1110100011',  '1110100001',  '100111000',  '110010011',  '1110001011',  '1000010010',  '1011010000',  '1011010000',  '1101101001',  '1110010000']  mutation\_rate = 0.01  def mutate\_Off\_Spring(off\_spring, mutation\_rate):      mutated\_list = list(off\_spring)      for i in range(len(off\_spring)):          r = random.random()          if r < mutation\_rate:              mutation\_point = random.randint(0, len(mutated\_list) - 1)              mutated\_list[mutation\_point] = random.choice(off\_spring)          return ''.join(mutated\_list)  mutated\_offspring = [mutate\_Off\_Spring(os,mutation\_rate) for os in off\_spring]  ## run the function  print(mutated\_offspring) |

1. Assuming that a salesperson who has to travel between the following three cities for his/her job - London, Barcelona, and New York, his/her goal is to minimize the traveling time so that he/she can be more efficient. A set of transition probabilities among three cities is given as follows and initial probabilities where he/she is as well. The salesperson starts his/her journey on Tuesday from a city and he/she has to plan something on Friday. Please calculate what is the probability that he/she will be in three cities on Friday by Python program?



*\*Hint: In the Markov chain, after 3 days from Tuesday, transition matrix A will be becoming from the state "Tuesday" to the state "Friday". And the probabilities in three cities - London, Barcelona and New York on Friday should be equal to π\*, and the size of π is 1*×*3 matrix, like*

*[P(London), P(Barcelona), P(New York)]*

**Solution:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Using the markov chain, given transition probablities matrix A, and initial probabilities as follows** | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  | **Transition Matrix A** | |  |  |  | **Initial Probabilites π (Today)** | | | |  |
|  |  | **London** | **Barcelona** | **New York** |  | **P(London) = 0.1** |  |  |  |  |
|  | **Landon** | **0.1** | **0.7** | **0.2** |  | **P(Barcelona) = 0.6** |  |  |  |  |
|  | **Barcelona** | **0.75** | **0.15** | **0.10** |  | **P(New York) = 0.3** |  |  |  |  |
|  | **New York** | **0.6** | **0.35** | **0.05** |  |  |  |  |  |  |

**The probabilities of being in any of the cities on Friday is obtained as follows using python code**

|  |
| --- |
| import numpy as np  # probabilities of being in London, Barcelona ,  New York on Tuesday  # Is 0.1, 0.6, 0.3 respectively  prob\_Tu = [0.1, 0.6, 0.3]  # after 3 days the probabilities of being on any of the city is :  tranA = np.matrix([[0.1,  0.7 ,0.2],[0.75,  0.15, 0.10], [0.6,  0.35, 0.05]])  prob\_Friday = prob\_Tu\*(tranA\*\*3)  print(prob\_Friday) |

**Result:**

****

**Hence the probability of being in London , Barcelona, and New York on Friday are 0.51825, 0.3577, and 0.1240 respectively**