AI Assignment 3

Theory

1. a) T → The event that a person has traveled.
   C → The event that a person has caught corona.
   O → The event that a person has caught a disease other than corona.
   D → The event that a person has died.
   M → The event that a person has a mild case.
   S→ The event that a person has a severe case.

   Now representing each statement in terms of random variable

   i) P($T \cap (C \cup O)$)
   ii) P($(M \cap C) \cap T$) = 0.15, P($(S \cap C)|T$) = 0.22
   iii) P($O|T$) = 0.485
   iv) P($O \cap D \cap T$) = 0.24
   v) P($\neg T \cap S \cap C$) = 0.025
   vi) P($S|\neg T$) = 0.457
   vii) P($D \cap C$) = 0.059
   viii) P($S \cup M$) = 0.7
   ix) P($T|S$) = 0.8
   x) P(C) = 0.5

   b) All probabilities should be greater than 0, they satisfy the axiom that P(A) >= 0 where A is a random variable
   c)
   D.

2. a) Yes we should switch the doors because there are ⅔ chances of winning in that case and if we do not switch the door the chances of winning are only ⅓. When we don't switch the door the probability of winning is ⅓ as the key is behind only 1 door whereas if we were initially wrong which has ⅔ chances as there are two empty doors, on switching in that case we win.
   b) Case 1
   We have chosen the correct door and do no not switch in this case the winning probability is ⅓
   Case 2
   We have chosen the door and the host reveals the correct door with probability ⅔ in this case we have to give up. Now if he reveals the other door with probability ⅓

and we decide to switch the chances of winning in this case are ⅔* ⅓ = 2/9
which is less than ⅓ **hence we should not switch. These are conditional cases(i.e given host reveals the door)**

c) Assumption: Host can make a mistake only when the player selects incorrect door
P(Host reveals loss | player selects incorrect door) = ⅓
P(Host reveals key | player selects incorrect door) = ⅔
P(Host reveals loss | player selects correct door) = 1
P(Host reveals key | player selects correct door) = 0
P(player selects incorrect door) = ⅔
P(player selects correct door) = ⅓

Player can win by switching only when he selects incorrect door initially

P(player selects incorrect door | Host reveals loss) = P(Host reveals los | player selects incorrect door)*P(player selects incorrect door)/P(Host reveals loss)

P(player selects incorrect door | Host reveals loss) = ⅔ *⅓)/P(Host reveals loss)

P(Host reveals loss) = P(Host reveals loss | player selects incorrect door)*P(player selects incorrect door) + P(Host reveals loss | player selects correct door) *P(player selects incorrect door)

= ⅔ *⅔ + ⅓ *1
= 5/9

P(player selects incorrect door | Host reveals loss) = ⅖ = 0.4
P(player selects correct door | Host reveals loss) = ⅗ = 0.6

d) Assumption: Calculating only when host reveals loss
Case 1 when he sticks
he will win only when he selects the correct door initially
Win = 1, Loss = 0

E[Win | Host reveals loss] = Win*P(player selects correct door | Host reveals loss) + Loss*P(player selects incorrect door | Host reveals loss) = 0.6

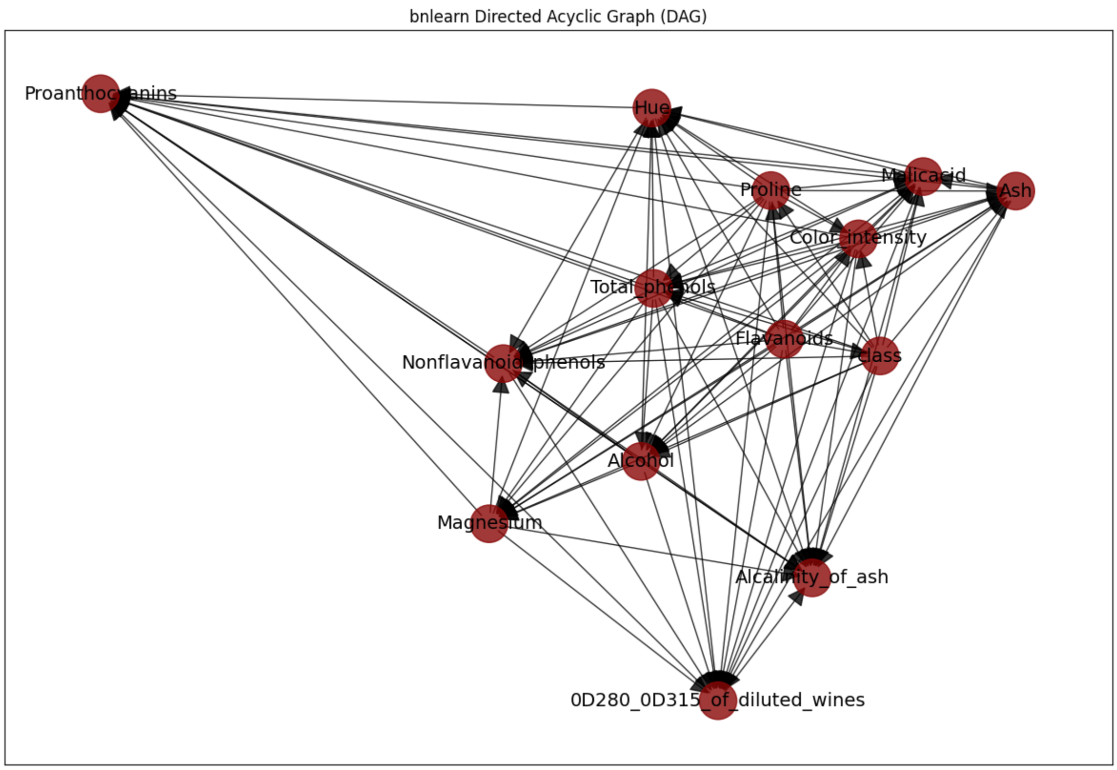
Case 2 when he switches
he will win only when he selects the incorrect door initially

E[Win | Host reveals loss] = Loss*P(player selects correct door | Host reveals loss) + Win*P(player selects incorrect door | Host reveals loss) = 0.4

Based on the expected values he should stick as concluded in part b.

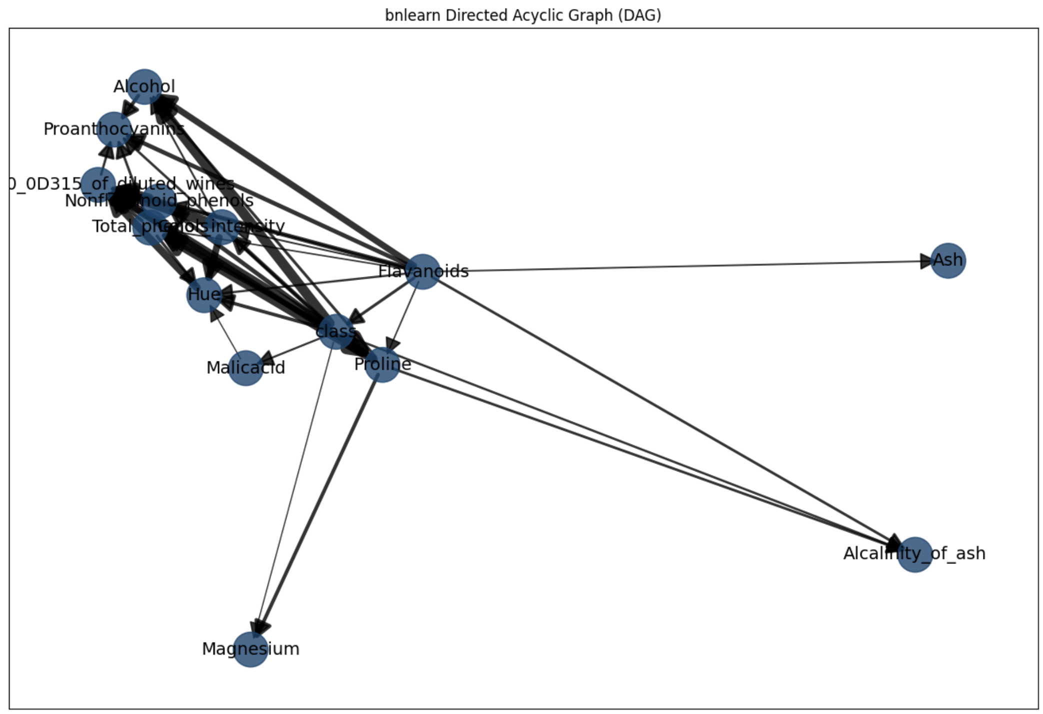
3. a) Pre-processing - I have done discretization of the continuous parameters in the dataset using KBinsDiscretizer function of sklearn with bin size = 4, this was necessary for the bayesian classifier as without that it was treating all the continuous variables as discrete which was clearly wrong. I also split the dataset with test ratio 0.2.

   b) Network A



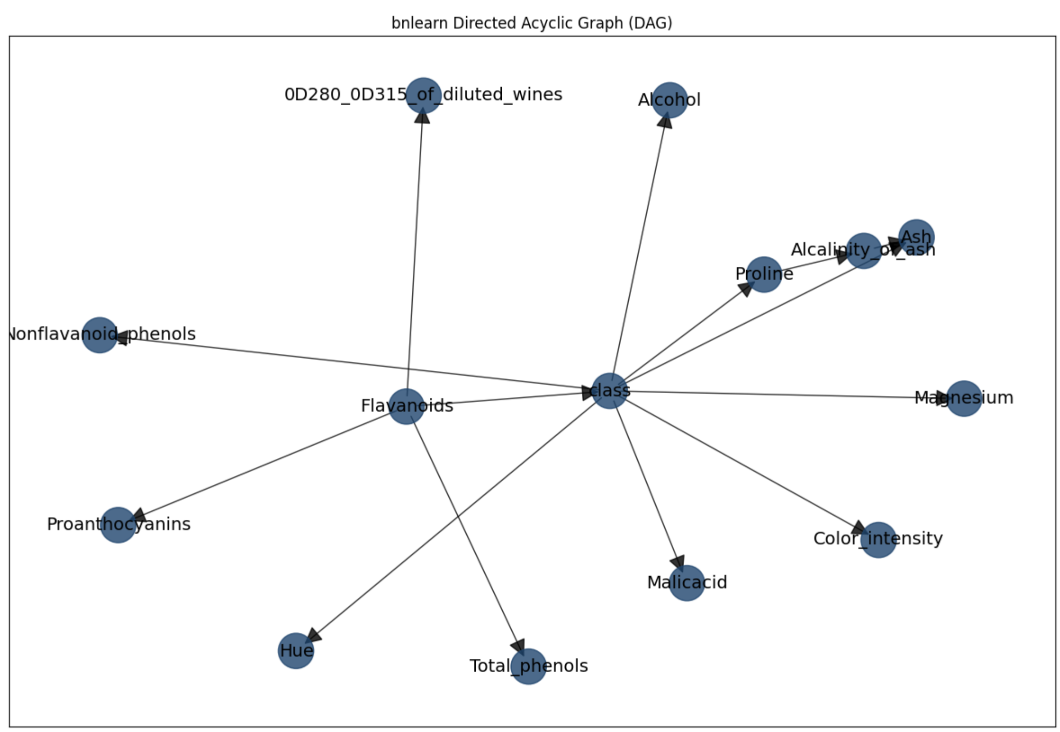
The graph was constructed using bn.structure_learning.fit(df_train, methodtype='hc', scoretype='k2'). From graph we can see that
“Magnesium” is parent of “Proanthocyanins”
“Magnesium” is parent of “0D280_0D315_of_diluted_wines”
“Alcohol” is parent of “0D280_0D315_of_diluted_wines”
“Class” is parent of “Alcalinity_of_ash”

Network B
I have pruned the network using bn.independence_test(network_A, df_train, alpha=0.0001, prune=True) It computes the edge strength using a statistical test of independence based using the model structure (DAG) and the data. For the pairs in the DAG (either by structure learning or user-defined), a statistical test “chi-square” is performed and it then removes the insignificant edges. The number of edges significantly declined from 91 in network A to 38.



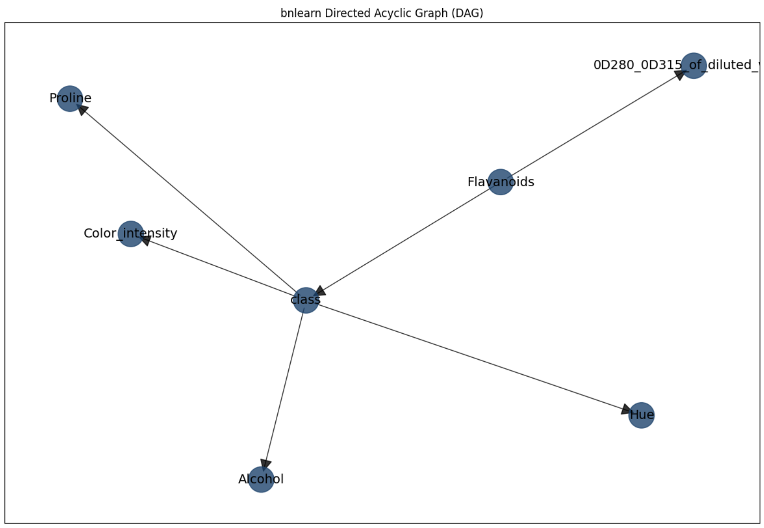
Improved Network A

I changed the scoring type from ‘k2’ to ‘bdeu’ incorporates a prior to account for the uncertainty in the parameter estimate and is more suitable for dealing with datasets with smaller sizes and prevents overfitting.



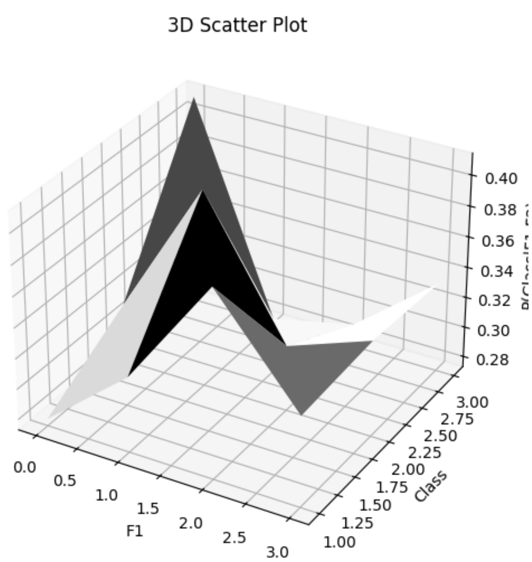
Network C

I used SelectKBest(f_classif, k=7).fit(X, y) to select the 7 most significant features ['Alcohol', 'Flavanoids', 'Color_intensity', 'Hue',''0D280_0D315_of_diluted_wines', 'Proline'], The Dag was constructed using “hc” and “bdeu”



Network C

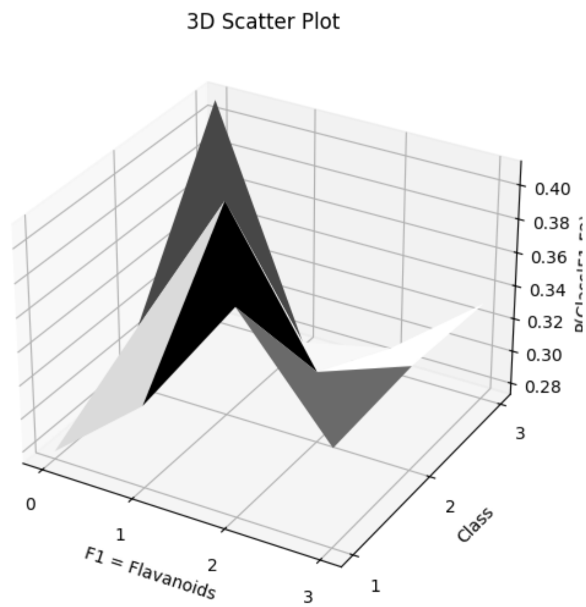
	class	F1	p
0	1	0.0	0.277778
1	1	1.0	0.322296
2	1	2.0	0.395617
3	1	3.0	0.332005
4	2	0.0	0.311111
5	2	1.0	0.401766
6	2	2.0	0.316032
7	2	3.0	0.335989
8	3	0.0	0.411111
9	3	1.0	0.275938
10	3	2.0	0.288351
11	3	3.0	0.332005



From CPD of ‘class’ we can infer that “class” is not dependent on F2 hence P(class|F1, F2) = P(class|F1)

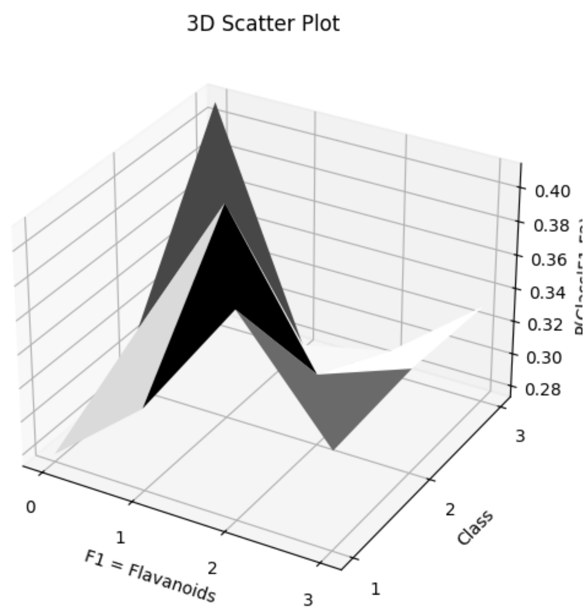
Network A improved

	class	Flavanoids	p
0	1	0.0	0.277778
1	1	1.0	0.322296
2	1	2.0	0.395617
3	1	3.0	0.332005
4	2	0.0	0.311111
5	2	1.0	0.401766
6	2	2.0	0.316032
7	2	3.0	0.335989
8	3	0.0	0.411111
9	3	1.0	0.275938
10	3	2.0	0.288351
11	3	3.0	0.332005



Network B

	class	Flavanoids	p
0	1	0.0	0.277778
1	1	1.0	0.322296
2	1	2.0	0.395617
3	1	3.0	0.332005
4	2	0.0	0.311111
5	2	1.0	0.401766
6	2	2.0	0.316032
7	2	3.0	0.335989
8	3	0.0	0.411111
9	3	1.0	0.275938
10	3	2.0	0.288351
11	3	3.0	0.332005



From the figures we can see that P(class | F1 = Flavanoids, F2) does not change as the parent-child dependency for ‘class’ variable did not change.

Inference

```
q1 = bn.inference.fit(network_A_imp_para, variables=['class'], evidence={'Malicacid' : 1,
'Flavanoids' : 2})
q2 = bn.inference.fit(network_A_imp_para, variables=['class'], evidence={'Alcohol' : 2,
'Flavanoids' : 0})
q3 = bn.inference.fit(network_A_imp_para, variables=['class'], evidence={'Malicacid' : 0,
'Flavanoids' : 2, 'Total_phenols' : 2})
q4 = bn.inference.fit(network_A_imp_para, variables=['class'], evidence={'Malicacid' : 0,
'Flavanoids' : 1, 'Magnesium':0})
```

q1

```
+----+---------+----------+
|    |   class |        p |
+====+=========+==========+
|  0 |       1 | 0.380019 |
+----+---------+----------+
|  1 |       2 | 0.317295 |
+----+---------+----------+
|  2 |       3 | 0.302686 |
+----+---------+----------+
```

q3

```
+----+---------+----------+
|    |   class |        p |
+====+=========+
|  0 |       1 | 0.426444 |
+----+---------+----------+
|  1 |       2 | 0.335766 |
+----+---------+----------+
|  2 |       3 | 0.237789 |
+----+---------+----------+
```

```
+----+---------+----------+
|    |   class |        p |
+====+=========+==========+
|  0 |       1 | 0.30524  |
+----+---------+----------+
|  1 |       2 | 0.274297 |
+----+---------+----------+
|  2 |       3 | 0.420462 |
+----+---------+----------+
```

q4

```
+----+---------+----------+
|    |   class |        p |
+====+=========+
|  0 |       1 | 0.296484 |
+----+---------+----------+
|  1 |       2 | 0.48135  |
+----+---------+----------+
|  2 |       3 | 0.222166 |
+----+---------+----------+
```

The class with the highest probability matches the domain knowledge.(I picked the evidence variables from df_test, the first four rows)

Train Metrics

Network	Accuracy
A	95.77
B	100
C	98.28

Test Metrics

Network	Accuracy
A	91.66
B	86.11
C	94.44