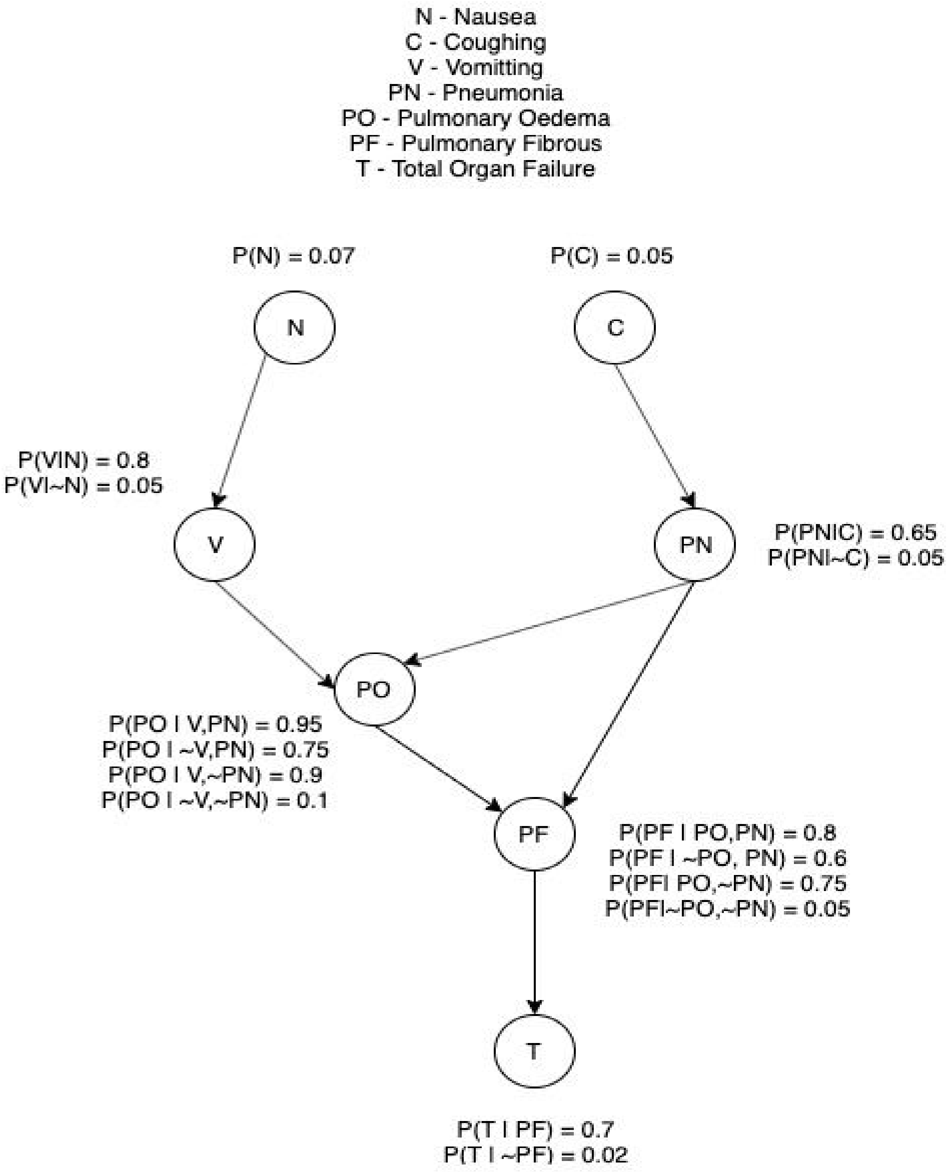
Assignment 5

Name: Adarsh Trivedi
Unity Id: atrived
Student Id: 200317604

Q1.

a. Bayes Network



1b.

Conditional Independencies

a. N is conditionally independent of other if V is given.

b. N is conditionally independent of C and P if PO is not given.

c. V is conditionally independent of C and P if PF and (PO or T) is not given.

d. V is conditionally independent of PF and T if PO not given or P is given.

e. PO is conditionally independent of N if V is known.

f. PO is independent of T if PF is not known.

g. PO is conditionally independent of C and PN if PF or T is not known.

h. T is conditionally independent of everything if PF is not given.

i. C is conditionally independent of everything if PN is known and is independent of N and V if PO or PF is not known.

j. PF is independent of N and V if PO is known.

k. PF is independent of C if PN is known.

l. Pneumonia is conditionally independent of N and V if PO is not known and independent of T if PF is known.

1c.

1. The symptoms include Nausea and Vomiting.

   We have to find P(N, V) = P(N) * P(V|N)
   = 0.07 * 0.8 = 0.056

2. The symptom is Coughing given that it is Total Organ Failure.

   P(C=true | T = True) = x * P(T = True, PF, PO, PN, V, N, C=True) eq.1
   P(C=false | T = True) = x * P(T = True, PF, PO, PN, V, N, C=True) eq.2

$$= \sum PF \sum PO \sum PN \sum V \sum N * P(T = True|PF, PO, PN, V, C = true, N) \; * \; P(PF|PO, PN, V, C = true, N) * P(PN|V, C = true$$
$$N) * P(N).P(C = true)$$

We can remove all the conditionally independent variables,

$$= \; \sum PF \sum PO \sum PN \sum V \sum N * \; P(T = true|PF) * P(PF|PO, PN) * P(PO|PN, V) * P(PN|C = true) * P(V|N) * P(N) * P(C)$$

Eliminating N:
Let $a1(V) = \sum_{N} P(V|N). P(N)$

$\sum PF \sum PO \sum PN \sum V{*}\ P(T = true|PF). P(PF|PO, PN). P(PO|PN, V). P(PN|C = true). a1(V). P(C = t)$

Eliminating V:
Let $a2(PO,PN) = \sum_{V} a1(V). P(PO|PN, V)$

$\sum PF \sum PO \sum PN{*}\ P(TOF = true|PF). P(PF|PO, PN). a2(PO, PN). P(PN|C = true). P(C = true)$

Eliminating PN:
Let $a3(PF,PO,C=true) = \sum_{Pn} P(PF|PO, Pn). a2(PO, Pn). P(Pn|C = true)$

$\sum PF \sum PO * P(T = true|PF). a2(PF, PO, C = true). P(C = true)$

Eliminating PO:
Let $a4(PF,C=true) = \sum_{PO} a3(PF, PO, C = true)$

$= P(C = true) * \sum PF * P(T = true|PF) * a4(PF,C=true)$

Solving the above equation for eq1,

=0.05[P(T=true|PF=true) * a4(PF=True,C=true) + P(T=True,PF=false) * a4(PF=false,C=true)[
= 0.05 [0.7*f(PF=true,C=true) + 0.02 * f(PF=False,C=true) **E1**

Similarly for equation 2:

= 0.95 [0.7*a4(PF=true,C=false) + 0.02*a4(PF=false,C=false)] **E2**

Solving the above equation by substituting values through a4, a3, a2 and a1 we get the following results:
(Calculations too long to be typed because of the shortage of time)

For equation 1:
We get P(C=true | T = True) = 0.0194
For equation 2:
We get P(C=false | T = True) = 0.5313

Normalizing the probability of 1 = 0.0194 / (0.0194 + 0.5313) = 0.0352

Hence, P(C|T) = 3.52%

3. The symptom is Pulmonary Oedema, ie : P(PO=true).

P(PO=TRUE) = $\sum P(N, V, C, PN, PO = TRUE)$

Expanding the above gives us:
P(PO=TRUE) = P(PO | V, PN) * P(V | N) * P(N) P(PN | C) * P(C) +
P(PO | V, PN) * P(V | ~N) * P(~N) * P(PN | C) * P(C) +
P(PO | V, PN) + P(V|N) + P(N) + P(PN | ~C) * P(~C) +
P(PO|V,PN) * P(V | ~N) * P(~N) * P(PN|~C) * P(~C) +
P(PO|~V,PN) * P(~V|N) * P(~N) * P(PN|C) * P(C) +
P(PO | ~V,PN) * P(~V|~N) * P(~N) * P(PN|C) * P(C) +
P(PO|~V,PN)*P(~V|N)*P(N) * P(PN|~C)*P(~C)+
P(PO|~V, PN) * P(~V|~N) * P(~N) * P(PN|~C) + P(~C) +
P(PO | ~V,~PN) * P(~V|N) * P(N) * P(~PN|C) * P(C) +
P(PO|~V,~PN) * P(~V|~N) * P(~N) * P(~PN|C) * P(C) +
P(PO | ~V, ~PN) * P(~V|N) * P(~N) * P(~PN | C) * P(C) +
P(PO | ~V, ~PN) * P(~V | ~N) * P(~N) * P(~PN|~C) * P(~C) +
P(PO|V, ~PN) * P(V|N) * P(N) * P(~PN|C) + P(C) +
P(PO | V, ~PN) * P(V|~N) * P(~N) * P(~PN|~C) * P(~C) +
P(PO | V, ~PN) * P(V | ~N) * P(~N) * P(~PN|C) * P(C) +
P(PO | V, ~PN) * P(V|N) * P(N) * P(~PN|~C) * P(~C)

Substituting the values of the probabilities from the probability table in the equation gives,
P(PO=True) = 0.229

P(PO=FALSE) = P(~PO | V, PN) * P(V | N) * P(N) P(PN | C) * P(C) +
P(~PO | V, PN) * P(V | ~N) * P(~N) * P(PN | C) * P(C) +
P(~PO | V, PN) + P(V|N) + P(N) + P(PN | ~C) * P(~C) +
P(~PO|V,PN) * P(V | ~N) * P(~N) * P(PN|~C) * P(~C) +
P(~PO|~V,PN) * P(~V|N) * P(~N) * P(PN|C) * P(C) +
P(~PO | ~V,PN) * P(~V|~N) * P(~N) * P(PN|C) * P(C) +
P(~PO|~V,PN)*P(~V|N)*P(N) * P(PN|~C)*P(~C)+
P(~PO|~V, PN) * P(~V|~N) * P(~N) * P(PN|~C) + P(~C) +
P(~PO | ~V,~PN) * P(~V|N) * P(N) * P(~PN|C) * P(C) +
P(~PO|~V,~PN) * P(~V|~N) * P(~N) * P(~PN|C) * P(C) +
P(~PO | ~V, ~PN) * P(~V|N) * P(~N) * P(~PN | C) * P(C) +

P(~PO | ~V, ~PN) * P(~V | ~N) * P(~N) * P(~PN|~C) * P(~C) +
P(~PO|V, ~PN) * P(V|N) * P(N) * P(~PN|C) + P(C) +
P(~PO | V, ~PN) * P(V|~N) * P(~N) * P(~PN|~C) * P(~C) +
P(~PO | V, ~PN) * P(V | ~N) * P(~N) * P(~PN|C) * P(C) +
P(~PO | V, ~PN) * P(V|N) * P(N) * P(~PN|~C) * P(~C)

Substituting the values of the probabilities from the probability table in the equation gives,
P(PO=False) = 0.771

The two probabilities add up to 1.

Hence, P(PO=true) = 0.229

Question 2:

Spect Heart Result:

**********Accuracy************************

Prediction accuracy 79.63%.

**********Confusion Matrix***************

True Negatives : 7
True Positives : 36
False Negatives : 7
False Positives : 4

Mushroom Result:

**********Accuracy************************

Prediction accuracy 99.73%.

**********Confusion Matrix***************

True Negatives : 695
True Positives : 431
False Negatives : 0
False Positives : 3

Expected Model file and Result files are generated as part of the assignment as mentioned in the Instructions.

Explanation of the model is provided in the “mode” file.

Question 3:

Note:

1. **Kindly provide the full path to the author directories.**
2. **If the code gives an exception, try running it again. Since it picks a random unigram, and then the next word using bigram, there is a possibility that this bigram is not present as a trigram.**
3. **The model runs fast. It’s the probability file generation that takes time because of formatting. (1min)**

Authors Selected:

Samuels Hopkins Adams
Burton Egbert Stevenson

Report for sentence generation:

The below is a sample from result file using extra credit validation:

Sentence: could bring something pink wear preferably coral came next time see without overcoat ll order hair shirt ve never seen

Sentence Probability Distribution:

```
Unigram Probability for word [could] is 0.0031054473354211395.

Bigram probability for word [bring] given the first word from unigram [could] is 0.0021141649048625794.

Trigram probability for word [something] given words [('could', 'bring')] is 1.0.
Trigram probability for word [pink] given words [('bring', 'something')] is 1.0.
Trigram probability for word [wear] given words [('something', 'pink')] is 1.0.
Trigram probability for word [preferably] given words [('pink', 'wear')] is 1.0.
Trigram probability for word [coral] given words [('wear', 'preferably')] is 1.0.
Trigram probability for word [came] given words [('preferably', 'coral')] is 1.0.
Trigram probability for word [next] given words [('coral', 'came')] is 1.0.
Trigram probability for word [time] given words [('came', 'next')] is 0.3333333333333333.
Trigram probability for word [see] given words [('next', 'time')] is 0.3333333333333333.
Trigram probability for word [without] given words [('time', 'see')] is 0.25.
Trigram probability for word [overcoat] given words [('see', 'without')] is 1.0.
Trigram probability for word [ll] given words [('without', 'overcoat')] is 1.0.
Trigram probability for word [order] given words [('overcoat', 'll')] is 1.0.
Trigram probability for word [hair] given words [('ll', 'order')] is 1.0.
Trigram probability for word [shirt] given words [('order', 'hair')] is 1.0.
Trigram probability for word [ve] given words [('hair', 'shirt')] is 0.5.
Trigram probability for word [never] given words [('shirt', 've')] is 1.0.
Trigram probability for word [seen] given words [('ve', 'never')] is 0.25.
```

Sentence Probability (Evaluating with books from other author): 0.0031054473354211395 * 0.0021141649048625794 * 1.0 * 1.0 * 1.0 * 1.0 * 1.0 * 1.0 * 1.0 * 0.3333333333333333 * 0.3333333333333333 * 0.25 * 1.0 * 1.0 * 1.0 * 1.0 * 1.0 * 0.5 * 1.0 * 0.25 = 2.2796624202938832e-08

Sentence Probability with other model: 0.005169222355440567 * 0.004069175991861648 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 0.2727272727272727 = 5.736675137823621e-142

1. Analyzing the sentence,

could bring something pink wear preferably coral came next time see without overcoat ll order hair shirt ve never seen

3 subparts:

1. **could bring something pink wear preferably coral**
2. **came next time see without overcoat**
3. **ll order hair shirt ve never seen**

It is evident that the generated sentences can be split into three sub sentences, which seem semantically correct (since stop words are missing they don't look grammatically).

This is happening possibly for the below reason:

Initially, a unigram is selected. Then the bigram selection also holds the sentence's semantic properly. As we move towards trigrams for sentence generation, we are ought to receive the next word which completely changes the context of the sentence. Since the textual data is not enormous which is being used for sentence generation it is highly probable that the next work for a bigram in a trigram means something completely different.

Consider the below sequence:

```
Trigram probability for word [something] given words [('could', 'bring')] is 1.0.
Trigram probability for word [pink] given words [('bring', 'something')] is 1.0.
Trigram probability for word [wear] given words [('something', 'pink')] is 1.0.
Trigram probability for word [preferably] given words [('pink', 'wear')] is 1.0.
Trigram probability for word [coral] given words [('wear', 'preferably')] is 1.0.
Trigram probability for word [came] given words [('preferably', 'coral')] is 1.0.
Trigram probability for word [next] given words [('coral', 'came')] is 1.0.
```

Here from the sentence “could bring something pink preferably coral **pink**” we could have expected word “pink” after “coral”, but the trigram “preferably coral” could have two words following it [“pink”, “came”] in a completely different context and from there again a meaningful sequence is generated for next 5-6 words and then again context changes.

Picking the bigram from the distribution here is the what is seen.

```
('preferably', 'coral'): { 'count': 2,
                         'list': ['came', ‘pink’],
                         'words': { 'came': { 'count': 1,
                                              'probability': 0.5},
                                    ‘pink’: { ‘count’: 1,
                                              ‘probability’: 0.5}}
```

2. Analyzing low probability when evaluating from other model:

Sentence Probability with other model: 0.005169222355440567 * 0.004069175991861648 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 1e-08 * 0.2727272727272727 = 5.736675137823621e-142

As evident from the expression, most of the bi-grams are not present in the second author’s book to generate the 3rd word. This is possible if the author’s picked write different genres or their style of writing is very different.

3. Analyzing high probabilities of words selected using tri-grams:

```
Trigram probability for word [something] given words [('could', 'bring')] is 1.0.
Trigram probability for word [pink] given words [('bring', 'something')] is 1.0.
Trigram probability for word [wear] given words [('something', 'pink')] is 1.0.
Trigram probability for word [preferably] given words [('pink', 'wear')] is 1.0.
Trigram probability for word [coral] given words [('wear', 'preferably')] is 1.0.
Trigram probability for word [came] given words [('preferably', 'coral')] is 1.0.
Trigram probability for word [next] given words [('coral', 'came')] is 1.0.
Trigram probability for word [time] given words [('came', 'next')] is 0.3333333333333333.
Trigram probability for word [see] given words [('next', 'time')] is 0.3333333333333333.
```

In the above seq, all the third words selected have high conditional probabilities since there are very fewer bigrams occurring together very frequently.

So the possibility of bigram ‘came next’ occurring together a lot and giving multiple 3rd words is very small considering the size of the data set.