HW 3: Question 1 (K-Means):

To implement K-Means, I wrote a Python script from scratch that performs K-Means and that stops after an appropriate number of iterations. This script is called kmeans.py. It gives output in the command line.

The final centroids are (1.5, 2.0) and (4.67, 4.33), which was achieved after two iterations. For my work intermediate steps, please see the program I wrote.

HW 3: Question 2 (ID3 Decision Tree):

I wrote another Python program from scratch that performs the ID3 decision tree and stops when it is appropriate to do so. Please see "ID3_clustering.py", which gives output in the command line.

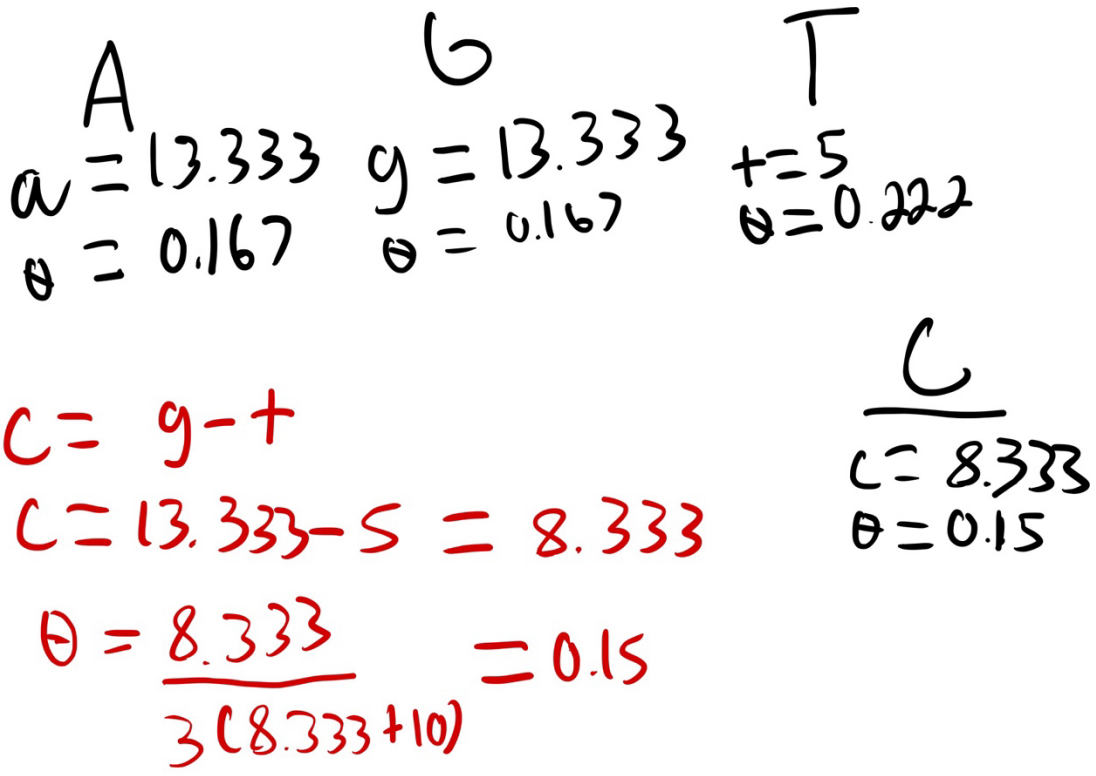
Below is the final decision tree.

For my work and intermediate steps, please see the program I wrote.

HW3: Question 4 (EM Algorithm):

I wrote a Python script to implement the EM algorithm. The script is called EM_algorithm.py, and it is attached in the submission. It gives output in the command line and to a CSV.

Below is work done by hand to finish up where my program left off. In the program (and continuing by hand), I initialized quantities of all bases as 10 and thus x & y each as 20.



For my work and intermediate steps, please see the program I wrote.

HW4: Question 2 (Hidden Markov Model):

Below is my work:

	State	Probability	Path	Probability		
	E(F,H)	0.5	**start -> E(F,H)**	0.5		
	E(B, H)	0.75	**E(F,H) -> E(B,H)**	0.4		
	E(B,T)	0.25	**E(B,H) -> E(B,T)**	0.6		
	E(F,T)	0.5	**E(B,T) -> E(F,T)**	0.4		
		0.046875		0.048		
	Total probability	=	0.046875	*	0.048	
	Total probability	=	**0.00225**			

HW4: Question 3 (mutual information):

First, I did preliminary calculations in Excel, as shown below:

		% both	% first	% second	num both
<25	High	0.5	0.5	0.625	4
<25	Low	0	0.5	0.375	0
≥25	High	0.125	0.5	0.625	1
≥25	Low	0.375	0.5	0.375	3
>50K	High	0.25	0.625	0.625	2
>50K	Low	0.375	0.375	0.375	3
<50K	High	0.375	0.625	0.625	3
<50K	Low	0	0.375	0.375	0
F	High	0.375	0.5	0.625	3
F	Low	0.125	0.5	0.375	1
M	High	0.25	0.5	0.625	2
M	Low	0.25	0.5	0.375	2

Then, I wrote a Python script called “mutual_information.py” to handle the rest. The Python script gives command line output, which is below:

```
roberthart@Methane Desktop % python3 mutual_test.py

        ********
        age
            mutual information = 0.5487949406953986


            chi squared = 4.8
        ********


        ********
        income
            mutual information = 0.34758988139079694


            chi squared = 4.66
        ********


        ********
        sex
            mutual information = 0.0487949406953985


            chi squared = 0.5333333333333333
        ********
```

All code & information on my GitHub: https://github.com/BinaryBrawler/MLHW