Assignment 4 Report **Anish Banerjee** 2021CS10134
COL333: Principles of Artificial Intelligence **Piyush Chauhan** 2021CS11010

Medical Diagnosis using Bayesian Networks

We have implemented the Expectation-Maximization algorithm as part of this assignment. You can access our code by following this link to our GitHub repository.
Initially, we organized our code into distinct files to enhance accessibility. Additionally, we utilized `unordered_maps` for the variables: `Names_to_index`, `Values_to_index` and `variable_nvalues`. By employing these structures, rather than iterating through the entire Bayesian network, we were able to retrieve the necessary values in $O(1)$. This experience underscored the insight that, particularly when dealing with extensive datasets, even minor optimizations can lead to substantial improvements in efficiency.
The basic idea behind our approach can be divided into three steps:

1. **Expectation:** In this phase, we iterate through each row in the dataset to determine the most likely value for the missing element. Subsequently, the dataset is updated, setting the missing element to the value associated with the highest probability.

2. **Maximization:** During this step, we compute the CPT values based on the dataset, employing a counting method. These counts undergo smoothing and normalization in the subsequent step.

3. **Normalization:** Our normalization process involves handling zero counts, applying smoothing factors, and averaging values for unreliable columns based on their dependencies with parent variables.

   The function first computes the sum of probabilities for each value, maintaining a `zeroflag` to check if any of the counts were 0. The smoothing factor was figured out by us by hit and trial:

   ```
   if (sum == 0.0f)     //  If the sum itself is zero then the sf= 1
       smoothing_factor = 1.0f;
   else if (zeroflag)  // Otherwise, if some counts are zero, use smaller sf
       smoothing_factor = ((sum) / (nvalues)) * 0.04f;
   else                // Smoothing not needed
       smoothing_factor = 0;
   ```

   Then we applied a basic normalization:

   ```
   newCPT[i][j]=(newCPT[i][j]+smoothing_factor)/(sum+nvalues*smoothing_factor)
   ```

   We call a column *unreliable* if it has sum 0. In the absence of counts for this column in the data, we cannot depend on the dataset to provide a reliable value for it. We identified unreliable columns in the `reliableCols` vector. Next, we calculated the specific combination of parent variable values for the current unreliable column and stored them in the `parent_values` vector.

   Now we perform additional averaging for these columns based on the reliable ones. For each parent variable and its possible values we calculated an index corresponding to the combination of parent variable values. If the column at this index is reliable it is added to a `rel_cols` vector. Then, for each value of the unreliable column, we calculated the average of corresponding values in the reliable columns. The average is then used to update the value in the unreliable column using the basic normalization. This step, surprisingly, led to a major improvement in the results.

Other approaches tried (did not work)

- Transitioning from Hard EM to Soft EM did not yield any notable enhancement in our results.
- While exploring alternatives to random initialization for CPTs, we considered using a counts table based on the frequencies of datapoints in the initial data. Our expectation was that this approach, given only one missing datapoint per row, would provide a more accurate estimate of probabilities and expedite convergence. Unfortunately, this strategy did not lead to any improvement in results.