1. Naive Bayes:

For naive Bayes classifier, it assumes conditional independence, which states that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. Thus for:

$$\underbrace{P(c|x)}_{\text{posterior probability}} = \frac{\overbrace{P(x|c)}^{\text{likelihood}} \cdot \overbrace{P(c)}^{\text{class prior probability}}}{\underbrace{P(x)}_{\text{predictor prior probability}}}$$

and $P(c|x) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$

For the HW5 dataset. we have 18 predictors (pstatus, medu, fedu, traveltime, studytime, failures, activities, nursery, higher, internet, romantic, famrel, freetime, goout, dalc, walc, health, absences), and 1 class (grade);

Step 1: The pstatus data are 'A' or 'T', and were converted to 0 and 1 respectively;
The activities data were 'yes' or 'no', and were converted to 0 and 1 respectively;
The nursery data were 'yes' or 'no', and were converted to 0 and 1 respectively;
The higher data were 'yes' or 'no', and were converted to 0 and 1 respectively;
The internet data were 'yes' or 'no', and were converted to 0 and 1 respectively;
The romantic data were 'yes' or 'no', and were converted to 0 and 1 respectively;
The python code for these conversions is HW5_naive_bayes.py.
The converted files are train.txt and test.txt.

Step 2: We first calculate the 21 prior probabilities for the class variable grade, and the vector (1X21) was saved as prior1; ← using the train.txt;

Step 3: We then calculate the likelihoods for each of the 18 predictors against each value of the class variable grade, and the matrix (138 X21) was saved as count1; ← using the train.txt;

Step 4: We then apply the prior1 and count1 to all the samples in the test.txt, and determine the maximum posterior probabilities for each sample and record the corresponding classification for the grade variable.
The result is saved in a vector (1X test sample size) as sample1;

Step 5: We then compare the predicted grade classifications with the observed ones in the test sample.

The accuracy is 9%, indicating that the performance is unreasonably low. This most likely is due to the fact that for this dataset, the conditional independence probably is not valid.

The matlab code is HW5_part1.m.

2. Structure Learning:

For the train.txt dataset, we have in total 19 variables (18 predictors and 1 class);

We first will calculate the empirical mutual information between each possible pair (171 pairs in total)

As an example; pstatus $\in \{0,1\}$, modu $\in \{1,2,3,4\}$;

Thus: Step 1: count singletons:

$N_{pstatus=0}$, $N_{pstatus=1}$, $N_{modu=1}$, $N_{modu=2}$, $N_{modu=3}$, $N_{modu=4}$;

Step 2: count pairwise:

$N_{(pstatus=0,\,modu=1)}$, $N_{(pstatus=0,\,modu=2)}$, $N_{(pstatus=0,\,modu=3)}$, $N_{(pstatus=0,\,modu=4)}$;

$N_{(pstatus=1,\,modu=1)}$, $N_{(pstatus=1,\,modu=2)}$, $N_{(pstatus=1,\,modu=3)}$, $N_{(pstatus=1,\,modu=4)}$;

Step 3: calculate the mutual information (MI)

$$I_{pstatus,\,modu} \propto \sum_{pstatus,\,modu} \left(N_{(pstatus,\,modu)} \cdot \log \frac{N_{(pstatus,\,modu)}}{N_{pstatus} \times N_{modu}} \right);$$

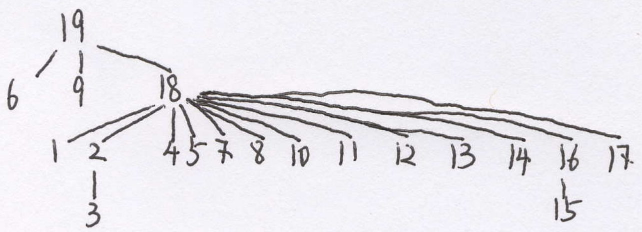
Once we collect all the MI between all pairs of nodes, we will use the Matlab minspantree to identify the tree T that maximizes $\max_{T} \sum_{i,j} I(x_i; x_j)$;

Note, since matlab only has minspantree, we will negate all the MI values, and then apply the minspantree function;

We determined that this chow-Liu tree has nodes and edges as:

1-18, 2-3, 2-18, 4-18, 5-18, 6-19, 7-18, 8-18, 9-19, 10-18, 11-18, 12-18, 13-18, 14-18, 15-16, 16-18, 17-18, 18-19;

One way to plot is as (use 19 as root):



note: for simplicity, all 1, 2, ⋯ 19 here denote $x_1, x_2 \cdots x_{19}$;

In this case, when trying to maximize the joint probability P;

$$P \propto P(19) \cdot P(6|19) \cdot P(9|19) \cdot P(18|19);$$

$$\propto \frac{P(6,19) \cdot P(9,19) \cdot P(18,19)}{P^2(19)}$$

Thus, we first use the train.txt to count M6_19, M9_19, M18_19 and M19, respectively;

Finally, we apply these maximization procedures on each of the test sample, and saved the predicted grades as result1 (vector: 1 X size of test.txt);

And then, we compared the predicted grades (result1) with observed values (result2).

The accuracy is 20%. We can see that the Structure Learning method produced a higher accuracy compared to that of naive bayes (9%).

The matlab code is HW5_part2.m.