Modeling and Analysis of Admission Process

Model Setup: For the first step of the exercise, I tried to generate a reasonable graphical model where each instance in the data corresponds to a student and an admission-relevant label regarding that student. A diagram of the DAG I generated along with a detailed description of the variables and their values can be found in the *generateData.ipynb* file. The 10 features I created were sex, diversity, legacy (whether the student has legacy for admission), SAT, GPA, number of extracurricular events, TOEFL, number of AP tests taken, a letter or recommendation score (a continuous score where 0 is the worst letter and 1 is the best letter), and number of subject tests taken. For the features, I tried to mimic appropriate values that could be realistic. For example, features that involved a count, I used an ordinal node so that it enforces the package to take integer values. For continuous features such as SAT score or GPA (whose values approximately follow a normal distribution), I chose to use Gaussian nodes.

In the DAG I created, I had admission be dependent on all 10 features, sex affecting SAT and GPA, and diversity affecting TOEFL. The reason I had some features like sex affecting the admission result along with SAT and GPA, is because I thought that these dependencies could exist in real admission data. In addition, I also thought that when validating the results, it would be interesting to compare the results of a classifier that is trained with and without the sex feature. Otherwise the rest of the dependencies I had in the DAG were dependencies that could exist in real-world admission data.

Limitations: One limitation of the package was that it could not enforce integer values or any lower/upper bounds from the Gaussian nodes. In other words for example, the SAT scores could take decimal values and because of the normal distribution some scores might be greater than the 2400 maximum. If desired, a possible future fix could be rounding the values that are lower than the minimum or larger than the maximum bound, they can be replaced respectively with the minimum/maximum bound value. Another limitation I noted while programming the DAG was that the package enforces categorical variables to strictly use one letter for the categories. If the categories are longer than one character the package will throw a key error.

Model Checking: After generating and saving the data, the next step was for me to validate the data to verify that it satisfies the properties of the DAG (this was done in *validateData.ipynb*). I first output some general EDA to verify that the ranges and values of the data were accurately generated from my DAG. To validate the independent categorical nodes in the DAG, I output the percentage of each unique value present in the data and checked if they were approximately equal (uniform as expected from DAG). Note that for the AP tests feature, since there was a probability associated with each range, and the scores were uniform within each range/bucket (as described in the header of *generateData.ipynb*), I first checked if the percentage of values that were in each range were approximately the same as specified in my DAG structure. Afterwards, I then checked to see if the percentage of each unique value in each range was uniform (approximately same percentage). For the letter of recommendation score feature, since it had continuous values I plotted a histogram and checked to see if the counts in each bin were approximately the same thus verifying the uniform distribution property of my DAG. For dependent continuous features (where parent is categorical), I first did a groupby on the parent feature in the DAG and then found summary statistics of each group. Using the outputted information, I then verified if the dependent feature followed the appropriate gaussian distribution (mean and standard deviation) for each group/category as specified in my DAG.

Machine Learning Testing: Finally after verifying this information, I wanted to check if the dependencies for example from sex to the admission label was incorporated in the generated data. Based on my DAG, I would expect that if we train a classifier with the sex feature and train another classifier without the sex feature, the accuracy would be approximately the same since gender is encoded in the SAT and GPA scores (which are still included when training the model). I chose to use logistic regression to try to predict the admission labels, and trained the two models as described above. I found that the accuracy for male and female with sex included was 0.6377 and 0.5429 respectively, and the accuracy without the sex feature was 0.6310 and 0.5025. The accuracies are approximately the same as we expected, again since even though we might think removing sex from the data during training would make the model fair, however we know from our DAG that sex affects SAT and GPA which are still included in the model.