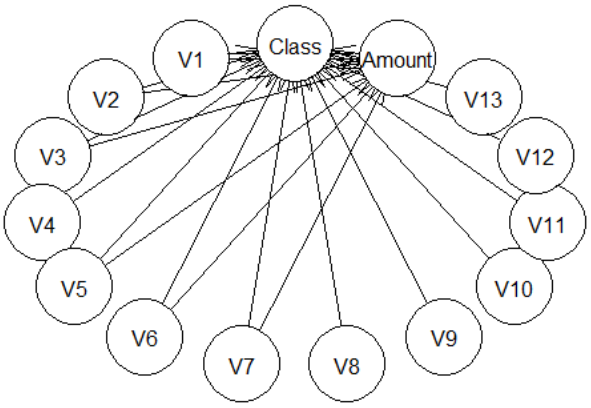
Methodology

In this section I will outline the complete methodology that I followed to produce my results. The data that I have used is titled "*Anonymized credit card transactions labeled as fraudulent or genuine"* and can be found at (https://www.kaggle.com/dalpozz/creditcardfraud). The dataset contains actual credit card transaction made by European cardholders in September 2013. The dataset contains 284807 total transactions, of which 492 are fraud cases (0.172%), and 284315 are non-fraud cases (99.828%). As you can see the dataset is highly class unbalanced, which is typical of a credit card fraud dataset. In terms of technologies used, I have used R to complete this project.

As previously mentioned, learning the structure of a Bayesian network is a NP-Hard problem. In order to learn the most optimal variation of the structure, I utilised a greedy search algorithm, known as hill climbing. The hill climbing algorithm iterates through varies Bayesian network structures, until it discovers the network with the highest statistically motivated score. After the structure of the network was determined, I manually removed edges between any of the principle components that are not correlated. In addition, I also added extra edges between the amount variable and the highest correlated principle components.



In order to perform Bayesian network inferencing on unseen data samples, I used the R package “BNlearn” and the cpquery function. The cpquery function estimates the conditional probability of the event given the evidence. This function uses a Monte Carlo simulation to estimate the conditional probability value. After experimenting with different iteration values, I determined that 10000 iterations provides an optimal result.

After computing the conditional probabilities of every unseen record in the testing test, the results were converted into binary predictions using a threshold value. After experimenting with different threshold values, I determined that any record with a conditional probability of greater than or equal to 0.3 would be considered as fraudulent. This threshold value was determined by selecting a value that would result in zero false positive predictions.

In order to compute the 10000 thousand iterations in parallel and thus reducing overall computational time, I used the R packages “parallel”, “doParallel”, and “foreach”. I executed the 10000 iterations of the cpquery function across three processing cores (~3333 iterations per core). I decided to parallelize across 3 processing cores, because the maximum number of cores that the R program can operate with on one machine is (total number of available cores -1).