NCAA Basketball Prediction:

Model: The first draft of the model consisted of using a perceptron and running the seasons A-Q through it to train the perceptron. From there, we randomized the remaining season and ran that through as testing data. Over the course of several iterations, the average and median accuracy was 67- 68%.

1. The model worked decently well, 67-68% of the time. We could not see any reason for when it produced outliers, but the model needs refinement so we can focus on that later. It was quick to perform, taking about a couple seconds to read in the .csv and getting the data processed and producing the result.
2. Feature generation was not too terrible, we used the win percentages of teams as well as the seed they teams were places in for the actual tournament for now. Generating the features took some time as we used Microsoft Excel to calculate win rates when they were not readily available. However, it worked well enough, only resulting in some out of memory problems with Excel.

Since then, we added opponent win % as a feature, to help separate the various win %. If a team had faced easier opponents, then that can mean their skill is not as high as their win percentage would indicate. We did this by using python to read in the win percentages of every team for each season, and go through their opponents and calculate the average win percentage. Win percentages as well as opponent win percentages are the strongest indicators of skill available to us with the training data. We do not necessarily know individual player strength, so we have to look at how the team performs as a whole.

Feature Selection: Feature selection was largely logic based. When examining two different teams, the main indicator of skill available to us was their win percentages on a season basis. As teams can greatly vary from season to season, there is very little correlation from one season to the next.

Loan Default Prediction:

Model: The first draft of the model involved splitting the training data into two subsets: no loss, and total loss. For each sub set the extremes (top and bottom 5%) were removed and the remainder of data was averaged into an average data point. We did not use some of the data in the training data (to be used for a testing set). A data point would be compared to each average, for each of the roughly 800 features; if it was closer to the no loss it received one point in that feature, if it was closer to the complete loss it lost one point for that feature. If the final tally was above zero, predict no loss; if it were below zero predict complete loss.

The final model used a modified version of the perceptron code used by the NCAA prediction. Surprisingly, (after over an hour of running the code) it ended up being more accurate than NCAA prediction.

1. Ignoring the percentage of loss, the algorithm correctly predicted whether the loan would default or not sixty percent of the time. The numbers produced [-400, 400] seemed to be random and could not be used to determine the percentage of default. Using this model also crashed our computers multiple times (and we have a Haswell processor with 16 GBs of RAM).
2. A lot of the features seem random and appear to not have any correlation with the amount of default. Using less features would make the algorithm faster (and easier to run) but (currently) we do not know of a good way of eliminating features with unsupervised learning.

Feature Selection: For this, given the large number of features given to us in the training data, we went through and found ones that would most likely have no impact (all being the same number or similar) and had gigantic gaps in values. From there, we went through and did some preliminary tests on various features in order to find the features that had the largest possible impact. We did not want to have too many features so we picked ones that seemed to affect the outcome on the loss. Due to this fact some human error may have been added (as we decided which features to use instead of letting a computer decide which features were most important).