I utilized pieces of Professor Rayid's "Magic Loop" and reworked it to fit my coding style and fit in some of the functions of the that I had previously written. First, I placed the grids in a second file, since they seemed to clog up space in the original file. I wrote a helper function to setup and write to the results dataframe within the loop. I also marked time at the beginning of running the model and the end, in order to determine the time elapsed. I replaced the split function from Pandas with my own, just because I already had it and because I knew how it worked and didn't seem to find any performance differences. For validation, I added the time elapsed, and recall and F1 scores at each k. I also included some print statements when each model finishes running, in order to notify the user of progress. This was handy for understanding on the fly which models were running quickly and which were not.

Expanding on my pipeline from last time, I added more methods for filling empty data to allow for categorical values in future data sets. I then wrote functions to plot histograms of each feature and a function to create a correlation matrix between features. I also wrote a piece to examine the missing values and determine if there is major differences in the mean of each feature for missing rows and all rows. This function notified me that there is a massive gap in the mean Debt Ratio of missing rows and all rows, with the missing row mean at 1673.39 and the mean for all rows at 353.00.

The correlation matrix indicated that there is significant correlation between each of the "NumberOfNDaysLate" features, though this is a bit intuitive and isn't very helpful. There was also a relationship between real estate loans and lines of credit. Otherwise, the matrix does not indicate a strong correlation between any of the other variables.

I started running my models by testing the test-grid and small-grid on my own computer. Both worked and the small grid took roughly ~7 hours the first time I ran all the way through. I tried to run the large grid, but after about 5 hours, it ended up bogging my computer down to unusable level and had not made it past the Random Forests models. Since I wanted to run this multiple times to iterate with new features, I spun up an EC2 instance with more RAM than my own computer. This ran the small-grid much faster, though it was still slow on the large-grid. Even after a few hours, it was still working through the Random Forests models, so I decided to kill the process and let it run the small-grid with code I had written in the interim. I would be interested in understanding more about exactly how quick/slow I can expect certain models to run given the size of a dataset, the model parameters and the specific hardware of the computer.

In terms of the actual results of the models, I found that gradient boosting and random forests delivered the higher area under the ROC curve. These models also ran reasonably fast in ~3-5 seconds. Given that my fastest models ran in less than a tenth of a second this is relatively 'slow', but on the other hand, the slowest models I ran took ~7 minutes. On my personal computer, some of these models took close to an hour, so I'm 3-5 seconds is not a bad speed in my opinion. The precision, recall and F1 values for these top gradient boosting and random forests models also tend to be higher than the average for each validation method.

The worst models under AUROC validation were also two gradient boosting models, though the bulk of the “bad” models for AUROC were actually K-Nearest Neighbors models. Some of these models performed similar to the models with the best precision, recall, F1 scores for k=10, but for the most part, K-Nearest Neighbors performed abysmally for most validation methods.

The top models for recall, precision and F1 were decision trees, which in some cases have perfect 1 values for recall. These models also tended to run the fastest, at least than a tenth of a second for each model. However, the area under the ROC curve for these models was worse than the average AUROC, which was 0.72. There is also a subset of the decision tree models that had exactly the same predictions, leading to the same AUROC and recall, precision and F1 scores. This also was matched by some of the AdaBoosting models, which utilized the decision tree classifier. I found this result interesting.