# Introduction

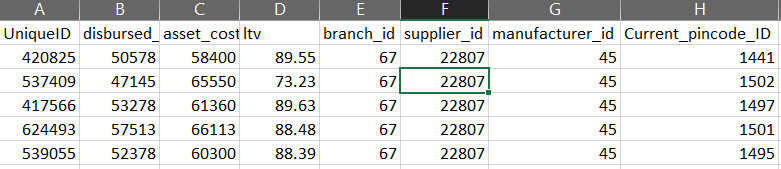
My Initial project was to determine the fail rate of hard drives with data provided by a cloud storage company called BackBlaze. The hard drives each have software in them that takes a snapshot of the day and stores it. This snapshot file contains about 70,000 to 80,000 rows of data per day for a whole year and by my calculations, a whole year of data would result in about 30,000,000 rows of data. So, this is huge data, but I found something quite interesting in my limitations in handling data. This years’ worth of data only contained about 13 times where the drive failed, so needless to say these are quite reliable hard drives. However, this was way to big for my computer to handle so I had to downsize.

I instead went with some data that is similar in that they are both classification problems, but instead of classifying whether a hard drive will fail I wanted to predict if the vehicle loan will default. I had to do some additional digging because I am not super familiar with loans and defaults, but I was able to fine enough information to understand how to prep the data.

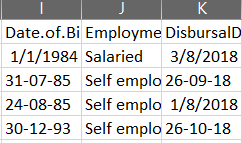
I used RapidMiner as the main tool for this project because 1) its quite simple and 2) I am not fluent in python enough to do any of the modeling I wanted to. With that being said I didn’t learn quite a bit about python in a rather short amount of time.

# Data Prep

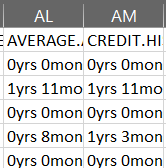
This dataset contains 41 rows and about 253,000 rows. It was still quite a bit of information and my computer ran for a total of 3 days to get all the models. The data was quite dirty and took most of my working time to clean it. Right away I see many ID fields telling me that this data came from a database but was not flattened before extraction.



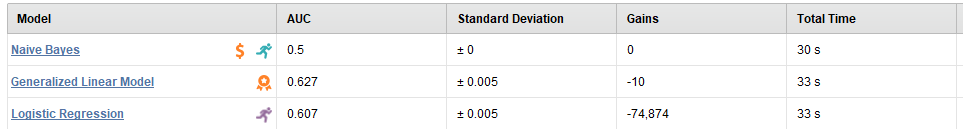
These are the first 8 columns and 5 of them are ID values. This made the data difficult to get a deep understanding so they were removed. The next problem I ran into were the date fields.

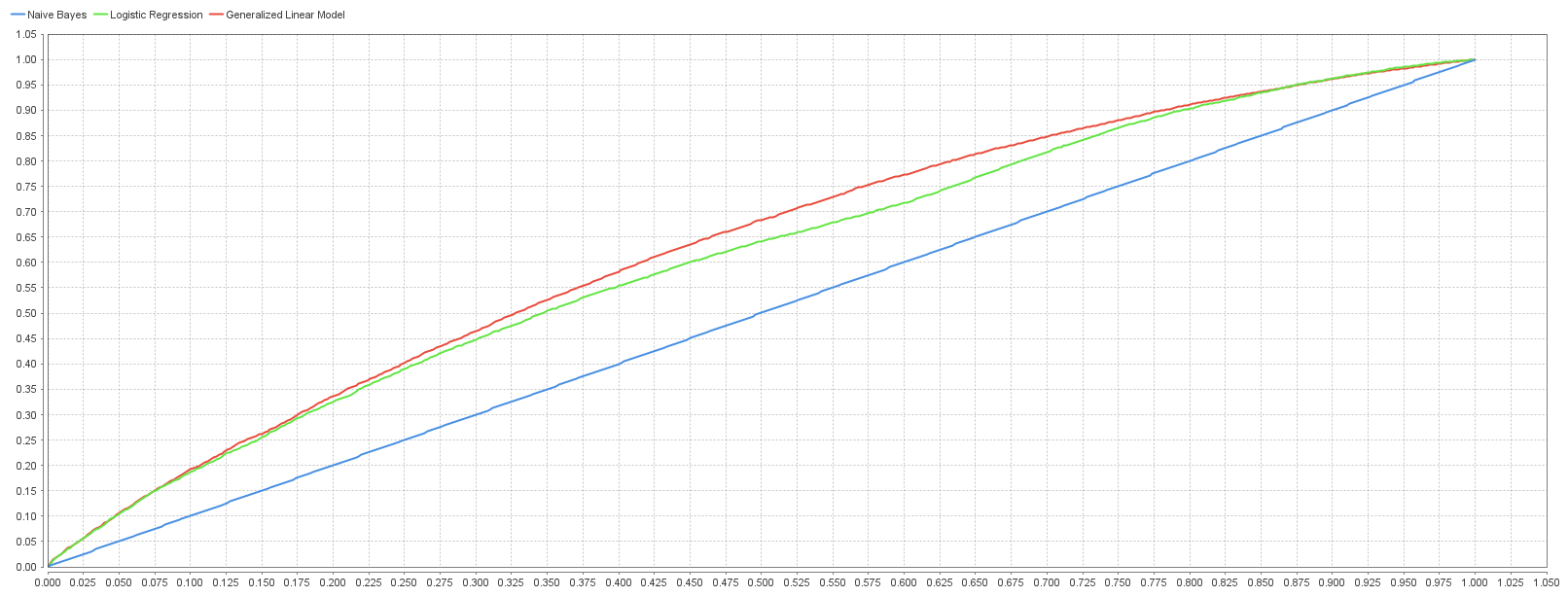


Even in the first 4 points you can see the problem. The dates were entered differently and ultimately removed from the dataset. The next 2 steps take requires some splitting and algebra



These were split and converted into days by a simple calculation to get better normalization of the days. There rest of the points were split, or dummy encoded if necessary. The data was normalized only for the third and fourth models. This was a lesson in futility as well as Occam’s razor. No matter what I did all the values seemed to be quite terrible in all of the models ran, but just as I am writing this I run a quick model and it is the simplest yet also the most accurate and a great deal fast.





# Models

These are the various model that were run on the dataset

## Model 1

This model was run on a Generalized Linear Model, Decision Tree, Random Forest, and Gradient Boosted Trees algorithms and all these returned error rates of about 21.7 %.

The other algorithms that this model was run on were Naïve Bayes, Logistic Regression, Deep Learning, and Support Vector Machines and returned errors. I expected some errors because the dataset used contained various categories and not all were numbers. Some were dates, some were numbers, some looked to be dummy encoded. Since it was a mixed bag of variable types, I simply ran an autocleaning step to only remove the useless variables from the dataset. After running this model, it is necessary to do more cleaning of the dataset

I allowed this model to run with feature extraction and auto feature engineering with no real necessity. The max number of features in this model was 3 with most of them containing 2. All the features weighted the most were the original features of the dataset.

Loan to value (ltv) parameter was weighted the highest at 0.096 and disbursed\_amount was the second with 0.071

This first run created 72,404 models with 16,556 feature sets and generated 934 features

This first run was quite terrible

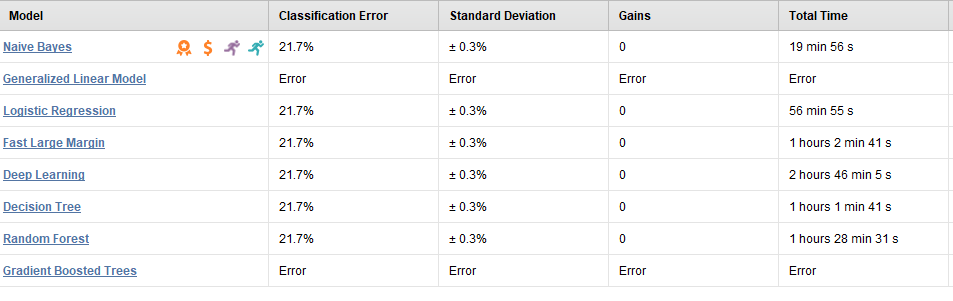
## Model 2

The second model was a little worse than the first run. To prep the data, I changed the years and months of credit age and account age to days. I dummy encoded a CNS score variable, which states how high a risk the person will be. With all the data prep I increase the number of columns from 41 to 53. This run had about the same results as the first one

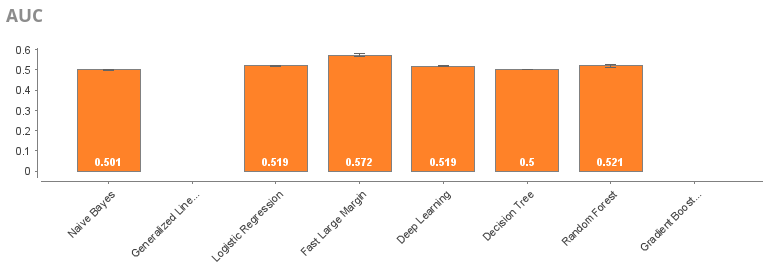
## Model 3

I used model 2 as the basis for this model. The difference being that all the values were normalized. This gave the dataset a better modeling ability. Model 1 and 2 ran 4 algorithms on the dataset while this one ran 6 of them. Noted above, model 1 created 72,000 models with almost similar results, while model 3 created 63,836 models with 17,333 feature sets. This tells me that the data prep was successful in achieving the almost the same, if not better, results than the first 2 models and with more prep it could be even more accurate.

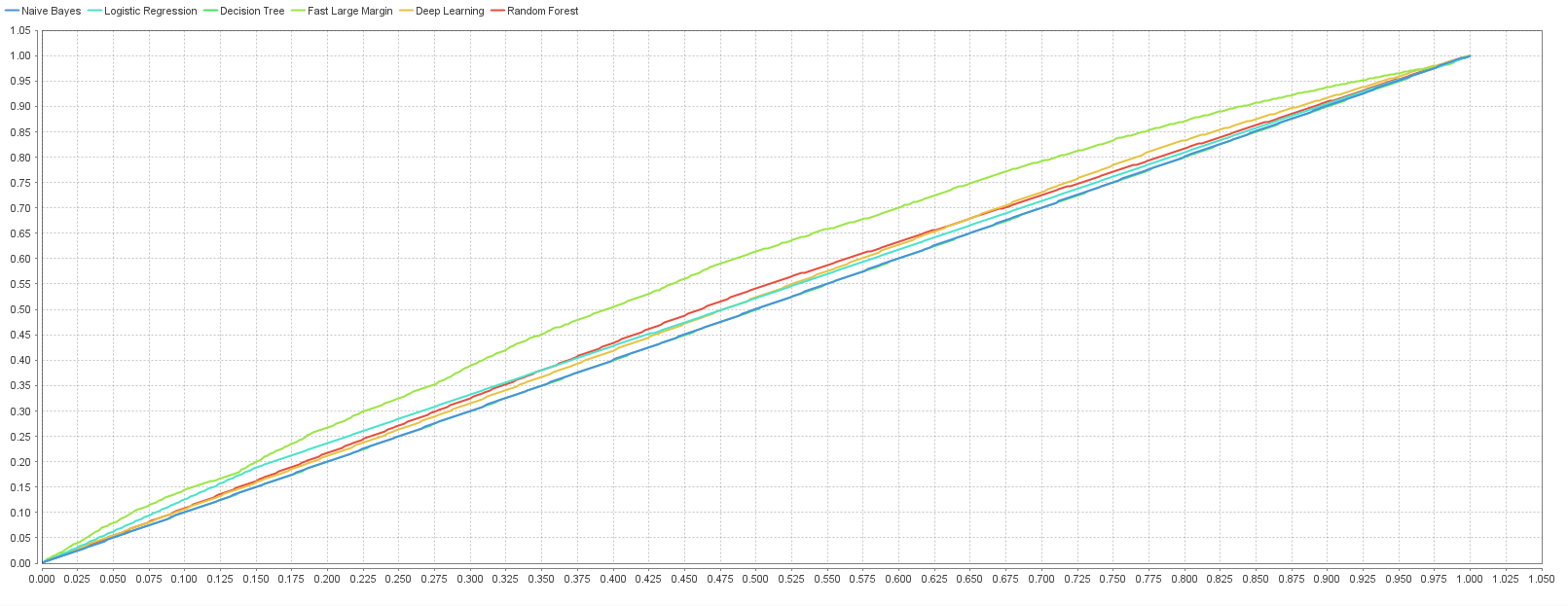
Model 1, 2, and 3 all had the same error rate of about 22%, but what is different is that run 3 was able to incorporate algorithms that the first 2 were not.



Looking at the AUC values we can derive some insights.



Right away we can see that there is room for much improvement, but Fast Large Margin was the most accurate while Naïve Bayes was the fastest and most cost efficient. This is confirmed by the ROC graph where green is Fast Large Margin and Naïve Bayes is in blue.



# Summary

As we all know, data preparation is the most time-consuming part, weeks or months long. The fun part, training and testing the model takes significantly less time. Preparing the data included creating new columns out of existing ones by multiplication or division, removing columns that had to great a variability or were mis entered, and splitting and dummy encoding the data. With the data cleaned it was time to created models. The first run was more error than good results. After the first run I went back to clean the data again this time creating more columns that the data originally had. The second run ended just as poorly as the first one did.

Model 3 was slightly better than models 1 and 2. The data this time was normalized along with all the previously mentions steps. This run still created many models but was 10,000 models less than the first 2.

The lessons learned here are data cleaning and prep are very important for accurate models. While I was never able to get any better that 22% error and .62 on the ROC curve the more data cleaning, I did the more algorithms were able to be used increasing from 3 to 6 algorithms. This all well and good, but we still need to consider Occam’s Razor, which states that sometimes the best models are the simplest models. Like I said, as I sit here are write this model running the creating the best results out of all and it’s the simplest one.