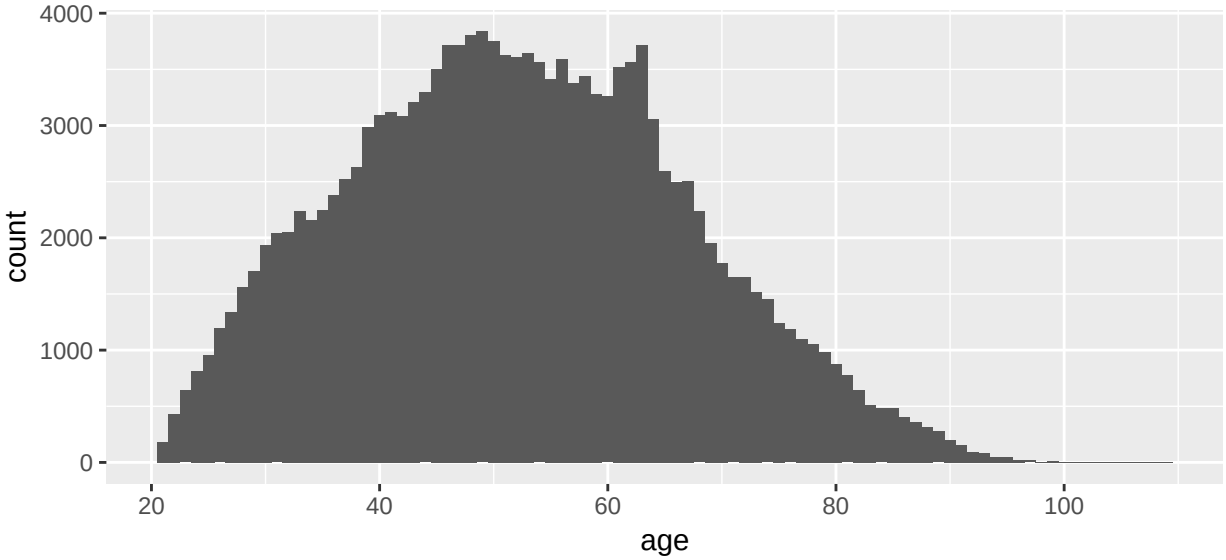
Kaggle Data Competition – Give Me Some Credit

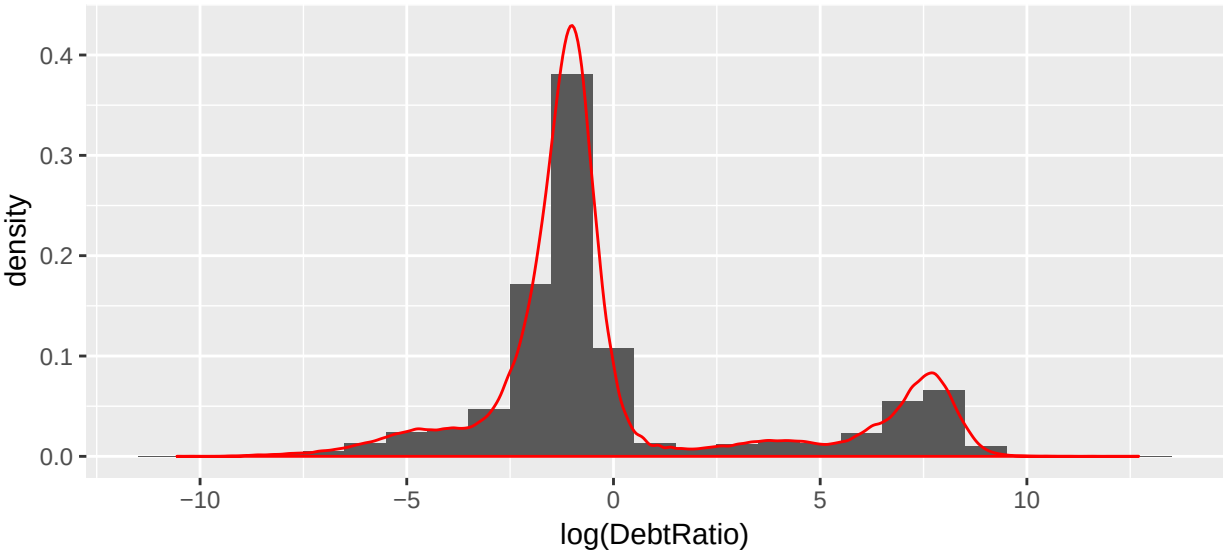
Nolan McCafferty, Jack Hanley, Jacob Niskey, Zach Senator

Data Exploration

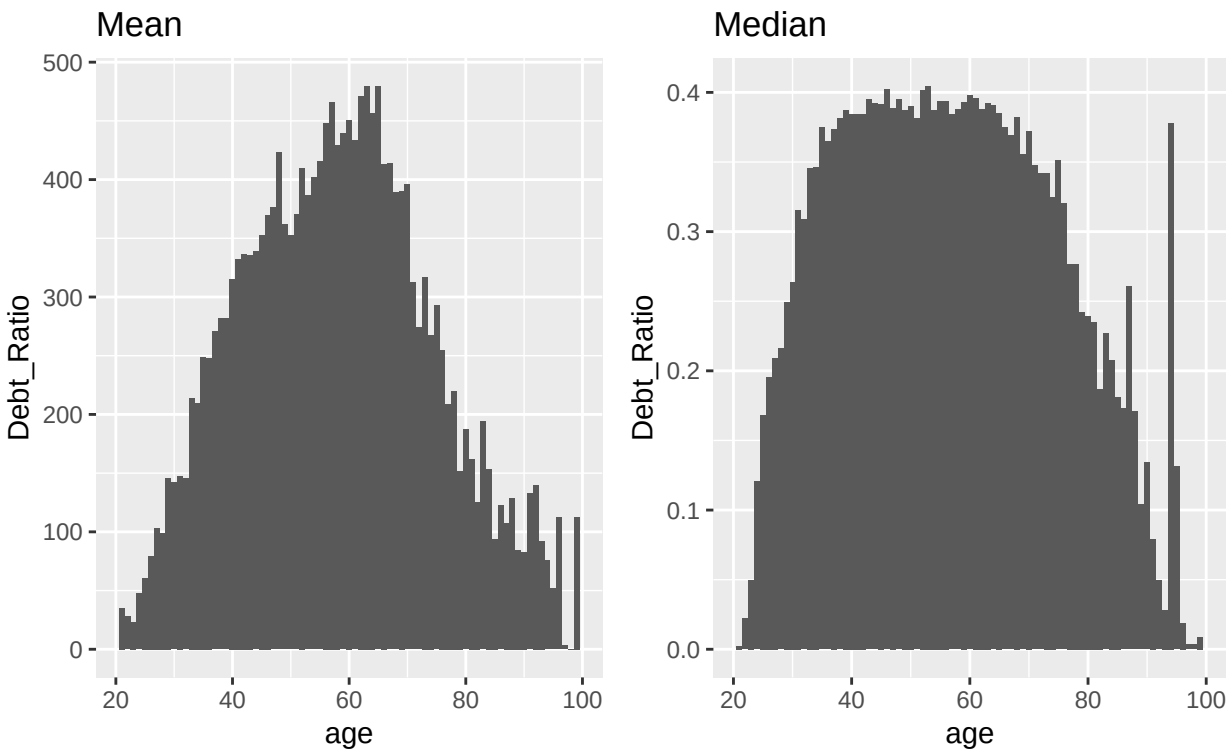
We started our project by looking at the data, and variables. The first thing we looked at was a histogram of ages. We noticed a somewhat normal distribution with a right skew, showing us that we were mainly lending to people between 30-70 years of age. The graph was increasing until about 60 years of age, and then started to decrease. This confirms our intuition that people start to take out loans to buy things such as cars and houses before they retire, and then settle with what they have when they are retired.



Next, we looked at DebtRatio.

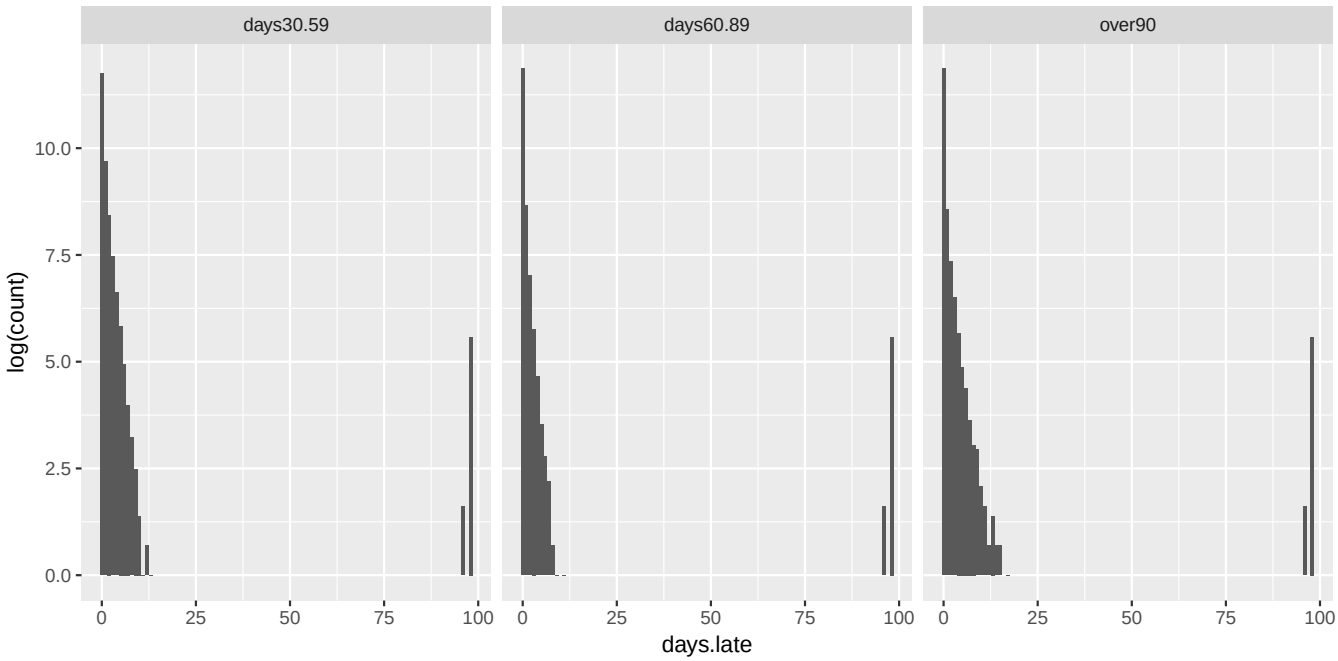


From analyzing the distribution of the log of DebtRatio above, we were able to draw some interesting conclusions. First, there was a spike right below 0, meaning lots of people barely had more income than debt. In other words they were making a tiny bit more money then they were spending on things like alimony, debt repayments, and living costs. To the contrary, there is another spike present at about 7.5, which corresponds to a debt ratio of 1800, which means that if you have more debt than you do income, you have a lot more debt than you do income. Further, the the debt is 1800 times the monthly income. We also explored age vs. DebtRatio:



This plot of age vs average debt ratio shows that the average ratio of debt to gross income steadily increases until around age 60. This is midly surprising given that students just graduating college, in their early 20s, would be expected to have lots of debt and a relatively low income. I would have expected more of a spike on the far left of this distribution with a slow decline as age increases, but I guess it makes sense that middle-aged people are more likely to have accumlated debt through the years of working a low income job and trying to support a family. An interesting thing to point out about this graphic is that if the median statistic was used instead of the mean, all of the values would be less than one because most people have a gross income that is larger than their debt (this is shown on the second plot). This second plot has a similar shape, except with more of a plateau from 35 to 65 at a debt ratio of 40%.

Next, we examined the number of days late variables:



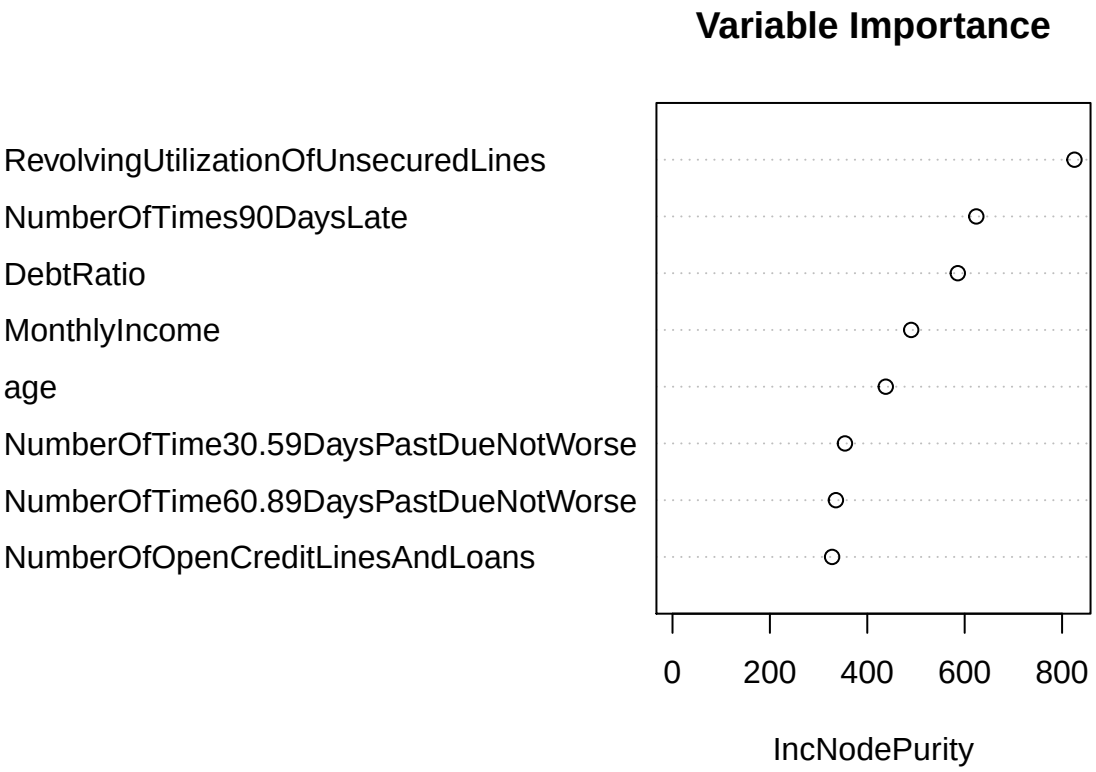
We can see that zero is the most popular number for days late and as the days increase the likelihood decreases. However, there are these very strange spikes on all three graphs at 96 and 98 days late. We have no way to explain this phenomenon, so we decided to change all of the 96 and 98 values to the smallest number of days late that has no observations. For example, for the over 90 days late variable we converted these outlier observations to 18 days late.

Model Building

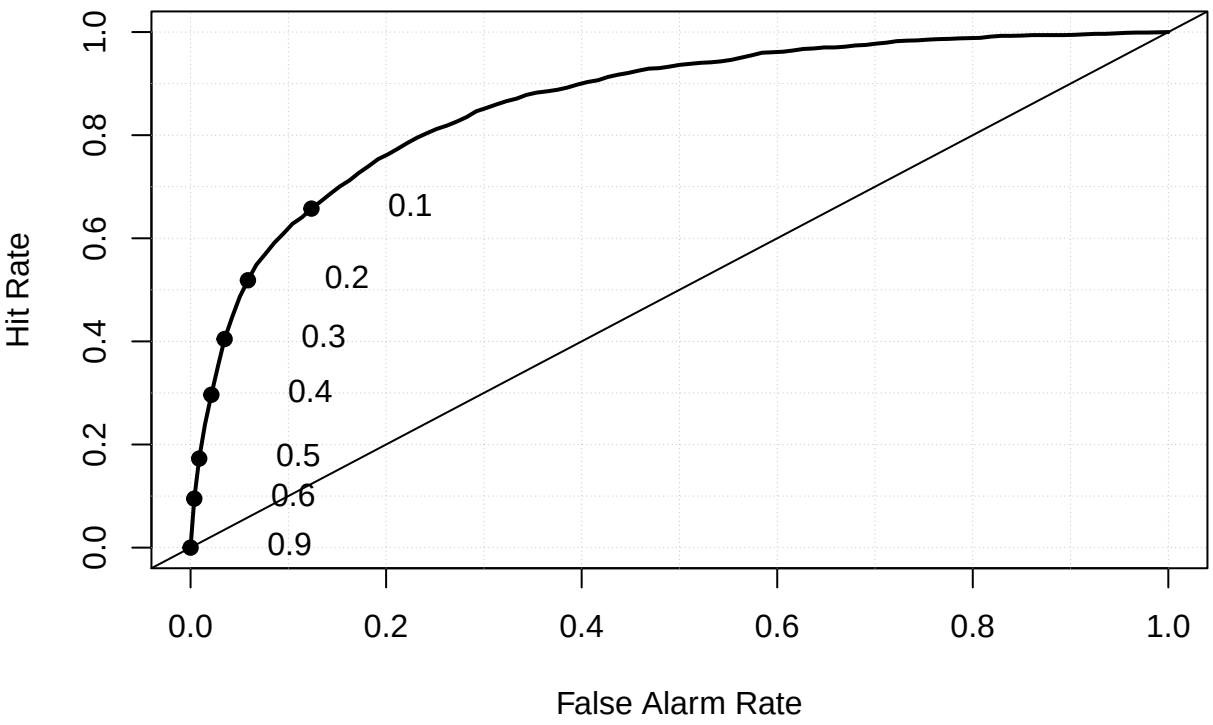
Random Forest

The median was the most successful method of imputation that we used to impute the missing monthly income values. We also tried to impute using the random forest implementation but that did not give us as good of predictions. Since there are not very many variables to consider, especially since we removed number of dependents and number of real estate loans or lines because they did not add value to the model, we were easily able to identify the optimal value of the `mtry` parameter for the random forest as 2 by checking the few possibilities. Next, we cross-validated to find the optimal value of `nodesize`, using k-fold cross-validation with 6 folds, which we found to be 7. To evaluate each model, we predicted the validation data and check the AUC of the ROC plot.

We built our random forest model using the imputed training data and got the following variable importance plot:



The plot shows that the most important varibale is the Revolving Utilization Of Unsecured Lines (RUUL). The definition of RUUL is the total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits. This makes sense because RUUL essentially evaluates how well somebody delas with small scale loans from credit cards and we are trying to predict if someone will default on their bank loan. The ROC plot from the validation predictions is below:



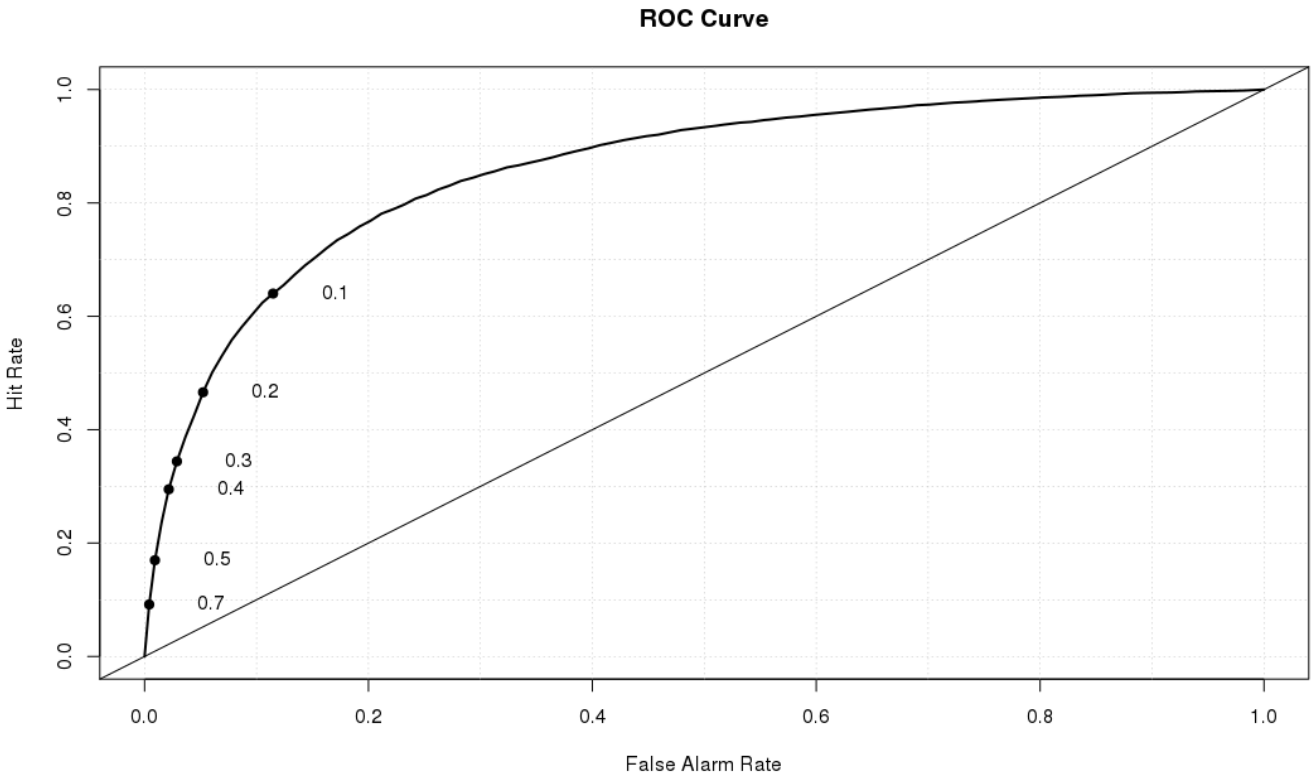

```
## [1] 0.8628996
```

Now, we will evaluate our best random forest model on the test data. To deal with the NAs in the monthly income variable of the test data, we used the same method as with the training data, replace with the median. It is interesting to note that there was no change in our model predictability when we cleaned the NumberOfDaysLate variables in our test data. However, the predictive power of our model did increase when we cleaned those same columns in the training data. Finally, we predicted the test data and submitted our predictions to Kaggle.

XGBoost

Another model we considered was boosting, primarily through the R package XGBoost. As a concept, boosting essentially uses weak learners that can only fit signal to generate a smarter learner. In addition to being successfully used in numerous other Kaggle competitions, XGBoost also does its own data imputation, allowing us to avoid the computationally heavy imputation methods from the randomForest package. Unfortunately, XGBoost does not explicitly state how it imputes data. With this in mind, along with the success of median imputation in the randomforest algorithm, we decided to do median imputation for both the training and test data. Additionally, the default objective of XGBoost is linear regression, which unfortunately produced some predictions over 1 and less than 0 for probabilities; this was eventually changed to logistic regression (still using XGBoost). To generate the best model, we first looped over the three different learners you can choose from in XGBoost, gblinear, gbtree, and dart. Both gbtree and dart are tree-based learners, while gblinear is a linear learner. We then looped over an array of 9 cost parameters ranging from 5 to around 316. To pick the best cost/learner combination, we performed k-fold cross validation with 6 folds and ran models with 50 rounds. These were chosen to reduce the computational strain; it's very likely that if we had either more time or computing power, we could have produced a better model by using more rounds and validating over more folds.

The ROC plot below looks very similar to the ROC plot for our random forest model.



Through this process we ultimately ended up selecting a dart learned model with a cost parameter equal to about 12. The mean AUC over the cross-validation process was 0.859, with a max of 0.861.

Results

These are the results we got after submitting our best random forest model to Kaggle:

Submission and Description	Private Score	Public Score
predictions.csv a few seconds to go by Nolan McCafferty final submission	0.862372	0.857990

A private score of 0.86237 puts us in... 485th place in the competition. Ouch.

We then trained an XGBoost model with the best parameters on the entire training set with 100 rounds. The result was a private score of 0.865845 and a public score of 0.859612. This gets us to... 185th place, not too bad.

The XGBoost algorithm outperformed the random forest model for this dataset.