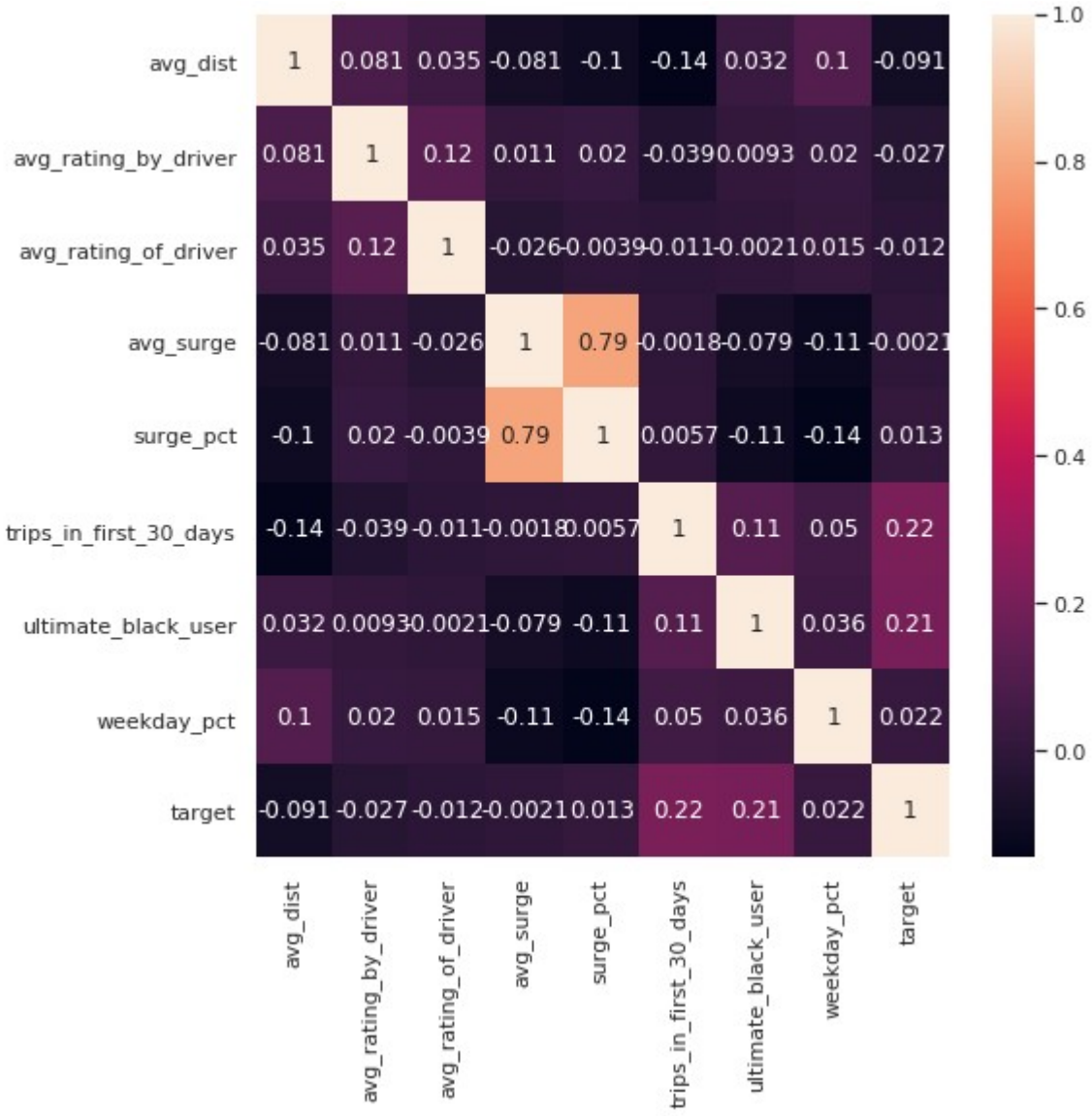
1. I made histograms of all the raw features to get an idea of what the data look like. Many of the features are heavily skewed and should be transformed before being used in a linear model. However, for tree models, transformation won't be necessary. Three of the features — avg_rating_by_driver, avg_rating_of_driver, and phone — have missing values. Since "phone" is an unordered categorical feature, the missing values can simply be a separate category. The others with missing values are numerical, and to use them in a linear model, it would be necessary either to drop the cases with missing values or to impute missing values. However, many tree-fitting algorithms are already set up to handle missing values. I also confirmed that all dates are consistent with the description of the data (All signup dates are in January 2014, and all last trip dates are January or later, with the last on July 1.)

It's potentially instructive (and pretty) to look at the correlations among the features:



Most of the correlations are low, with the exception of surge_pct and avg_surge, which for obvious reasons are highly correlated. ("Target" is an indicator for the retention criterion—users whose last trip was on or after June 3.)

Of all 50,000 users in the sample, 64.2 percent were retained.

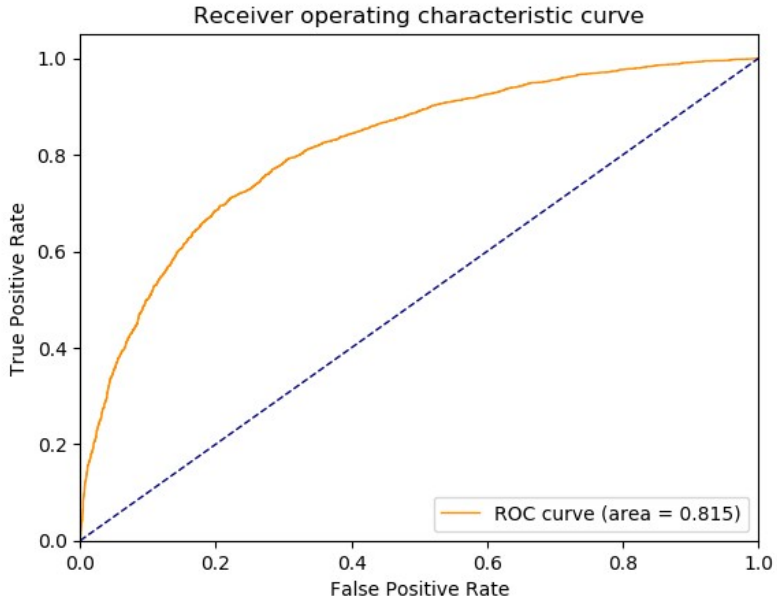
However, there was another data issue I needed to address before proceeding to predictive modeling. It seemed likely to me that "trips in first 30 days" would be an important predictor. However, using it as a predictor on the full sample would create a data leak. It is likely that, for some of the users who had relatively few trips in their first 30 days, the reason they had few trips is that they left Ultimate early. That is, they were not "retained" even through the first 30 days. Since retention is what we're trying to predict, it would be cheating to use as a predictor something that directly reflects lack of retention.

I chose to eliminate from my sample those users whose last trip was within 30 days of their signup date. (If one were to pursue this problem further, one could create a separate model for those users, without using the "trips in first 30 days" feature.) This decision had the pleasant side effect of producing a balanced sample: among users who stayed at least 30 days, 51.5 percent were subsequently retained. I like balanced samples particularly because they allow one to express model performance using the intuitively straightforward accuracy metric.

2. Since the charge was merely to "build a predictive model," not to "build a really awesome predictive model that will make better predictions than anyone expected," and not to "build a predictive model of which every aspect has a straightforward and easily expressed interpretation," I chose the approach I felt was most likely to produce a well-performing model with least effort—namely the gradient boosted decision tree approach (using the LightGBM package). If I were prioritizing interpretability, I might have chosen a logistic regression model instead, since its coefficients allow for a more straightforward interpretation. If I were prioritizing superior performance, I would have used an ensemble of models (starting by adding more GBDT models with different hyperparameters but likely continuing by adding other types of models, such as support vector machines and random forests, chosen based on validation performance). Also, I would have thought more about feature engineering and experimented with alternative feature specifications. As it is, I think my vanilla LightGBM model did pretty well.

I held out 20% of the data for evaluation of the final model, but for hyperparameter selection I used the CV routine included with LightGBM. Rather than pursuing a particular systematic method of hyperparameter selection, I used an intuitive trial-and-error approach, with which I've been successful in the past.

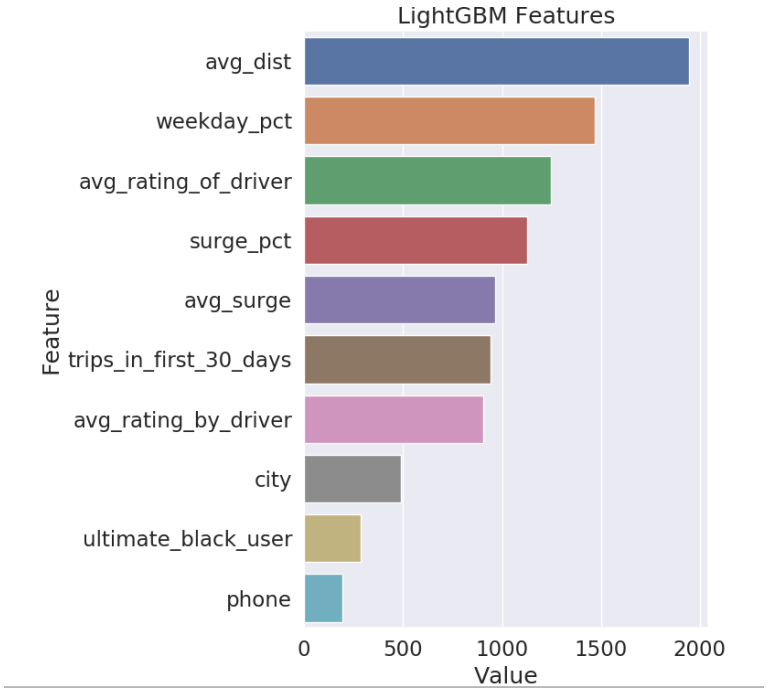
The results look pretty good, I think. Here's the ROC curve for my test data:



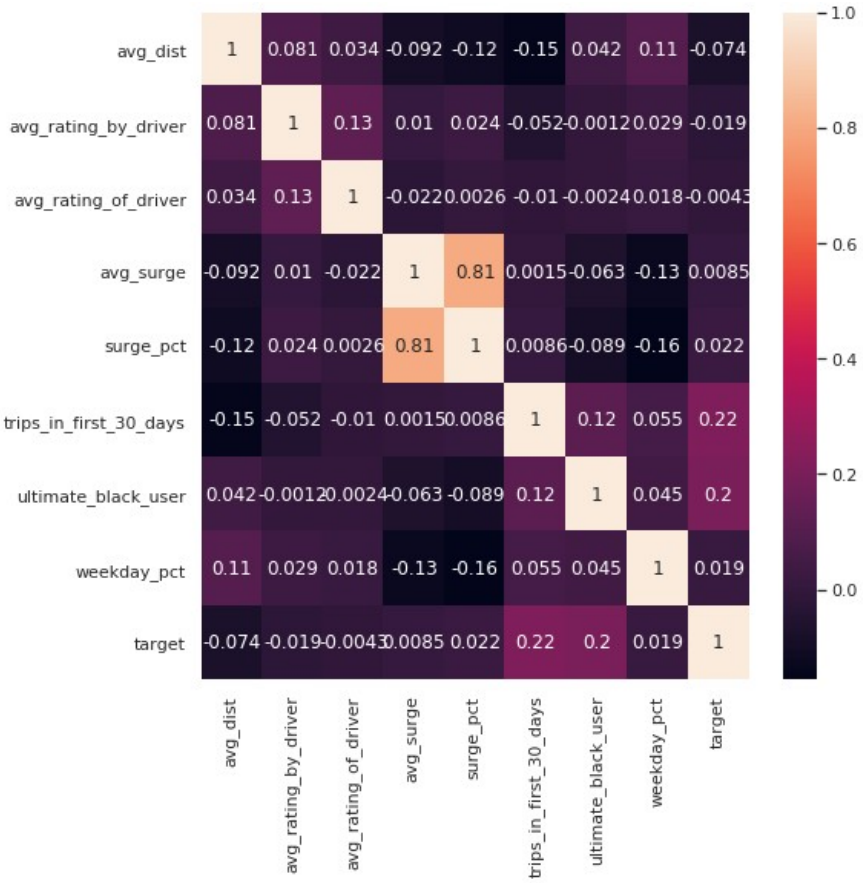
It appears that the model does a good job of discriminating between those who were likely to be retained and those who weren't. The test set accuracy (using a 0.5 probability cutoff) was 73.9%. (Note that the test set was randomly selected from a set of data which were now well-balanced after eliminating 30-day dropouts, as described earlier.) The F1 score was 73.4%.

But as usual, I'm less than 100% confident in the purity of the test set. The users in the test set came from the same time period as those in the training set. There is no guarantee that the model will generalize well to another time period. Ideally, I'd at least like to see the experiment repeated in a different time period and the model tested on the new data (but this could be done at the same time as attempting to apply the results).

3. The feature importance chart looks like this:



It's interesting to think about this chart in the light of the correlation heatmap. (The map below differs slightly from the earlier one in that it includes only the data used for modeling, which exclude users who left in their first 30 days.)



The most important features are not necessarily what I would have expected. Also, interpreting them in the light of the correlation heatmap, the impact of many of these features does not appear to be straightforward. The most important features are not highly correlated with the target and don't always have the sign one might expect (for example, users who rated drivers more highly are, overall, slightly less likely to be retained). Understanding what is going on will require a more in-depth analysis.

But the model does suggest at least one way to improve rider retention. Although it doesn't directly model how responsive riders will be to incentives, it does suggest which ones are most likely to be influenced. Riders with a low predicted retention probability aren't easily going to be pushed over the hump, and riders with a high predicted retention probability will not need to be cajoled; those with a moderate retention probability (near 50%) are likely the ones who could be most cost-effectively influenced.

Code for Part 3 is in "predictive.ipynb"