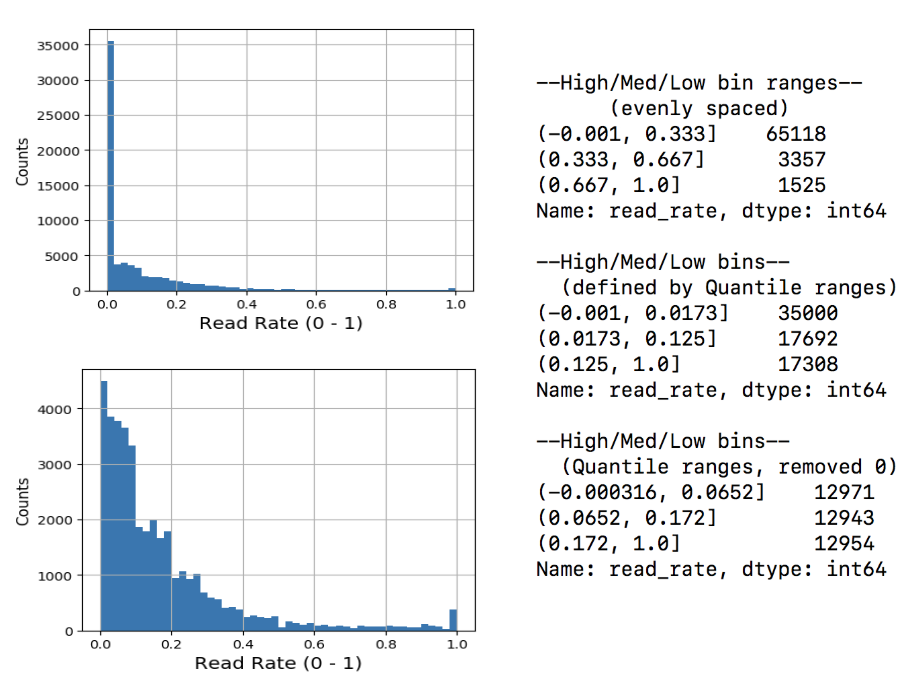
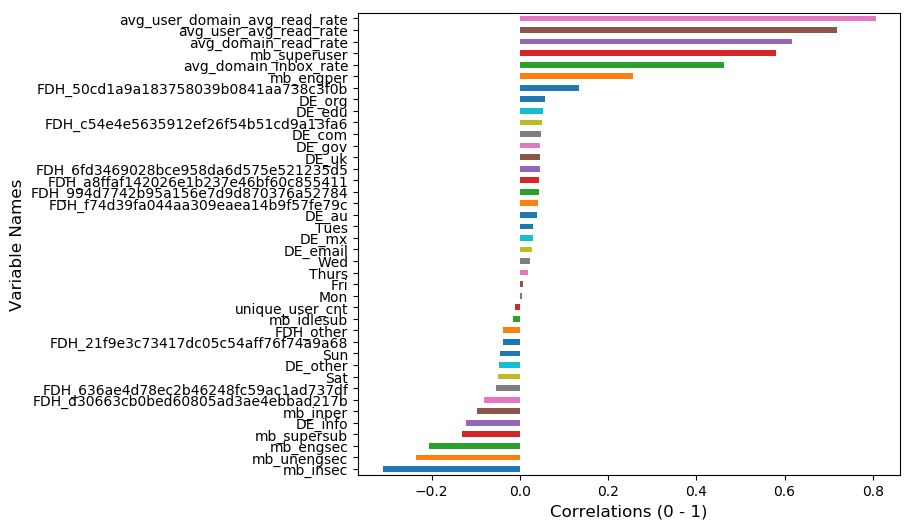
Coding Challenge Summary Report, Robert Manriquez

1. **Data Prep and EDA**

To start this project, I was interested in which variables had the highest correlations with the target variable, read\_rate. It was necessary to one-hot encode the non-numeric variables, particularly “day”, “from\_domain\_hash”, and “Domain\_extension.” Converting day was straightforward, but since the latter two variables had so many unique values, I only took the top 10 most correlated and labeled the rest as “other.”

Average read rates based on users, sending domain, and users for that domain seemed to be the most positively correlated along with recipients labeled as “super users” or “engaged personal accounts,” which is expected. Users and domains that have been profiled to have high read rates would naturally have higher read rates on new campaigns. The most negative correlations were user profiles identified as historically being less engaged, and surprisingly “super subscribers” as well. The least correlated variables were the domain dummies (“DE”) and from domain hash dummies (“FDH”), which could be just contributing noise since the average read rates of domains and user are already accounted for in the “avg” variables. Below is the bar graph of feature correlations with read\_rate, as well as read\_rate distributions with and without zero.



To convert the target variable to discrete “High/Med/Low” values, I determined bins based on the distribution of read\_rates. Splitting the bins into even ranges did not seem appropriate, since more than 90% of the values would fall below 0.33. Since almost half of the read\_rate values were “0” (~35k values), they were skewing the distribution significantly; quantile ranges with zero included forced the “Low” range to be extremely narrow (<1.73%) and the “High” range to be very wide (>12.5%). Splitting while ignoring zero seemed more appropriate by widening the “Low” range and narrowing “High” slightly, but also balancing our classes more. I compared models using the “even” ranges and the quantile ranges with zero removed to determine which method of converting read\_rate would be the most appropriate.

**II. Modeling and Scoring**

To perform the initial model screening, I tried both linear models and non-linear models. I chose first to use Random Forest, which benefits from robust parameter tuning, built-in ensembling to mitigate bias and variance, and the ability to handle high-dimensional data sets as well as multi-colinearity. Second, I wanted to compare with Logistic Regression, which has built-in regularization to handle noisy variables, relatively simple parameter optimization, and convenience for quick prototyping. I also prefer starting with these two models, since they both offer a method for pulling out feature importances / coefficients to help the user determine which variables are influencing the model decision making the most. This, along with the correlation bar graph, can help guide investigating which variables are the most important as well as somewhere to start pruning the least important variables if necessary. Two RF models were trained and optimized using RandomizedSearchCV (to save computation time), one with the “even” bins and the other with “quantile” bins. Two LR models were trained similarly, but using scaled values and comparing with and without PCA. All were split with 70/30 train-test ratio and the same random state for comparison.

Of the four models, RF using “even” split bins for read\_rate had the highest test accuracy, but the RF model using “quantile” bins scored better in terms of precision and recall. Both models can classify “Low” fairly well, but RF with quantile bins is classifying “High” and “Med” significantly better at the cost of worse performance on “Low.” Logistic Regresssion performed notably worse than both RF models. Incorporating PCA did reduce model training time appreciably, but at the cost of ~1% accuracy.

|  |  |
| --- | --- |
| Random Forest Scores (Quantile bins):  Train Accuracy : 0.883  Test Accuracy : 0.832  precision recall f1-score support  High 0.80 0.82 0.81 3902  Low 0.94 0.88 0.91 13224  Med 0.57 0.68 0.62 3874  avg / total 0.85 0.83 0.84 21000 | Random Forest Scores (Even bins):  Train Accuracy : 0.936  Test Accuracy : 0.920  precision recall f1-score support  High 0.75 0.73 0.74 473  Low 0.99 0.93 0.96 19492  Med 0.37 0.79 0.51 1035  avg / total 0.95 0.92 0.93 21000 |
| Log Reg Scores:  Train Accuracy : 0.817  Test Accuracy : 0.813  precision recall f1-score support  High 0.74 0.83 0.78 3902  Low 0.91 0.90 0.90 13224  Med 0.55 0.49 0.52 3874  avg / total 0.81 0.81 0.81 21000 | Log Reg with PCA Scores:  Train Accuracy : 0.805  Test Accuracy : 0.806  precision recall f1-score support  High 0.69 0.84 0.76 3902  Low 0.89 0.92 0.91 13224  Med 0.56 0.37 0.45 3874  avg / total 0.79 0.81 0.79 21000 |

**III. Conclusions:**

Of these models, Random Forest using quantile-defined bin edges for discretizing the target variable seemed to work the best, producing a model that is 83.2% accurate on the test set (but somewhat overfit the training set, 88.3%), which is an improvement over the baseline accuracy of 63% for the majority class, “Low.” However, all of these models still had trouble classifying the “Med” category, since it was somewhat tricky to define the range for. The true positive rate (recall) for “Med” is still only 68% with 57% precision, and considering the confusion matrix almost 30% of these values were mispredicted as “High” or “Low” equally.

Further work would include determining the most appropriate method of defining these bins, since some subjectivity may be required. Although it’s simple to intuitively judge if a campaign is successful using business sense, it can be difficult for a model. Although objectively a campaign read rate of 5% may be categorized as “Low,” if it’s still reaching 1000’s of users we may subjectively still consider that a success, whereas a 50% read rate for a much smaller campaign target at “engaged” users could classified as “High” but may not meet our expectations reaching a highly active user base. Also, addressing the class imbalance by using down-sampling could be a useful method to mitigate this data’s bias to the zero read\_rate value, but would be subject to experimentation. Lastly, confusion matrix printouts are saved in the “/img” folder as well as feature importance / coeff tables are saved in “/temp” for viewing, which mostly reiterate what was found from the correlation graph.