Dhruval Bhatt & Keertana Chidambaram

**Summary of Prediction Challenge and Learning**

**Notes on Learning**

This process provided a start to finish experience of working with given data to predict an outcome. An important step is to really get to know the data, so we familiarized ourselves with features, missingness and outcome to gain an intuition for types of model to consider. Upon exploring the data, a robust classification algorithm would be a good choice. We focused on using a decision tree, random forest, gradient boosting and ensemble methods to minimize the mean squared error (MSE). Of the models tested, gradient boosting regressor gave the lowest MSE. We learned that while running a simple decision tree is a much quicker, the results are not optimized and can be highly variable. Using methods such as bagging, boosting and finally an ensemble of models a, b, c improved the performance significantly.

Given the high dimensional data with a large number of missing values (~69%), another important question is how to handle missing data. We tried coding a new missing data indicator variable for all variables (0 if missing, 1 if not) as well as imputing the mode of a feature for missing data. The imputation was only done for data that had less than 80% missing data to ensure a reasonable estimate. However, neither helped with the performance. Both gave similar MSE as tuned random forest on unaltered train data. We suspect this process could be made more efficient by using a theoretical basis to group the features and impute types of variables strategically versus using the same method on all features.

Finally, we learn that in addition to statistical tradeoffs in determining best model for a given purpose, there are practical tradeoffs to consider such as, computational time. While the ensemble methods are more powerful, it takes much longer to run on mid-performance computer making tuning time consuming. One must consider, do we have the time or money for high power computing to afford higher accuracy or would a lower but faster performance be more acceptable?

**Unanswered Questions**

In this assignment, the number of missing data was high so clearly a strategy to address them would be helpful. However, based on the strategies employed, we didn’t see an improvement. The remaining questions here are: Why didn’t the missing variable indicator help? Is there an error in implementation or understanding of the effect of features on the data?

We ran a feature importance and found the top 2000 features and only used those in prediction using decision tree as well as reducing features using LASSO regression. The effect on the performance was marginal. A more robust tuning would probably help that further. However, we would like to further understand: Is there any value in reducing the number of features? In applying a method to select features – should that be on theoretical basis or algorithmically driven and no reduction applied?

**Anticipated test-data MSE**

Based on k-fold cross validation of final model on test data, the anticipated test data MSE is: XX