Predicting Survival Outcome on Hospital Data

Jacob Mitchell, Dan Mrachko

STAT 6440: Data Mining

Dr. Shuchismita Sarkar

April 24, 2022

1. Introduction

1.1 Motivation

Throughout our lives, sickness naturally occurs, and sometimes tragic accidents arise. As a result, we or our loved ones can end up at hospitals to treat our ailments. In more severe cases, patients can even stay at a hospital for several days or longer to treat their issues. These stays can include operational procedures or treatments for severe diseases. Sometimes, different underlying factors and conditions can have an impact on one's recovery process. Through data collection and analysis, we can narrow down which factors may have a positive correlation with a hospital stay resulting in full recovery or conversely ending in death.

1.2 Objective

Using R with an arsenal of classification algorithms and data imputation methods, the goal was to use collected data, data imputation and machine learning techniques to develop models for predicting the outcome of patients admitted to the hospital. A secondary objective was to study and compare the usefulness of different missing value imputation techniques with medical hospital stay data. A model with sufficient accuracy and sensitivity to *death result* could be used by medical professionals across the country for the purpose of understanding which factors may predict a patient's death during a hospital stay.

1.3 Data

To study the complexity of a patient hospital stay, we used a dataset called *Patient Survival Prediction,* as posted by Mtisha Agarwal. This dataset was curated with the intention of

developing a predictive model for all-cause-in hospital mortality among admitted patients. This was made through MIT's Global Open Source Severity of Illness Score initiative, and notably has an emphasis on the chronic condition of diabetes. This dataset is extensive. It has 91,713 observations of 85 features. The included features contain information that is collected when a patient is admitted to the hospital and throughout their stay. This information includes but is not limited to the patient's weight, BMI, age, gender, and specific information about medical history such as prior diseases, max heart rate, reason of admittance, etc. All of this information was gathered in the USA in the year 2021. A complete description of this dataset can be found at the link in our references for *Patient Survival Prediction* at Kaggle.com.

2. Methodology

2.1 Data Cleaning

Out of the 91,713 observations in the dataset, 34,115 of them were missing at least one feature. Because the features are important, containing medical information potentially critical to analysis, a method of dealing with the missing data was developed. Simply imputing missing data with the mean could lead to adverse predictive results. For instance, replacing missing values in BMI or max heart rate with the mean of all respective observations may be misleading when used as predictors. To develop models that are built off of data that is as accurate as possible, we realized it was necessary to strategically deal with the missing values. Some of the dataset contained binary indicators of whether or not a patient has a history of the following diseases or preexisting conditions: leukemia, cirrhosis, AIDS, solid tumor with metastasis, immunosuppression, and hepatic failure. Looking at the distribution of these features on all

observations, it was clear to see that most observations were not flagged with these features, as they were rare conditions. Since the distribution of these rarer conditions was leaning heavily towards 0, these features were imputed to be 0 if they were not present in the dataset. Imputing the other medical data was less straightforward and potentially more problematic. Columbia Public Health states that missing data can add bias to our results and can distort prediction results. (“Columbia Public Health,” 2019) To model the data with high accuracy and accomplish the secondary objective, we chose to create three datasets using different data imputation techniques and compared the results.

2.1.2 Drop NA Dataset

A common way of dealing with missing values is to simply drop the observations containing missing values, which doesn't add bias if the missing values are missing at random.(“Columbia Public Health,” 2019) This technique is simple to implement and adds simplicity to the model by reducing the total number of observations used for training. The primary detriment of this technique is the loss of all records containing missing values. For this dataset, that is roughly 37% of all observations. A reduced subset of the data is not ideal because it may not adequately represent the full set, leading to bias in the analysis.

2.1.3 Mean and Mode Imputation

An alternative to dropping the missing data is to fill in continuous numeric features with mean values and categorical features with modal values. This method assumes that missing numeric values can be represented by the average of the population, and the missing categoric values can be represented by the most frequently occurring values in the population. Imputing with this technique is only slightly more complex than dropping the values, but may also

introduce bias into the analysis. When dealing with something as sensitive as death in a hospital setting, misrepresenting data could lead to inaccurate model development. For instance, filling in a person's max heart rate with the mean value of the data would represent them as healthy. If the truth were that they were an outlier case with abnormal heart rate measurements, this incorrect assumption could have the potential to lead to model misidentification. For this reason a third technique of missing data imputation was chosen.

2.1.4 Multivariate Imputation by Chained Equations (*MICE*)

The third missing data imputation technique chosen was Multivariate Imputation by Chained Equations (MICE). MICE is an imputation technique that assumes that values in the dataset are missing at random. It uses chained equations and an iterative regression approach to dealing with missing values.

The general steps of MICE are as follows: a place-holder imputation is used for all missing data values. One parameter at a time, the values are removed from the set and regressed on. Those missing values are then replaced with the result of the regression, and this process is repeated for all variables that have missing data. (Hamzah 2020) This method of imputation was chosen with the hope of filling the missing values with data that would enable a more accurate model to be developed. However this method rests on the primary assumption that the data is missing at random, and is less likely to perform accurately when the missing data is the result of some bias or conditional behavior.

2.1.5 Final Datasets

Three datasets were generated from the original raw data using the above mentioned imputation techniques: Drop NA, Mean-Mode Imputation, and MICE imputation. Each machine

learning modeling technique chosen was performed over each of the three datasets to enable comparison of the imputation techniques while searching for an accurate model.

2.3 Constructing Models

2.3.1 Modeling Techniques

To develop models which could accurately predict patient death in a hospital setting, multiple approaches to modeling were employed. 10-fold cross validation was employed using three different types of classification models.

Classification Trees were used as the first model type. In this approach, the data is regularly partitioned to optimize a cost-reduction function. Each partition is a split in the tree, leading to two branches. At each branch an impurity metric is computed to determine the homogeneity of each new partition. The tree will continue to "grow" by partitioning each branch into new branches until the impurity metric is below a minimal threshold, indicating the dataset has been adequately partitioned. 10-fold cross validation with cost complexity pruning was used to iteratively develop a model for each imputed data set.

In addition to Classification Trees, Regularized Regression with Lasso technique was employed to model the result of a patient's hospital stay. Lasso technique estimates regression coefficients by minimizing the Sum of Square Deviations plus a Penalty factor. In this case the penalty factor is defined as:

$$\lambda \sum_{j=1}^{k} \left|\beta_j\right|$$

where lambda is the tuning parameter. This technique is useful for large datasets with the potential to suffer from multi-colinearity because it has the potential to force coefficients to zero,

simplifying the model and reducing the number of features involved in prediction. Since the dataset for this project was suitably large, this was deemed an appropriate method.

The third modeling technique chosen was Random Forest Classification trees. Random Forest technique is one of several Ensemble modeling methods which aims to reduce variance without increasing bias. This is done by performing bootstrapping, a sampling technique of repeatedly drawing samples with replacement. Each bootstrap is then further divided into groups of randomly chosen predictors. Each group of data will contain a subset of the overall data, as well as a subset of all predictors. Each group is used to train a tree, then the trees are combined to form a final model. The number of predictors used in each tree was cross-validated over (2, 8, 9, and 40) to determine the optimal parameter to maximize kappa. In addition this method was employed with a 10-fold cross validation approach. Although this method can lead to highly accurate models, it has the drawback of being computationally expensive.

2.3.2 Optimizing Cutoff

Our dataset is heavily unbalanced, with most patients in cases of hospital admittance surviving their respective operations. As a result, models may become very good at predicting patient survival, without being sensitive to patient deaths. We would like to create a model that is more accurate with predicting both classes, but more importantly, cases of death. To combat this imbalance, we designed a function to optimize the value of alpha (probability cut-off value when classifying as “1” or “0”) to more accurately predict both cases. This function iterates over different cutoff values of alpha ranging from 0.1 to 0.9, and records the performance in terms of overall accuracy, kappa, and the cutoff value at that iteration. This function then returns the cutoff values corresponding to the maximum kappa and maximum overall accuracy. This function was used to help us find which value of alpha works best for predicting both cases of

survival and death on our test data in all three model types and all 3 data imputation techniques. For reference, see appendix 7.2.2.

2.3.3 Classification Tree with 10 Fold Cross Validation

Classification trees were constructed and trained using the r package *r part*. 10-fold cross-validation with an initial data partition of 70:30 (training : testing) split was employed on each of our datasets for testing and validation. To find the best alpha cutoff value, we employed the optimizing function and generated the following plots. For the max kappa value, cutoff value was determined to be 0.28, 0.14, 0.18 for the MICE, Dropped NA, and Mean-Mode datasets respectively.

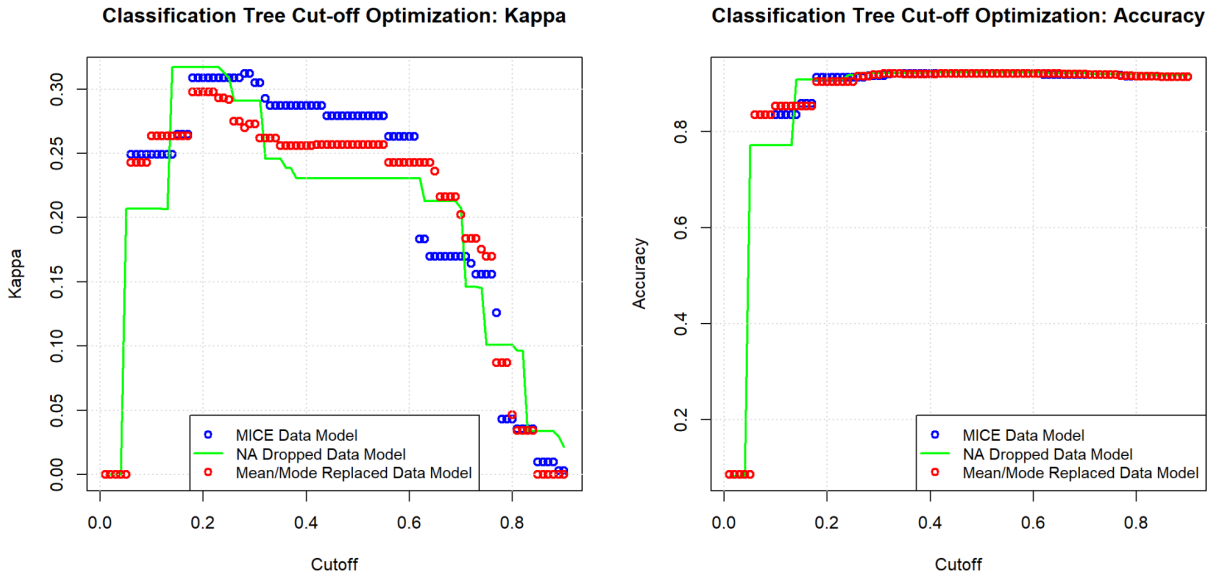

Figure 1. Classification Tree Cutoff Optimization

2.3.4 Logistic Regression Using 10 Fold Cross Validation

The r package *glmnet* was employed to fit a lasso logistic regression model with 10-fold cross-validation. The metric used to measure predictive power was Kappa. We chose Kappa to ensure we could get our highest balance of sensitivity and specificity. Death in the hospital is less frequent than living, so it is important with such dire consequences to create a model that can predict both classes sufficiently. Similar to the Classification tree, the datasets were partitioned using a 70:30 split, and iteratively cross-validated over 10 folds. The caret library in r was used to specify the 10 fold cross validation over the 70% partition training data. After training the model, the alpha optimization functions was used to determine the optimal cutoff values. The value of alpha returned by our cutoff function was used to evaluate the performance of our model on our test data. Below shows a plot of the value alpha vs kappa and of alpha vs accuracy for each of our three datasets.

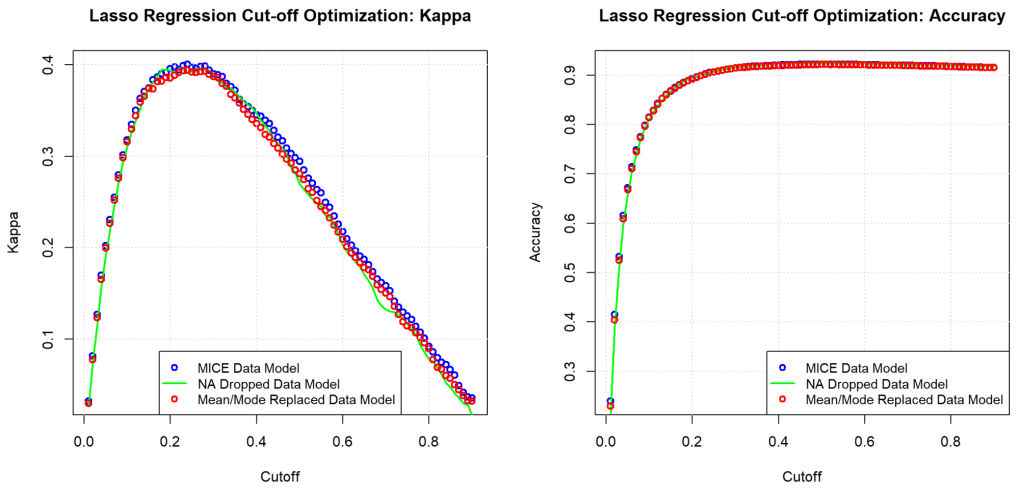

Figure 2. Lasso Regression Cutoff Optimization

Optimal cut-off values were determined to be 0.24, 0.23, 0.24 for the MICE, Dropped NA, and Mean-Mode datasets respectively.

2.3.5 Random Forest with 10 Fold Cross Validation

The third model type created was a random forest model. Our dataset had 79 predictors, so we chose to cross-validate over tree depth values of 2, 8, 9, 40. The number of trees was chosen as 100 to ensure accuracy at computational expense. The resulting models iteratively optimized and selected tree depth of 40 as the best value. Performing cutoff value optimization yielded cutoff values of 0.33, 0.31, and 0.33 for the MICE, Dropped NA, and Mean-Mode datasets respectively. The results of this optimization are depicted in the figures below.

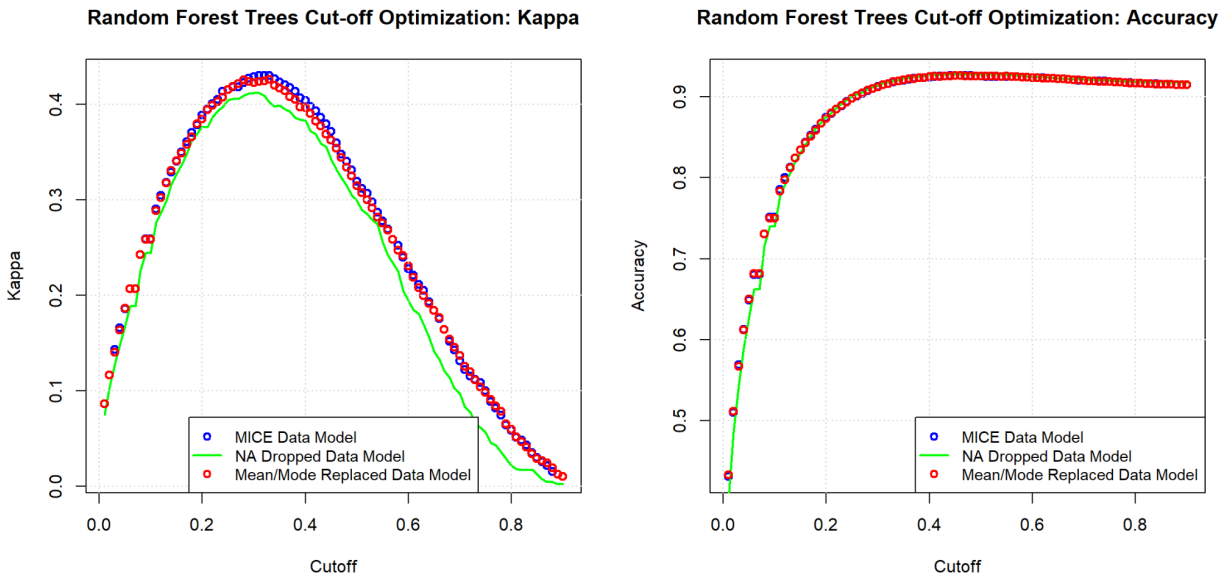

Figure 3. Random Forest Cutoff Optimization

3. Results

3.1 Model Accuracy

Table 1 below has a breakdown of the accuracy measures of each of our constructed models.

Model	MICE Data	NADropped	MeanMode
Lasso Logistic Regression	Accuracy: 0.9051 Kappa: 0.4007 Sensitivity: 0.4569 Specificity: 0.9472	Accuracy: 0.9038 Kappa: 0.3969 Sensitivity: 0.47552 Specificity: 0.94237	Accuracy: 0.9044 Kappa: 0.3939 Sensitivity: 0.4480 Specificity: 0.9474
Classification Tree	Accuracy: 0.9166 Kappa: 0.3123 Sensitivity: 0.2631 Specificity: 0.9780	Accuracy: 0.9086 Kappa: 0.3173 Sensitivity: 0.31894 Specificity: 0.96162	Accuracy: 0.904 Kappa: 0.2979 Sensitivity: 0.2982 Specificity: 0.9610
Random Forest	Accuracy: 0.9183 Kappa: 0.4304 Sensitivity: 0.42893 Specificity: 0.96425	Accuracy: 0.9164 Kappa: 0.4121 Sensitivity: 0.42655 Specificity: 0.96046	Accuracy: 0.9184 Kappa: 0.4263 Sensitivity: 0.42090 Specificity: 0.96517

Table 1

3.2 Discussion of Results

3.2.1 Model Comparison

Each of the datasets performed with over 90% overall accuracy, which shows that these features do have correlation with hospital death. However, the relatively low kappa value in each model's results indicate the models are not highly sensitive to the death condition. This is further demonstrated by the values listed in Table 2 (see Appendix 7.1 Table of Results). Overall, the classification tree was the weakest performer, with the lowest overall kappa values and sensitivity values across all three datasets. The logistic regression models on each dataset had slightly lower kappa values than the random forest models, but they were comparable. On first glance, the random forest models appear to be the best with highest accuracy and kappa values, indicating the best sensitivity to patient death. However it should be noted the lasso logistic regression models predicted fewer false-survivals than the random forest models.

This finding is further reinforced by the ROC charts depicted below in Figure 4. These images indicate that although the random forest method jumps up sooner on the chart (has a higher kappa value) the lasso regression method has a slightly higher overall sensitivity value (the y-axis scale). This means the random forest model was more balanced at predicting both life and death, but the lasso regression technique was more sensitive to predicting death.

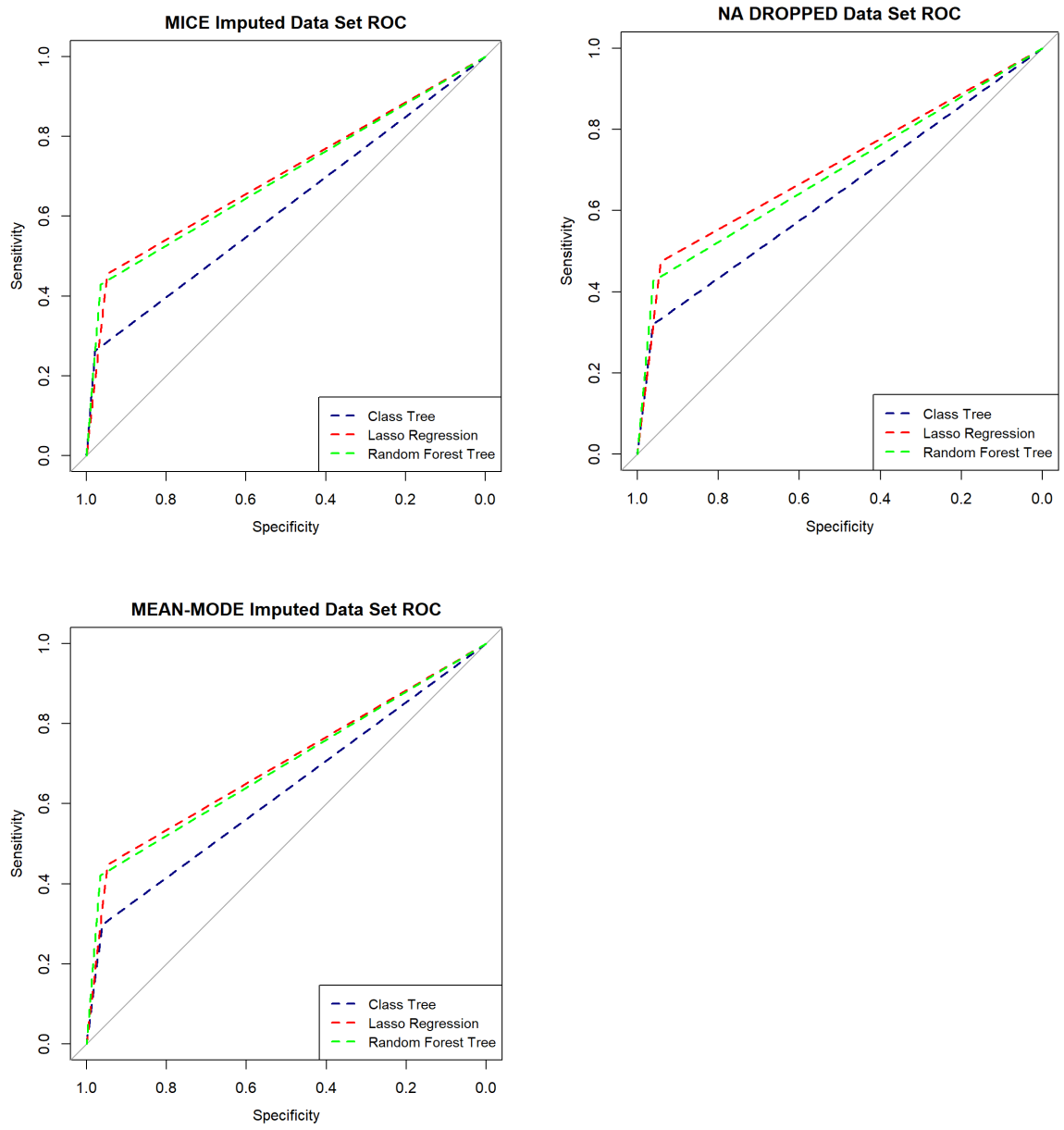

Figure 4. ROC Charts

3.2.2 Imputation Performance Comparison

The imputation technique comparison is best done by looking at the results in Table 2 (see Appendix 7.1). For all three modelling techniques, the greatest kappa values were achieved when using the MICE imputed dataset. Additionally the best accuracy was achieved with the MICE imputed datasets, except in the case of the random forest model on the Mean-Mode data set which improved on accuracy by only 2 hundredths of a percent.

4. Conclusions

Through our study, we have found that data can be useful in the prediction of death and survival in the medical setting. With all of our models performing with over 90% overall accuracy, it would seem that data can have an impact on predicting this outcome. It has also been shown, however, that more work is to be done to improve the accuracy of predicting cases of death.

Additionally, our results show that using a multivariate imputation to impute missing data may be the best option in this setting. Our MICE imputed datasets outperformed the other imputation methods in every model, leading us to believe that multivariate imputation may be the best way to fill in missing values in the medical setting.

5. Limitation of the analysis and future direction

5.1 Limitation of the analysis

One limitation of this analysis was computational power. We were unable to train a neural network given the power of our machines, and even running our random forest models and MICE imputations took a lot of time and resources. Another notable limitation of our study is

that our data is only comprised of hospital data from the USA during a pandemic, which may lead to varying predictive power if used in other areas of the world at different times.

Without a "gold-standard" dataset that does not contain missing values for comparison, it is difficult to assess the true accuracy with which data imputations replaced the data. Here imputation techniques were compared based on the results of the models built over them.

5.2 Future direction

It is difficult to compare the models to each other without an un-tainted "gold-standard" dataset missing no values for testing of models. Future work could focus on securing such a dataset for the purpose of testing, training, validation, and imputation technique comparison.

While the models performed with relatively high accuracy, they did not perform with as high of a sensitivity towards the death condition as was desired. The data was imbalanced, leaning towards instances where patients survived (negative outcome to parameter *hospital_death*). As a result, the models were very good at predicting survival, and less accurate when predicting patient death. Future studies should aim at exploring techniques for balancing the data prior to analysis to create models less biased towards a single outcome.

Another method of addressing this issue is the use of neural networks. Given the magnitude of the dataset and the computational complexity of training a neural network, this could potentially require outside resources such as the Ohio SuperComputer. In addition, several other reduction methods such as Principal Component Analysis could be performed to improve computational need.

Additional work would include studying the selected variables used in the final models. Each of the models used different features in their final models. A combination of knowing the

most important features and a medical professional who understands the implications of those features could help to determine parameters that have a higher influence on survival rate, and maybe save lives!

6. References

Agarwal, Mitisha. “Patient Survival Prediction.” *Kaggle*, 26 Dec. 2021, https://www.kaggle.com/datasets/mitishaagarwal/patient.

Azur, Melissa J, et al. “Multiple Imputation by Chained Equations: What Is It and How Does It Work?” *International Journal of Methods in Psychiatric Research*, John Wiley & Sons, Ltd, Mar. 2011, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3074241/.

Khan, Shahidul Islam, and Abu Sayed Md Latiful Hoque. 2020. "SICE: an improved missing data imputation technique." Journal of Big Data 7 (37). Accessed December 2021. doi:10.1186/s40537-020-00313-w.

Hamzah, Fatimah Bibi, Firdaus Mohd Hamzah, Siti Fatin Mohd Razali, Othman Jaafar, and Norhayati Abdul Jamil. 2020. "Imputation methods for recovering streamflow observation: A methodological review." Cogent Environmental Science 6 (1). Accessed December 2021. doi:10.1080/23311843.2020.1745133.

“Missing Data and Multiple Imputation.” *Columbia Public Health*, https://www.publichealth.columbia.edu/research/population-health-methods/missing-data-and-multiple-imputation.

7. Appendix

7.1 Table of Results

Model	Imputation	Kappa	Accuracy	CutoffVal	TruePositive	TrueNegative	FalsePositive	FalseNegative
ClassTree	MICE	0.312274705	0.916587919	0.28	0.022606673	0.893981246	0.020098859	0.063313222
ClassTree	NADropped	0.317307715	0.90856383	0.14	0.026329787	0.882234043	0.035212766	0.056223404
ClassTree	MeanMode	0.29792427	0.904048848	0.18	0.025623319	0.878425529	0.035654576	0.060296576
LassoRegres	MICE	0.400671347	0.905066512	0.24	0.039252744	0.865813768	0.048266337	0.046667151
LassoRegres	NADropped	0.396938478	0.903829787	0.23	0.039255319	0.864574468	0.05287234	0.043297872
LassoRegres	MeanMode	0.393891297	0.904448644	0.24	0.038489496	0.865959148	0.048120957	0.047430399
RandomForest	MICE	0.430383503	0.918259795	0.33	0.036853965	0.88140583	0.032674275	0.04906593
RandomForest	NADropped	0.412138009	0.916382979	0.31	0.035212766	0.881170213	0.036276596	0.047340426
RandomForest	MeanMode	0.426342909	0.918405176	0.33	0.036163408	0.882241768	0.031838337	0.049756488

Table 2. Overall Accuracy Results by Model and Dataset

7.2 User Defined R-Functions

7.2.1 Mean and Mode Imputation Function: *R-Script*

```
hosp_mode_mean <- function(hospital) {
  cols = length(hospital)

  # iterate over columns of data table
  for (i in 1:cols) {
    # check if column is a factor
    if (is.factor(hospital[,i])) {
      # iterate over values, compute mode, replace NAs with mode.
      samp <- c()
      for (j in 1:length(hospital[,i])) {
        if (!is.na(hospital[j,i])) {
          samp <- append(samp, hospital[j,i])
        }
      }
      uniq_samp <- unique(samp)
      mode_samp <- uniq_samp[which.max(tabulate(match(samp, uniq_samp)))]
      hospital[,i] <- replace_na(hospital[,i], replace = mode_samp)
    } else {
```

```
    # in this case the values are numeric and need to be replaced with mean value
    hospital[is.na(hospital[,i]),i] <- mean(hospital[,i], na.rm=TRUE)
   }
  }

  return(hospital)
}
```

7.2.2 Cutoff Value Optimization Function: *R-Script*

```
cutoff_opt <- function(test_data, userpred, pos_val) {
  # pos_val = "1" or "0"
  # userpred = prediction values from the model
  # usermod = the model

  cutoffs = as.vector(seq(0.01, 0.9, by = 0.01))

  res_mat <- matrix(vector(), nrow = length(cutoffs), ncol = 3, dimnames = list(c(), c("Cutoff",
"Accuracy", "Kappa")))

  for (i in 1:length(cutoffs)) {

    the_cut = cutoffs[i]

    model_preds <- ifelse(userpred[,2] > the_cut, "1", "0")
    confMat <- confusionMatrix(as.factor(model_preds),
                               as.factor(test_data$hospital_death),
                               positive = pos_val)
    res_mat[i,1] = the_cut
    res_mat[i,2] = confMat$overall[1]
    res_mat[i,3] = confMat$overall[2]

  }

  best_acc <- res_mat[which.max(res_mat[,2]),]
  best_kap <- res_mat[which.max(res_mat[,3]),]

  return(list(best_kappa = best_kap, best_acc = best_acc, res_matrix = res_mat))

}
```