Hospital Length of Stay Prediction and Bed Allocation Optimization

Abstract

Elongated and often miscalculated length of stay (LOS) costs hospitals a significant amount of money every year in terms of suboptimal resource allocation. For this reason, hospitals start to rely on analytical approaches, specifically machine learning methods, to make accurate predictions on LOS. Following such predictions comes with optimizing bed allocation within each hospital, thus maximizing occupancy levels and enabling resources to be efficiently utilized. In this paper, we predicted patients' LOS using multiple machine learning models, including the well-known Classification and Regression Trees (CART), Random Forests, Boosted Trees, as well as the state-of-the-art Optimal Regression Trees (ORT), in which we also compared the advantages and limitations of each model. Finally, we used Mixed Integer Optimization (MIO) to maximize bed utilization on a synthetic dataset to show how our predictions can help hospitals reduce empty beds.

1 Introduction

The term average length of stay is widely used to gauge the efficiency of a healthcare facility. According to the Agency for Healthcare Research and Quality, in the United States, the national average for a hospital stay is 4.5 days per patient, at an average cost of $10,400 per day. While this situation does not apply to all hospitals across the world, it is generally the case that longer and more unpredictable LOS costs hospitals more resources. We believed that this cost can be reduced through the following 2 ways:

- Predict the next move of each patient, such as the probability for a given patient to need an intensive care bed in the next 24 hours (Bertsimas et al. 2019), so that the hospital can provide the patient with the most proper treatment and allocate resources accordingly.
- Obtain a more accurate forecast of each patient's LOS before or when the person first admits to the hospital so that bed allocation can be optimized ahead of the time.

Due to limited computation resources and availability of the data, the first solution is out of the scope of this paper and we thus focused on the second solution that we suggested. Consequently, our models would not consider any events of a particular patient that happened after his or her bed assignment. In other words, we would only predict each patient's LOS once and do not update it with respect to time. The results of this paper can be extended to, first of all, helping the hospitals to assign emergency patients to available beds based on current and possibly future occupancy levels, and secondly, allowing the hospitals take reservations of non-emergency patients and allocate them to the best admission dates given their medical situations. Ultimately, this not only reduces costs of the hospitals but also benefits the patients in a way that, as the medical resources being optimized, the hospitals can accept more in-patients, thus alleviating many hospitals' sporadic overcrowding problems.

2 Dataset and Features

2.1 Data Source

The data we used for this paper comes from the MIMIC-III Critical Care Database, a large, publicly-available data source comprising de-identified health-related data associated with over 40 thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center in Boston, Massachusetts, between 2001 and 2012.

There are 26 tables within the MIMIC database, and each of them contains thousands to millions rows of data. After carefully reviewing all variables in each table, we decided to only use 4 tables, which include data on patients' demographic information, types of visit to the hospital, diagnoses, and most importantly, the admit and discharge time of each visit. Then we joined the tables together to create a master data frame through two primary keys: patient ID and hospital visit ID (*Figure 1*).

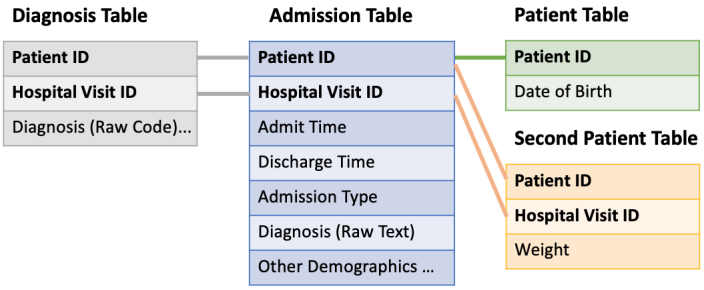

Figure 1: Table Relationships

2.2 Feature Engineering

Although the data provided by the MIMIC database is relatively structured, many data cleaning and feature engineering procedures were still necessary before feeding the data into our machine learning models. The following showcase some of the most important engineering steps:

- Transformed strings of time to correct date format and calculated length of stay and age as:
  - Length of Stay = Discharge Time − Admit Time
  - Age = Admit Time − Date of Birth

  There is an additional caveat on the time-related variables: to protect patient confidentiality, all dates in the database have been shifted to a future date. While these dates are internally consistent for the same patient, there exists another time shift on date of birth for children under 12 years old for extra protection. We discovered that the second shift is exactly 300 years and thus recovered the correct age for young kids.
- Categorized and pattern-searched on diagnosis:
  1. All the raw diagnosis codes are of length 6, for example, 158297. After some investigation, we realized that the correct format of the code is a three digits number followed by some decimals, which represent their subcategories. Since we are only interested in the broader categories of the disease, we extracted the first three digits of each raw diagnosis code.
  2. We created 18 broader health problems groups based on the ICD-9 codes from the World Health Organization and categorized each patient's diagnosis according to these groups. One thing to notice is that a patient might have multiple diseases across multiple groups, so this is not quite a one-hot-encoding engineering.
  3. To make sure the ICD-9 codes classified the diagnoses correctly, we cross-checked with the raw diagnosis texts written by the doctors.
- We realized that, for example, the *Ethnicity* variable has many values that indeed represent the same group of people. For this type of categorical variables, we cleaned them up using regular expressions and combined similar categories into the same group.
- Applied one-hot-encoding to all categorical variables.

3 Machine Learning Models

We built four machine learning models to predict a patient's length of stay. They are Classification and Regression Trees (CART), Random Forests, Boosted Trees, and Optimal Regression Trees (ORT). As our dataset is quite complete and we didn't have many missing data, we assumed that the data are missing at random. To deal with missing data, we trained each model on dropping missing data, mean imputation, and optimal imputation, in which we found out that dropping missing data provided the best out-of-sample performance. Consequently, we decided to drop rows with missing values. Below is a detailed description of how these models are implemented:

3.1 Classification and Regression Trees

We used 10-fold cross-validation to select the best complexity parameter (cp) of 10^{-4} for our CART model because we would like to capture the richness of our data but at the same time also have a relatively simple and interpretable tree. Due to the intrinsic complexity of this problem, our final CART tree still has about 20 layers in total, so we would instead like to display a simpler version of the CART tree, which has only four layers and is built by setting cp to be 0.01.

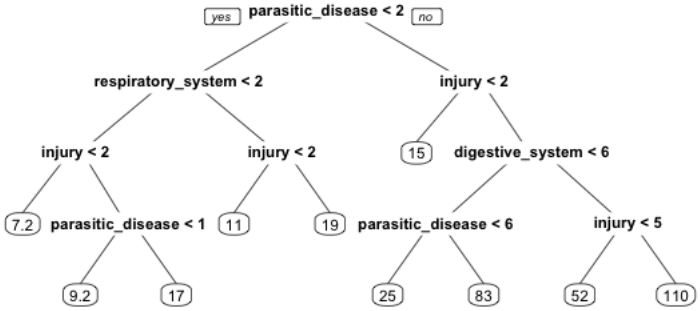

Figure 2: CART with Complexity Parameter of 0.01

Following the path from the top node of the tree to the leaf node, we can easily interpret the decision rules of the tree. For example, if a patient has more than one parasitic disease but less than two injuries, this CART model predicts the patient to stay in the hospital for 15 days.

3.2 Random Forests

For Random Forests, we applied the out-of-bag score to validate our hyperparameters such as the number of variables available for splitting at each tree node (mtry), in which we found a sweet spot when mtry is 11. The following table describes the 6 most important variables found by our Random Forests model:

Table 1: Top 6 Variables Found by Random Forests

	injury	**parasitic**	**digestive**	**respiratory**	**age**	**weight**
IncNodePurity	131594.94	128862.44	104827.47	82427.40	67679.96	63442.35

From the above table, we can observe that our Random Forests model also emphasizes the importance of the number of injuries, parasitic diseases, and diseases related to the digestive system and respiratory system, which perfectly corresponds to what we have found from the CART model.

3.3 Boosted Trees

Due to the computational complexity associated with boosting, grid search was not performed for parameter selection. Instead, we manually calibrated the values of our hyperparameters. The 6 most important variables selected by the Boosted Trees model are in decreasing order of patient's age, number of parasitic diseases, number of injuries, patient's weight, number of digestive and respiratory system diseases, which also work in concert with findings of the previous two models.

Table 2: Top 6 Variables Found by Boosted Trees

	age	parasitic	injury	weight	digestive	respiratory
rel.inf	11.27	10.00	9.73	9.72	7.95	5.92

3.4 Optimal Regression Trees

Optimal Regression Trees (ORT) is a state-of-the-art decision trees model that is proved to be globally optimal, thus obtaining more predictive power than the traditional top-down, greedy decision trees such as CART. Moreover, ORT is also extremely interpretable, as it can achieve the same level of performance as CART with much less complexity in the tree (Bertsimas and Dunn 2017).

We found the optimal values of maximum depth [10], complexity parameter [0.003], and the minimum number of observations in any terminal node [10] for OCT through validation. The complete tree is displayed below.

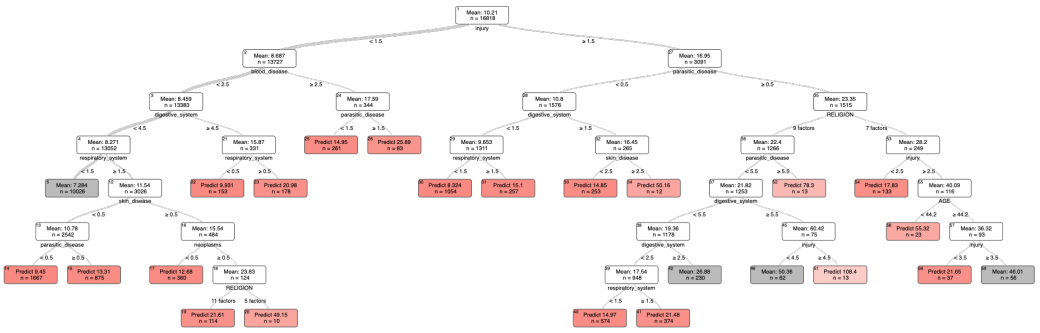

Figure 3: Optimal Regression Trees

From the above, ORT has fewer layers than the complete CART model, which makes it more interpretable. Notice the top splits of the ORT model are number of injuries, blood disease, and parasitic disease, which are similar but also slightly different than those of the CART. This is expected since CART uses a greedy approach for each split while ORT takes consideration of global optimality.

4 Machine Learning Results

From the table below we can see that among the 4 decision tree models, Boosted Trees performs the best across all metrics, and Random Forests is slightly worse but still very close to the best. Although ORT does not outperform Boosted Trees and Random Forests, it beats CART in every category.

Table 3: Out of Sample Performance of the 4 Models

	CART	Random Forests	Boosted Trees	ORT
R^2	0.2968	0.5323	**0.5583**	0.3051
MAE	5.6206	4.7313	**4.5645**	5.5653
RMSE	9.2911	7.5772	**7.3637**	8.9327

To choose which model to select, it is important to determine what is the main goal of the analysis. If, for instance, interpretability is a key success factor, ORT should be selected since it is more interpretable than Random Forests and Boosted Trees and performs better than CART on the test set. However, this model is not the best in terms of prediction accuracy. On the other hand, if prediction accuracy is the main goal of the analysis, Boosted Trees is the to-go choice, since it consistently achieves high predictive performance among the 4 models.

In our case, having an accurate prediction for LOS is very important because the ultimate goal of the project is to optimize bed allocation for patients. For this reason, Boosted Trees was selected to predict LOS.

5 Optimization Methods

5.1 Mixed Integer Optimization Formulation

Assumptions:

We assumed that patients come on a reservation basis, which means that we can assign each of them a date for admission or inform the person that there is no bed available for him or her, based on our model's prediction on the patient's LOS. We also assumed that patients should stay in single-gender wards. Moreover, for a particular ward, its assigned gender can change over time; our model ensures that all patients in the same ward must be of the same gender every day. As we did not have the domain knowledge about which diseases should be separated from others, we do not consider this in our constraint. However, these constraints can be easily incorporated into our formulation once we obtain this information. Finally, we assumed that there are no patients before the first day and all patients need to leave on the final day; nevertheless, this can also be easily modified dependent on the hospital's situation.

Sets:

- $i = 1, \cdots, I$: Patient
- $j = 1, \cdots, J$: Room
- $t = 1, \cdots, T$: Day

Parameters:

- l_i: Expected length of stay (LOS) of $patient_i$
- g_i: Gender of $patient_i$ such that: $g_i = 1$ if $patient_i$ is female, and -1 otherwise.
- c_j: Capacity (beds) of $room_j$

Decision Variables:

- x_{ijt}: Binary variable such that $x_{ijt} = 1$ if $patient_i$ stays in $room_j$ on day_t and 0 otherwise.
- y_{ij}: Binary variable such that $y_{ij} = 1$ if $patient_i$ is assigned to $room_j$ and 0 otherwise.

Formulation:

$$\text{maximize} \quad \sum_i \sum_j \sum_t x_{ijt} \tag{1}$$

$$\text{subject to} \quad \sum_j y_{ij} \leq 1 \qquad \forall i \tag{2}$$

$$x_{ijt} \leq y_{ij} \qquad \forall i, j, t \tag{3}$$

$$\sum_j \sum_t x_{ijt} = l_i \cdot \sum_j y_{ij} \qquad \forall i \tag{4}$$

$$\sum_j \sum_{t=2}^{T} s_{ijt} + \sum_j (x_{ij1} + x_{ijT}) \leq 2 \qquad \forall i \tag{5}$$

$$s_{ijt} \geq x_{ijt} - x_{ijt-1} \qquad \forall i, j, t \tag{6}$$

$$s_{ijt} \geq x_{ijt-1} - x_{ijt} \qquad \forall i, j, t \tag{7}$$

$$\sum_i x_{ijt} \leq \sum_i g_i \cdot x_{ijt} + p_{jt} \cdot M \qquad \forall j, t \tag{8}$$

$$\sum_i x_{ijt} \leq -\sum_i g_i \cdot x_{ijt} + (1 - p_{jt}) \cdot M \qquad \forall j, t \tag{9}$$

$$\sum_i x_{ijt} \leq c_j \qquad \forall j, t \tag{10}$$

$$p_{jt} \in \{0, 1\} \qquad \forall j, t \tag{11}$$

Explanation of the Formulation:

(1) Maximizing on total number of beds utilized from day 1 to day T;

(2) Each patient can only be assigned to at most 1 room;

(3) Patients can only stay in the room assigned to them for their entire stay;

(4) If a patient gets assigned, we make sure that the patient stays for a length equal to his or her expected LOS; if the patient is not assigned a room, the length is just 0;

(5) - (7) Each stay needs to be consecutive (there cannot be gaps in between a patient's single stay);

(8) - (9) All patients in the same room must have the same gender for a given day;

(10) Each room cannot exceed its capacity;

5.2 Optimization Results

Our optimization formulation can be implemented to maximize occupancy levels based on the machine learning predictions of patients' LOS. More constraints of patients' medical conditions can be imposed in a similar fashion as the gender constraint.

In terms of runtime, we can see that our MIO formulation can handle a 95-105% overcrowded situation in a relatively reasonable time (105% overcrowded means that there are 105 beds needed while the hospital only has 100 beds). For example, for 100 patients and a 20-day scheduling period, our algorithm finished running in about 4 minutes, which is practically tractable considering that we are scheduling these 100 patients at the same time. The time will be vastly reduced if we are scheduling patients one-by-one.

Table 4: Runtime (Sec) of MIO (Beds Numbers Adjusted to Create 95-105% Overcrowded Situation)

	10 Patients	25 Patients	50 Patients	100 Patients
10 Days	0.1045	0.5579	2.1107	16.2869
20 Days	3.3346	3.8063	29.7983	259.4201
30 Days	6.4041	10.7130	112.2601	600.2212

6 Future Work

To improve the prediction accuracy of our machine learning models, more features related to patient's medical conditions can be collected. For example, data such as patient's blood pressure, body temperature, body mass index (BMI), and historical medical and hospitalization records should be readily available when the person first admits to the hospital. However, we did not have these critical features in our dataset, which greatly limited our modeling power. In addition, as we mentioned in the *Introduction* section, we only considered predicting LOS once at the time of the admission. To have dynamic and more accurate predictions, our model can also include time-series data after patient's admission, such as treatments, lab results, and heart rate monitoring, so that our model can update its prediction every 12-24 hours, depends on the needs of the hospital.

Since we have made many assumptions about our optimization model, in order for it to match the real-world complexity, we need to add more constraints and consider more cases. For example, segregating patients according to their illness is crucial. Patients that are infectious must be separated from noncontagious patients. Also, infrastructure requirements must be met. For instance, a patient with a head injury needs to be allocated to a room that has an MRI machine. Beyond that, instead of maximizing the total number of beds utilized, we can also conduct more research on what types of patients' are more urgent in needing a bed allocation, and thus can give them high priorities in our optimization. In other words, we can work with doctors and develop a system of reward rules, and consequently optimize with respect to total rewards. Lastly, our estimate on patients' LOS is definitely not perfect, and there can be cases such that we predict a patient to stay in the hospital for 7 days while the patient ends up staying for 10 days. Therefore, for future work, we need to take this uncertainty into consideration and extend our MIO formulation to robust optimization.

7 Conclusion

In this paper, we demonstrated how machine learning models can be used to help hospitals make an accurate forecast on the patient's length of stay. Among the 4 models we built for prediction, Boosted Trees and Random Forests obtained better predictive performance, while CART and ORT provided better interpretability. The choice of which model to use depends on the business situation. Moreover, with the prediction of length of stay in hand, we showed how Mixed Integer Optimization can allocate beds in the most optimal fashion, thus greatly reduces empty beds in the hospital. Finally, future work is necessary to transform our methodology into a more tailored and realistic setting based on different hospital's needs.

References

Bertsimas, Dimitris, et al. "Predicting inpatient flow at a major hospital using interpretable analytics."

Johnson, A., Pollard, T., Mark, R. (2016). MIMIC-III Clinical Database. PhysioNet. doi:10.13026/C2XW26

Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database. Scientific Data, 3, 160035.

Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

International Statistical Classification of Diseases and Related Health Problems. World Health Organization, 2016.

Bertsimas, D. and Dunn, J. (2019). Machine Learning Under a Modern Optimization Lens. Part II, Optimal Trees for Classification and Regression. Chapter 10, Optimal Regression Trees with Constant Prediction, pp. 179-202. Chapter 12, Optimal Regression Trees with Linear Predictions, pp. 203-218.

Appendix

Link to the code: https://github.com/mikejin96/Hospital-Length-of-Stay-Prediction

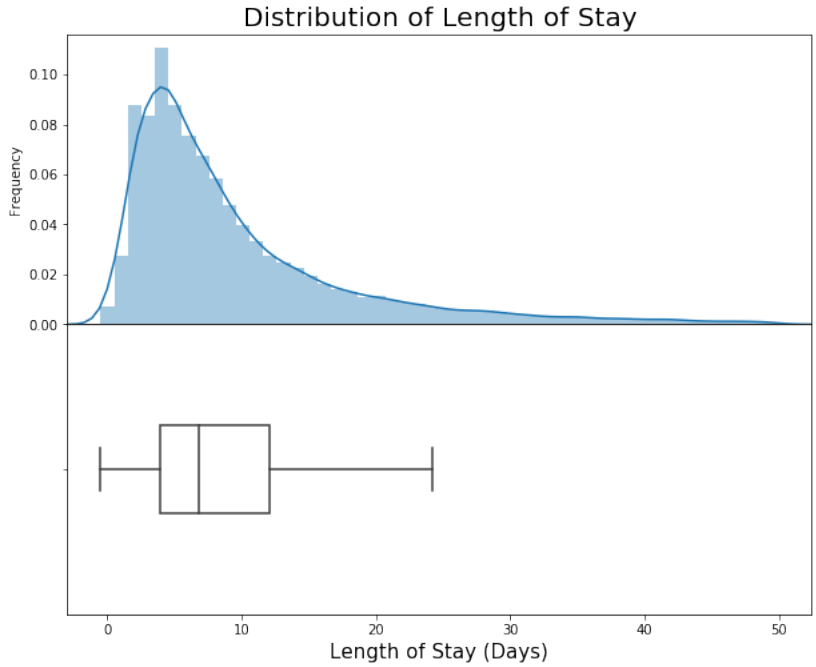

Figure 4: Distribution of LOS

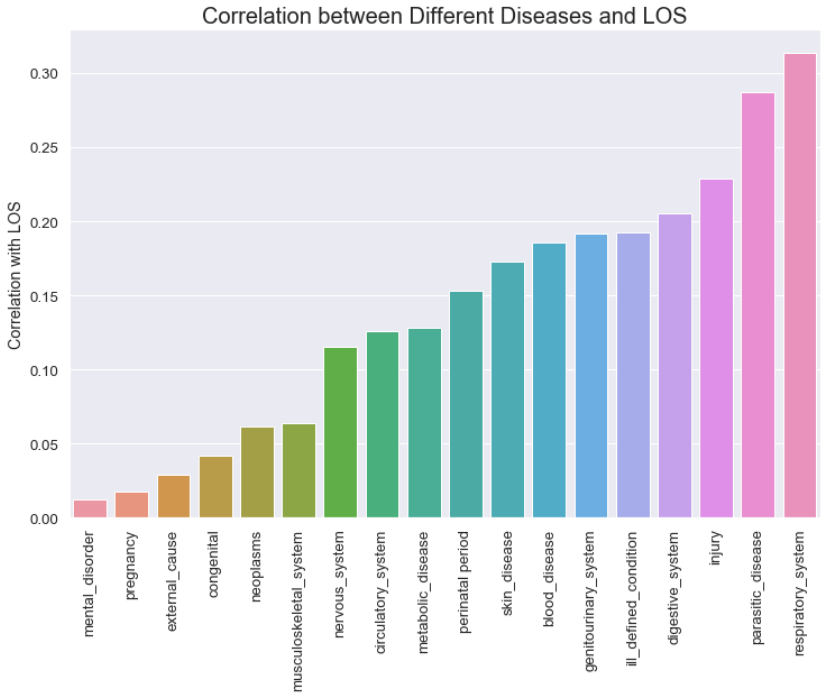

Figure 5: Correlation between Different Diseases and LOS

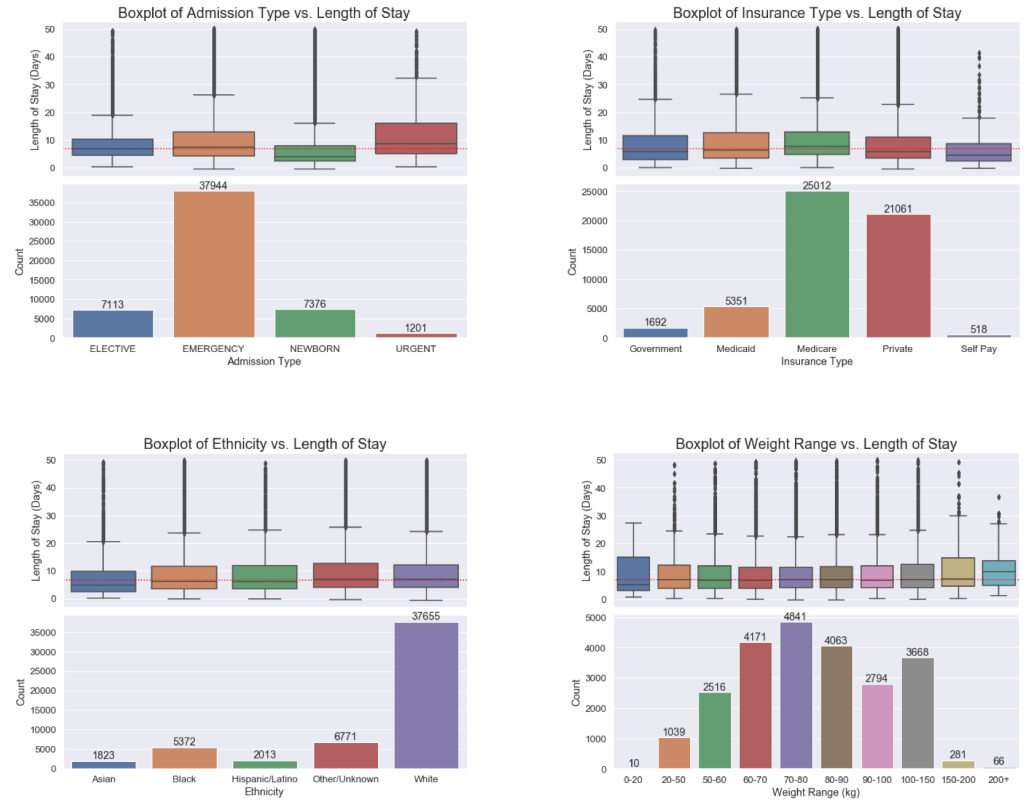

Figure 6: Boxplots of Admission Type (Top Left), Insurance Type (Top Right), Ethnicity (Bottom Left), and Weight (Bottom Right) vs. LOS

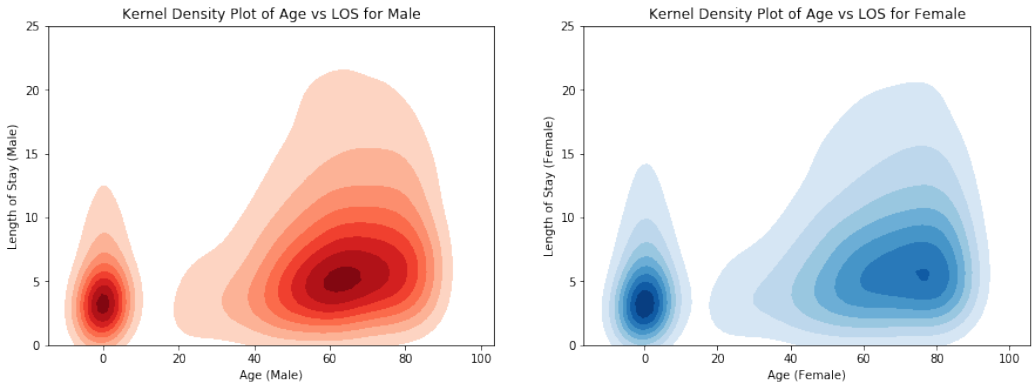

Figure 7: Kernel Density Plot of Age vs. LOS for Male (Left) and Female (Right)

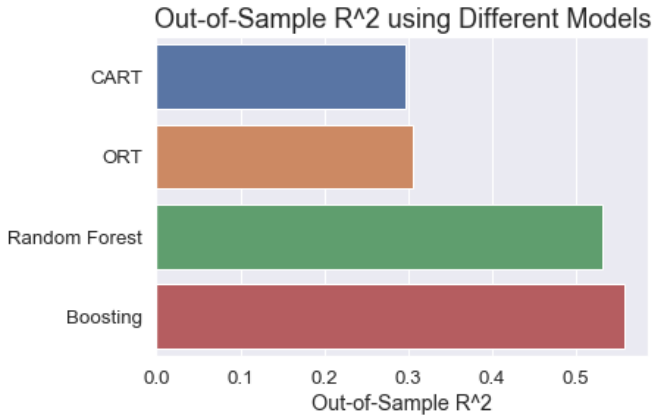

Figure 8: Out of Sample R-squared for the 4 Models

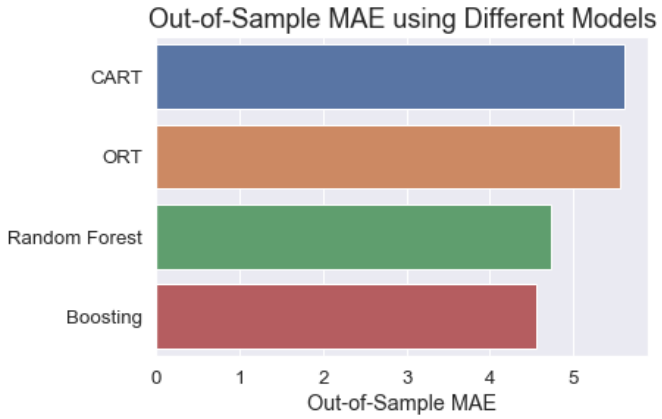

Figure 9: Out of Sample MAE for the 4 Models

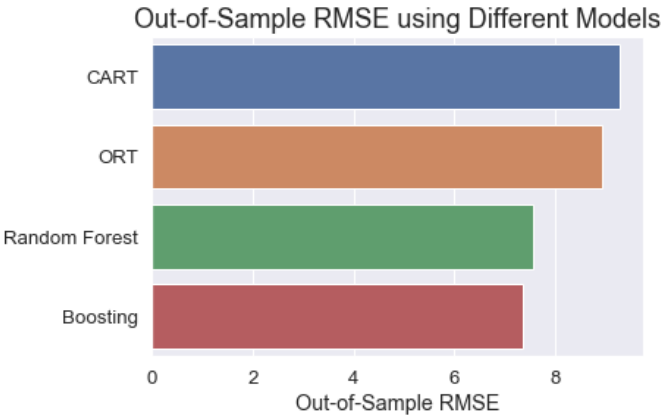

Figure 10: Out of Sample RMSE for the 4 Models

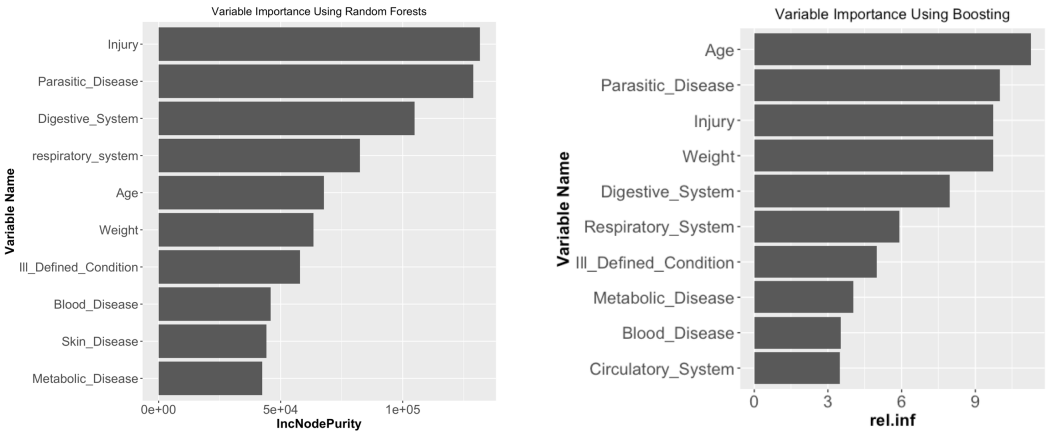

Figure 11: Variance Importance from Random Forests (Left) and Boosting (Right)

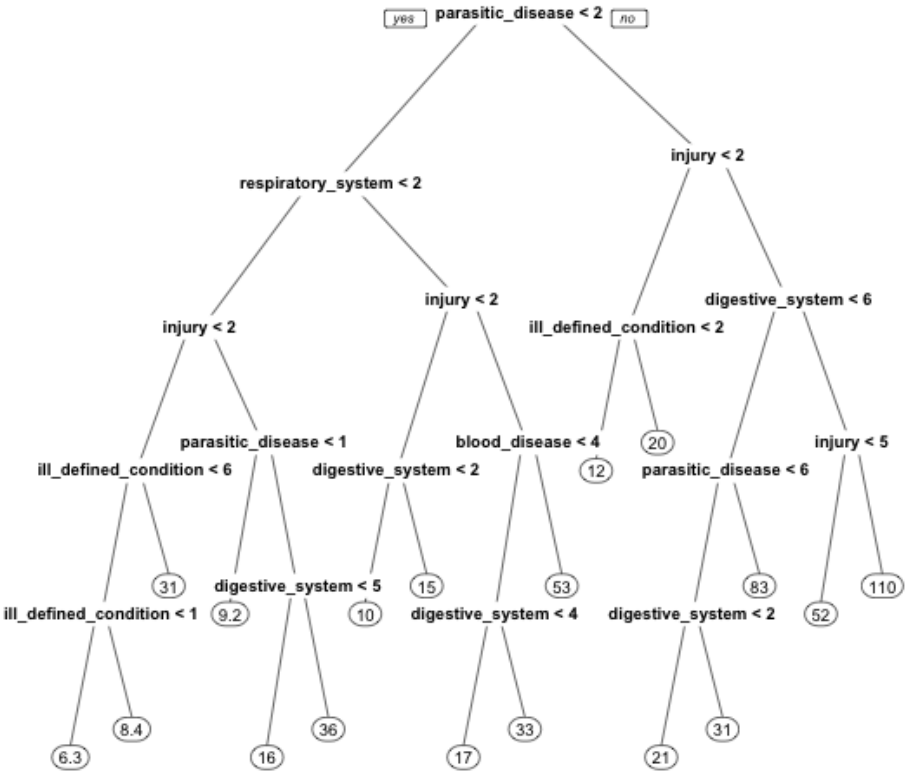

Figure 12: CART with Complexity Parameter of 0.005