Supplementary Materials

Database

We evaluated deidentified 8,947 patients who underwent cardiac surgery between April 2006 and January 2018 at Yale University affiliated hospital. The patient data and outcomes were collected based on the STS Adult Cardiac Surgery Database collection tool (version 2.81) which includes 765 variables related to preoperative (287 variables + 13 variables for demographics), operative (384 variables), and postoperative (81 variables) assessment (Fig S1).

The purpose of this study was to evaluate the power of machine learning (ML) technique for predicting patient outcomes in cardiac surgery. However, our data source is limited to 8,947 patients from a single institution. The sample size is one of the primary factors affecting the model performance. On average, 10,000 – 100,000 samples are required to achieve effective prediction. The STS risk models are built based on 100,000 – 500,000 samples\textsuperscript{1,2}. From 2006 to 2018, data collection form has been updated four times. The figure above shows distribution of patient populations used in our analysis. With our data, if we were to evaluate the effects of database amendment, five separate models need to be made with less than 1,000 samples, which
is extremely small sample size for building a predictive model. To accommodate all 8,947 patients and achieve our research goal of evaluating the power of ML technique in cardiac surgery, we refined data based on a single version (Version 2.81) which may have generated missing values attributed to the discrepancies by version updates (e.g. new variables added or variables deleted) – Version 2.81 was used to test the most recent STS risk models (2018 risk models) which was developed based on Version 2.73. We evaluated the impact of missing values on the predictive modeling using different data imputation techniques and demonstrated robustness of the ML model (e.g XGboost, Fig S9, Fig S10).

We would like to address that even with considerable amount of missing data generated due to the data refinement with respect to the single version of data collection tool, we were able to show successful ML performance, demonstrating its potential.

Why binary classification?

We used one versus all approach (binary classification) to evaluate the ML performance for predicting one of the three outcomes (‘Discharged’, ‘Died’, and ‘Readmission’). If it were multiclassification, the probability that the patient died, readmitted, and discharged will result in the probability added up to 100%, which can be inappropriate approach from clinical and medical perspective since these outcomes often overlie with each other. It will be more of interest to know the probability of patient died, or was readmitted, or discharged respectively, in which binary classification is more likely to be appropriate and provides more insight and information.

Data processing

Step 1. Data cleaning
The data cleaning step included simplifying categories, feature engineering, and eliminating variables with 100% missing values (Fig 1A). For example, redundant categories were merged into a single category. For the categorical variable having more than two categories except ‘No’ and ‘Unknown’, the number of each category was investigated and the category which occupied less than 2% of the entire data (< 200) was merged into the parent category if available. For feature engineering, among 13 categories related to health insurance information in data collection tool we considered only three types of health insurance: Medicare, Medicaid, and Commercial Health Insurance composed of ‘Yes’ and ‘No’ categories to simplify the information. Among three variables related to patient locations: zipcode, address, and country, since the zipcode includes address and country information only the zipcode was considered as a valid predictor. Two new variables were created based on the zipcode, one composed of first two digits representing state and city and the other composed of the first digit only representing state. Datetime variables were modified into duration variables. For example, ‘Admit Date’, ‘Date of Surgery’, and ‘Date of Discharge’ were newly formatted as ‘Admission to surgery’, ‘Surgery to discharge’, and ‘Admission to discharge’ which represent durations from admission to discharge, surgery to discharge, and admission to discharge respectively.

Step 2. Data imputation

Data distribution for preoperative, operative, and postoperative variables shown in Fig S1 shows substantial amount of missing values in the Raw data. The Filtered data where variables with 100% missing values were eliminated had 73% missing values overall. Three different data sets were prepared by applying different imputation techniques: 1. Data imputation with a rough approximation (e.g. mean imputation and statistical imputation) for all variables (Fig S3), 2.
Data imputation with singular value decomposition (SVD)\textsuperscript{3} for the selected variables where missing values were less than 30\%, and 3. Data imputation with SVD\textsuperscript{3} and the random-forest-based imputation method (i.e. MissForest)\textsuperscript{4} for all variables (Fig 1, the \textit{Full data}).

For the initial analysis to investigate impact of data processing on ML prediction performance with RUSBoosted Trees algorithm, mean imputation and statistical imputation were utilized in the process of data imputation (Fig S3). For more detailed analysis including data visualization and classification (Fig 1), singular vector decomposition technique was utilized for data imputation (Fig S9, Fig S10).

\textbf{Mean imputation} was performed for numerical variables. For example, there were 14 missing values in Surgery to discharge and Admission to discharge which is equivalent to 0.16\% of total values. Four of them were classified as ‘Discharged’ and the rest were ‘Died’. There were significant differences among mean values of ‘Admission to surgery’, ’Surgery to discharge’, and ‘Admission to discharge’ for ‘Discharged, ‘Died’, and ‘Readmission’. Those 14 missing values were replaced with the mean value based on patient outcomes. Missing values in LwstTemp were replaced with the mean values based on LwstTempSrc as there were significant differences in mean values of ‘LwstTemp’ for each category of ‘LwstTempSrc’: ‘Bladder’, ‘Other’, and ‘Unknown’. Missing data for height and weight were replaced with the mean value of corresponding gender. Missing data for other numerical variables (e.g. WBC, Hct, TotBlrbn) were filled up with their individual mean values.

\textbf{Statistical imputation} was performed for several categorical variables. For example, the imputation for the 48\% missing values of admit source (denoted as AdmitSrc in the STS data collection tool version 2.81) were performed based on Admission to surgery variable. Unbalanced one-way ANOVA was used to compare mean values of the four different AdmitSrc
categories (‘Elective admission’, ‘Emergency department’, ‘Other’, and ‘Transfer in from another acute care facility’) under Admission to surgery. All groups except the means for ‘Other’ and ‘Transfer in from another acute care facility’ were significantly different from each other (p-value<0.05). Hence, categories of ‘Other’ and ‘Transfer in from another acute care facility’ were merged into ‘Other’. The missing AdmitSrc were filled up with the category that provides the minimum difference between the corresponding Admission to surgery values and the mean Admission to surgery value of each category.

If the variable has ‘Unknown’ or ‘Not documented’ as the category, its missing values were replaced with this category. If the categories are composed of ‘Yes’ or ‘No’ only, a new category ‘Unknown’ has been introduced for missing values.

**Singular value decomposition (SVD)**\(^3\) was utilized to fill up missing values and process the *Filtered* data to the *Full* data (Fig 1). To apply SVD, the missing values denoted by NaN were first initialized to their respective column means. Then SVD was performed on the filled-in matrix and the data matrix was then reconstructed from the SVD. The convergence was tested using relative change in trace norm (changes in the sum of singular values which are square root of the eigenvalues representing stretching or shrinking magnitude). This procedure of missing value approximations using SVD technique was repeated until convergence (1E-3).

In addition to SVD, the **random-forest-based imputation method (RF, i.e. MissForest)**\(^4\) was used for missing value imputation. The RF is a non-parametric method which can cope with different types of variables simultaneously. It is known to perform very well under barren conditions like high dimensions, complex interactions and non-linear data structure.\(^4, 5\) Similar to SVD, the missing values denoted by NaN are first initialized to their respective column means followed by iterative imputation scheme. The algorithm adjusts predictions on missing values
each iteration and the iterations continues until the stopping criteria is met (i.e. when the
difference between iteration i and i+1 of the imputed dataframes begins to increase). We used the
MissForest function from the missingpy (i.e. missingpy 0.2.0) in the python library.

**ML models**

**Logistic regression (LR)** is a classification algorithm that transforms linear combinations of
variables into probability values between 0 and 1 using logistic sigmoid function. The data is
then classified into two groups at the decision boundary of 0.5, answering the binary questions,
yes or no.

**Adaptive Boosted (AdaBoosted) Trees, Boostrap Aggregating (Bagged) Trees, Subspace
discriminant, Subspace K-nearest neighbor (KNN), and Random under-sampling Boosted
(RUSBoosted) Trees** are ensemble models which utilize collection of multiple base models
instead of drawing result from one single base model (e.g. LR, Decision tree). The ensemble
model first splits the data into several subsets based on the selected variables (e.g. age) and their
thresholds (e.g. age of 50). Each subset produces predictive values following the chosen learner
type (e.g. Decision tree, Discriminant, Nearest neighbors). Then, the model provides the final
predictive value by combining the results from the subsets. The type of the model algorithm (e.g.
AdaBoost, Bag, Subspace, RUSBoost) determines the way it creates subsets (e.g. data splitting
rules) and aggregates the result (e.g. taking average at the end of the process vs. taking weighted
average in a sequential way). For example, Bag splits the data at the optimal split point chosen
by scanning all variables and thresholds while AdaBoost splits the data in a random way. The
Boosting algorithms (e.g. AdaBoost, RUSBoost) model subsets sequentially based on the results
from the previous subsets and aggregate the result by computing weighted average, while Bag or
Subspace processes the subsets in parallel and aggregates the results at the end. The Subspace algorithm is similar to Bag except that it splits data at the randomly selected variable rather than the specific variable selected as the optimized split points. The RUSBoost is designed to handle imbalance problem by combining the standard boosting algorithm AdaBoost with random under-sampling (RUS) technique, which randomly removes samples from majority class until the desired balance is achieved. The Discriminant learner type classifies the data by finding linear or non-linear boundaries that maximizes variance between the classes. The K-nearest neighbor (KNN) classifies the data based on a similarity measure (e.g. distance) within its K number of nearest neighbors.

Boosting algorithms (e.g. AdaBoost, RUSBoost) utilize gradient descent (partial derivative of loss function) to minimize the error. The gradient gives directions to the minimum of the loss function by assigning correspondent weight values. The size of each descending step is given by the parameter called learning rate. **XGboost** is also a boosting algorithm but known for extreme gradient boosting algorithm because it uses second-order gradients (second partial derivatives of the loss function) that provides more information in terms of the directions to achieve the minimum of the loss function, therefore can improve the performance with further reduced error.

In this paper, we used XGboost python package at [https://pypi.org/project/xgboost/](https://pypi.org/project/xgboost/). The descriptions of XGBoost parameters used in this study are shown in Table S2.

**Data visualization**

The correlation matrix heatmap was utilized to visualize correlations among the variables. The correlation of each variable with respect to the three outcome variables (‘Discharged’, ‘Died’, and ‘Readmission’) at multi-level (e.g. ‘Discharged’=0, ‘Died’=1, ‘Readmission’=2) (Fig 4) and binary level (e.g. positive class=1 and negative class =0) (Fig S5) was measured. By ranking the
correlation values, total of 15 top ranking variables were selected (five from preoperative, operative, and postoperative respectively).

For clustering, t-Distributed Stochastic Neighbor Embedding (t-SNE)\(^6\) - an unsupervised, non-linear technique was used with respect to different variable sets selected as shown in Table 1. These variable sets were determined based on the data fulfillment ratio for each cohort labeled by eight types of designated surgical procedures in the STS report: (1) Isolated CABG, (2) Isolated AVR, (3) Isolated MVR, (4) AVR+CABG, (5) MVR+CABG, (6) AVR+MVR, (7) MV repair, (8) MV repair + CABG. The hypothesis behind the clustering analysis was that each cohort has distinctive features or characteristics contributing to the clustering of the patient populations by the cohort.

**Classification**

For classification, cohort-level and patient-level analysis were performed with respect to the patient outcomes classified into ‘Discharged’, ‘Died’, and ‘Readmission’ following the flow chart shown in Fig S2. In the cohort-level analysis, STS risk scores calculated for ‘Predicted Risk of Mortality (PredMort)’\(^2\) were compared with prediction scores produced by XGboost model. In the patient-level analysis, six other ML models (LR model, AdaBoosted Trees, Bagged Trees, Subspace discriminant, Subspace KNN, and RUSBoosted Trees) in addition to the XGboost model were tested and compared in terms of their predictive performance. For XGboost model, the Bayesian hyperparameter optimization\(^7\) was applied to see its influence on the classification performance. Three to five-fold cross-validation was used for the cohort-level analysis (due to small number of positive classes) (Table S1) and five to ten-fold for the patient-
level analysis. The setting details and descriptions of each ML model are provided in Table 2, Table S2, and the ML models section above.

As an initial analysis, the systematic impact of data cleaning, imputation, and data formats on classification was investigated using RUSBoosted Trees with the four data types (Raw data: Raw, Data before filling up missing variables: BeforeImp, Data after filling up missing variables: AfterImp, and Data composed of all numeric values by converting categorical variables to numerical: AfterImp_numeric) (Fig S3). With the ‘AfterImp_numeric’ data type, the predictive performance of the different ML models were systematically investigated with six different data sets constructed based on the combinations of preoperative, operative and postoperative variables such that: 1. Preoperative variables only (Pre), 2. Operative variables only (Op), 3. Postoperative variables only (Post), 4. Pre+Op, 5. Pre+Post, and 6. Pre+Op+Post.

Further analysis and results

a. Predictability by data processing

The data processing including data cleaning, imputation, and format conversion showed systematic improvement of predictive performance for all outcomes such that increase in AUC-ROC from 0.56 to 0.77 for ‘Discharged’, 0.60 to 0.94 for ‘Died’, and 0.56 to 0.75 for ‘Readmission’ (Fig S3).

b. Predictability by data sets

According to the ROC curve (Fig S11), Post performed the best (p<0.05) for predicting ‘Discharged’ and ‘Died’ compared to Pre and Op, while Pre performed the best (p<0.05) for predicting ‘Readmission’ compared to Op and Post. Adding Op to Pre improved the performance for ‘Discharged’ and ‘Died’, but not for ‘Readmission’. Adding Post to either Pre
or *Op* improved the performance (p<0.05) for all outcomes up to 0.80, 0.98, and 0.76 for ‘Discharged’, ‘Died’, and ‘Readmission’ respectively. Adding *Pre* or *Pre+Op* to *Post* improved the performance (p<0.05) for ‘Discharged’ and ‘Readmission’, but not for ‘Died’. The *Op* showed the lowest predictability (p<0.05) for all outcomes compared to *Pre* and *Op*. The *Pre+Op+Post*, which had the greatest number of predictor variables does not necessarily performed the best, demonstrating that increasing variable does not always improve the performance. For example, adding *Op* to *Pre+Post* for predicting ‘Died’ or to either *Pre* or *Pre+Post* for predicting ‘Readmission’ had no impact, while improved predictability for ‘Discharged’. This result shows the importance of the predictor selection which would impact the computational time of the analysis.

The PR curve showed different predicting trend from ROC curve (Fig S12). While ROC curve ranked the predictability for each outcome in the order of ‘Died’ (AUC 0.98), ‘Discharged’ (AUC 0.80), and ‘Readmission’ (AUC 0.76), PR curve ranked in order of ‘Discharged’ (AUC 0.95), ‘Died’ (AUC 0.74), and ‘Readmission’ (AUC 0.18). Adding variables made no difference in predicting ‘Discharged’ and ‘Readmission’. Adding *Pre, Post*, or the both to *Op* had impact on predicting ‘Died’ and ‘Readmission’ because *Op* itself had poor performance. Adding *Op* or *Post to Pre* had impact on predicting ‘Died’, but no impact on either ‘Discharged’ or ‘Readmission’, which may suggest that adding good predictors potentially improve the performance while adding poor predictors not necessarily worsen the performance but has no impact for improvement. For predicting ‘Died’ and ‘Readmission’ with *Op* only was poor (p<0.05) compared to *Pre* and *Op*, but showed comparative predictability for ‘Discharged’. The PR curve also showed that *Pre+Op+Post* does not necessarily perform the best, addressing the importance of predictor selection which could be related to optimizing the computational time.
c. *Predictability by six different ML models*

In addition to XGboost model, six different ML models (LR and five ensemble models: AdaBoosted Trees, Bagged Trees, Subspace Discriminant, Subspace KNN, and RUSboosted Trees, were tested for the patient-level analysis.

Among five ensemble models, Subspace KNN required substantial amount of computational time (i.e. training time of ~3000 sec, 100 fold more training time than Bagged Trees) with relatively poor performance (AUC 0.68 for ‘Discharged’, 0.88 for ‘Died’, 0.62 for ‘Readmission’ vs. average AUCs from other ensemble models: 0.79 for ‘Discharged’, 0.97 for ‘Died’, 0.74 for ‘Readmission’). Subsequently, Subspace KNN was not considered for further analysis.

All ensemble methods outperformed the classic LR model for all six data sets (*Pre, Op, Post, Pre+Op, Pre+Post, and Pre+Op+Post*) according to both ROC and PR curves (Fig S11 & S12). The ensemble methods combined with hyperparameter optimization may further improve the performance as demonstrated in the cohort-level study (Fig 5).

Similar to the XGboost model, all additional ML models tested showed different performance results according to which model metrics used whether ROC curve or PR curve. While the ROC curve showed outstanding ML performance for predicting ‘Died’ (AUC 0.97) followed by ‘Discharged’ (AUC 0.79) and ‘Readmission’ (AUC 0.74), the PR curve showed relatively poor performance with different ranks: the highest predictability for ‘Discharged’ (AUC 0.87) followed by ‘Died’ (AUC 0.68) and ‘Readmission’ (AUC 0.17).

According to the ROC curve, AdaBoosted Trees performed the best for predicting ‘Discharged’ while RUSBoosted Trees performed the best for predicting ‘Died’. For predicting
'Readmission’, there were no significant differences among AdaBoosted Trees and RUSBoosted Trees.

According to the PR curve, Subspace Discriminant and RUSBoosted Trees performed the best for predicting ‘Discharged’ while AdaBoosted Trees performed the best for predicting ‘Died’. For predicting ‘Readmission’, there were no significant differences among AdaBoosted Trees, Subspace Discriminant, and RUSBoosted Trees. The predictability by data processing and predictability by data sets are shown in Fig S3, Fig S11, and Fig S12.

**ROC curve vs PR curve**

The study utilized two metrics to provide quantitative measurements of predictive performance, the ROC curve and the PR curve. The PR curve is recommended when dealing with a highly skewed dataset as the PR curve may provide more informative pictures regarding the relevant results while the ROC curve may present an overly optimistic view of the ML performance. In this study, the excellent performance seen from the ROC curve did not agree with the PR curve. The ROC curve reflects the influence of the true negative (patients not readmitted being correctly classified), while the PR curve focuses on the positive class (patients actually readmitted) and unconcerned about the true negative. Therefore, if the patient who got readmitted is misclassified as not readmitted, it will create large changes in the PR curve but small changes in the ROC curve. If a large number of patients not readmitted is correctly classified (true negative), the influence of such effect can be relatively large in the ROC curve compared to the result of false positive (patients readmitted & incorrectly classified), while the PR curve would have minimal effects with increased true negatives. The study also shows that algorithms that optimize the AUC-ROC do not optimize the AUC-PR as demonstrated by Davis et al\(^8\). For example, Subspace Discriminant showed comparable performance with RUSBoosted Trees and
AdaBoosted Trees according to the ROC curve, but Subspace Discriminant performed significantly worse than RUSBoosted Trees and AdaBoosted Trees according to the PR curve (Fig S11 and S12). However, the curve dominates in PR space dominates in ROC space.

Study Limitations

Some of the missing outcomes were filled up with assumptions following the flow chart shown in Fig S2, which can produce bias in the model. For example, all missing at ‘Discharge’, ‘Readmission’, and ‘Mortality’ record were 12 patients and they were considered as ‘Died’. Patients having ‘Discharge’ data but neither ‘Readmission’ nor ‘Mortality’ were considered as ‘Discharged’. The variables collected based on the STS Adult Cardiac Surgery Database collection tool present a large portion of missing data (73%) (Fig S1). This can be attributed to refining data based on a single version (2.81) of the STS data collection tool where data collection form has been updated four times from 2006 to 2018. However, if we were to evaluate the effects of database amendment throughout years, five separate models need to be made with less than 1,000 samples, which is extremely small sample size for building a predictive model. To accommodate all 8,947 patients and achieve our research goal of evaluating the power of machine learning technique in cardiac surgery, we used a single version (Version 2.81) to preprocessing the data. Further detailed rationale for using the database of a single version is provided in ‘Database’ section in this supplementary material. The variables related to a specific procedure (e.g. valve procedure) can also be a reason for generating missing values since patients associated with valve procedure is relatively small number than those with the non-valvular procedure. Nonetheless, the XGboost model, an extreme gradient boosting algorithm showed
successful predictive performance with missing values (Fig 6), and data imputation had no impact (Fig S9 and Fig S10).
Tables

Table S1. Percentage and number of endpoint events by cohorts. Histogram of the population distributions for nine cohorts and the endpoint events are shown in Fig 2.

<table>
<thead>
<tr>
<th>Endpoint Events</th>
<th>Discharged</th>
<th>Died</th>
<th>Readmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n = 8,947)</td>
<td>8,015 (89.6%)</td>
<td>230 (2.6%)</td>
<td>702 (7.8%)</td>
</tr>
<tr>
<td>Isolated CABG (n = 2,971)</td>
<td>2,695 (90.7%)</td>
<td>54 (1.8%)</td>
<td>222 (7.5%)</td>
</tr>
<tr>
<td>Isolated AVR (n = 957)</td>
<td>866 (90.5%)</td>
<td>8 (0.8%)</td>
<td>83 (8.7%)</td>
</tr>
<tr>
<td>Isolated MVR (n = 236)</td>
<td>195 (82.6%)</td>
<td>6 (2.6%)</td>
<td>35 (14.8%)</td>
</tr>
<tr>
<td>AVR+CABG (n = 473)</td>
<td>424 (89.7%)</td>
<td>12 (2.5%)</td>
<td>37 (7.8%)</td>
</tr>
<tr>
<td>MVR+CABG (n = 69)</td>
<td>58 (84.1%)</td>
<td>3 (4.3%)</td>
<td>8 (11.6%)</td>
</tr>
<tr>
<td>AVR+MVR (n = 56)</td>
<td>48 (85.7%)</td>
<td>2 (3.6%)</td>
<td>6 (10.7%)</td>
</tr>
<tr>
<td>MV Repair (n = 322)</td>
<td>288 (89.4%)</td>
<td>1 (0.3%)</td>
<td>33 (10.3%)</td>
</tr>
<tr>
<td>MV Repair + CABG (n = 83)</td>
<td>68 (81.9%)</td>
<td>3 (3.6%)</td>
<td>12 (14.5%)</td>
</tr>
<tr>
<td>Other procedures (n = 3,780)</td>
<td>3,373 (89.2%)</td>
<td>141 (3.7%)</td>
<td>266 (7.1%)</td>
</tr>
</tbody>
</table>
Table S2. **Hyperparameter tuning for XGboost model.** The Bayesian hyperparameter optimization was applied using the python library. Hyperparameters associated with the learning process in XGboost model were tuned with respect to the ROC curve to improve prediction scores. The following values are examples of parameters used for predicting mortality in the AVR + CABG cohort where the hyperparameter tuning successfully improved prediction scores from 0.5 up to 0.71 (Fig 5). Number of learners: Number of subsets, Learning_rate: Step size for gradient descent, Max_depth: Maximum number of splits, Min_child_weight: Minimum sum of instance weight needed to split a node, Gamma: Minimum loss reduction required to make a further split, Reg_alpha: Regularization term on weights, Reg_lambda: Regularization term on weights, Subsample: The fraction of the data to be used in training, Scale_pos_weight: Control the balance of the classes.

<table>
<thead>
<tr>
<th></th>
<th>Default (hyperparameter)</th>
<th>Tuning with respect to ROC curve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of learners</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>Learning_rate</td>
<td>0.05</td>
<td>0.4032083917998946</td>
</tr>
<tr>
<td>Max_depth</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Min_child_weight</td>
<td>5.0</td>
<td>4</td>
</tr>
<tr>
<td>Gamma</td>
<td>0.5</td>
<td>0.00015936523535755285</td>
</tr>
<tr>
<td>Reg_alpha</td>
<td>0.0</td>
<td>0.1611980387486336</td>
</tr>
<tr>
<td>Reg_lambda</td>
<td>1.0</td>
<td>387</td>
</tr>
<tr>
<td>Subsample</td>
<td>0.9</td>
<td>0.8391548832503206</td>
</tr>
<tr>
<td>Scale_pos_weight</td>
<td>1.0</td>
<td>171</td>
</tr>
<tr>
<td><strong>Prediction scores</strong></td>
<td><strong>ROC 0.5±0.0</strong></td>
<td><strong>ROC 0.71±0.13</strong></td>
</tr>
</tbody>
</table>
Figures

Fig S1. Distributions of preoperative, operative, and postoperative variables in the Raw data. The histogram visualizes the proportion of missing values in each time series data.
Fig S2. Patient outcome classification flow chart. Outcomes of total 8,947 patients were classified into ‘Discharged’, ‘Died’, and ‘Readmission’ based on the data recorded under the section of ‘Mortality’, ‘Discharge’, and ‘Readmission’ in the Society of Thoracic Surgeons (STS) Adult Cardiac Surgery Database (version 2.81). The resultant population distributions are shown in Fig 2.
Fig S3. Data processing effects on ML predictability – preliminary results. RUSBoosted Trees ML model was used to evaluate data processing effects on ML predictability. The receiver operating characteristic (ROC) curve was used for the performance measurement. Data processing steps of data cleaning, imputation, and data format conversion were involved in creating four data types, Raw, BeforeImp, AfterImp, and AfterImp_numeric as shown in the flow chart (left). The area under the ROC curves following the data processing steps demonstrate systematic improvements on predictability for all outcomes, ‘Discharged’, ‘Died’, and ‘Readmission’ (right).
**Fig S4. Systematic approach for clustering study.** Each data point was color-coded by the corresponding cohort to visualize commonalities that may exist among different cohorts. The clustering technique applied to all 748 variables, which visualizes 8,947 patients characterized by 748 dimensions in a two-dimensional space (tsne_x and tsne_y), results in numerous sparse clusters (#1). Selected variables based on non-missing value proportion by each cohort as specified in Table 1 were systematically applied in the clustering study. The application of a variable set in the clustering study combined sparsely distributed clusters (#1) as shown in #2. The sequential additions of the variable set such as $a_8 + a_7$ (#3), $a_8 + a_7 + a_5$ (#4), $a_8 + a_7 + a_5 + a_4$ (#5), $a_8 + a_7 + a_5 + a_4 + a_3$ (#6), $a_8 + a_7 + a_5 + a_4 + a_3 + a_2$ (#7), and $a_8 + a_7 + a_5 + a_4 + a_2 + a_1$ (#8) gradually separated data points and formed coherent clusters. Note that $a_6$ is excluded since the number of selected variables for $a_6$ was zero (Table 1). The addition of the first two majority variable set, $a_8$ and $a_2$, resulted in eight coherent clusters (#9). The detailed anatomy of this #9 cluster can be seen in Fig 3.
Fig S5. Correlation matrix created by individual outcomes at binary-level. The top five variables which have high correlations with the outcome were selected for each time series (preoperative, operative, and postoperative) data, thus a total of fifteen variables were used to create a correlation matrix for each outcome. The selected variables for ‘Died’ and ‘Readmission’ are similar (11 out of 15 variables are identical). The correlation matrices of these two outcomes also show similar patterns and trends where the selected variables are highly correlated to each other while their correlations to the corresponding outcome are relatively low. However, no common variables exist between those selected for ‘Discharged’ and the other two outcomes (‘Died’ and ‘Readmission’) and its correlation matrix shows a different pattern such that low correlations overall.
Fig S6. Cohort-level predictive performance by ML for predicting readmission. Predictive performance for each cohort is shown by the receiver operating characteristic (ROC). The performance for predicting readmission was significantly low compared to predicting mortality (Fig 5) for all cohorts (AUC-ROC > 0.6). While the predictability measured by the ROC curve was relatively good, that by the PR curve showed poor performance (AUC less than 0.5), suggesting that predicting patients who are not readmitted (majority group) can be more achievable than predicting patients readmitted (minority group).
Fig S7. Predictive performance in ‘Other procedures’ cohort. The ‘Other procedures’ cohort includes patients who underwent other procedures (e.g. aortic aneurysm, aortic dissection) not associated with the eight designated procedures in STS reports. The majority of patients were classified into the ‘Other procedures’ cohort (3,780 out of 8,947 as shown in Table S1 and Fig 2) and no STS risk model exists for predicting mortality for the ‘Other procedures’ cohort.

Predicting readmission in the ‘Other procedures’ cohorts showed significantly better performance than in any other cohorts (Fig S7).
Fig S8. Predictive performance by selected variables. The predictive performance is represented by the receiver-operating characteristic (ROC) curve. The area under the ROC curves for predicting mortality using preoperative, operative, and postoperative variables respectively under three different conditions: (A) using all variables (295 for preoperative, 369 for operative, and 84 for postoperative); (B) using selected variables from feature importance plot (Fig 7: 30 for preoperative, 21 for operative, and 16 for postoperative) and masking missing values; and (C) using selected variables from feature importance plot (Fig 7) and filling up missing values using singular value decomposition (SVD) method. The area under the ROC curves remained the same for the three conditions.
Fig S9. Data imputation on the variables with less than 30% missing values. The singular value decomposition (SVD) technique was used to fill up the missing values. 114 variables out of 295 in the Filtered data of preoperative dataset were less than 30% missing values. Data imputation had no significant impact on improving ML predictability compared to the ML performance without data imputations (Fig 6).
Fig S10. No imputation vs data imputation on all missing values for predicting ‘Died’. Two different algorithms were utilized to fill up the missing values, the singular value decomposition (SVD) technique and the random-forest-based imputation method (i.e. MissForest). Neither method had an impact on improving ML predictability. While the data imputation on the variables with less than 30% missing values slightly improved the predictability compared to data without imputation (Fig S9), the data imputation on all missing values rather decreased predictability which can be possibly due to bias introduced.
Fig S11. ML performance measured by the receiver operating characteristic (ROC) curve for different data sets. The six different data sets were constructed based on the combinations of preoperative, operative and postoperative variables: Set 1. Preoperative variables only (Pre), Set 2. Operative variables only (Op), Set 3. Postoperative variables only (Post), Set 4. Pre+Op, Set 5. Pre+Post, and Set 6. Pre+Op+Post. In each data set, five different ML models were applied for pairwise comparison (Logistic regression (LR), AdaBoosted Trees, Bagged Trees, Subspace Discriminant, and RUSBoosted Trees). AdaBoosted Trees performed the best for predicting ‘Discharged’ while RUSBoosted Trees performed the best for predicting ‘Died’. For predicting ‘Readmission’, there was no significant difference between AdaBoosted Trees and RUSBoosted Trees. Post performed the best (p<0.05) for predicting ‘Discharged’ and ‘Died’ compared to Pre and Op, while Pre performed the best (p<0.05) for predicting ‘Readmission’ compared to Op and Post. Adding Op to Pre improved the performance for ‘Discharged’ and ‘Died’, but not for ‘Readmission’. Adding Post to either Pre or Op improved the performance
(p<0.05) up to 0.80, 0.98, and 0.76 for ‘Discharged’, ‘Died’, and ‘Readmission’ respectively. Adding $Pre$ or $Pre+Op$ to $Post$ improved the performance (p<0.05) for ‘Discharged’ and ‘Readmission’, but not for ‘Died’. The $Op$ showed the lowest predictability (p<0.05) for all outcomes compared to $Pre$ and $Op$. The $Pre+Op+Post$, which had the greatest number of predictor variables did not perform the best, demonstrating that increasing variables does not always improve the performance. For example, adding $Op$ to $Pre+Post$ for predicting ‘Died’ or to either $Pre$ or $Pre+Post$ for predicting ‘Readmission’ had no impact, while it improved predictability for ‘Discharged’. This result showed the importance of the predictor selection which would impact the computational time of the analysis. ‘Boost’ label shown in legend indicates ‘AdaBoost’ algorithm.
Fig S12. ML performance measured by the precision-recall (PR) curve for different data sets. The six different data sets were constructed based on the combinations of preoperative, operative and postoperative variables: Set 1. Preoperative variables only (Pre), Set 2. Operative variables only (Op), Set 3. Postoperative variables only (Post), Set 4. Pre+Op, Set 5. Pre+Post, and Set 6. Pre+Op+Post. In each data set, five different ML models were applied for pairwise comparison (Logistic regression (LR), AdaBoosted Trees, Bagged Trees, Subspace Discriminant, and RUSBoosted Trees). The PR curve showed the different predicting trends from the ROC curve (Fig S11). While ROC curve ranked the predictability for each outcome in the order of ‘Died’ (AUC 0.98), ‘Discharged’ (AUC 0.80), and ‘Readmission’ (AUC 0.76), the PR curve ranked in order of ‘Discharged’ (AUC 0.95), ‘Died’ (AUC 0.74), and ‘Readmission’ (AUC 0.18). The Subspace Discriminant and RUSBoosted Trees performed the best for predicting ‘Discharged’ while Boosted Trees performed the best for predicting ‘Died’. For predicting ‘Readmission’, there was no significant difference among Boosted Trees, Subspace Discriminant, and RUSBoosted Trees. Adding variables made no difference in predicting
‘Discharged’ and ‘Readmission’. Adding Pre, Post, or the both to Op had an impact on predicting ‘Died’ and ‘Readmission’. Adding Op or Post to Pre had an impact on predicting ‘Died’, but no impact on predicting either ‘Discharged’ or ‘Readmission’. For predicting ‘Died’ and ‘Readmission’ with Op only was poor (p<0.05) compared to Pre or Post only, but predicting ‘Discharged’ with Op only had similar performance to that with Pre or Post only. Similar to the ROC curve (Fig S11), the PR curve also showed that Pre+Op+Post did not perform the best, addressing the importance of predictor selection which will be associated with optimizing the computational time. ‘Boost’ label shown in legend indicates ‘AdaBoost’ algorithm.
Supplementary References


