

Seminar

Predicting Mortality of ICU Patients:
The PhysioNet/Computing in Cardiology Challenge 2012

Bruno T. Scodari

Outline

Topic	Page
Overview	3
Section 1: Explore	7
(A) Patient attributes & outcomes	8
(B) Aggregated features	9
(C) Longitudinal features	14
Section 2: Wrangle	19
Section 3: Train	23
Section 4: Evaluate	27
Conclusions	32

Overview

The goal of this project was to develop a model to predict in-hospital death using the PhysioNet 2012 data

Motivation

- Prediction of patient-specific outcomes is useful for 1) clinical decision making, 2) improving patient outcomes, and 3) optimizing resource allocation
- Modeling the trajectory of disease or patient outcomes using demographic and/or hospital data is a challenging task, especially for Intensive Care Unit (ICU) patient populations

Objectives

- Using the 2012 PhysioNet dataset (N=4000), the goal is to predict **if a patient will die in the hospital** using baseline patient characteristics and time series data collected over the span of 48 hours
- Specifically, teams in this challenge were evaluated by their ability to maximize ***min(Recall, Precision)***; in other words, we are interested in predicting the positive class but must balance FPs and FNs

Features

There were 4 baseline and 37 time series features, recorded at various time points across 48 hours

Baseline Age (years), Gender (0: female, or 1: male), Height (cm),
ICUType (1: Coronary Care Unit, 2: Cardiac Surgery Recovery Unit, 3: Medical ICU, or 4: Surgical ICU)

Dynamic	<u>Albumin</u> (g/dL)	<u>HCT</u> [Hematocrit (%)]	<u>PaCO2</u> [partial pressure of arterial CO ₂ (mmHg)]
	<u>ALP</u> [Alkaline phosphatase (IU/L)]	<u>HR</u> [Heart rate (bpm)]	<u>PaO2</u> [Partial pressure of arterial O ₂ (mmHg)]
	<u>ALT</u> [Alanine transaminase (IU/L)]	<u>K</u> [Serum potassium (mEq/L)]	<u>pH</u> [Arterial pH (0-14)]
	<u>AST</u> [Aspartate transaminase (IU/L)]	<u>Lactate</u> (mmol/L)	<u>Platelets</u> (cells/nL)
	<u>Bilirubin</u> (mg/dL)	<u>Mg</u> [Serum magnesium (mmol/L)]	<u>RespRate</u> [Respiration rate (bpm)]
	<u>BUN</u> [Blood urea nitrogen (mg/dL)]	<u>MAP</u> [Invasive mean arterial blood pressure (mmHg)]	<u>SaO2</u> [O ₂ saturation in hemoglobin (%)]
	<u>Cholesterol</u> (mg/dL)	<u>MechVent</u> [Mechanical ventilation respiration (0:false, or 1:true)]	<u>SysABP</u> [Invasive systolic arterial blood pressure (mmHg)]
	<u>Creatinine</u> [Serum creatinine (mg/dL)]	<u>Na</u> [Serum sodium (mEq/L)]	<u>Temp</u> [Temperature (°C)]
	<u>DiasABP</u> [Invasive diastolic arterial blood pressure (mmHg)]	<u>NIDiasABP</u> [Non-invasive diastolic arterial blood pressure (mmHg)]	<u>Tropl</u> [Troponin-I (µg/L)]
	<u>FiO2</u> [Fractional inspired O ₂ (0-1)]	<u>NIMAP</u> [Non-invasive mean arterial blood pressure (mmHg)]	<u>TropT</u> [Troponin-T (µg/L)]
	<u>GCS</u> [Glasgow Coma Score (3-15)]	<u>NISysABP</u> [Non-invasive systolic arterial blood pressure (mmHg)]	<u>Urine</u> [Urine output (mL)]
	<u>Glucose</u> [Serum glucose (mg/dL)]		<u>WBC</u> [White blood cell count (cells/nL)]
	<u>HCO3</u> [Serum bicarbonate (mmol/L)]		<u>Weight</u> (kg)

Prevailing modeling strategy

Rather than modeling “long” data, my prevailing hypothesis was that discretizing time and transforming the data into “wide” format would be an optimal strategy primarily due to the sample size and computational constraints

“Long” data

		X_1	X_2	X_3
P_1	T_1			
	T_2			
	T_3			
	T_4			
P_2	T_1			
	T_2			
	T_3			
	T_4			

P_i : the i th person
 T_i : the i th continuous time point
 X_i : the i th feature
 B_i : the i th discrete (binned) time point

Form discrete
time “bins”

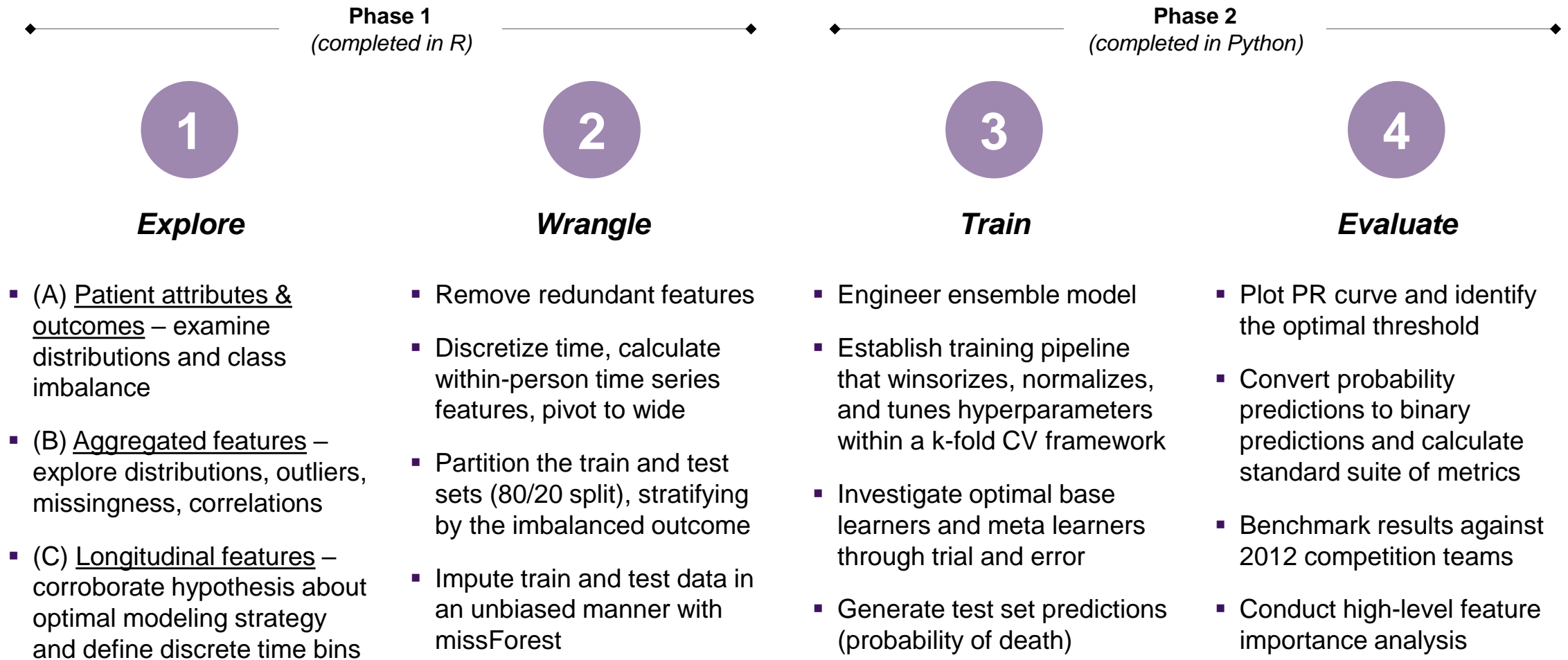
			X_1	X_2	X_3
P_1	B_1	T_1			
		T_2			
	B_2	T_3			
		T_4			
P_2	B_1	T_1			
		T_2			
	B_2	T_3			
		T_4			

“Wide” data, where X_i have been summarized by some function

	B_1			B_2		
	X_1	X_2	X_3	X_1	X_2	X_3
P_1						
P_2						

Approach

This project was carried out in four distinct steps: explore, wrangle, train, and evaluate



Section 1: Explore

(A) Patient attributes by outcome

As a first step, I started by examining the distributions of patient attributes by the outcome

	Survived (N=3446)	Died (N=554)	P-value ¹
Age			<0.01
<50	764 (22.2%)	72 (13.0%)	
50-59	549 (15.9%)	61 (11.0%)	
60-69	599 (17.4%)	83 (15.0%)	
70-79	698 (20.3%)	117 (21.1%)	
80+	836 (24.3%)	221 (39.9%)	
Gender			0.34
Female	1498 (43.5%)	253 (45.7%)	
Male	1946 (56.5%)	300 (54.2%)	
Missing	2 (0.1%)	1 (0.2%)	
Height			0.01
Short (<165 cm)	569 (16.5%)	94 (17.0%)	
Average (165-180 cm)	895 (26.0%)	111 (20.0%)	
Tall (180+ cm)	381 (11.1%)	56 (10.1%)	
Missing	1601 (46.5%)	293 (52.9%)	
ICU Type			<0.01
Cardiac Surgery Recovery Unit	831 (24.1%)	43 (7.8%)	
Coronary Care Unit	496 (14.4%)	81 (14.6%)	
Medical ICU	1206 (35.0%)	275 (49.6%)	
Surgical ICU	913 (26.5%)	155 (28.0%)	

¹Calculated via a chi-squared test for independence

- There is severe class imbalance, with 86% of the patients surviving and 14% of the patients dying in-hospital; this requires special attention during train/test splits and training
- The distributions of patient baseline features vary significantly across the different outcome classes
- A lot of patients are missing height values, which should be considered for imputation

(B) Aggregating the dynamic features for exploration

Given the prevailing modeling strategy, dynamic features were aggregated by taking the mean (ignoring missing values) for each person; this simplifies exploration of the data, as now we have one row per patient

“Long” data

		X_1	X_2	X_3
P_1	T_1			
	T_2			
	T_3			
	T_4			
P_2	T_1			
	T_2			
	T_3			
	T_4			

P_i : the i th person

T_i : the i th continuous time point

X_i : the i th feature

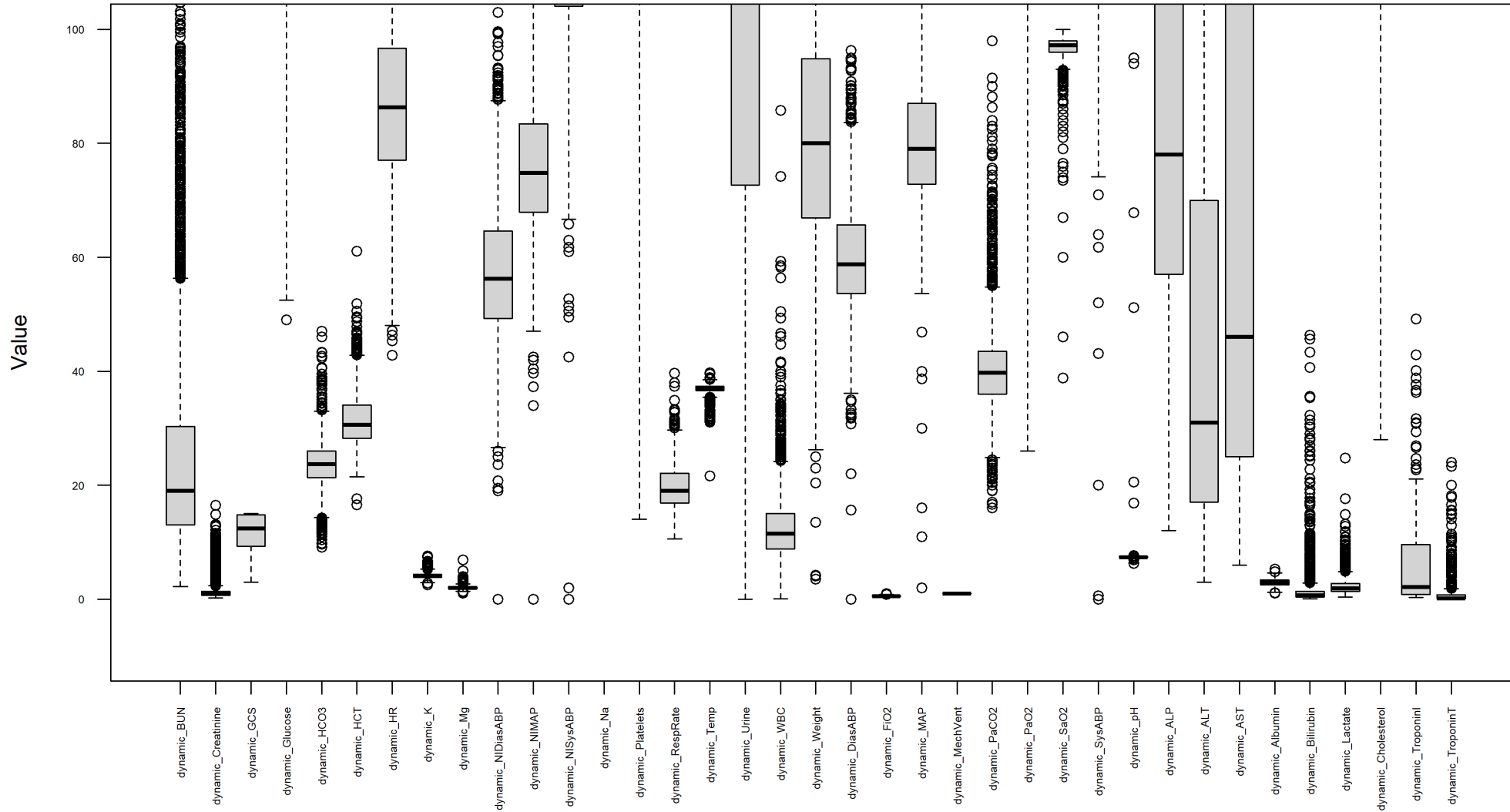
“Wide” data, where X_i were obtained by taking the mean

	X_1	X_2	X_3
P_1			
P_2			

This simplifies exploration of the dynamic features and allows us to look at things like pairwise correlations, etc.

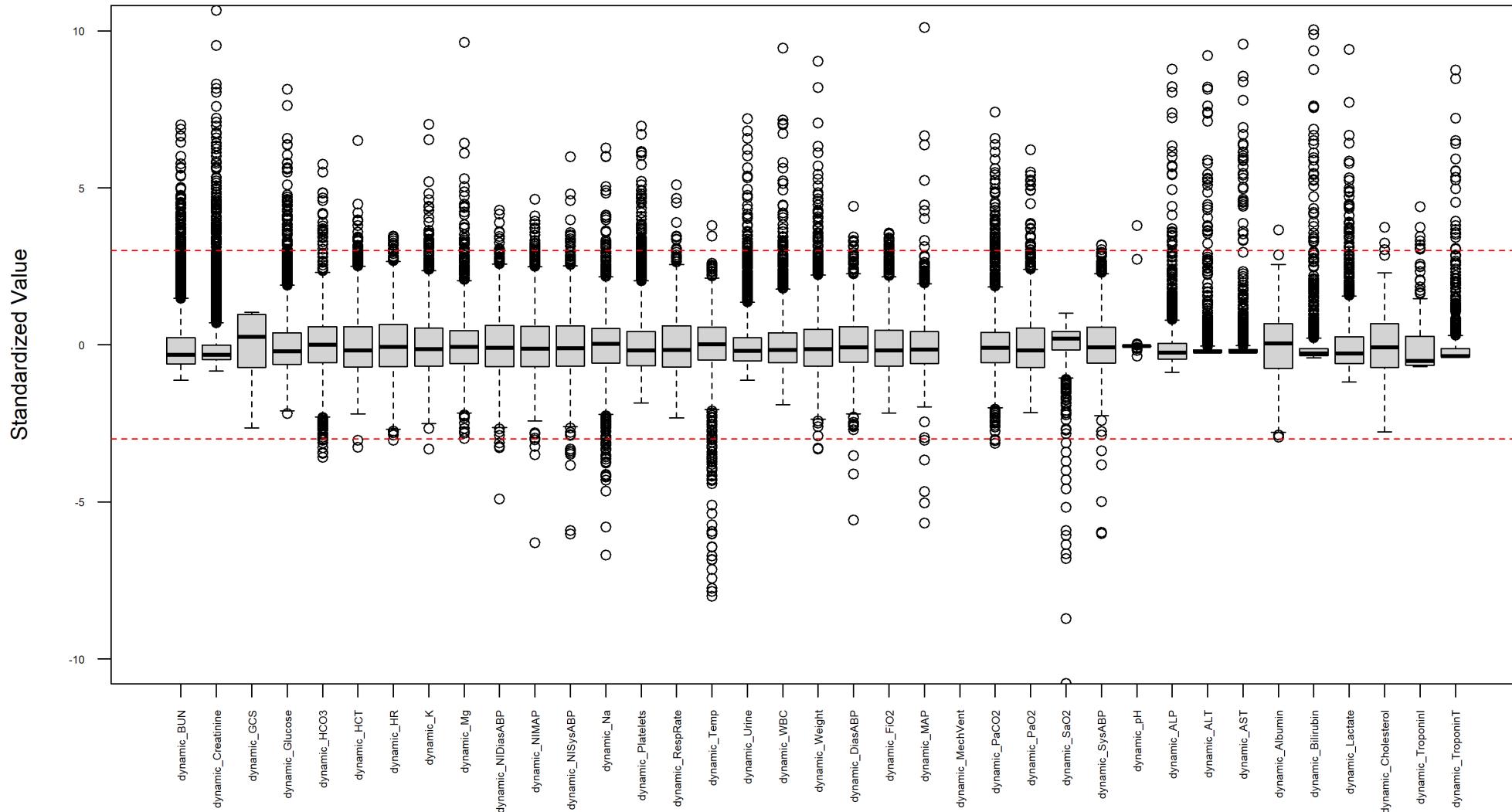
(B) Distributions of dynamic features (1 of 2)

Analyzing the distributions of each feature reveals that they are on completely different scales; normalization during modeling is key, otherwise features with higher variation will dominate



(B) Distributions of dynamic features (2 of 2)

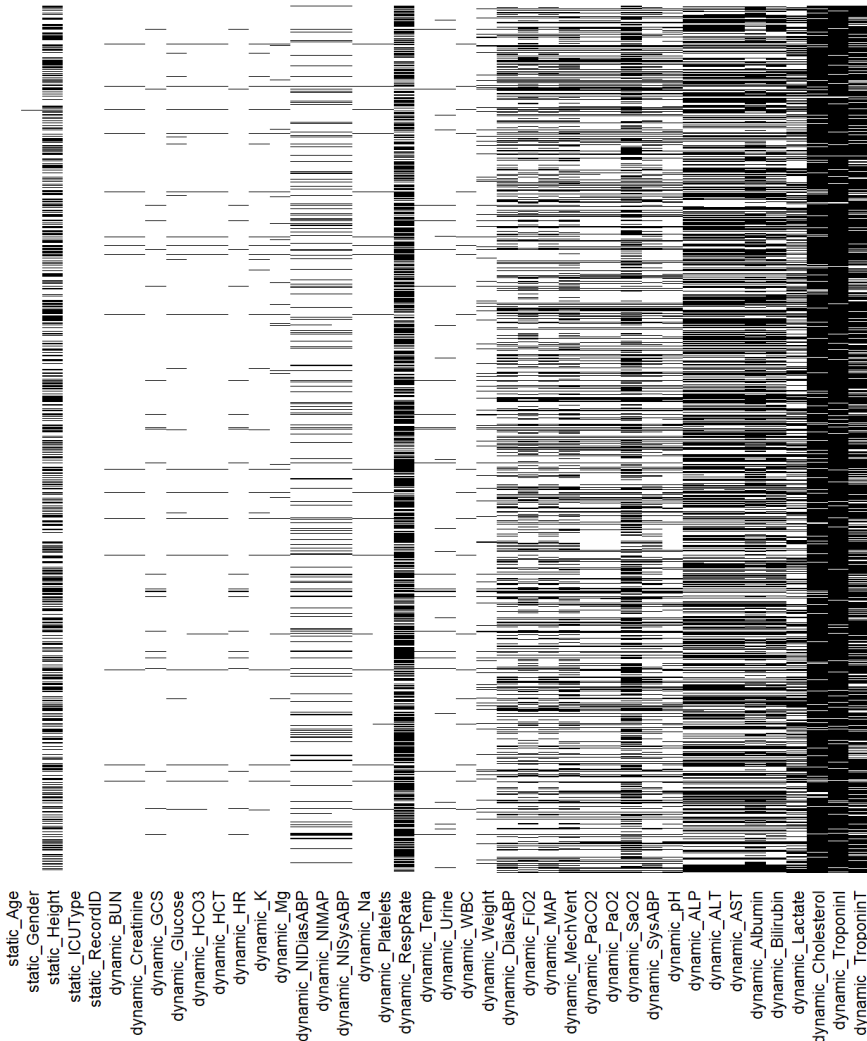
Upon standardizing the features (Z-score), much of the data still appeared to be “outliers” >3 deviations away from the mean; winsorization or clipping of values during modeling should help deal with this issue



(B) Missingness patterns

An analysis of missingness revealed that some features may be removed and that the data may be MAR

Heatmap of missingness (in black) by feature

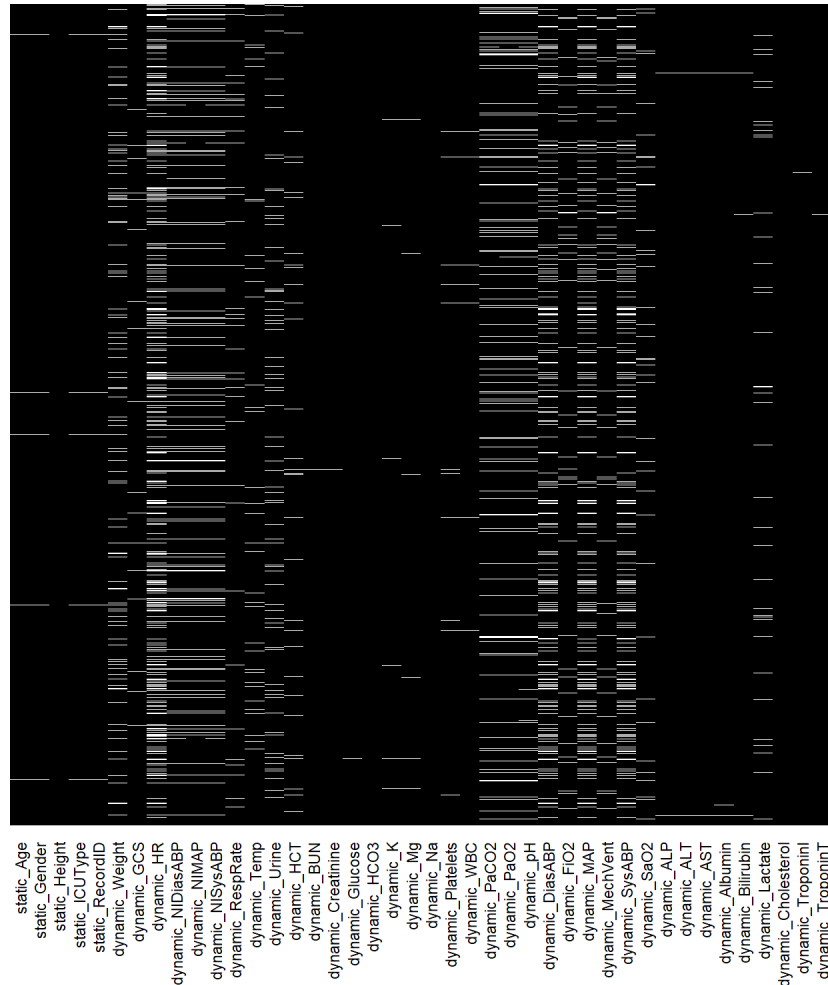


- Extreme missingness observed in Troponin I/T variables and Cholesterol ($\approx 80\%+$)
- Some evidence that data may be missing at random (MAR), or in other words, the missingness in one variable is explained by missingness in another (i.e., they are correlated)
- This motivates potentially 1) removing Troponin I/T and Cholesterol, and 2) using imputation methods tailored for MAR data

(C) Corroborating the modeling strategy

Lastly, to understand if modeling “long” data was viable, I examined the available dynamic features across time

Heatmap of missingness (in black) by feature, ordered by patient and continuous time



- There is substantial missingness in this data, with most features having 80%+ missingness
- Due to the sparsity and number of observations, imputing this data would be very challenging
- Evidence supports transforming to wide format and leveraging tabular ML algorithms

(C) Going from “long” to “wide”

Recall that transforming the data from “long” to “wide” requires careful selection of a discrete time bin

“Long” data

		X_1	X_2	X_3
P_1	T_1			
	T_2			
	T_3			
	T_4			
P_2	T_1			
	T_2			
	T_3			
	T_4			

Choosing a discrete time bin is critical

			X_1	X_2	X_3
P_1	B_1	T_1			
		T_2			
	B_2	T_3			
		T_4			
P_2	B_1	T_1			
		T_2			
	B_2	T_3			
		T_4			

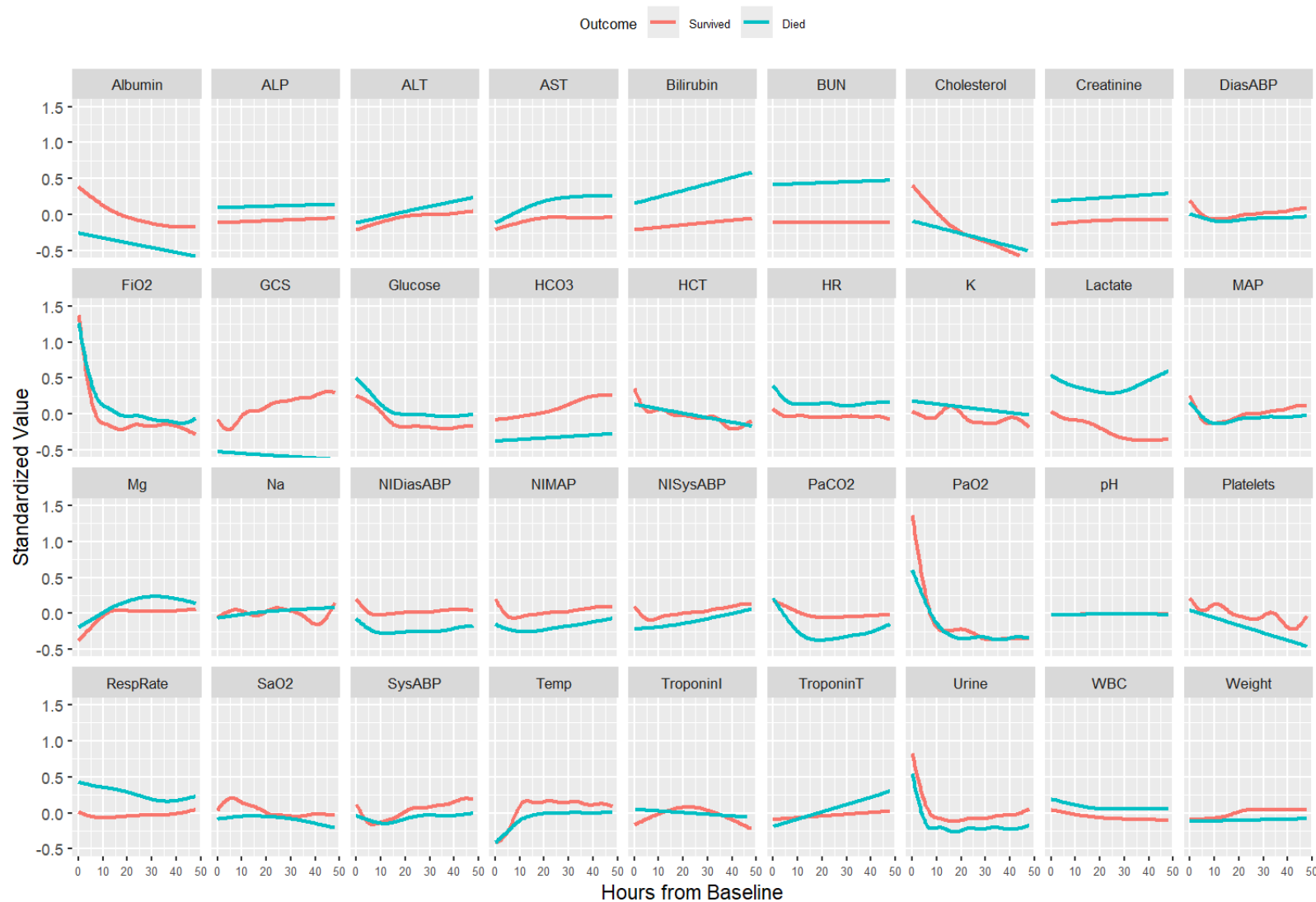
P_i : the i th person
 T_i : the i th continuous time point
 X_i : the i th feature
 B_i : the i th discrete (binned) time point

“Wide” data, where X_i have been summarized by some function

	B_1			B_2		
	X_1	X_2	X_3	X_1	X_2	X_3
P_1						
P_2						

(C) Identifying a discrete time window (1 of 2)

To inform the optimal discrete time window, I looked for trends in the dynamic features across continuous time



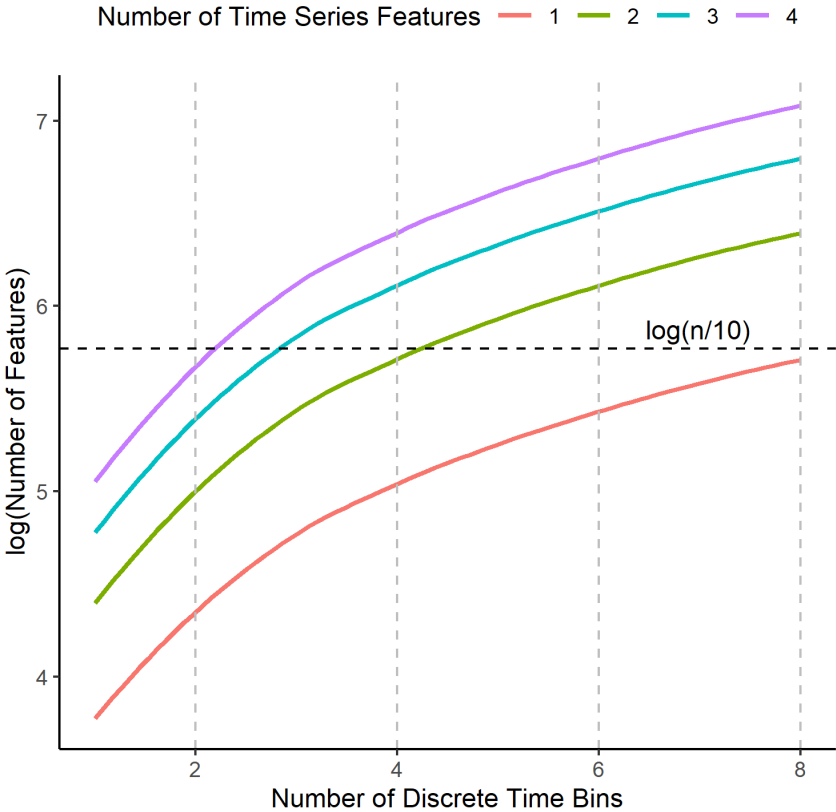
- I plotted the average standardized values for patients in both classes across continuous time, using splines
- There was some evidence of oscillatory or cyclic patterns in some variables (e.g., K, Platelets)
- Can make an argument that discretizing time by two 24-hour bins would capture these cycles

(C) Identifying a discrete time window (2 of 2)

Given the explosion in cardinality and diminishing completeness that is observed when increasing the number of discrete time bins, I elected to go with two 24-hour bins

As you increase the number of discrete time bins, the number of features in the training data (cardinality) explodes...

As you increase the number of discrete time bins, the proportion of dynamic features with at least one complete observation decreases...



	One 48-hr bin	Two 24-hr bins	Four 12-hr bins	Six 8-hr bins	Eight 6-hr bins
Bin 1	0.70	0.69	0.62	0.57	0.53
Bin 2		0.62	0.52	0.49	0.45
Bin 3			0.49	0.44	0.44
Bin 4			0.48	0.42	0.38
Bin 5				0.44	0.38
Bin 6				0.40	0.40
Bin 7					0.40
Bin 8					0.35
Avg	0.70	0.65	0.53	0.46	0.42

Summary of exploration

Data type	Visualization(s)	Question(s)	Implications & decisions
(A) Patient attributes & outcomes	Table 1	<ul style="list-style-type: none"> Is there outcome imbalance? Are patient attributes balanced across the outcome? Are there missing patient attributes? 	<ul style="list-style-type: none"> Need to deal with outcome imbalance in train/test split and modeling Distribution of attributes vary Height is often missing – consider imputing
	Boxplots	<ul style="list-style-type: none"> Are the features on the same scale? Are there any outliers that need to be dealt with? 	<ul style="list-style-type: none"> Need to normalize during modeling Need to winsorize during modeling
	Heatmap	<ul style="list-style-type: none"> Are there missing features? If so, are there any missingness patterns to be gleaned? 	<ul style="list-style-type: none"> Tropinin I/F and Cholesterol have substantial missingness – consider removing Data looks MAR – consider regression-based imputation
(B) Aggregated features	Correlation matrix	<ul style="list-style-type: none"> Are the continuous variables correlated with the outcome and amongst each other? 	<ul style="list-style-type: none"> Tropinin I/F and Cholesterol are barely correlated with outcome, so okay to remove Others are correlated, which supports regression-based imputation
	Heatmap	<ul style="list-style-type: none"> How much missingness is there across time? Can this inform the optimal modeling strategy? 	<ul style="list-style-type: none"> Substantial missingness, which would be difficult to impute Best approach is to discretize time, pivot wide, and use tabular machine learning methods
	Spaghetti plots	<ul style="list-style-type: none"> Are there meaningful trends across time that may inform the discretization strategy? 	<ul style="list-style-type: none"> Can make an argument that two 24-hour bins captures adequate signal in the data
(C) Longitudinal features	Line plot, frequency table	<ul style="list-style-type: none"> Are there other things we should consider to inform the discretization strategy? Can we confirm an optimal time bin? 	<ul style="list-style-type: none"> As a function of the number of bins, consider cardinality and the proportion of columns with at least one observation Confirmed that two 24-hour bins is optimal for this problem

Section 2: Wrangle

Wrangling pipeline

 Incomplete data  Complete data

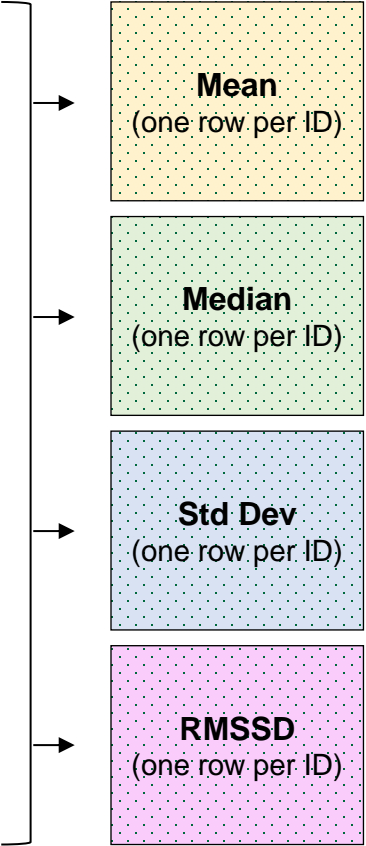
Split & Remove

Split dynamic and static features, remove Troponin I/F and Cholesterol, collapse static features to one row per patient



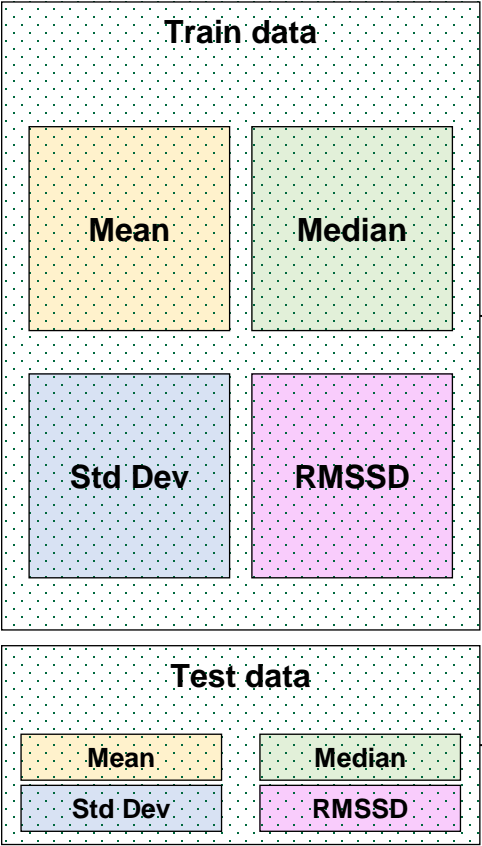
Engineer

Discretize time into two 24-hour bins, compute features for each patient-bin pair, pivot wide, join back static features



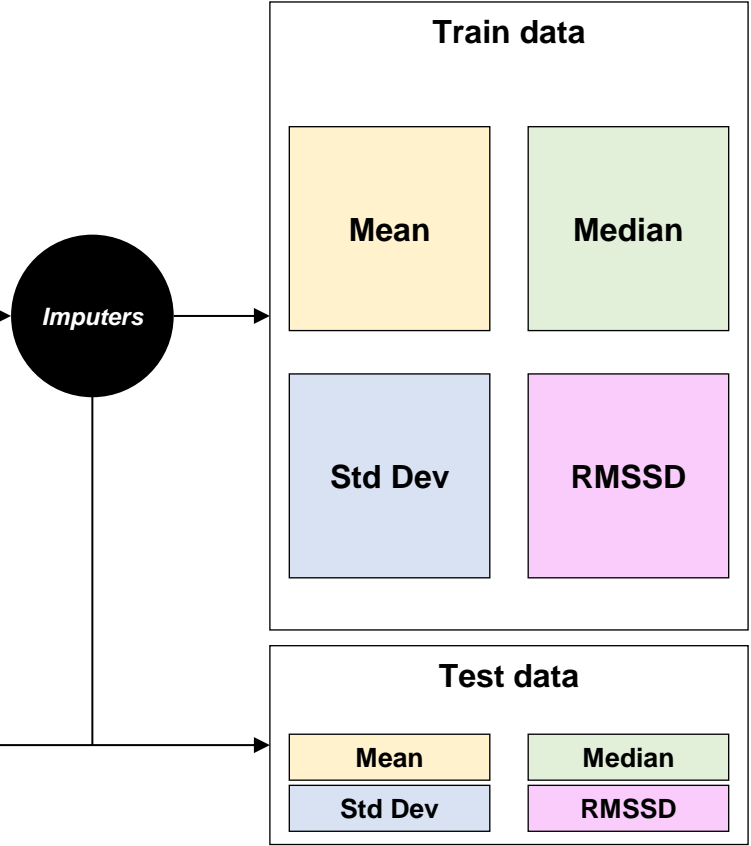
Partition

Set a seed and randomly partition the transformed data into train/test splits (80/20), stratifying by the imbalanced outcome



Impute

Fit missForest imputer objects to each train dataset; use the same object to impute the corresponding test data



Feature engineering (1 of 2)

The “engineer” phase involved 1) discretizing time into two 24-hour bins, 2) calculating four time series features for each patient-bin pair, and then 3) pivoting the data into wide format

Discretize time into two 24-hour bins

ID	Time	Time Bin	Predictor
A	1.3	1	x_1
A	4.5	1	x_2
A	7.8	1	x_3
A	12.0	1	x_4
A	21.0	1	x_5
A	24.6	2	x_6
A	29.0	2	x_7
A	40.0	2	x_8
A	42.1	2	x_9

Calculate time series features for each patient-bin pair

ID	Time Bin	Predictor – Mean	Predictor – Median	Predictor – Std Dev	Predictor – RMSSD
A	1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
A	2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$

Pivot the data into wide format, where there is one row per patient and 8 columns per predictor

ID	Predictor – mean (bin 1)	Predictor – median (bin 1)	Predictor – sd (bin 1)	Predictor – rmssd (bin 1)	Predictor – mean (bin 2)	Predictor – median (bin 2)	Predictor – sd (bin 2)	Predictor – rmssd (bin 2)
A	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$	$x_{1,6}$	$x_{1,7}$	$x_{1,8}$

- The static features are joined back to this data prior to imputing
- Some patients may not have observations in both periods, which induces missingness

Imputation strategy

I used missForest to impute train and test data, as it is a robust state-of-the-art method for MAR data

Requires:

X , our $n \times p$ matrix for imputation

γ , the stopping criterion (it works in iterations, stopping when the difference between iteration i and $i + 1$ of the imputed dataframes begins to increase for categorical and numeric variables, or after 10 iterations with the default `maxiter` parameter)

Algorithm:

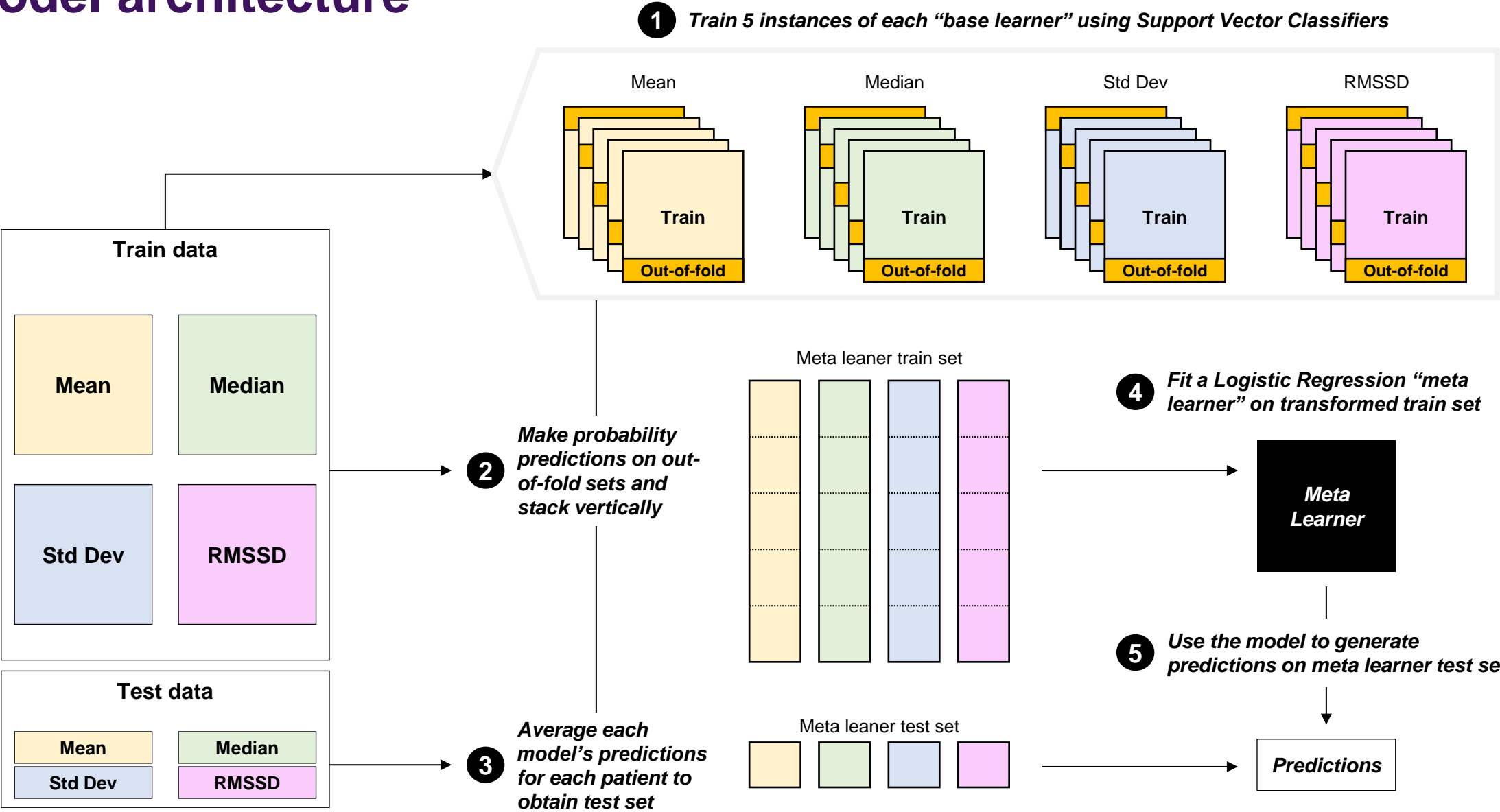
1. Make an initial guess for all missing categorical/numeric values (e.g. mean, mode)
2. $k \leftarrow$ vector of column indices in X , sorted in ascending order of % missing
3. while not γ do:
4. $X_{old}^{imp} \leftarrow$ store previous imputed matrix
5. for s in k do:
6. Fit a random forest predicting the non-missing values of X_s : $y_{obs}^{(s)} \sim x_{obs}^{(s)}$
7. Use this to predict the missing values of X_s : predict $y_{mis}^{(s)}$ using $x_{mis}^{(s)}$
8. $X_{new}^{imp} \leftarrow$ update imputed matrix, using the predicted $y_{mis}^{(s)}$
9. end for
10. update γ
11. end while
12. return the final imputed matrix X^{imp}

In words...

- *Impute all missing data using the mean/mode*
- *For each variable with missing values, fit a random forest on the observed part and then predict the missing part*
- *Repeat iteratively until a stopping criterion is met, or a maximum number of user-specified iterations is reached*
- *After the object is fit, apply it to the test data to preclude any “data leakage” (applies the same process using the same mean/modes and models)*

Section 3: Train

Model architecture



Training considerations

I considered the following components when training machine learning models

	Transformations ¹	Model choice	Tuned parameters	Tuning strategy	Metric optimized
<i>Base learners</i>	Winsorization, Z-score normalization	Support Vector Classifier (SVC)	Tuned: C Preset: kernel (rbf), class weight (balanced)	Grid search, within 5-fold CV framework	Average precision
<i>Meta learner</i>	Min-max normalization	Logistic Regression (LR)	Preset: class weight (balanced)	N/A – preset parameters	N/A – preset parameters
<i>Reasons</i>	<ul style="list-style-type: none"> Winsorization used to clip outliers Z-score is a common choice for continuous data on different scales Min-max is a common choice for probabilities 	<ul style="list-style-type: none"> SVC: outperformed the standard suite of models upon experimentation LR: linear model is typically chosen for meta learner, as the base learners are already flexible 	<ul style="list-style-type: none"> SVC: C controls regularization, which is commonly tuned; other parameters preset based on domain knowledge LR: class weight set to balanced based on the outcome imbalance 	<ul style="list-style-type: none"> Hyperparameter space was small enough to utilize a grid search on a local computer 	<ul style="list-style-type: none"> Average precision accounts for the trade-off between FPs and FNs when trying to predict TPs amidst class imbalance

¹The same transformations were made to train and test data to prevent “data leakage”

Optimizing for average precision

I trained my models to optimize for average precision, as this metric focuses on predicting TPs while minimizing both FNs and FPs in cases of class imbalance

		<u>Actuals</u>	
		D^+	D^-
<u>Predictions</u>	T^+	TP	FP
	T^-	FN	TN

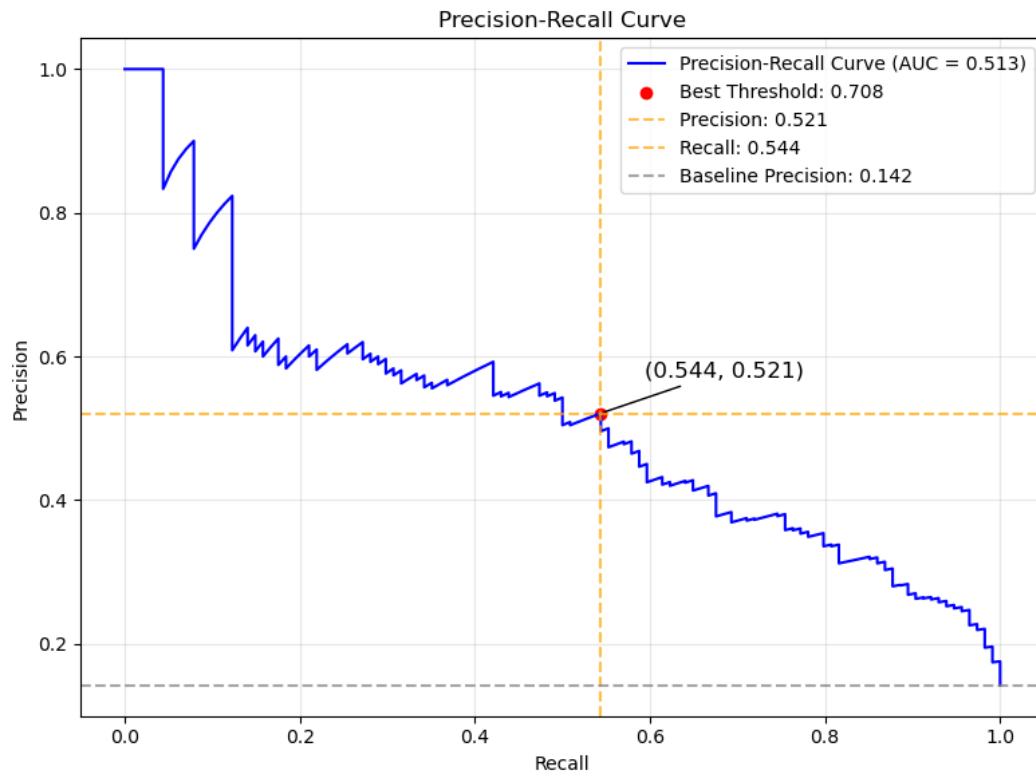
Metric	Formula
<i>Recall</i>	$P(T^+ D^+) = TP / (TP + FN)$
<i>Precision</i>	$P(D^+ T^+) = TP / (TP + FP)$
<i>Avg Precision</i>	$\sum_n (Recall_n - Recall_{n-1}) \times Precision_n$ <p>where n corresponds to a threshold</p>

As average precision is **threshold-agnostic**, I maximize it during training and then identify the threshold that maximizes $\min(Recall, Precision)$ in the test set

Section 4: Evaluate

Results

The average precision in the test set was 0.513 and the optimal threshold was 0.708, corresponding to a $\min(\text{Recall}, \text{Precision})$ of 0.521



Given the optimal threshold, the probabilities can be discretized and a confusion matrix + all metrics of interest can be analyzed

		<u>Actuals</u>	
		D^+	D^-
<u>Predictions</u>	T^+	62	52
	T^-	57	629

Precision = 0.521

Specificity = 0.924

Recall = 0.544

NPV = 0.917

F1 = 0.532

Accuracy = 0.864

Benchmarks

Despite a different test set, our results are ballpark to those of the top performers

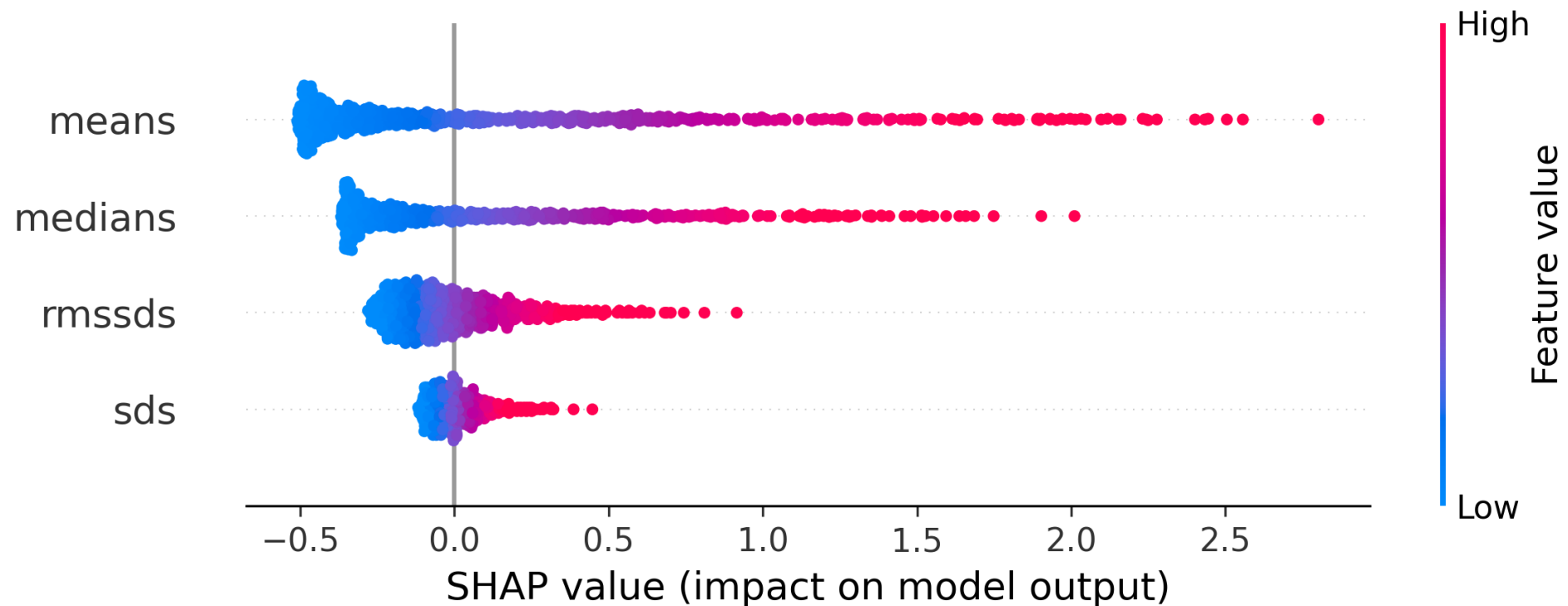
Team	$\min(\text{Recall}, \text{Precision})$
Alistair Johnson, Nic Dunkley, Louis Mayaud, Athanasios Tsanas, Andrew Kramer, Gari Clifford	0.5353
Luca Citi, Riccardo Barbieri	0.5345
Srinivasan Vairavan, Larry Eshelman, Syed Haider, Abigail Flower, Adam Seiver	0.5009
Martin Macas, Michal Huptych, Jakub Kuzilek	0.4928
Henian Xia, Brian Daley, Adam Petrie, Xiaopeng Zhao	0.4923
Steven L Hamilton, James R Hamilton	0.4872
Natalia M Arzeno, Joyce C Ho, Cheng H Lee	0.4821
Chih-Chun Chia, Gyemin Lee, Zahi Karam, Alexander Van Esbroeck, Sean McMillan, Ilan Rubinfeld, Zeeshan Syed	0.4564
Alexandros Pantelopoulos	0.4544
Deep Bera, Mithun Manjnath Nayak	0.4513
<i>Sample and random predictors</i>	
SAPS-I (in m-code)	0.3125
SAPS-I (in C)	0.3097
86% randomly predicted to survive	0.1386

A $\min(\text{Recall}, \text{Precision})$ score of **0.521**, ranks **3rd** out of the original teams who participated in the challenge

Feature importance (1 of 2)

While the ensemble is a black box, the meta learner's coefficients can be used to understand which time series features were most influential

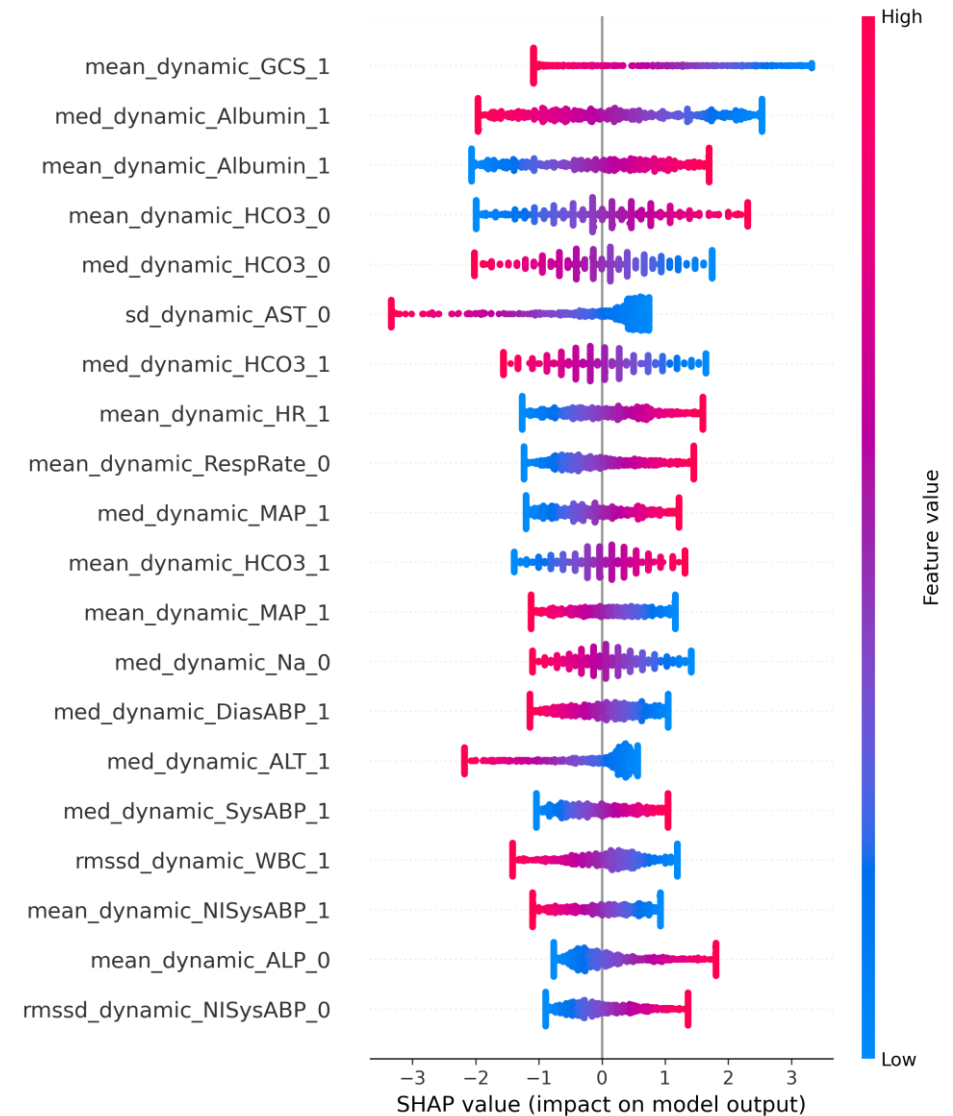
Higher values of the time series features positively influence the model to make an in-hospital death prediction, while lower values have the opposite influence



Feature importance (2 of 2)

Alternatively, the features fed into the individual base learners can be concatenated + explored together using a simple Logistic Regression model (SVC is expensive)

- **Transformations:** mean and median transformations are the most influential, which reflects the prior slide
- **Discrete time bins:** 13/20 of these features span the 24-48 hour period
- **Individual predictors:** Predictors related to Albumin and HCO3 (serum bicarbonate) appear to be very “important”
 - For both predictors, low medians but high means influence the model’s decision to make an in-hospital death prediction
 - This reveals that sharp spikes in these features may be indicative of severe underlying health issues



Limitations

There are some limitations that should be recognized

- Discretizing time into bins distorts longitudinal signal that may be valuable
- May be missing important time series features that account for unexplained variation in the outcome
- The selected model is not super interpretable, but rather more of a “black box”
- Results are evaluated on one held out test set; in practice, it would be better to have multiple held out sets and perhaps even an external test set
- Unable to do an apples-to-apples comparison to the 2012 teams

Future directions

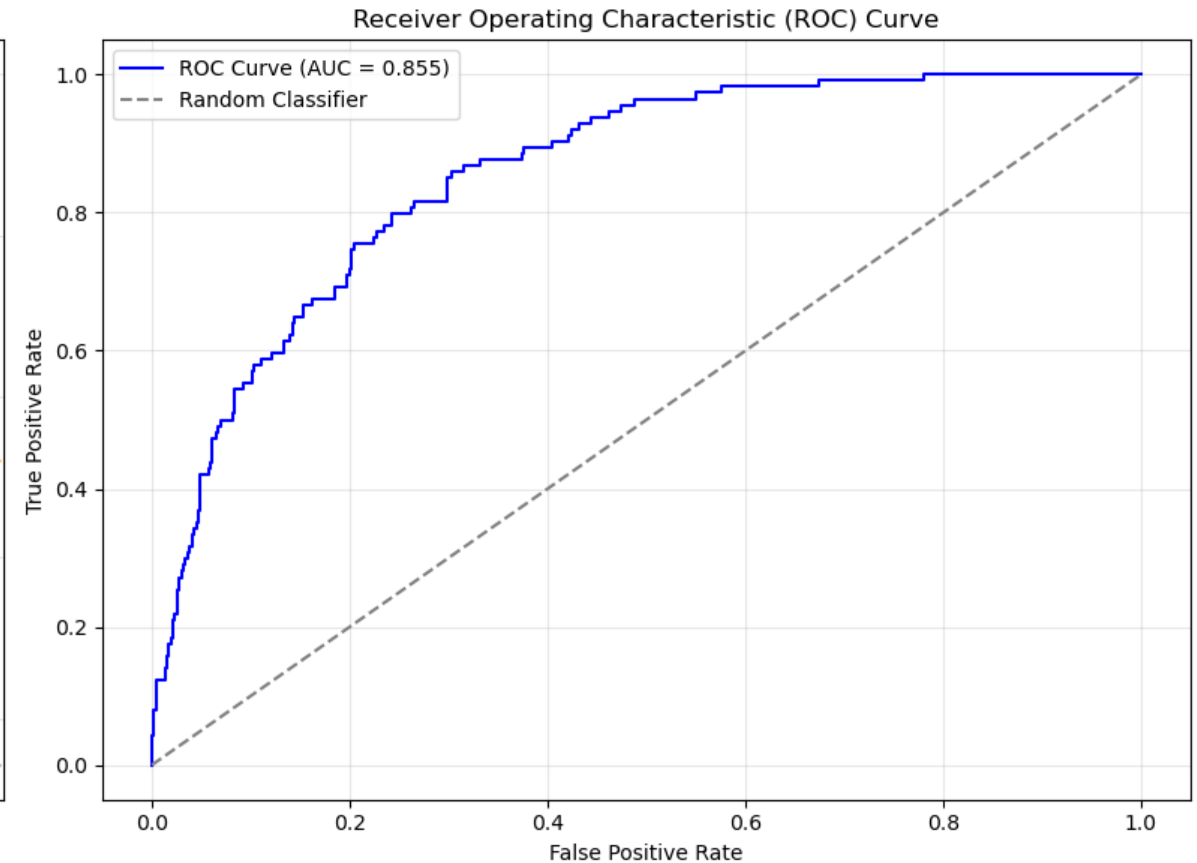
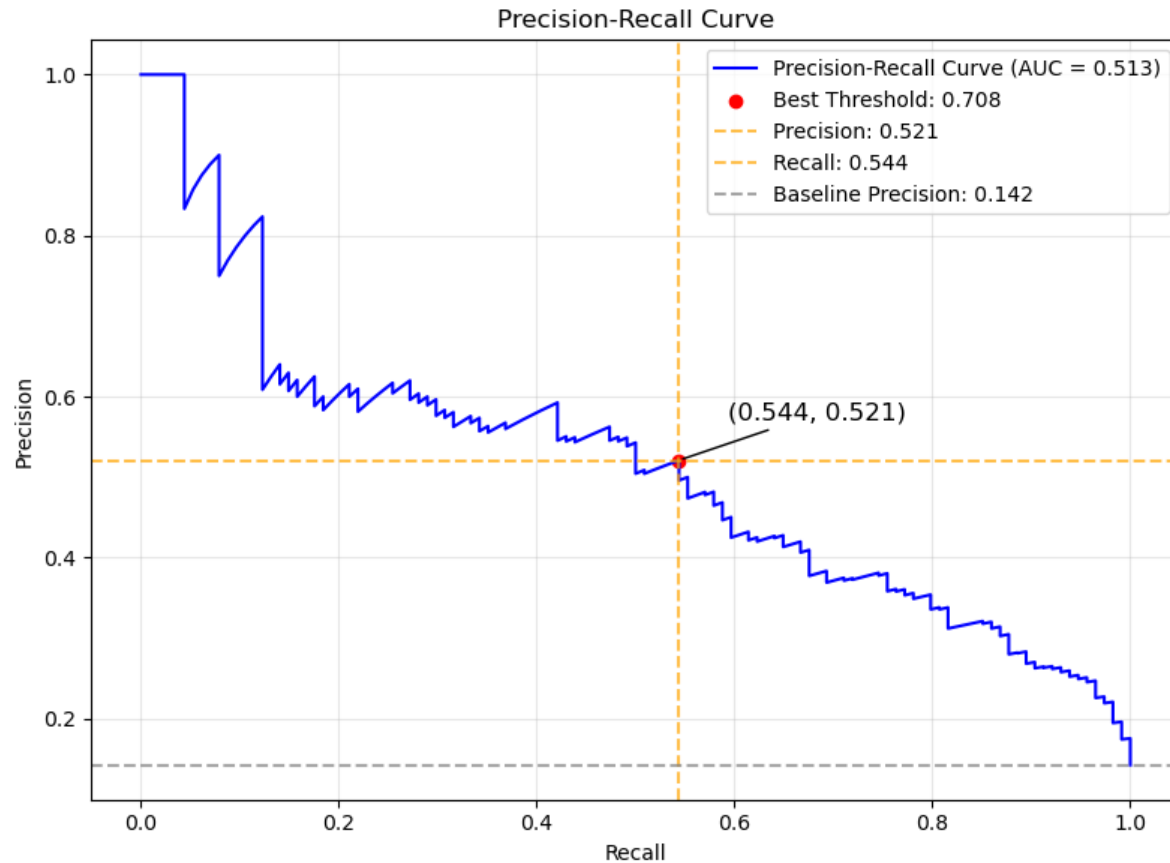
If I had more time, I would do the following...

- Explore other discrete time bins + more diverse set of time series features
- Spend more time on model selection (including deep learning models) and hyperparameter tuning
- Experiment with upsampling the minority class or downsampling the majority class during training
- Evaluate pipeline within a nested cross-validation or bootstrapped framework
- Develop deployment code to produce predictions on new unseen data

Appendix

PR vs. ROC curves

The PR curve tells a far different story than the ROC curve, as the PR curve prioritizes TPs while the ROC curve considers both TPs and TNs



Pros & cons of modeling “long” vs. “wide” data

The primary reasons for this hypothesis were due to the relatively small sample size (N=4000) and availability of computational resources; I approached my exploratory data analysis with this strategy in mind

Strategy	Example	Pros	Cons
Use ML approaches that can accommodate “long” data (i.e. repeated observations per patient)	Recurrent neural networks (e.g., LSTM)	<ul style="list-style-type: none">✓ Leverages longitudinal data✓ No feature engineering✓ Lower bias	<ul style="list-style-type: none">✗ Computationally expensive to train✗ Requires many samples✗ Higher variance
Transform the data into “wide” format (i.e. one observation per patient) and use tabular approaches	Standard suite of ML classifiers (e.g., Logistic Regression)	<ul style="list-style-type: none">✓ Computationally less expensive✓ More robust to smaller sample size✓ Lower variance	<ul style="list-style-type: none">✗ Potential for increased cardinality✗ Must engineer features carefully✗ Higher bias