Exploring Value-Based Care:

Predictive Analytics for Patient Outcomes, Operational Efficiency, and Health Disparities

Loyola University Maryland

DS 796 – Spring 2025: Capstone

Jeff Jay

Abstract

This study explores the integration of clinical and socioeconomic data to improve prediction of key hospital outcomes, including 30-day readmission, 30-day mortality, in-hospital mortality, and length of stay, within the context of value-based care (VBC). Using de-identified patient records from MIMIC-IV and MIMIC-IV-ED combined with socioeconomic indicators from IPUMS ACS, we constructed and evaluated predictive models to understand how social determinants of health (SDoH) influence care outcomes.

We implemented an end-to-end data science workflow using SQL and R, incorporating advanced preprocessing, statistical testing, and model tuning. Models included logistic regression, Random Forest, XGBoost, and GLMs, with performance evaluated across standard metrics including AUC, F1, RMSE, and recall. Tree-based models consistently outperformed others for mortality and length of stay. In contrast, linear models were most effective for predicting readmission. Across all targets, clinical variables such as ICU exposure, Charlson Comorbidity Index, and key medications were the strongest predictors.

Although SDoH features like race and insurance status showed statistically significant associations with outcomes, their inclusion yielded inconsistent effects on model performance. These findings suggest that while socioeconomic data offer valuable context, clinical data remain the primary drivers of predictive accuracy in hospital-based models.

This work supports CMS priorities around VBC and health equity. It highlights both the promise and the current limitations of using SDoH in hospital outcome modeling. Future work should refine patient subgroup analysis and validation methods to better isolate disparities and improve fairness in prediction.

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1 Motivation

Health outcomes are shaped by more than just medical care, they are influenced by education, income, and the broader environment. Studies have shown that people with lower incomes and less education tend to have worse health, higher rates of chronic disease, and greater difficulty accessing quality care (Braveman & Gottlieb, 2014; Bhavsar et al., 2018). Education also plays a key role, affecting health literacy, access to preventive care, and long-term well-being (Cutler & Lleras-Muney, 2006). These disparities show how early-life circumstances shape unequal health outcomes.

We are particularly interested in better understanding how these social factors shape patient outcomes in hospitals. Many predictive models used in healthcare focus mainly on medical factors and do not fully account for the role of Social Determinants of Health (SDoH) (Cantor & Thorpe, 2018). By studying hospital-wide predictions with multiple targets and different levels of SDoH data, we will explore the link between equality of opportunity and equality of outcome. This project will examine how factors like income and education influence key health metrics, helping to build a clearer picture of disparities in patient care.

1.2 Project Background

The Centers for Medicare & Medicaid Services (CMS) oversee Value-Based Care (VBC) programs designed to improve care quality, promote equity, and reduce healthcare costs. Key initiatives such as the Hospital Readmissions Reduction Program (HRRP), Hospital Value-Based Purchasing (VBP) Program, and Hospital-Acquired Condition Reduction Program (HACRP) encourage hospitals to improve patient outcomes while addressing disparities in healthcare delivery (Centers for Medicare & Medicaid Services, n.d.).

Recognizing that social and economic factors influence health, CMS has incorporated health equity as a priority in its Framework for Health Equity 2022–2032. This initiative emphasizes the importance of SDoH data, including race, ethnicity, income, and education, to better understand disparities in healthcare access and patient outcomes (Centers for Medicare & Medicaid Services, 2022).

This study takes an exploratory approach by leveraging de-identified clinical data from MIMIC-IV, MIMIC-IV-ED, and eICU to develop predictive models for hospital-wide outcomes, including readmission rates, length of stay, mortality, and complications. To incorporate SDoH-related factors, individual-level socioeconomic indicators, such as household income and education, are estimated from IPUMS ACS data. These sample-based socioeconomic proxies are linked to individual patient records to assess how estimated income, education, and other household characteristics influence patient outcomes. This approach enables a more granular examination of SDoH effects than models relying solely on broad geographic or community-level averages.

While this research does not aim to produce definitive conclusions, it serves as a learning exercise to explore how individual socioeconomic factors influence hospital outcomes. The findings may provide supporting insights for future studies and contribute to ongoing discussions on integrating SDoH into predictive models to better understand disparities in healthcare.

1.3 Objectives

This project aims to address the following key questions:

1. What are the significant predictors for critical clinical outcomes, such as length of stay, readmission rates, mortality, and complications?
2. How do social determinants of health (e.g., race, gender, marital status, insurance type, socioeconomic status) influence these outcomes?

The primary objectives are to:

- Identify and validate predictors of clinical outcomes using advanced statistical and machine learning models.
- Incorporate social determinants of health into predictive models for deeper insights and equity-driven care.
- Develop predictive models, including classification and logistic regression, to enhance decision-making.
- Support alignment with Medicare's (CMS) Value-Based Care Programs

1.4 Data Sources

This project uses three de-identified datasets made available to credentialed researchers through PhysioNet: MIMIC-IV, MIMIC-IV-ED, and eICU-CRD. These datasets provide detailed information about patient care and outcomes, supporting predictive modeling and analysis while ensuring data privacy.

MIMIC-IV includes data from over 540,000 hospital admissions at Beth Israel Deaconess Medical Center (BIDMC) from 2008 to 2022 (Johnson et al., 2023a). It is organized into modules such as hosp, which contains hospital-wide data like demographics, diagnoses, and medications, and icu, which focuses on ICU-specific clinical details and outcomes. MIMIC-IV-ED captures over 400,000 emergency department visits at BIDMC between 2011 and 2019 (Johnson et al., 2023b). This dataset includes triage details, vital signs, discharge diagnoses, and emergency department outcomes, offering valuable insights into acute care settings.

The eICU-CRD dataset contains data from more than 200,000 ICU admissions across 208 U.S. hospitals from 2014 to 2015 (Pollard et al., 2018). It provides information on vital signs, care plans, and severity scores, offering a broader view of patient care across different healthcare facilities. Because the ICU datasets share similar features, eICU will validate models developed using MIMIC-IV ICU data, ensuring that findings are generalizable beyond a single institution.

To enhance the analysis of SDoH, this project will use data from IPUMS USA, a research dataset offering detailed information on income, education, race, insurance, and marital status. IPUMS USA combines data from the U.S. Census and American Community Survey (ACS) to offer a more detailed view of socioeconomic factors than county-level averages (IPUMS USA, n.d.).

This data can be connected to de-identified features from MIMIC, like age, race, and insurance type, to better understand how socioeconomic factors affect patient outcomes. This approach supports more accurate modeling and aligns with CMS's focus on improving health equity.

2. Data Science Analysis and Research Platform

2.1 Overall Analysis

This project was designed to analyze four clinical outcomes: 30-day readmission, in-hospital mortality, 30-day mortality, and length of stay using data from MIMIC-IV, MIMIC-ED, and IPUMS. Although the targets were planned from the start, working with three different datasets, each organized in its own way, added significant complexity. In total, the work involved joining over a dozen tables, making data aggregation more complex than expected.

The pipeline was built using SQL for data extraction and R for cleaning, modeling, and visualization. To generate additional clinical features, such as ICU severity scores, scripts from the MIMIC code library were integrated and customized. As the analysis expanded, the original monolithic notebook was split into smaller scripts and utility functions to manage preprocessing, figure generation, and model outputs more cleanly.

A wide range of R packages was used to support data management, modeling, and evaluation. Exploring multiple approaches and producing consistent outputs shaped the modular workflow described below.

2.2 Data Access and Integration

Data access and integration were key initial steps in building the project's analytical foundation. A Google Cloud Platform (GCP) project was created to host BigQuery as the primary environment for interacting with large-scale clinical datasets. Access to MIMIC-IV and MIMIC-ED was granted through PhysioNet, which required credentialing as a researcher and completion of the appropriate data use training. Once approved, the datasets were linked to the GCP environment and queried directly through BigQuery.

Clinical data was extracted using structured SQL queries across a dozen interconnected tables, covering patient demographics, admissions, procedures, diagnoses, vitals, and outcomes. This enabled efficient feature extraction, filtering, and early aggregation across over 400,000 patient records. Significant time was spent researching ICD codes and designing join logic to construct features such as prior visits, readmission flags, length of stay, and ICU usage. These definitions required careful attention to temporal relationships in the data to avoid data leakage and ensure proper alignment of features with outcome windows.

To engineer additional clinical indicators, such as ICU severity scores, a copy of the open-source MIMIC Code Repository was cloned from GitHub. Several scripts were customized to reflect schema updates in MIMIC-IV and to replace deprecated functions. This work required both SQL and shell-level testing before uploading the derived tables into a separate schema within BigQuery for integration with the main dataset.

For social determinants of health, relevant variables were sourced from the IPUMS USA (American Community Survey) platform. After registering for access, a custom extract was designed to include household-level features such as median income, education attainment, and insurance coverage. The dataset was then downloaded, cleaned, and aligned using demographic identifiers before being merged with clinical records in R.

2.3 Data Preparation and Statistical Workflow

After data extraction, the R environment was used for data cleaning, exploratory analysis, and statistical testing. Data preparation involved filling missing values with defaults (typically zero), simplifying and recoding categorical variables, and scaling continuous features as needed. These tasks were moved into standalone scripts to ensure consistency and reusability across all modeling stages.

A core challenge during this phase was managing four different prediction targets, three binary and one continuous, which required applying a variety of exploratory and statistical techniques suited to each outcome type. This included learning and testing different EDA and statistical testing methods in R, such as correlation analysis, group comparisons, and significance testing, to better understand how features behaved across targets. To support this process and avoid redundancy, a function library was developed to centralize key operations like preprocessing steps, test routines, and reusable plotting functions.

To keep the workflow organized, exploratory data analysis and statistical testing were handled separately from the core preprocessing. Summary statistics, correlation checks, and group-wise comparisons were performed to assess patterns across outcome groups. Common statistical tests, including t-tests, Wilcoxon, ANOVA, and chi-square, were used to evaluate feature relevance and support early decision-making.

Utility functions automated figure saving, dataset exports, and formatted table outputs. This supported reproducibility and helped streamline the process as the analysis grew in size and complexity.

2.4 Statistical Models and Machine Learning Algorithms

The modeling phase of the project involved two primary workflows, one for binary classification tasks (mortality and 30-day readmission) and one for the continuous regression task (length of stay). While the binary workflows followed a shared structure, the regression workflow required different preprocessing logic and evaluation metrics. These differences informed the structure of the modeling functions and training scripts.

Classification models logistic regression, random forest, XGBoost, support vector machines (SVM), and naïve Bayes. For regression, linear models, tree-based regressors, and XGBoost were applied. All models were trained using a shared function framework that supported consistent model calls, stratified cross-validation, and standardized output formats.

As the project scaled, support functions were developed to prepare modeling inputs, including dynamic feature selection, target assignment, and filtering options. Models were trained using various feature groups to enable comparison, starting with clinical and administrative data and expanding to include SDoH features from IPUMS. These grouped feature sets were passed to the modeling functions as configurable inputs, allowing efficient comparisons between baseline and SDoH-enhanced models.

In addition to feature groups, subsetting logic was implemented to support targeted training on specific cohorts such as ICU-only patients or those meeting CMS program criteria. This required flexible dataset construction and parameterized filtering, all integrated into the modeling workflow.

Key R libraries included caret, yardstick, e1071, pROC, and SHAPforxgboost. The platform-level design ensured that models could be rapidly configured and tested across multiple combinations of targets, filters, and feature sets.

2.5 Validation, Visualization and Reporting

Model evaluation was structured to provide a consistent comparison across algorithms, feature sets, and prediction targets. For classification tasks, including mortality and 30-day readmission, models were evaluated using accuracy, precision, recall, F1-score, sensitivity, specificity, Cohen's kappa, ROC-AUC, and PR-AUC. For the continuous outcome of length of stay, performance metrics included root mean squared error (RMSE), mean absolute error (MAE), R-squared, AIC, and BIC.

To support interpretability and reproducibility, evaluation outputs were generated using modular utility functions. These included standardized reporting of metrics, grouped result tables, and automated figure export. Visualizations were used to supplement numeric evaluation. For classification, ROC and PR curves were plotted, along with predicted probability distributions across outcome classes. For regression, residual plots and actual vs. predicted scatterplots were used to assess model fit and identify patterns in prediction errors.

Evaluation results were compared across different model types, filtered datasets, and feature configurations (e.g., clinical-only vs. SDoH-enhanced). These comparisons informed the selection of final models and provided insight into the influence of socioeconomic features on predictive performance.

3. Related Work

Recent studies have examined how social determinants of health (SDoH) influence predictive modeling in healthcare. Many studies suggest that incorporating SDoH can improve predictions for key outcomes such as mortality, readmission, and length of stay, but challenges remain in integrating high-quality SDoH data into clinical models.

Several studies have explored the role of SDoH in risk prediction using electronic health records (EHRs). A meta-analysis found that incorporating SDoH features improved models for emergency department revisits and hospital readmissions, reinforcing the idea that socioeconomic factors play a significant role in patient outcomes (Chen et al., 2020). Additionally, Ghassemi et al. (2023) demonstrated that integrating probabilistically inferred SDoH features from MIMIC-III improved ICU outcome predictions, highlighting the potential

for external socioeconomic data to enhance clinical models. These findings suggest that when SDoH data is properly incorporated, it can meaningfully contribute to risk stratification, particularly in critical care settings.

Other studies have examined whether external socioeconomic data provides additional predictive power beyond what is already captured in EHR-derived variables like insurance status and race. Some have found that traditional clinical risk factors can serve as proxies for broader socioeconomic conditions, limiting the benefit of external SDoH data (Bhavsar et al., 2018). However, others have shown that linking patient records to census-based socioeconomic indicators, such as income and education, enhances model performance (Braveman & Gottlieb, 2014). These mixed findings indicate that the way SDoH is incorporated, whether at the individual, household, or community level, may significantly impact its predictive value.

This work builds on prior research but differs in key ways. Rather than focusing solely on ICU populations or specific diagnoses, it examines hospital-wide outcomes, capturing a broader range of patient experiences. It also incorporates individual and household-level SDoH features by linking MIMIC-IV patient records with IPUMS ACS data, rather than relying on national or community-level aggregates. This allows for an evaluation of socioeconomic influences at a more granular level. Additionally, this study considers multiple hospital-wide targets, including mortality, readmission, and length of stay, rather than a single outcome, providing a broader perspective on how SDoH interacts with clinical predictors.

This project compares baseline clinical models to SDoH-enhanced models to assess the impact of socioeconomic data on predictive performance and fairness across hospital outcomes. While this analysis does not aim to resolve disparities, it explores how incorporating external socioeconomic data into hospital-based models may refine risk assessment and outcome prediction.

4. Data Analysis

4.1 Research Questions

The analysis was guided by two primary questions:

1. What patient-level and system-level features are most predictive of clinical outcomes including as 30-day readmission, in-hospital mortality, and length of stay?
2. To what extent do social determinants of health (SDoH), including race, insurance status, and marital status, influence these outcomes in the context of value-based care?

4.2 Hypotheses

The following hypotheses were formulated:

H1: Certain clinical variables (e.g., Charlson Comorbidity Index, age) are significant predictors of adverse outcomes.

H2: SDoH features such as race and insurance type are statistically associated with differences in outcome predictions, suggesting potential disparities in care.

H3: Models incorporating both clinical and socioeconomic variables will demonstrate improved predictive performance over those using clinical variables alone.

4.3 Data Pre-Processing

To prepare the data for modeling, a comprehensive SQL query was used to extract and join information from multiple MIMIC-IV tables using Google BigQuery. This included sampling patients while retaining all their hospital admissions. The query assembled a wide range of variables such as demographics, diagnoses, procedures, ICU and ED visits, medications, and outcomes including 30-day readmission, length of stay, and 30-day mortality. Key clinical features were engineered during this step, including Charlson Comorbidity Index, ICU usage metrics, and CMS-tracked condition flags.

Once the data was extracted, it was brought into R for further cleaning and transformation. Socioeconomic variables from IPUMS were merged in based on race, insurance type, and marital status. Missing numeric values were replaced with zeros, and missing categorical values were labeled as “Unknown.” Categorical variables were converted into factors

or numeric encodings where needed, and continuous variables were standardized. Age was also categorized into groups for modeling purposes.

Finally, exclusion rules were applied to remove pediatric patients and others not relevant to the study targets. Subsets of the data were created for ICU patients and for admissions related to CMS-monitored conditions. The cleaned datasets were saved as .rds files to support consistent downstream analysis.

Feature Category	Example Features	Source Table(s)
Demographics	age, gender, race_group, marital_status_category	patients, admissions, IPUMS_data
Clinical Outcomes	readmitted, length_of_stay, mortality_30_day	admissions, patients (derived)
Comorbidities	congestive_heart_failure, renal_disease, aids, charlson_comorbidity_index	diagnoses_icd
ICU Events	ICU_flag, total_icu_hours, first_icu_admit, max_sapsii	icustays, sapsii, lods, apsiii, icu_detail
Procedures	major_surgery_flag, dialysis_flag, tracheostomy_flag	procedures_icd
Medications	num_medications, pressor_med_flag, sedative_med_flag	prescriptions
Emergency Department Use	ed_acuity, ed_chief_complaint, total_ed_visits	edstays, triage
Socioeconomic Factors	insurance_category, IPUMS income, IPUMS education	admissions, IPUMS_data
Target Variables	readmitted, mortality_30_day, cms_flag	Derived / Constructed

Table 1 - Summary of Feature Categories and Examples

4.4 Exploratory Data Analysis

As a first step in exploratory data analysis, we examined the distribution of the primary outcome variables to better understand the structure of our prediction targets. This included one continuous variable (length of stay) and three binary targets: 30-day readmission, in-hospital death, and 30-day mortality. Understanding their distributions helps guide model design, especially in the presence of class imbalance or skewed values.

The length of stay (LOS) distribution was highly right-skewed, with the majority of patients discharged within 10–15 days, but a small subset experiencing prolonged hospitalizations beyond 30 days. This imbalance may impact modeling by placing greater influence on rare but extreme values.

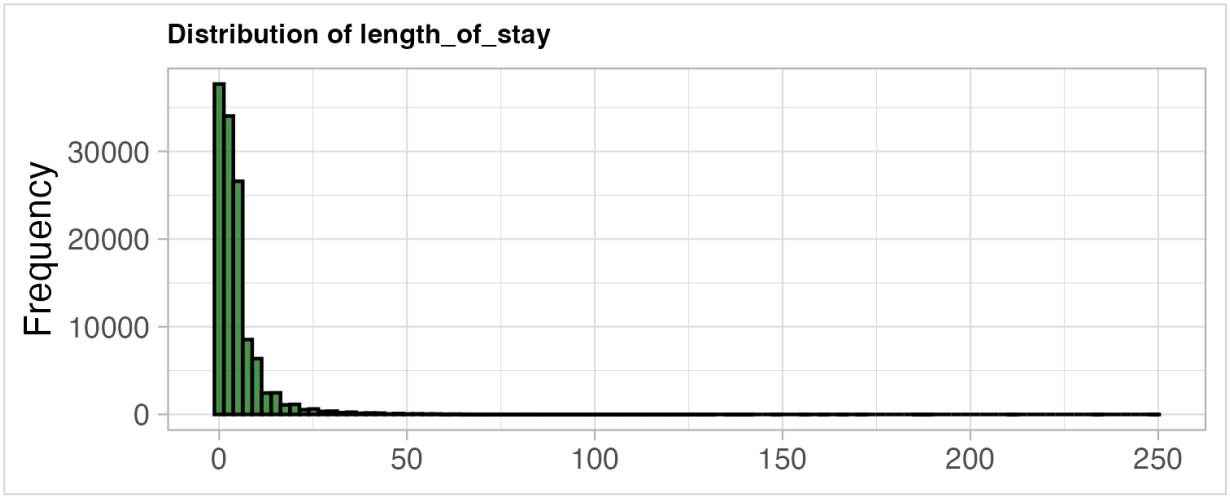


Figure 1 - Length of Stay Distribution

For the binary outcomes, including 30-day readmission, 30-day mortality, and in-hospital death the distributions were also imbalanced. A large proportion of patients were not readmitted and survived past 30 days, while the positive classes (readmitted or deceased) represented a smaller share of the dataset. This imbalance necessitates the use of evaluation metrics beyond accuracy and may require techniques such as resampling or class weighting in model development.

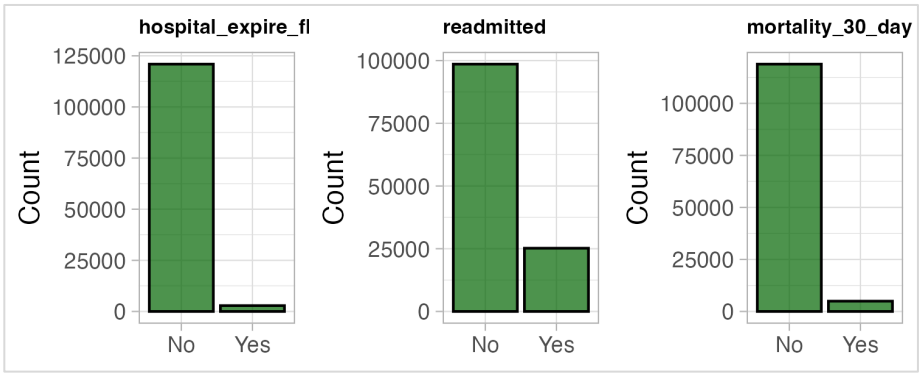


Figure 2 - Target Class Distribution

Binary Target Summary Table

Target	Category	Count	Proportion
hospital_expire_flag	No	120934	0.9769129
hospital_expire_flag	Yes	2858	0.0230871
readmitted	No	98585	0.7963762
readmitted	Yes	25207	0.2036238
mortality_30_day	No	118845	0.9600378
mortality_30_day	Yes	4947	0.0399622

Table 2 - Target Class Proportions

We reviewed the distribution of continuous predictors by grouping them based on whether they apply to ICU patients only or to the full hospital population. ICU-related features were analyzed using the ICU subset, since non-ICU patients have missing or zero values for these variables. Within the ICU group, we first looked at count-based measures such as length of stay, total ICU hours, and the number of ICU stays. These showed heavily skewed distributions, with most patients clustered at lower values and a long right tail.

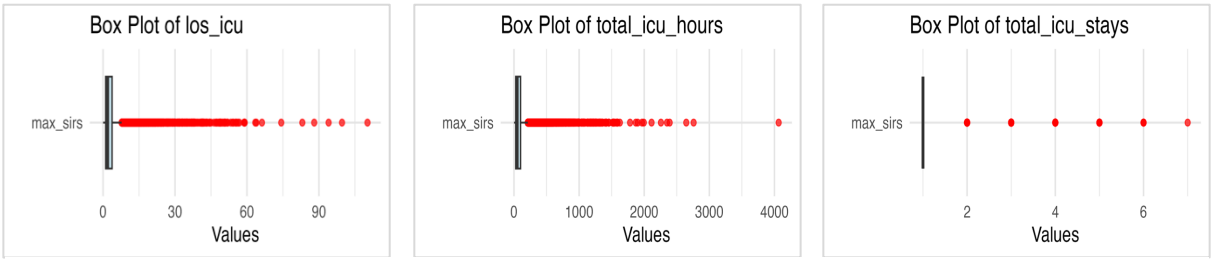


Figure 3 - ICU count predictor boxplots

Severity scores such as SAPS II, LODS, and OASIS followed a more normal patterns, with some extreme values on the higher end.

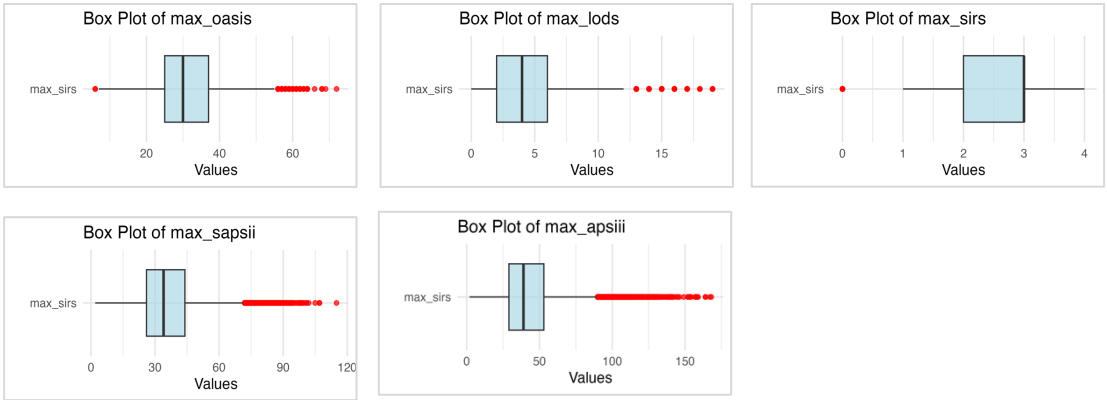


Figure 4 - ICU severity index predictor boxplots

For hospital-wide continuous features, distributions often showed a narrow interquartile range with a cluster of significant outliers primarily on the higher end. This pattern was observed in variables such as the number of procedures, number of medications, prior hospital visits.

The same was true for several weighted socioeconomic indicators from IPUMS. Although some features appeared relatively symmetric, most demonstrated enough skew or extreme values to warrant formal normality testing and potential transformation (e.g., log-scaling) prior to model training.

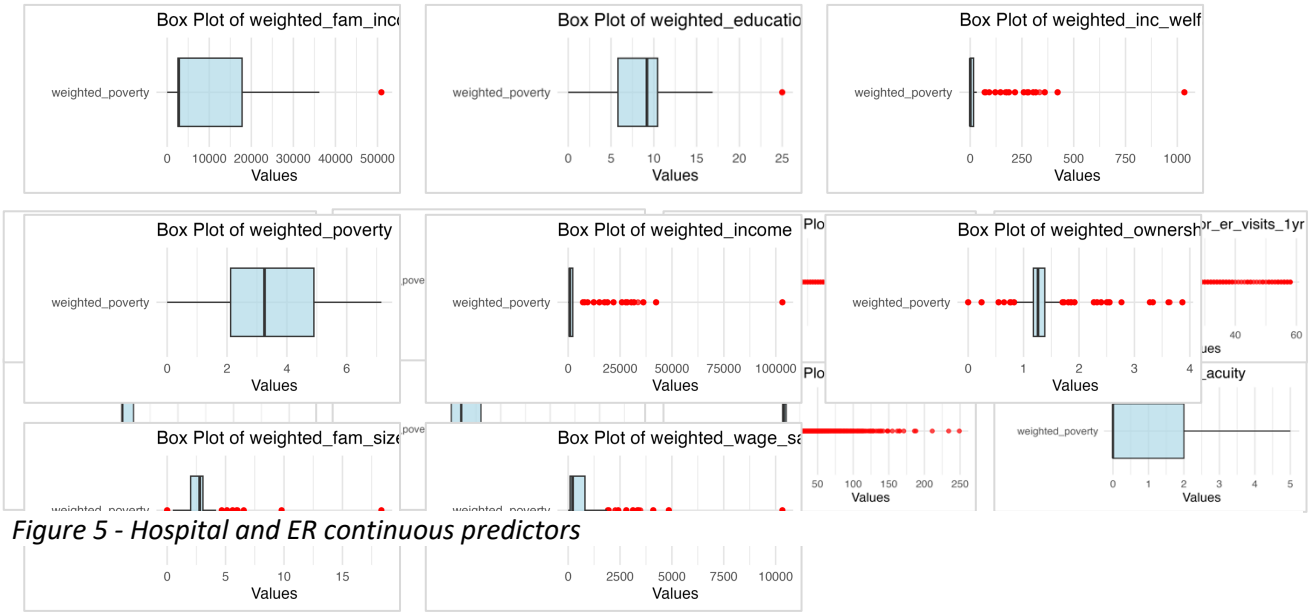


Figure 5 - Hospital and ER continuous predictors

Figure 6 - IPUMS weighted SE predictors

Following the review of continuous predictors, we examined patient-level readmission patterns to understand how often individuals return to the hospital. The distribution of admission counts per patient shows that most patients had only one or two visits, but a small subset experienced frequent hospitalizations (10 or more).

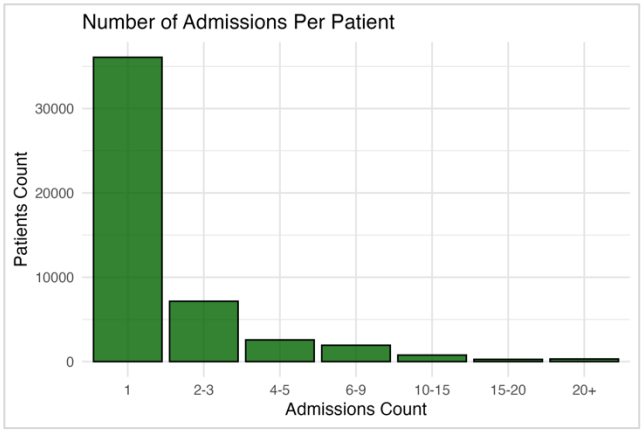


Figure 7 - Patient Counts by Admission Count

After reviewing readmission patterns across patient groups, we extended the analysis to examine how all predictor variables relate to each of the binary targets: 30-day readmission, in-hospital mortality, and 30-day mortality. For each target, we visualized categorical features using stacked bar charts, continuous features with density plots, and binary features with proportional bar plots. These comparisons provide an initial view of potential signal strength and feature relevance for downstream modeling.

Readmitted Examples:

For readmission we can see discharge location, specifically to a psych facility increases the chance of readmission considerably

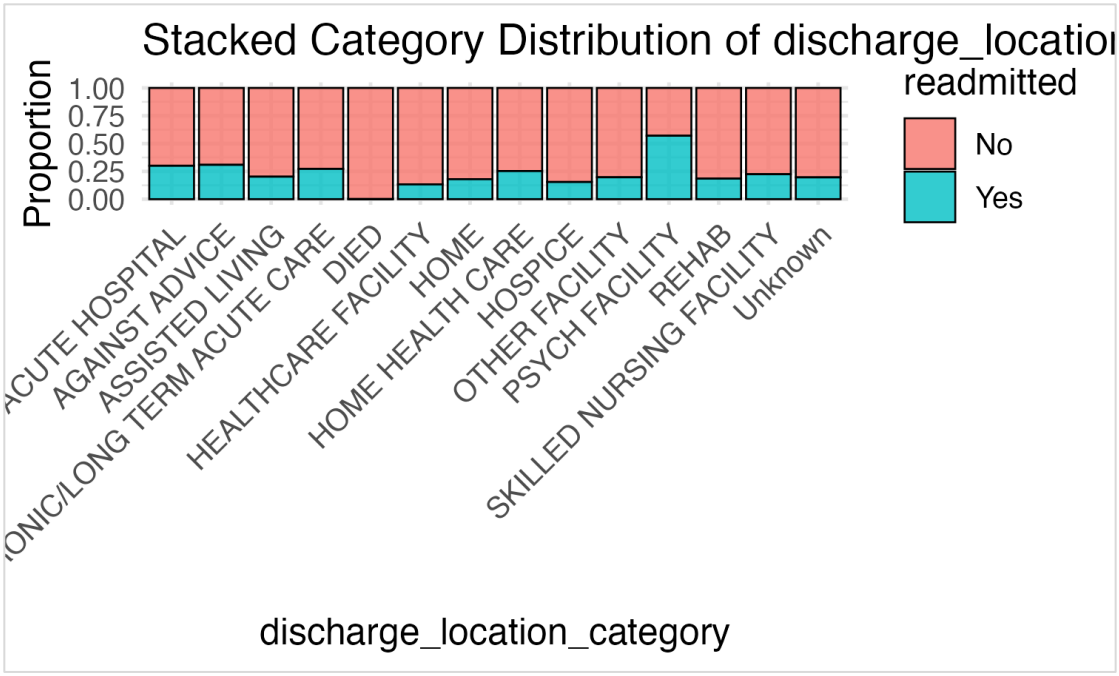


Figure 8 - Readmission by Discharge Location

Binary feature for Corticosteroids has a small but significant difference in readmission rates.

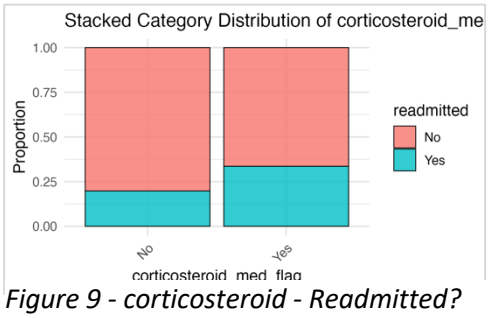


Figure 9 - corticosteroid - Readmitted?

Readmission clearly increases as the CCI increases

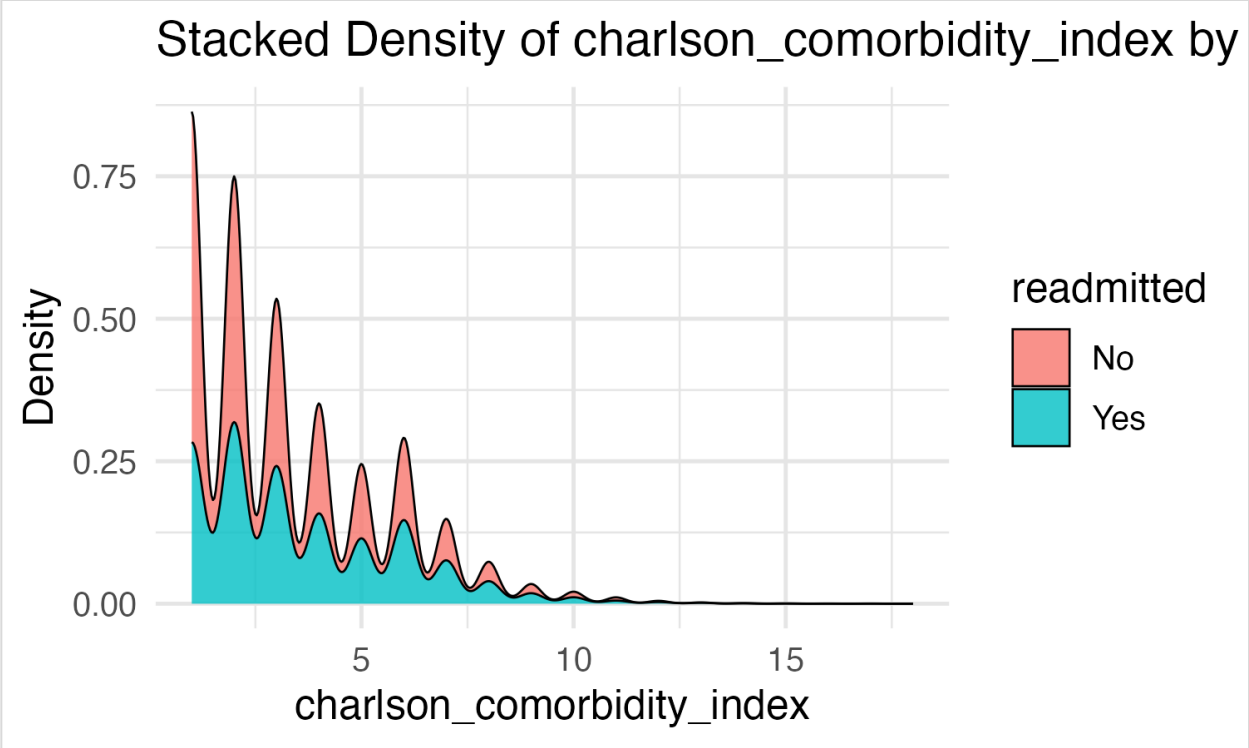


Figure 10 - CCI Readmitted?

Small but significant differences, (e.g., Medicaid versus private insurance).

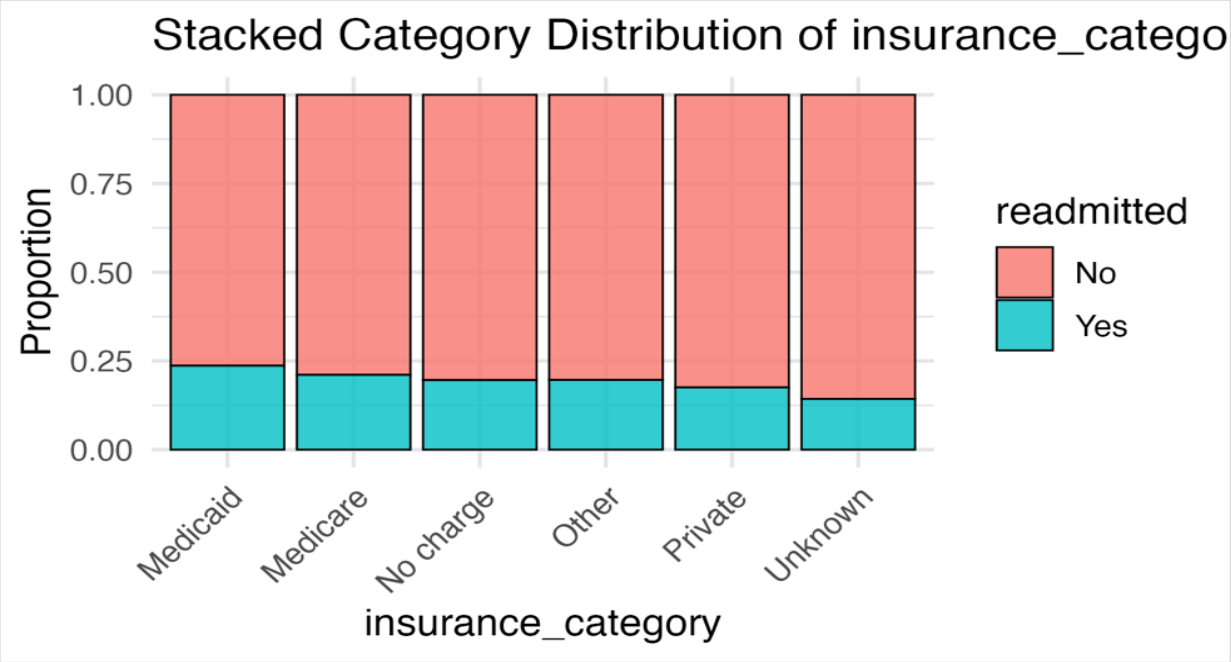


Figure 11 - Insurance type - Readmitted?

In Hospital Mortality Examples:

Clearly Sepsis diagnosis increases the changes of in hospital mortality

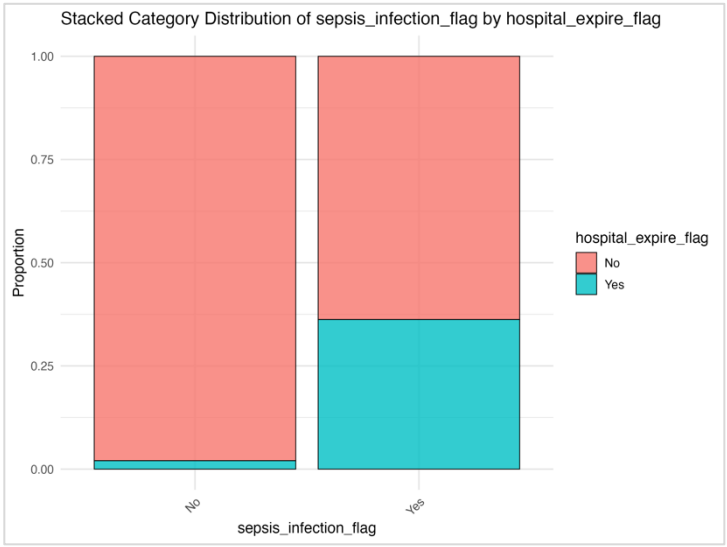


Figure 12 - Hospital Expire - Sepsis Flag.

Transfer from certain facilities can be indicative of higher chances of hospital mortality

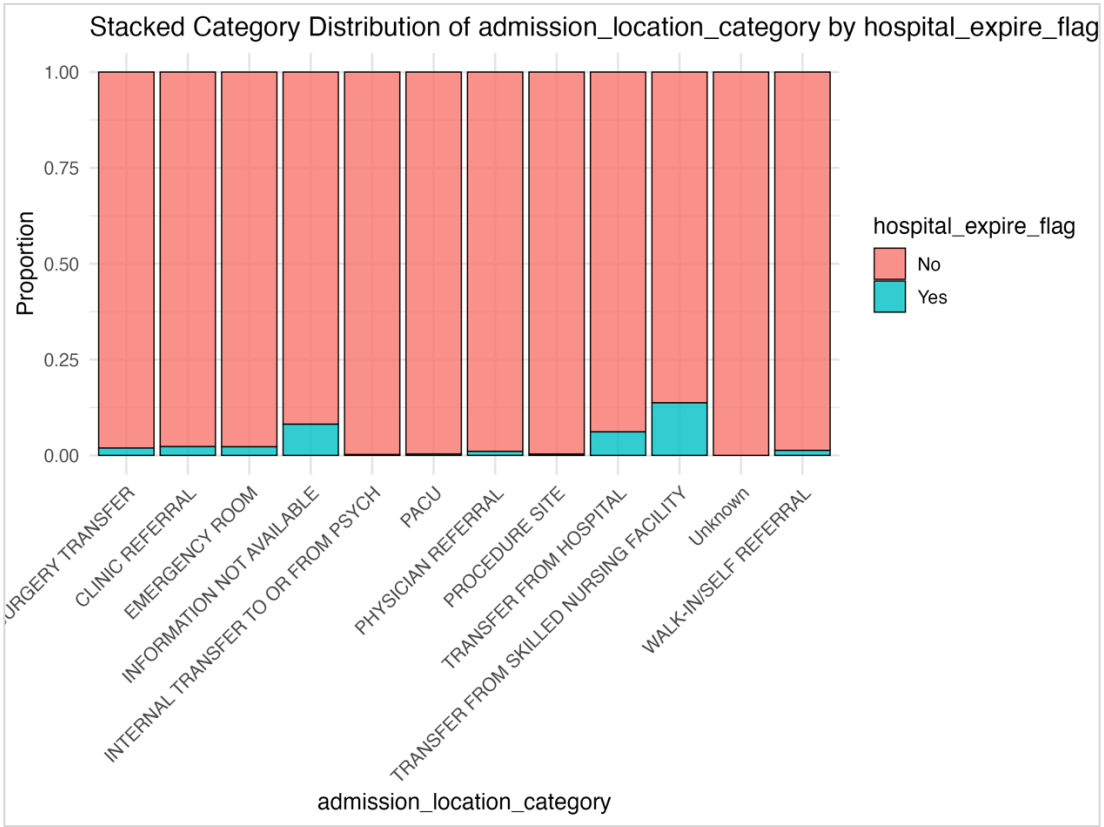


Figure 13 - Admission location - Hospital Expiration

This severity score appears to be highly predictive of in hospital mortality.

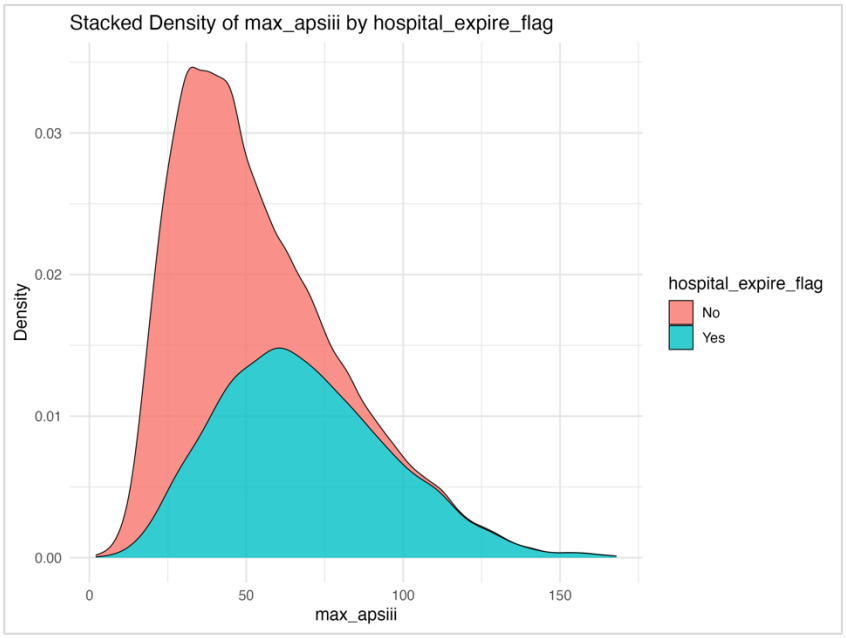


Figure 14 - ICU Severity APS III - Hospital Mortality

A full set of feature-wise plots across all binary targets is included in Appendix for reference.

The exploratory analysis identified several patterns relevant to modeling. Continuous features such as comorbidity scores, ICU hours, and prior hospitalizations exhibited right-skewed distributions with outliers, suggesting the need for transformation. Density plots also revealed meaningful separation between readmitted and non-readmitted patients across severity and medication-related variables. These findings guide the next phase—Data Transformation—where we apply normalization techniques and statistical tests to improve model readiness.

4.5 Data Transformation

To address distributional issues identified during exploratory analysis, we conducted a formal assessment of normality using the Shapiro-Wilk test, applied to a random sample of patients. This test evaluates the null hypothesis that a variable is normally distributed, and is especially useful for identifying features where standard modeling assumptions may not hold.

Many machine learning and statistical methods such as regression, SVM's, and Neural Networks benefit from predictors with more normal distributions so based on the results from the

Shapiro-Wilk normality test (table 3) features with a test statistic below 0.90 were considered substantially non-normal and selected for log transformation.

Before Transformation

Shapiro-Wilk Normality Test Results

	Feature	statistic	p.value
W...1	total_icu_stays	0.4227925	4.7380e-16
W...2	length_of_stay	0.5570714	1.2338e-32
W...3	weighted_income	0.5694344	2.4106e-32
W...4	weighted_inc_welfare	0.5857056	3.4750e-30
W...5	prior_er_visits_1yr	0.6333018	7.1010e-16
W...6	total_icu_hours	0.6357677	1.0675e-12
W...7	prior_hospital_visits_1yr	0.6593355	1.3924e-21
W...8	los_icu	0.6846124	9.7246e-12
W...9	weighted_wage_sal	0.7119511	2.3030e-27
W...10	num_procedures	0.7290672	1.0827e-20
W...11	ed_acuity	0.7940098	7.8467e-15
W...12	weighted_ownership	0.8025744	1.8195e-23
W...13	weighted_fam_income	0.8196651	1.5611e-22
W...14	complication_index	0.8318621	1.3338e-11
W...15	charlson_comorbidity_index	0.8542257	6.6366e-17
W...16	num_medications	0.8562594	1.5453e-19
W...17	weighted_education	0.8760696	6.9977e-19
W...18	max_sirs	0.8778995	2.3281e-06
W...19	weighted_poverty	0.9038477	1.3147e-16
W...20	weighted_fam_size	0.9121838	7.8149e-16
W...21	max_apsiii	0.9192977	9.7399e-05
W...22	max_lods	0.9275242	3.8860e-04
W...23	max_sapsii	0.9681945	4.5810e-02
W...24	max_oasis	0.9906534	8.4000e-01

After Transformation:

Shapiro-Wilk Normality Test Results

	Feature	statistic	p.value
W...1	total_icu_stays_log	0.4258148	5.2033e-16
W...2	weighted_inc_welfare_log	0.7173061	7.7333e-26
W...3	prior_er_visits_1yr_log	0.7489676	4.8509e-13
W...4	ed_acuity_log	0.7750324	1.5897e-15
W...5	prior_hospital_visits_1yr_log	0.8125208	3.1247e-16
W...6	weighted_education_log	0.8437904	4.2909e-21
W...7	weighted_income_log	0.8505126	1.1565e-20
W...8	max_sirs_log	0.8521237	2.8892e-07
W...9	weighted_ownership_log	0.8701424	2.5649e-19
W...10	num_procedures_log	0.8715315	2.9290e-14
W...11	complication_index_log	0.8914514	6.9926e-09
W...12	weighted_fam_income_log	0.9013151	7.8194e-17
W...13	charlson_comorbidity_index_log	0.9094269	4.3535e-13
W...14	length_of_stay_log	0.9115964	7.4930e-16
W...15	los_icu_log	0.9340559	5.1270e-04
W...16	weighted_wage_sal_log	0.9457914	6.2969e-12
W...17	total_icu_hours_log	0.9722598	8.2647e-02
W...18	num_medications_log	0.9947028	1.3990e-01

Table 3 - Shapiro-Wilk Normality Test Results

To support these findings visually, we also generated Q-Q plots for selected variables before and after transformation, allowing for a direct comparison of distributional improvements.

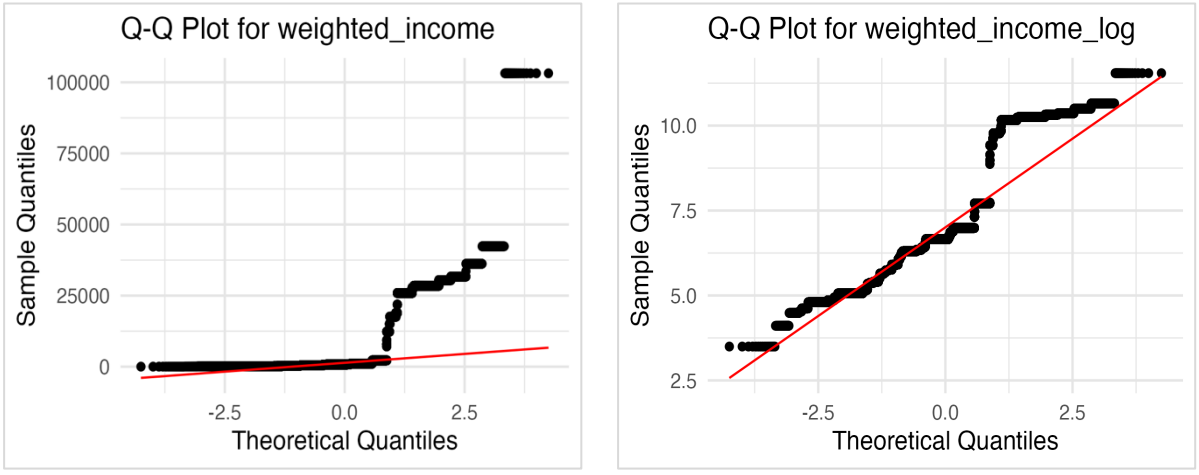


Figure 15 - Weighted Income QQ Plot - Before & After

Features such as weighted income, number of medications,and number of procedures, increased their normality statisitcs significantly.

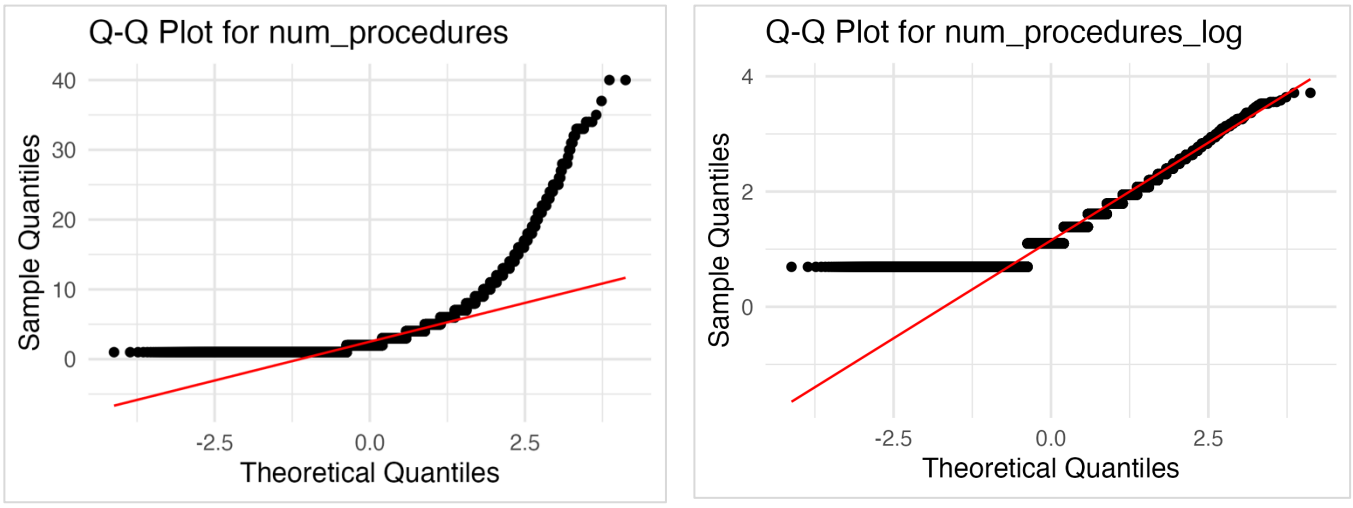


Figure 16 - Num of Procedures - QQ Plot - Before & After

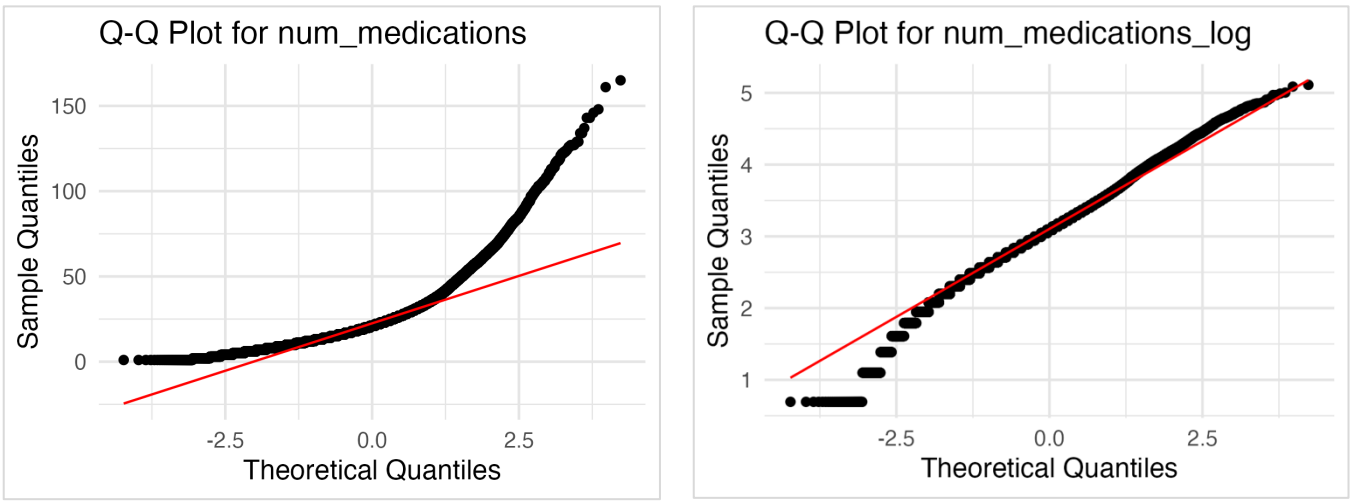


Figure 17 - Num of Medications -QQ Plot - Before and After

A full set of feature-wise plots across all binary targets is included in Appendix for reference.

Following the results of the Shapiro-Wilk normality test, we applied log transformations to 18 continuous features that exhibited strong right-skew and low normality scores. Visual inspection via Q-Q plots confirmed improved symmetry and reduced outlier impact. With normalized distributions in place, we next turn to statistical testing to formally assess associations between predictors and our key outcomes. This step will help identify significant variables and inform feature selection for modeling.

4.6 Statistical Testing

4.6.1 Length Of Stay

To evaluate relationships between predictor variables and length of stay (LOS), we conducted parametric and non-parametric statistical tests. This step aimed to identify features with significant associations to inform model development and reduce noise from irrelevant or collinear predictors.

We began with linear regression modeling, examining variance inflation factors (VIFs) to screen for collinearity (Table 4). Features showing high VIFs or known to represent data leakage (e.g., ICU duration variables or directly derived LOS measures) were excluded in a refined model, which produced a cleaner coefficient table (Table 5).

Predictors for: length_of_stay

Rank	term	estimate	statistic	p.value
1	num_procedures_log	2.1069	75.6822	0.0000
1	num_medications_log	1.0433	49.7661	0.0000
1	gastrostomy_tube_flag	11.9984	45.3414	0.0000
1	antifungal_med_flag	6.7764	61.1776	0.0000
2	sepsis_infection_flag	8.0190	37.8737	0.0000
3	diuretic_med_flag	1.6887	36.3407	0.0000
4	central_line_flag	3.2674	31.0703	0.0000
5	corticosteroid_med_flag	2.3322	29.7699	0.0000
6	major_surgery_flag	2.7218	22.6949	0.0000
7	max_apsiii	0.0594	17.4860	0.0000
8	admission_location_categoryINTERNAL TRANSFER TO OR FROM PSYCH	11.6956	15.3080	0.0000
9	discharge_location_categoryCHRONIC/LONG TERM ACUTE CARE	3.4671	13.3461	0.0000
10	max_oasis	-0.0724	-11.6313	0.0000
11	cms_flag	0.5294	11.2010	0.0000
12	colostomy_ileostomy_flag	5.5426	11.1898	0.0000
13	ed_acuity_log	0.2809	9.5661	0.0000
14	prior_hospital_visits_1yr_log	0.3249	8.8823	0.0000
15	inotrope_med_flag	2.5978	8.8661	0.0000
16	broad_spectrum_antibiotic_med_flag	1.0079	8.6689	0.0000
17	admission_location_categoryUnknown	46.8551	8.6530	0.0000
18	prior_er_visits_1yr_log	-0.3880	-8.5478	0.0000
19	discharge_location_categoryREHAB	2.0891	8.4983	0.0000
20	discharge_location_categoryAGAINST ADVICE	-2.4207	-8.0976	0.0000

Table 4 - LR Model - LOS - Top 20

VIF Summary for: length_of_stay

Rank	term	GVIF	Df	GVIF^(1/(2*Df))
1	total_icu_hours_log	212.4978	1	14.5773
2	weighted_education_log	56.8558	1	7.5403
3	ICU_flag	55.4788	1	7.4484
4	total_icu_stays_log	55.4000	1	7.4431
5	weighted_income_log	52.8595	1	7.2705
6	los_icu_log	46.5842	1	6.8253
7	weighted_ownership_log	33.8869	1	5.8212
8	max_oasis	29.5060	1	5.4319
9	weighted_inc_welfare_log	29.2742	1	5.4106
10	max_sapsii	25.0420	1	5.0042
11	hospital_expire_flag	24.2550	1	4.9249
12	weighted_fam_income_log	24.0728	1	4.9064
13	weighted_poverty	22.8927	1	4.7846
14	weighted_wage_sal_log	19.8668	1	4.4572
15	max_apsiii	16.4320	1	4.0536
16	max_sirs_log	15.4759	1	3.9339
17	complication_index_log	10.8665	1	3.2964
18	max_lods	10.5437	1	3.2471
19	complication_binary	8.7834	1	2.9637
20	broad_spectrum_antibiotic_med_flag	8.6481	1	2.9408

Table 5 - LOS VIF Summary – Top 20

Next, we used ANOVA tests to assess the relationship between categorical features and LOS. Several demographic variables, such as insurance category and marital status, were found to have statistically significant differences in LOS distributions across their levels (Table 6).

ANOVA: Predictors of length_of_stay

Predictor	P_Value
admission_location_category	0
discharge_location_category	0
race_group	0
age_category	0
marital_status_category	0
insurance_category	0
gender	0

Table 6 - LOS - ANOVA Results

For binary variables, we applied the Wilcoxon rank-sum test. Most binary features were significantly associated with LOS, with a few exceptions (e.g., wound debridement, stroke prevention), which were excluded from further modeling (Table 7).

Wilcoxon Test Results

Rank	Feature	Target	P_Value
1	length_of_stay	neurosurgery_flag	0.7853
2	length_of_stay	stroke_prevention_flag	0.2439
3	length_of_stay	shock_management_flag	0.1574
4	length_of_stay	wound_debridement_flag	0.1535
5	length_of_stay	prolonged_ventilation_flag	0.0695
6	length_of_stay	cardiac_surgery_flag	0.0031
7	length_of_stay	ecmo_flag	0.0009
8	length_of_stay	bowel_resection_flag	0.0000
9	length_of_stay	orthopedic_surgery_flag	0.0000
10	length_of_stay	blood_transfusion_flag	0.0000
11	length_of_stay	emergency_surgery_flag	0.0000
12	length_of_stay	cpr_flag	0.0000
13	length_of_stay	tracheostomy_flag	0.0000
14	length_of_stay	colostomy_ileostomy_flag	0.0000
15	length_of_stay	is_white	0.0000
16	length_of_stay	dialysis_flag	0.0000
17	length_of_stay	readmitted	0.0000
18	length_of_stay	inotrope_med_flag	0.0000
19	length_of_stay	hospital_expire_flag	0.0000
20	length_of_stay	gastrostomy_tube_flag	0.0000
21	length_of_stay	complication_binary	0.0000
21	length_of_stay	mortality_30_day	0.0000
21	length_of_stay	ICU_flag	0.0000
21	length_of_stay	cms_flag	0.0000
21	length_of_stay	major_surgery_flag	0.0000
21	length_of_stay	sepsis_infection_flag	0.0000
21	length_of_stay	central_line_flag	0.0000
21	length_of_stay	respiratory_support_flag	0.0000
21	length_of_stay	pressor_med_flag	0.0000
21	length_of_stay	broad_spectrum_antibiotic_med_flag	0.0000
21	length_of_stay	antifungal_med_flag	0.0000
21	length_of_stay	antiarrhythmic_med_flag	0.0000
21	length_of_stay	nephrotoxic_med_flag	0.0000
21	length_of_stay	diuretic_med_flag	0.0000
21	length_of_stay	corticosteroid_med_flag	0.0000
21	length_of_stay	sedative_med_flag	0.0000

Table 7 - LOS - Wilcoxon Test Results

Complementing ANOVA, the Kruskal-Wallis test was used to confirm patterns across categorical variables without assuming normality (Table 8). Finally, we generated a correlation heatmap for continuous predictors (Figure 18), highlighting both redundant and complementary signals across the normalized feature set.

Kruskal-Wallis Test: Categorical Features vs LOS

Rank	Feature	Target	P_Value
1	marital_status_category	length_of_stay	0
2	insurance_category	length_of_stay	0
3	gender	length_of_stay	0
4	race_group	length_of_stay	0
5	age_category	length_of_stay	0
5	admission_location_category	length_of_stay	0
5	discharge_location_category	length_of_stay	0

Table 8 - Kruskal Test Results - LOS

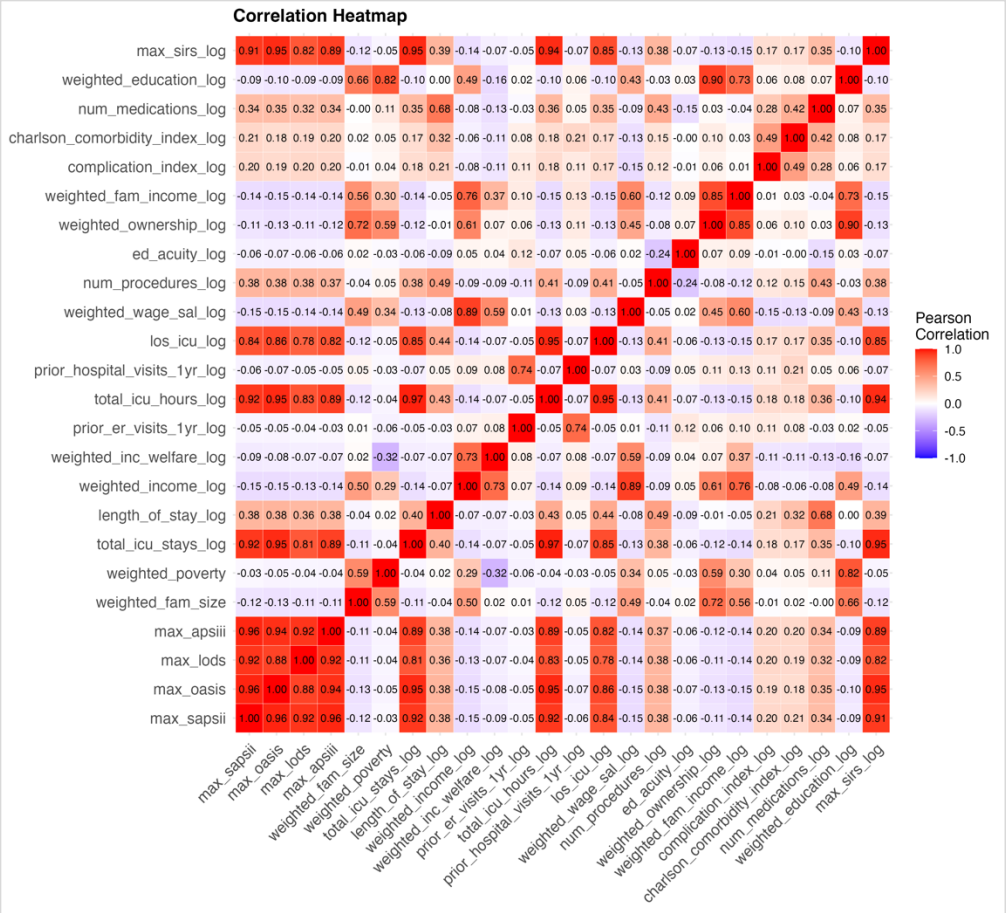


Figure 18 - Correlation Matrix - Pearson

As a result of this analysis, we flagged a subset of features for exclusion based on collinearity, redundancy, data leakage, or lack of statistical association. This refined feature set will guide model specification in later phases.

4.6.2 Binary Targets: 30 Day Mortality, Hospital Expiration, 30 Day Readmitted

We evaluated predictors for each of the three binary outcomes: readmission, 30-day mortality, and in-hospital mortality. Both parametric and non-parametric tests were used. Logistic regression models were run using the full feature set, excluding the target variables. Variance Inflation Factor (VIF) analysis (Table 9) identified high collinearity among certain socioeconomic indicators, particularly weighted_income and weighted_inc_welfare. These features were removed to reduce multicollinearity and improve model stability.

VIF Summary for: mortality_30_day

Rank	term	GVIF	Df	GVIF^(1/(2*Df))
1	weighted_income	9387.2928	1	96.8880
2	weighted_inc_welfare	8279.1722	1	90.9900
3	weighted_wage_sal	197.1348	1	14.0405
4	weighted_education	78.0539	1	8.8348
5	weighted_poverty	38.9004	1	6.2370
6	weighted_ownership	30.0633	1	5.4830
7	max_sapsii	28.3512	1	5.3246
8	max_oasis	26.9727	1	5.1935
9	ICU_flag	25.0836	1	5.0084
10	weighted_fam_size	18.0762	1	4.2516

VIF Summary for: hospital_expire_flag

Rank	term	GVIF	Df	GVIF^(1/(2*Df))
1	weighted_income	5332.9026	1	73.0267
2	weighted_inc_welfare	4658.8046	1	68.2554
3	weighted_wage_sal	135.8222	1	11.6543
4	weighted_education	76.7458	1	8.7605
5	weighted_poverty	38.4799	1	6.2032
6	weighted_ownership	32.9971	1	5.7443
7	max_sapsii	25.8468	1	5.0840
8	max_oasis	24.8592	1	4.9859
9	weighted_fam_size	19.6228	1	4.4298
10	ICU_flag	19.0671	1	4.3666

VIF Summary for: readmitted

Rank	term	GVIF	Df	GVIF^(1/(2*Df))
1	weighted_income	15886.1299	1	126.0402
2	weighted_inc_welfare	13698.6610	1	117.0413
3	weighted_wage_sal	315.2956	1	17.7566
4	weighted_education	43.2428	1	6.5759
5	max_oasis	28.6653	1	5.3540
6	weighted_poverty	28.3670	1	5.3261
7	ICU_flag	27.9272	1	5.2846
8	max_sapsii	23.5263	1	4.8504
9	max_apsiii	16.2902	1	4.0361
10	total_icu_stays	14.8898	1	3.8587

Table 9 - Variance Inflation Factor - LOS

Predictors for: mortality_30_day

Rank	term	estimate	OR	statistic	p.value
1	charlson_comorbidity_index	0.2605	1.2976	28.3005	0.0000
2	discharge_location_categoryDIED	5.1223	167.7261	22.2121	0.0000
3	length_of_stay	-0.0984	0.9062	-17.9220	0.0000
4	discharge_location_categoryHOSPICE	3.2489	25.7618	14.4547	0.0000
5	age_category6	1.7115	5.5374	9.1790	0.0000
6	corticosteroid_med_flag	0.6903	1.9943	8.3484	0.0000

Table 10 - Top 6 predictors logistic model

Refined logistic models (Table 10) revealed several significant predictors for each outcome. However, features such as discharge_location_categoryDIED and discharge_location_categoryHOSPICE showed extreme effects. These were excluded from the final models (Table 11) due to their close relationship with the outcome variables, which would otherwise introduce data leakage.

Predictors for: mortality_30_day

Rank	term	estimate	OR	statistic	p.value
1	charlson_comorbidity_index	0.2605	1.2976	28.3005	0.0000
2	length_of_stay	-0.0984	0.9062	-17.9220	0.0000
3	age_category6	1.7115	5.5374	9.1790	0.0000
4	corticosteroid_med_flag	0.6903	1.9943	8.3484	0.0000
5	discharge_location_categoryHOME	-1.7426	0.1751	-7.6757	0.0000
6	age_category5	1.2369	3.4450	6.7041	0.0000
7	(Intercept)	-7.6915	0.0005	-5.9823	0.0000
8	age_category4	0.9610	2.6143	5.3437	0.0000
9	num_medications	0.0151	1.0152	5.3273	0.0000
10	diuretic_med_flag	0.2938	1.3416	5.2709	0.0000

Table 11 - Updated logistic model output

Wilcoxon rank-sum tests were used to compare continuous predictors across each binary target group. Most features were significant for readmission and 30-day mortality with one exception was prior_hospital_visits_1yr, which was not significant for in-hospital mortality and was flagged for removal (Table 12).

Wilcoxon Test: Hospital Mortality

Rank	Feature	Target	P_Value
1	prior_hospital_visits_1yr	Hospital Mortality	0.9285
2	prior_er_visits_1yr	Hospital Mortality	0.0507
3	weighted_ownership	Hospital Mortality	0.0002
4	weighted_education	Hospital Mortality	0.0000
5	ed_acuity	Hospital Mortality	0.0000
6	weighted_poverty	Hospital Mortality	0.0000
7	weighted_fam_size	Hospital Mortality	0.0000
8	weighted_fam_income	Hospital Mortality	0.0000
9	weighted_inc_welfare	Hospital Mortality	0.0000
10	weighted_income	Hospital Mortality	0.0000
11	length_of_stay	Hospital Mortality	0.0000
12	weighted_wage_sal	Hospital Mortality	0.0000
13	complication_index	Hospital Mortality	0.0000
14	charlson_comorbidity_index	Hospital Mortality	0.0000
14	total_icu_stays	Hospital Mortality	0.0000
14	total_icu_hours	Hospital Mortality	0.0000
14	num_procedures	Hospital Mortality	0.0000
14	num_medications	Hospital Mortality	0.0000
14	max_sapsii	Hospital Mortality	0.0000
14	max_sirs	Hospital Mortality	0.0000
14	max_oasis	Hospital Mortality	0.0000
14	max_lods	Hospital Mortality	0.0000
14	max_apsiii	Hospital Mortality	0.0000
14	los_icu	Hospital Mortality	0.0000

Table 12 - Wilcoxon Test Results - Hospital Mortality

Chi-squared tests were then applied to evaluate categorical predictors against each binary outcome (Table 13 & 14). Discharge location remained a dominant predictor, reinforcing concerns about its predictive overlap with mortality targets.

Chi-Squared Results: All Binary Targets

	Rank	Feature	Target	Chi-Squared Statistic	P-Value
X-squared72	1	gender	hospital_expire_flag	36.8146	0
X-squared12	2	insurance_category	hospital_expire_flag	633.5225	0
X-squared52	3	readmitted	hospital_expire_flag	744.1749	0
X-squared10	4	race_group	hospital_expire_flag	1827.6200	0
X-squared22	5	marital_status_category	hospital_expire_flag	1923.1306	0
X-squared32	6	admission_location_category	hospital_expire_flag	2144.7050	0
X-squared82	7	mortality_30_day	hospital_expire_flag	53116.0700	0
X-squared42	8	discharge_location_category	hospital_expire_flag	118611.0354	0
X-squared81	9	gender	mortality_30_day	62.6859	0
X-squared61	10	readmitted	mortality_30_day	93.1390	0
X-squared11	11	insurance_category	mortality_30_day	969.6899	0
X-squared9	12	race_group	mortality_30_day	1045.0675	0
X-squared21	13	marital_status_category	mortality_30_day	1369.8286	0
X-squared41	14	admission_location_category	mortality_30_day	2384.9945	0
X-squared31	15	hospital_expire_flag	mortality_30_day	53116.0700	0
X-squared51	16	discharge_location_category	mortality_30_day	65684.8776	0
X-squared8	17	mortality_30_day	readmitted	93.1390	0
X-squared7	18	gender	readmitted	158.5731	0
X-squared4	19	admission_location_category	readmitted	268.9225	0
X-squared1	20	insurance_category	readmitted	384.3832	0
X-squared	21	race_group	readmitted	492.4633	0
X-squared2	22	marital_status_category	readmitted	553.8507	0
X-squared3	23	hospital_expire_flag	readmitted	744.1749	0
X-squared5	24	discharge_location_category	readmitted	1938.9041	0

Table 13 - Chi-Squared Results - Prelim

Chi-Squared Results: All Binary Targets

	Rank	Feature	Target	Chi-Squared Statistic	P-Value
X-squared52	1	gender	hospital_expire_flag	36.8146	0
X-squared12	2	insurance_category	hospital_expire_flag	633.5225	0
X-squared42	3	race_group	hospital_expire_flag	1827.6200	0
X-squared7	4	marital_status_category	hospital_expire_flag	1923.1306	0
X-squared32	5	admission_location_category	hospital_expire_flag	2144.7050	0
X-squared51	6	gender	mortality_30_day	62.6859	0
X-squared11	7	insurance_category	mortality_30_day	969.6899	0
X-squared41	8	race_group	mortality_30_day	1045.0675	0
X-squared6	9	marital_status_category	mortality_30_day	1369.8286	0
X-squared31	10	admission_location_category	mortality_30_day	2384.9945	0
X-squared5	11	gender	readmitted	158.5731	0
X-squared3	12	admission_location_category	readmitted	268.9225	0
X-squared1	13	insurance_category	readmitted	384.3832	0
X-squared4	14	race_group	readmitted	492.4633	0
X-squared	15	marital_status_category	readmitted	553.8507	0

Table 14 - Chi-Squared Results - Final

Lastly, Fisher's exact tests were used to examine associations between binary predictors and outcomes (Table 15). Several late-stage interventions, including CPR, ECMO, pressor use, and prolonged ventilation, were highly associated with mortality. Although these features may not support early interventions, they remain valuable for retrospective analysis and interpretation in outcome modeling. We retained them for now but testing alternate models that exclude these features should be included in future work.

Fisher's Exact Test Results

Rank	Feature	Target	Odds Ratio	CI Lower	CI Upper	P-Value
1	cpr_flag	hospital_expire_flag	77.8490	58.2444	105.4282	0.0000
2	ecmo_flag	hospital_expire_flag	77.8156	26.3581	256.2754	0.0000
3	pressor_med_flag	hospital_expire_flag	56.7041	51.4159	62.4895	0.0000
4	prolonged_ventilation_flag	hospital_expire_flag	42.3454	0.5391	3218.1204	0.0456
5	sepsis_infection_flag	hospital_expire_flag	27.2947	23.7629	31.3651	0.0000
6	respiratory_support_flag	hospital_expire_flag	26.5872	23.6882	29.8323	0.0000
7	ICU_flag	hospital_expire_flag	23.0416	20.9896	25.3200	0.0000
8	sedative_med_flag	hospital_expire_flag	16.9752	15.7051	18.3384	0.0000
9	bowel_resection_flag	hospital_expire_flag	9.6313	2.8473	26.0883	0.0003
10	broad_spectrum_antibiotic_med_flag	hospital_expire_flag	9.1455	8.4515	9.9032	0.0000
11	nephrotoxic_med_flag	hospital_expire_flag	9.0567	8.3781	9.7957	0.0000
12	central_line_flag	hospital_expire_flag	8.7538	7.8722	9.7194	0.0000
13	inotrope_med_flag	hospital_expire_flag	8.6537	6.4449	11.4606	0.0000
14	cms_flag	hospital_expire_flag	7.2596	6.6654	7.9139	0.0000
15	dialysis_flag	hospital_expire_flag	5.6013	4.0886	7.5340	0.0000
16	tracheostomy_flag	hospital_expire_flag	5.4714	2.2606	11.4779	0.0002
17	antiarrhythmic_med_flag	hospital_expire_flag	4.9464	4.4316	5.5118	0.0000
18	diuretic_med_flag	hospital_expire_flag	4.1649	3.8621	4.4911	0.0000
19	antifungal_med_flag	hospital_expire_flag	3.4304	2.9198	4.0108	0.0000
20	colostomy_ileostomy_flag	hospital_expire_flag	3.4077	1.5172	6.7141	0.0021
21	emergency_surgery_flag	hospital_expire_flag	3.1691	1.7388	5.3626	0.0002
22	complication_binary	hospital_expire_flag	2.8512	2.6439	3.0754	0.0000
23	corticosteroid_med_flag	hospital_expire_flag	2.5415	2.2318	2.8856	0.0000
24	gastrostomy_tube_flag	hospital_expire_flag	2.5028	1.6138	3.7248	0.0001
25	cardiac_surgery_flag	hospital_expire_flag	1.3544	0.6690	2.4579	0.2877
26	major_surgery_flag	hospital_expire_flag	1.3236	1.0169	1.6964	0.0309
27	blood_transfusion_flag	hospital_expire_flag	0.9198	0.0228	5.3906	1.0000
28	is_white	hospital_expire_flag	0.7864	0.7280	0.8497	0.0000
29	orthopedic_surgery_flag	hospital_expire_flag	0.3664	0.1467	0.7594	0.0031

Table 15 - Fishers Exact Test Results

This testing phase informed the removal of collinear, non-significant, and potentially biased predictors. Final filtered feature sets were saved for downstream modeling.

4.7 Feature Selection & Model Tuning

After completing statistical testing, we performed feature selection for each of the three binary outcomes and for length of stay using a 10% sample of the data. The goal of this step was to reduce the dimensionality of the dataset, improve model performance, and remove features that were redundant or statistically insignificant. For each target, we evaluated both the SDOH-

inclusive and SDOH-excluded feature sets independently. We applied stepAIC for linear models and retained features with variable importance greater than zero for tree-based models. Model hyperparameter tuning was performed using a 25% sample of the dataset.

During final model training and evaluation, we identified a critical issue in the cross-validation process that introduced unintentional data leakage. Specifically, the original k-fold validation strategy did not group admissions by patient ID as intended, allowing visits from the same patient to appear in both training and validation folds. This is particularly important for outcomes like readmission, and mortality to a lesser degree, where patient-specific patterns can recur across admissions. As a result, performance metrics reported during tuning and feature selection may overestimate model performance.

Although we were not able to re-run the entire tuning process, we plan to perform a targeted validation step on a final selected models using corrected patient-level splits. This will allow us to assess whether key feature selections and hyperparameter settings remain stable under proper evaluation. It is worth noting that tree-based models, which were chosen for their strong performance, are generally less susceptible to overfitting through data leakage. However, the performance metrics reported in the benchmark tables should be interpreted with caution, as they reflect the original, non-grouped folds.

4.7.1 Length of Stay

To establish baseline performance for Length of Stay prediction, we evaluated six models: Generalized Linear Models (GLM), regularized regression via GLMNet, Bayesian GLM, and three tree-based models, XGBoost, Gradient Boosting Machines (GBM), and Random Forests. These models were selected for their ability to handle continuous targets effectively. Model performance across all variants fell within a narrow RMSE range of 4 to 6, and an actual error between 2-3 days, indicating consistency in predictive accuracy. Differences between models trained with and without SDOH features were minimal.

Data_Set	Target	Model	Feature_Set	Validation_Type	Split_Type	Train_Count	Test_Count	RMSE	Rsquared	MAE	AIC	BIC
Samp_10	length_of_stay	glm	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.744483	0.403484	2.934701	313705.465	314269.32
Samp_10	length_of_stay	glm	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.759794	0.400255	2.94167	313983.435	314397.516
Samp_10	length_of_stay	glmnet	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.765271	0.399738	2.9136	NA	NA
Samp_10	length_of_stay	glmnet	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.774995	0.397661	2.924014	NA	NA
Samp_10	length_of_stay	bayesglm	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.744457	0.403489	2.93469	313709.466	314290.941
Samp_10	length_of_stay	bayesglm	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.759768	0.40026	2.94166	313987.436	314419.138
Samp_10	length_of_stay	xgbTree	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	4.688193	0.60764	2.239044	NA	NA
Samp_10	length_of_stay	xgbTree	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	4.687728	0.607549	2.241987	NA	NA
Samp_10	length_of_stay	gbm	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	4.713656	0.598825	2.292673	NA	NA
Samp_10	length_of_stay	gbm	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	4.71592	0.598245	2.294599	NA	NA
Samp_10	length_of_stay	rf	Base_SDOH	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.560382	0.491133	2.784318	NA	NA
Samp_10	length_of_stay	rf	Base	4-Fold Cross-Validation	Patient-Split (70% Train / 30% Test)	34285	15240	5.444284	0.507702	2.726312	NA	NA

Table 16 - Length of Stay - Baseline Performance

Tree-based approaches tended to retain more features overall, particularly among clinical indicators and comorbidity flags, reflecting their ability to manage high-dimensional inputs without overfitting (Table 17). Given their strong performance and built-in feature handling, we selected the three tree-based models to carry forward in subsequent modeling stages. In the next phase, we will apply additional filtering to remove unselected features as part of the hyperparameter optimization process.

	bayesglm	bayesglm-sdoh	gbm	gbm-sdoh	glm	glm-sdoh	glmnet	glmnet-sdoh	rf	rf-sdoh	xgbTree	xgbTree-sdoh
age	X	X	-	-	X	X	-	X	-	-	-	-
antiarrhythmic_med_flag	X	X	-	-	X	X	X	X	-	-	-	-
blood_transfusion_flag	X	X	X	X	X	X	X	X	-	-	-	-
bowel_resection_flag	-	-	X	X	-	-	-	-	-	-	-	X
cardiac_surgery_flag	-	-	X	X	-	-	-	-	-	-	-	-
charlson_comorbidity_index_log	-	X	-	X	-	X	X	X	-	-	-	-
colostomy_ileostomy_flag	-	-	X	X	-	-	-	-	-	-	-	-
complication_binary	-	-	X	X	-	-	X	-	-	-	X	X
complication_index_log	X	X	-	-	X	X	X	X	-	-	-	-
cpr_flag	-	-	-	-	-	-	X	X	-	-	-	-
dialysis_flag	X	X	X	X	X	X	X	X	-	-	-	-
diuretic_med_flag	-	-	X	X	-	-	-	-	-	-	-	-
ecmo_flag	-	-	X	X	-	-	-	-	-	-	-	-
emergency_surgery_flag	X	X	X	X	X	X	X	X	-	-	-	-
marital_status_numeric	-	-	-	X	-	-	-	X	-	-	-	-
max_lods	-	-	-	-	-	-	X	X	-	-	-	-
max_oasis	-	-	-	-	-	-	X	X	-	-	-	-
max_sapsii	-	-	-	-	-	-	X	X	-	-	-	-
max_sirs_log	-	-	-	-	-	-	X	X	-	-	-	-
nephrotoxic_med_flag	-	-	X	X	-	-	-	-	-	-	-	-
orthopedic_surgery_flag	-	-	X	X	-	-	X	X	-	-	-	-
prior_er_visits_1yr_log	-	-	-	X	-	-	X	X	-	-	-	-
prolonged_ventilation_flag	X	X	X	X	X	X	X	X	X	X	X	X
respiratory_support_flag	-	-	-	-	-	-	X	X	-	-	-	-
sepsis_infection_flag	-	-	X	X	-	-	-	-	-	-	-	-
tracheostomy_flag	-	-	X	X	-	-	-	-	-	-	-	-
weighted_education_log	-	-	-	-	-	-	-	X	-	-	-	-
weighted_fam_income_log	-	X	-	-	-	X	-	X	-	-	-	-
weighted_fam_size	-	-	-	-	-	-	-	X	-	-	-	-
weighted_inc_welfare_log	-	-	-	X	-	-	-	-	-	-	-	-
weighted_income_log	-	X	-	-	-	X	-	X	-	-	-	-
weighted_ownership_log	-	X	-	X	-	X	-	X	-	-	-	-
weighted_wage_sal_log	-	-	-	X	-	-	-	X	-	-	-	-

Figure 19 - Removed Features by Model - Length of Stay

For Length of Stay prediction, hyperparameter tuning focused on tree-based models (Random Forest, XGBoost, GBM). Across these models, deeper trees and a higher number of estimators generally improved performance, suggesting that increased model complexity helps capture non-linear interactions in the data. However, due to computational constraints, we limited the maximum depth and number of trees explored. Diminishing returns were evident even at moderate complexity settings, indicating that similar predictive performance could be achieved with more economical configurations. As such, we prioritized parameter sets that balanced model accuracy with training efficiency.

Data_Set	Target	Model	Feature_Set	Tuning_Params	RMSE	Rsquared	MAE
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 5	5.243636	0.526311	2.739698
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 10	5.240782	0.528236	2.738361
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 5	4.913722	0.551563	2.505217
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 10	4.906552	0.551734	2.497611
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 5	4.836141	0.570507	2.454533
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 10	4.831401	0.571078	2.446221
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 5	4.612592	0.596767	2.309789
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 10	4.621712	0.594714	2.309368
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 5	4.710652	0.589676	2.37543
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 50; shrinkage = 0.05; n.minobsinnode = 10	4.703931	0.589776	2.370923
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 5	4.512421	0.612272	2.24942
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 10	4.52995	0.608906	2.251288
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 5	4.913296	0.553015	2.511087
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 10	4.91044	0.552202	2.499558
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5	4.747453	0.572475	2.377217
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 1; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 10	4.740752	0.573856	2.369977
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 5	4.615373	0.596488	2.31264
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 10	4.612108	0.596553	2.306046
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5	4.489449	0.614727	2.228772
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 3; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 10	4.500676	0.613024	2.229482
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 5	4.512528	0.612486	2.254593
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 100; shrinkage = 0.05; n.minobsinnode = 10	4.517125	0.611727	2.24937
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5	4.409891	0.628261	2.189652
Samp_25	length_of_stay	gbm	Base_SDOH	interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 10	4.427401	0.62503	2.193205
Samp_25	length_of_stay	rf	Base_SDOH	mtry = 2	5.328237	0.503772	2.704013
Samp_25	length_of_stay	rf	Base_SDOH	mtry = 4	4.736256	0.590802	2.310073
Samp_25	length_of_stay	rf	Base_SDOH	mtry = 6	4.535841	0.61695	2.185634
Samp_25	length_of_stay	rf	Base_SDOH	mtry = 8	4.444558	0.62819	2.144546
Samp_25	length_of_stay	rf	Base_SDOH	mtry = 10	4.389407	0.635106	2.128654
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 3; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	5.490049	0.562636	2.411056
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 3; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.372638	0.634425	2.1597
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 3; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.296537	0.646463	2.127712
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 6; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	5.269321	0.608458	2.336881
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.289731	0.647618	2.095776
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 6; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.370011	0.635497	2.116342
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 9; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	5.226874	0.619438	2.305463
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 9; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.318633	0.643244	2.086937
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 100; max_depth = 9; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.464265	0.622011	2.161076
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 3; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.815672	0.585217	2.232057
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 3; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.30193	0.645652	2.128369
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 3; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.293541	0.646956	2.117548
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 6; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.577625	0.621984	2.122473
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.285518	0.648454	2.08835
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 6; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.410654	0.629938	2.134606
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 9; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.535097	0.628285	2.077103
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 9; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.325538	0.642565	2.09131
Samp_25	length_of_stay	xgbTree	Base_SDOH	nrounds = 200; max_depth = 9; eta = 0.3; gamma = 0; colsample_bytree = 1; min_child_weight = 1; subsample = 1	4.518828	0.614747	2.211394

Table 17 - Hyperparameter Tuning Results - Length of Stay

4.7.2 Binary Targets

For the binary outcomes (readmission, 30-day mortality, and in-hospital death) we trained a set of models to establish baseline performance and begin feature selection. Models

included XGBoost, GBM, Random Forest, GLM, GLMNet, Bayesian GLM, Naive Bayes, KNN, and a basic neural network. This mix allowed us to compare linear and non-linear methods with varying complexity.

30 Day Mortality and In Hospital Mortality

In baseline testing, the tree-based models (Random Forest, GBM, and XGBoost) achieved the strongest performance, with F1 scores in the range of 0.33 to 0.46. Recall values ranged from 0.23 to 0.65, while precision scores fell between 0.27 and 0.63. These results indicated that, although precision and recall varied across models, overall performance favored ensemble methods for this target. Based on this outcome, we selected the tree models to carry forward into full dataset training and final tuning.

Model	Feature_Set	Validation_Type	Train_Count	Accuracy	Kappa	ROC	PR_AUC	Specificity	Precision	Recall	F1
xgbTree	Base	4-Fold Cross-Validation	34285	0.963271	0.318919	0.896709	0.374686	0.992751	0.57088	0.23706	0.334473
xgbTree	Base	4-Fold Cross-Validation	34285	0.963271	0.318919	0.896709	0.374686	0.992751	0.57088	0.23706	0.334473
xgbTree	Base_SDOH	4-Fold Cross-Validation	34285	0.963513	0.322825	0.895926	0.382729	0.992919	0.578251	0.23913	0.338276
gbm	Base	4-Fold Cross-Validation	34285	0.948915	0.311414	0.886742	0.295455	0.973862	0.341738	0.334369	0.337976
gbm	Base_SDOH	4-Fold Cross-Validation	34285	0.952408	0.319295	0.888088	0.295592	0.978106	0.374105	0.319358	0.343784
rf	Base	4-Fold Cross-Validation	34285	0.92422	0.308814	0.884886	0.317897	0.941021	0.260089	0.510352	0.344542
rf	Base_SDOH	4-Fold Cross-Validation	34285	0.932983	0.310965	0.87877	0.284488	0.952556	0.278486	0.450828	0.344192
nnet	Base	4-Fold Cross-Validation	34285	0.782494	0.165274	0.877007	0.297483	0.782111	0.128648	0.791925	0.221293
nnet	Base_SDOH	4-Fold Cross-Validation	34285	0.754225	0.151166	0.877938	0.31419	0.751077	0.119588	0.831781	0.209066
glm	Base	4-Fold Cross-Validation	34285	0.803776	0.183651	0.885441	0.341442	0.804572	0.140089	0.784161	0.237685
glm	Base_SDOH	4-Fold Cross-Validation	34285	0.809409	0.192062	0.888214	0.347618	0.810035	0.145116	0.793996	0.245355
glmnet	Base	4-Fold Cross-Validation	34285	0.788369	0.167209	0.871727	0.292043	0.788792	0.130073	0.77795	0.222875
glmnet	Base_SDOH	4-Fold Cross-Validation	34285	0.789763	0.167519	0.872201	0.292637	0.79041	0.130347	0.77381	0.223105
bayesglm	Base	4-Fold Cross-Validation	34285	0.803897	0.183904	0.885938	0.3414	0.804677	0.140235	0.784679	0.237918
bayesglm	Base_SDOH	4-Fold Cross-Validation	34285	0.809349	0.192124	0.88903	0.347658	0.809951	0.145142	0.794513	0.245418
naive_bayes	Base	4-Fold Cross-Validation	34285	0.951459	0.306563	0.845023	0.283712	0.97756	0.388402	0.308489	0.330505
naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.942837	0.29194	0.832435	0.271188	0.967223	0.321231	0.342133	0.320534
knn	Base	4-Fold Cross-Validation	34285	0.826229	0.179437	0.811945	0.258872	0.832391	0.140413	0.674431	0.232424
knn	Base_SDOH	4-Fold Cross-Validation	34285	0.825482	0.176219	0.805815	0.277412	0.831971	0.138603	0.665631	0.229417

Table 18 - 30 Day Mortality - Baseline Model Performance

Model	Feature_Set	Validation_Type	Train_Count	Accuracy	Kappa	ROC	PR_AUC	Specificity	Precision	Recall	F1
xgbTree	Base	4-Fold Cross-Validation	34285	0.980777	0.446323	0.950215	0.513353	0.995187	0.634241	0.356903	0.455302
xgbTree	Base_SDOH	4-Fold Cross-Validation	34285	0.980919	0.454327	0.949384	0.520742	0.995104	0.63346	0.366731	0.463297
gbm	Base	4-Fold Cross-Validation	34285	0.967733	0.382387	0.943736	0.369431	0.979177	0.345157	0.472245	0.398525
gbm	Base_SDOH	4-Fold Cross-Validation	34285	0.968763	0.395901	0.942541	0.37953	0.979982	0.358806	0.482965	0.411572
rf	Base	4-Fold Cross-Validation	34285	0.95471	0.374712	0.943303	0.411346	0.961658	0.282408	0.653802	0.39442
rf	Base_SDOH	4-Fold Cross-Validation	34285	0.962847	0.393504	0.944558	0.398052	0.971822	0.320436	0.574187	0.411077
nnet	Base	4-Fold Cross-Validation	34285	0.878971	0.207248	0.92654	0.267703	0.880493	0.140244	0.813086	0.237802
nnet	Base_SDOH	4-Fold Cross-Validation	34285	0.845795	0.166283	0.921923	0.29831	0.845848	0.11333	0.84342	0.199456
glm	Base	4-Fold Cross-Validation	34285	0.892741	0.227685	0.935267	0.407004	0.894375	0.152389	0.821973	0.2571
glm	Base_SDOH	4-Fold Cross-Validation	34285	0.893226	0.230391	0.934176	0.414007	0.894705	0.153972	0.829122	0.259698
glmnet	Base	4-Fold Cross-Validation	34285	0.888561	0.214061	0.928869	0.359734	0.890677	0.144134	0.79695	0.244111
glmnet	Base_SDOH	4-Fold Cross-Validation	34285	0.889026	0.214653	0.929114	0.362731	0.891173	0.144546	0.796057	0.244662
bayesglm	Base	4-Fold Cross-Validation	34285	0.89268	0.227576	0.936092	0.407939	0.894313	0.152318	0.821973	0.256998
bayesglm	Base_SDOH	4-Fold Cross-Validation	34285	0.893286	0.230924	0.9352	0.415571	0.894726	0.15427	0.830911	0.260211
naive_bayes	Base	4-Fold Cross-Validation	34285	0.874851	0.195246	0.904176	0.424796	0.876382	0.131755	0.808554	0.226497
naive_bayes	Base	4-Fold Cross-Validation	34285	0.964059	0.368477	0.925035	0.32039	0.974818	0.316576	0.498169	0.385941
naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.855487	0.172542	0.900548	0.433499	0.856219	0.117279	0.823768	0.205251
naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.971732	0.352697	0.923962	0.323777	0.985746	0.372974	0.364852	0.367079
knn	Base	4-Fold Cross-Validation	34285	0.898698	0.214542	0.86622	0.313955	0.902741	0.146783	0.723595	0.244026
knn	Base_SDOH	4-Fold Cross-Validation	34285	0.902373	0.218519	0.856109	0.325594	0.90679	0.149981	0.711092	0.247657

Table 19 - Hospital Mortality - Baseline Model Performance

Selected features will be explored in the findings, and a full feature selection table for each target are provided in Appendix. These tables include all retained and removed predictors for models trained on both the SDOH and non-SDOH feature sets. In the main text, we focus on overall trends across models and targets. Generally, tree-based models retained a larger number of features, including clinical and comorbidity indicators, while linear models filtered more aggressively. SDOH variables were included in some models but often ranked lower in

Target	Model	Tuning_Params	Precision	Recall	F1
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.142037	0.701419	0.235702
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.466919	0.273088	0.344485
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.637344	0.220332	0.327195
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.210015	0.501714	0.295457
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.647996	0.222758	0.331424
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.6361	0.252069	0.360969
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.283714	0.375372	0.321404
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.641171	0.233875	0.34257
mortality_30_day	xgbTree	nrounds = 100; max_depth	0.604755	0.255704	0.359213
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.205719	0.573465	0.302059
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.606978	0.216896	0.319542
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.648204	0.236099	0.345959
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.312244	0.392962	0.347582
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.662056	0.236303	0.348104
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.61412	0.258335	0.363523
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.438177	0.291689	0.350052
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.646966	0.242366	0.352524
mortality_30_day	xgbTree	nrounds = 200; max_depth	0.598199	0.262782	0.365123
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.130184	0.680199	0.218473
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.129279	0.681208	0.217209
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.146143	0.676764	0.240332
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.147044	0.677371	0.241594
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.176669	0.650889	0.277721
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.177826	0.648464	0.279005
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.260026	0.499889	0.341866
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.257514	0.504136	0.340847
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.24	0.522928	0.328303
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.243074	0.516462	0.33017
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.331033	0.387098	0.356845
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.332093	0.377798	0.35341
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.146294	0.677574	0.240584
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.146521	0.683232	0.241246
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.182126	0.620973	0.281595
mortality_30_day	gbm	interaction.depth = 1; n.tr	0.181251	0.617333	0.280165
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.256028	0.505549	0.339733
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.258541	0.507974	0.342455
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.364505	0.335551	0.348987
mortality_30_day	gbm	interaction.depth = 3; n.tr	0.359092	0.342424	0.350511
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.334442	0.379617	0.355488
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.333593	0.379619	0.355086
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.459269	0.269053	0.339259
mortality_30_day	gbm	interaction.depth = 5; n.tr	0.455156	0.26177	0.332306
mortality_30_day	rf	mtry = 2	0.269976	0.478465	0.345057
mortality_30_day	rf	mtry = 4	0.505124	0.251462	0.335388
mortality_30_day	rf	mtry = 6	0.606098	0.219726	0.322258

Target	Model	Tuning_Params	Precision	Recall	F1
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.199171	0.738985	0.313688
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.478513	0.45906	0.468539
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.682084	0.375091	0.483811
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.296829	0.644514	0.406407
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.675095	0.375443	0.482326
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.685314	0.397132	0.502636
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.450392	0.509104	0.477265
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.692571	0.388393	0.497488
hospital_expire_flag	xgbTree	nrounds = 100; max_dep	0.677454	0.39399	0.49791
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.23193	0.699799	0.348323
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.626293	0.379281	0.472211
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.693668	0.391885	0.500588
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.393415	0.550029	0.458662
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.705669	0.386288	0.49895
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.671654	0.401336	0.502225
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.545163	0.460125	0.498811
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.698791	0.392592	0.502417
hospital_expire_flag	xgbTree	nrounds = 200; max_dep	0.678597	0.398185	0.501555
hospital_expire_flag	gbm	interaction.depth = 1; n	0.163917	0.75123	0.268948
hospital_expire_flag	gbm	interaction.depth = 1; n	0.163558	0.74948	0.26835
hospital_expire_flag	gbm	interaction.depth = 1; n	0.1705	0.74143	0.277223
hospital_expire_flag	gbm	interaction.depth = 1; n	0.172697	0.742478	0.280206
hospital_expire_flag	gbm	interaction.depth = 3; n	0.196519	0.74108	0.310637
hospital_expire_flag	gbm	interaction.depth = 3; n	0.197417	0.740731	0.311713
hospital_expire_flag	gbm	interaction.depth = 3; n	0.254753	0.651507	0.366281
hospital_expire_flag	gbm	interaction.depth = 3; n	0.253655	0.643457	0.363837
hospital_expire_flag	gbm	interaction.depth = 5; n	0.214917	0.722186	0.331205
hospital_expire_flag	gbm	interaction.depth = 5; n	0.216496	0.719735	0.332837
hospital_expire_flag	gbm	interaction.depth = 5; n	0.313337	0.593421	0.410021
hospital_expire_flag	gbm	interaction.depth = 5; n	0.3173	0.580475	0.410115
hospital_expire_flag	gbm	interaction.depth = 1; n	0.172752	0.743181	0.280314
hospital_expire_flag	gbm	interaction.depth = 1; n	0.171521	0.742127	0.278636
hospital_expire_flag	gbm	interaction.depth = 1; n	0.201516	0.70574	0.313489
hospital_expire_flag	gbm	interaction.depth = 1; n	0.20142	0.707139	0.313503
hospital_expire_flag	gbm	interaction.depth = 3; n	0.251334	0.655009	0.363241
hospital_expire_flag	gbm	interaction.depth = 3; n	0.251884	0.652907	0.363452
hospital_expire_flag	gbm	interaction.depth = 3; n	0.36218	0.524482	0.428381
hospital_expire_flag	gbm	interaction.depth = 3; n	0.369956	0.521339	0.432763
hospital_expire_flag	gbm	interaction.depth = 5; n	0.312369	0.585375	0.407347
hospital_expire_flag	gbm	interaction.depth = 5; n	0.316947	0.586777	0.411527
hospital_expire_flag	gbm	interaction.depth = 5; n	0.455532	0.451706	0.453453
hospital_expire_flag	gbm	interaction.depth = 5; n	0.458485	0.448552	0.452914
hospital_expire_flag	rf	mtry = 2	0.341889	0.609875	0.438079
hospital_expire_flag	rf	mtry = 4	0.651959	0.362155	0.465263

Table 20 - Tuning Results - Mortality Targets

importance relative to clinical variables. The most consistent predictors across targets included ICU-related interventions, severity scores, and prior visit history.

Model performance was evaluated using precision, recall, and F1 score, given the need to balance identifying high-risk outcomes while minimizing false alarms. Recall measures the ability to correctly

identify true positive cases, while precision reflects how many predicted positives were correct. The F1 score provides a single value that balances both. These metrics were selected due to their interpretability and relevance in imbalanced clinical prediction tasks, where both missed cases and over-alerting carry significant consequences. The parameters with results that best balanced these metrics were selected for the final models (table 21).

30 Day Readmission

Model performance for the 30-day readmission target showed a different trend compared to the mortality outcomes. In this case, linear models (GLM, GLMNet, and Bayesian GLM) performed the best, achieving F1 scores in the range of 0.39 to 0.41, with relatively stable precision and recall (table 22). These results stood out against lower-performing tree-based and non-linear models, which produced F1 scores ranging from the mid-teens to mid-thirties.

Interestingly, some of the linear models dropped a large number of features during selection. In one case, almost all predictors were removed, suggesting instability in the underlying signal or possibly sensitivity to class imbalance and encoding (table 23). These observations were an early indication that the cross-validation strategy may have introduced data leakage, later confirmed during final model validation.

Model	Feature_Set	Validation_Type	Train_Count	Accuracy	Kappa	ROC	PR_AUC	Specificity	Precision	Recall	F1	AIC	BIC
xgbTree	Base	4-Fold Cross-Validation	34285	0.798889	0.148612	0.667622	0.368819	0.965536	0.513463	0.1431	0.223817	NA	NA
xgbTree	Base_SDOH	4-Fold Cross-Validation	34285	0.796911	0.147731	0.674454	0.373972	0.962066	0.495242	0.146987	0.22656	NA	NA
gbm	Base	4-Fold Cross-Validation	34285	0.78213	0.211783	0.665693	0.371192	0.912636	0.438797	0.268561	0.333023	NA	NA
gbm	Base_SDOH	4-Fold Cross-Validation	34285	0.785664	0.200385	0.672217	0.374405	0.923652	0.447134	0.242651	0.3143	NA	NA
rf	Base	4-Fold Cross-Validation	34285	0.721979	0.20535	0.670358	0.33906	0.797493	0.347785	0.424813	0.382423	NA	NA
rf	Base_SDOH	4-Fold Cross-Validation	34285	0.742372	0.204923	0.677632	0.359163	0.837503	0.36541	0.368013	0.366549	NA	NA
nnet	Base	4-Fold Cross-Validation	34285	0.622837	0.174147	0.671329	0.363901	0.622715	0.295789	0.623321	0.40109	NA	NA
nnet	Base_SDOH	4-Fold Cross-Validation	34285	0.635416	0.173416	0.66453	0.337371	0.647529	0.299485	0.587743	0.394836	NA	NA
glm	Base	4-Fold Cross-Validation	34285	0.664331	0.200051	0.674701	0.374219	0.687794	0.317665	0.571999	0.408473	101554.75	102027.874
glm	Base_SDOH	4-Fold Cross-Validation	34285	0.662918	0.20297	0.680923	0.372278	0.683414	0.318492	0.582264	0.41175	100471.728	101102.56
glmnet	Base	4-Fold Cross-Validation	34285	0.626128	0.168186	0.665846	0.371963	0.63254	0.294044	0.6009	0.394464	NA	NA
glmnet	Base_SDOH	4-Fold Cross-Validation	34285	0.644119	0.179248	0.665845	0.37195	0.66037	0.302687	0.58017	0.397818	NA	NA
bayesglm	Base	4-Fold Cross-Validation	34285	0.664351	0.20017	0.674692	0.374216	0.687769	0.317724	0.572199	0.408573	101564.116	102074.348
bayesglm	Base_SDOH	4-Fold Cross-Validation	34285	0.662958	0.202928	0.680916	0.372278	0.683515	0.318486	0.582065	0.411695	100483.103	101160.32
naive_bayes	Base	4-Fold Cross-Validation	34285	0.25048	0.008089	0.627109	0.334947	0.074019	0.205912	0.944892	0.338133	NA	NA
naive_bayes	Base	4-Fold Cross-Validation	34285	0.776113	0.064745	0.64256	0.355203	0.941482	0.750611	0.125395	0.115562	NA	NA
naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.282625	0.020621	0.62191	0.326741	0.118587	0.211103	0.928151	0.343968	NA	NA
naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.69912	0.056305	0.636181	0.338863	0.815152	0.62994	0.242609	0.150046	NA	NA
knn	Base	4-Fold Cross-Validation	34285	0.547118	0.097817	0.614549	0.316856	0.52745	0.251402	0.624515	0.358486	NA	NA
knn	Base_SDOH	4-Fold Cross-Validation	34285	0.540192	0.103503	0.62306	0.3309	0.511623	0.253519	0.652616	0.36516	NA	NA

Table 21 - 30 Day Readmission Baseline results

After selecting the linear models for 30-day readmission, we continued with hyperparameter tuning and re-ran the models on the 25% training set. The results looked even better than the initial baseline. Recall values ranged from 0.56 to 0.64, showing strong sensitivity in identifying readmitted patients. Precision was lower, between 0.29 and 0.32, which is expected given the class imbalance and the nature of the problem, catching more target class outcomes based on leakage. These results further supported the choice to continue with the linear models for this target.

	bayesglm_base	bayesglm_base_sdoh	gbm_base	gbm_base_sdoh	glm_base	glm_base_sdoh	glmnet_base	glmnet_base_sdoh	naive_bayes_base	naive_bayes_base_sdoh	rnet_base	rnet_base_sdoh	rf_base	rf_base_sdoh	xgbTree_base	xgbTree_base_sdoh
admission_location_numeric	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
age	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
age_category	X	X	X	X	X	X	X	X	-	-	X	X	X	X	X	X
antiarrhythmic_med_flag	-	-	-	X	-	-	X	X	-	-	-	-	-	-	-	-
antifungal_med_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
blood_transfusion_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
bowel_resection_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
broad_spectrum_antibiotic_med_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
cardiac_surgery_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
central_line_flag	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
colostomy_ileostomy_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
complication_binary	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
complication_index_log	X	X	-	-	X	X	X	X	-	-	-	-	-	-	-	-
corticosteroid_med_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
cpr_flag	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
dialysis_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
diuretic_med_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
ecmo_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	X	X
ed_acuity_log	X	X	-	-	X	X	X	X	-	-	-	-	-	-	-	-
emergency_surgery_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
gastrostomy_tube_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
gender	X	X	X	X	X	X	X	X	-	-	X	X	X	X	X	X
inotrope_med_flag	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
insurance_numeric	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
is_white	-	X	-	X	-	X	-	X	-	-	-	-	-	-	-	X
los_icu_log	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
major_surgery_flag	X	X	-	-	X	X	X	X	-	-	-	-	-	-	-	-
marital_status_numeric	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
max_apsiii	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
max_lods	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
max_oasis	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
max_sapsii	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
max_sirs_log	X	X	-	-	X	X	X	X	-	-	-	-	-	-	-	-
nephrotoxic_med_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
neurosurgery_flag	X	X	X	X	X	X	X	X	-	-	-	-	-	-	X	X
num_medications_log	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
num_procedures_log	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
orthopedic_surgery_flag	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
pressor_med_flag	-	-	-	X	-	-	X	X	-	-	-	-	-	-	-	-
prior_er_visits_1yr_log	-	-	-	-	-	-	X	X	-	-	-	-	-	-	-	-
prolonged_ventilation_flag	X	X	X	X	X	X	X	X	X	X	X	-	X	X	X	X
race_group	-	X	-	X	-	X	-	X	-	-	-	X	-	X	-	X
respiratory_support_flag	X	-	X	X	X	-	X	X	-	-	-	-	-	-	-	-
sedative_med_flag	X	X	-	-	X	X	X	X	-	-	-	-	-	-	-	-
sepsis_infection_flag	-	X	X	X	-	X	X	X	-	-	-	-	-	-	-	-
shock_management_flag	X	X	X	X	X	X	X	X	X	X	-	-	X	X	X	X
stroke_prevention_flag	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
total_icu_hours_log	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
total_icu_stays_log	X	X	X	X	X	X	X	X	-	-	-	-	-	-	-	-
tracheostomy_flag	-	-	X	X	-	-	X	X	-	-	-	-	-	-	-	-
weighted_education_log	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
weighted_fam_income_log	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
weighted_fam_size	-	X	-	X	-	X	-	X	-	-	-	-	-	-	-	-
weighted_inc_welfare_log	-	-	-	-	-	-	-	X	-	-	-	-	-	-	-	-
weighted_income_log	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
weighted_ownership_log	-	X	-	-	-	X	-	X	-	-	-	-	-	-	-	-
weighted_poverty	-	X	-	X	-	X	-	X	-	-	-	-	-	-	-	-
weighted_wage_sal_log	-	-	-	X	-	-	-	X	-	-	-	-	-	-	-	-
wound_debridement_flag	X	X	X	X	X	X	X	X	X	X	-	-	X	X	X	X

Figure 20 - Removed Features - Readmission

Target	Model	Tuning_Params	Train_Count	Test_Count	Accuracy	Kappa	ROC	PR_AUC	Sensitivity	Precision	Recall	F1
readmitted	nnet	size = 3; decay = 0.01	86398	37394	0.632844	0.17766	0.661034	0.331628	0.604873	0.300368	0.604873	0.401331
readmitted	nnet	size = 3; decay = 0.1	86398	37394	0.667119	0.199955	0.671766	0.354069	0.56302	0.320045	0.56302	0.407347
readmitted	nnet	size = 3; decay = 0.5	86398	37394	0.652005	0.196822	0.677108	0.356533	0.596778	0.313866	0.596778	0.411068
readmitted	nnet	size = 5; decay = 0.01	86398	37394	0.641624	0.175501	0.653544	0.34481	0.571153	0.305818	0.571153	0.393302
readmitted	nnet	size = 5; decay = 0.1	86398	37394	0.625461	0.180104	0.674193	0.370492	0.627996	0.300977	0.627996	0.405405
readmitted	nnet	size = 5; decay = 0.5	86398	37394	0.61559	0.176935	0.676644	0.372746	0.644225	0.29782	0.644225	0.406031
readmitted	nnet	size = 7; decay = 0.01	86398	37394	0.628514	0.183147	0.674706	0.357395	0.626732	0.302738	0.626732	0.407314
readmitted	nnet	size = 7; decay = 0.1	86398	37394	0.630226	0.186599	0.680126	0.370697	0.630182	0.304617	0.630182	0.409741
readmitted	nnet	size = 7; decay = 0.5	86398	37394	0.641931	0.191476	0.681675	0.379997	0.610941	0.309139	0.610941	0.410102
readmitted	glm	Default	86398	37394	0.665649	0.208081	0.68412	0.382591	0.584481	0.322747	0.584481	0.415853
readmitted	bayesglm	Default	86398	37394	0.665665	0.208101	0.68412	0.382593	0.584481	0.322761	0.584481	0.415864

Table 22 - Hyper tuning Results - Readmission

4.8 Model Results & Evaluation

30 Day Mortality

For 30-day mortality prediction, the GBM model achieved the best overall balance of performance, with the highest F1 score across both the Base and Base_SDOH feature sets. GBM offered modest recall while maintaining reasonable precision, outperforming the other models on this combined metric (Table 23). Although the Random Forest model had lower overall F1 scores compared to GBM, its Precision-Recall curve showed stronger precision across clinically important recall ranges (approximately 0.1–0.5)(Figure 22). This suggests that Random Forest may be preferable if operational goals prioritize early identification of high-risk patients while maintaining reasonable precision.

Target	Model	Feature_Set	Validation_Type	Accuracy	Kappa	ROC	PR_AUC	Specificity	Precision	Recall	F1
mortality_30	xgbTree	Base_SDOH	Final Model (No CV)	0.963321	0.1782	0.87606	0.354826	0.998667	0.766169	0.105915	0.186103
mortality_30	xgbTree	Base	Final Model (No CV)	0.963335	0.178735	0.875991	0.35555	0.998667	0.766749	0.106259	0.186651
mortality_30	gbm	Base_SDOH	Final Model (No CV)	0.963934	0.254922	0.892103	0.377049	0.996853	0.684211	0.165406	0.266408
mortality_30	gbm	Base	Final Model (No CV)	0.963852	0.253621	0.892375	0.379153	0.996796	0.679433	0.164718	0.265154
mortality_30	rf	Base_SDOH	Final Model (No CV)	0.962777	0.129822	0.835791	0.387865	0.999433	0.84252	0.07359	0.135357
mortality_30	rf	Base	Final Model (No CV)	0.962913	0.135539	0.80758	0.386068	0.999433	0.848485	0.077029	0.141236

Table 23 - 30 Day Mortality Final Results

While ROC curves were generated for completeness, they are less informative in the context of highly imbalanced outcomes like mortality and readmission. Due to the rarity of positive events, ROC

AUC values tend to remain artificially high, even when model discrimination is limited.

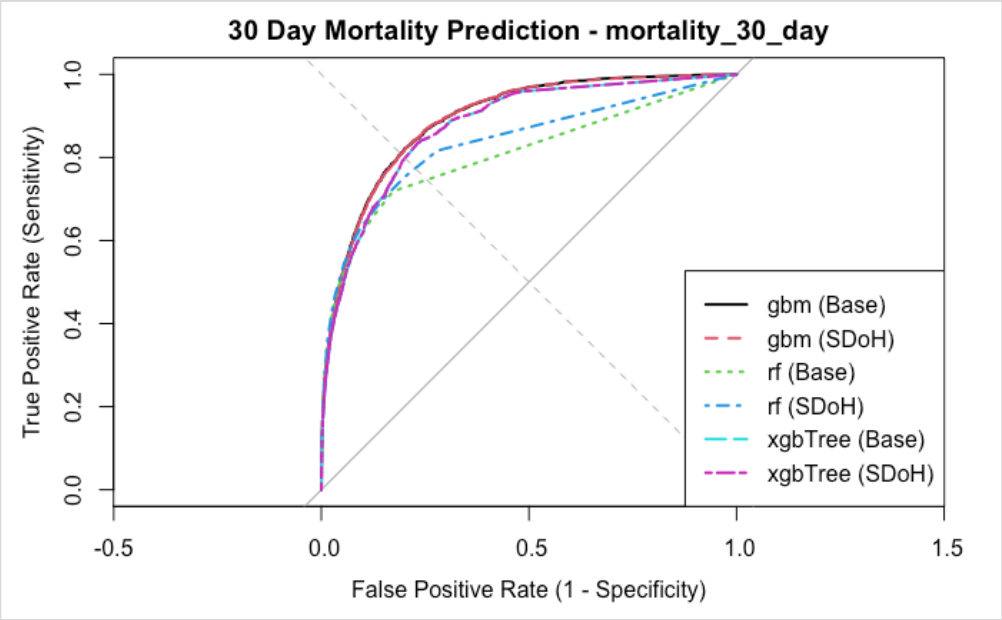


Figure 21 - ROC AUC - 30 Day Mortality

Random Forest exhibited some divergence from XGBoost and GBM in the ROC curve at higher sensitivity levels (above 70%), as shown in Figure 21. However, because operational use cases are more likely to prioritize recall ranges between 30% and 50%, this difference was not considered critical for model selection.

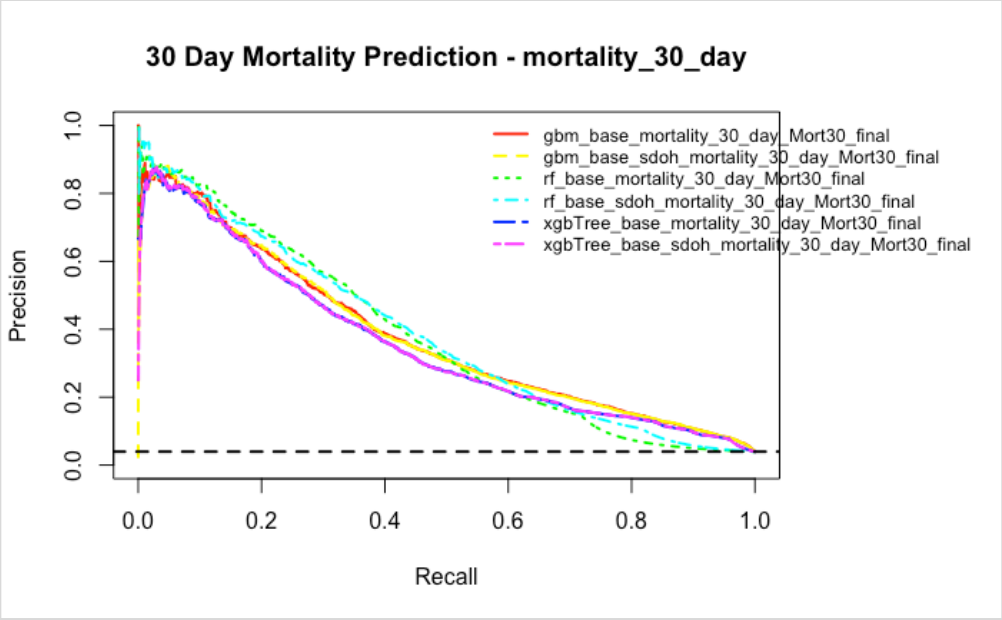


Figure 22 - PR AUC - 30 Day Mortality

Hospital Mortality

Target	Model	Feature_Set	Validation_Type	Accuracy	Kappa	ROC	PR_AUC	Specificity	Precision	Recall	F1
hospital_expire_flag	xgbTree	Base_SDOH	Final Model (No CV)	0.981252	0.418406	0.942213	0.531927	0.997867	0.769578	0.294524	0.426011
hospital_expire_flag	xgbTree	Base	Final Model (No CV)	0.981034	0.416811	0.938939	0.528927	0.997602	0.749271	0.296254	0.424618
hospital_expire_flag	gbm	Base_SDOH	Final Model (No CV)	0.980612	0.394412	0.943539	0.515513	0.997657	0.74034	0.276081	0.402183
hospital_expire_flag	gbm	Base	Final Model (No CV)	0.980626	0.385121	0.937493	0.50037	0.997936	0.756579	0.26513	0.392659
hospital_expire_flag	rf	Base_SDOH	Final Model (No CV)	0.979292	0.235808	0.926504	0.536925	0.999624	0.899254	0.138905	0.240639
hospital_expire_flag	rf	Base	Final Model (No CV)	0.979523	0.247591	0.91316	0.533219	0.999679	0.916968	0.146398	0.252485

Table 24 - Hospital Mortality Final Results

For the hospital mortality target, the XGBoost model achieved the highest F1 scores (0.426 for Base_SDOH, 0.425 for Base), followed closely by GBM. Random Forest achieved the highest precision across all models (approximately 0.90–0.92), but at the cost of substantially lower recall (around 0.14), resulting in lower F1 scores overall (Table 24). Random Forest diverged from XGBoost and GBM at low sensitivity levels (around 20%), indicating weaker ability to capture positive cases early (Figure 23). In contrast, XGBoost and GBM maintained stronger true positive rates and largely overlapped on the ROC curve across most thresholds. XGBoost achieved the highest Precision-Recall AUC, further supporting its selection as the preferred model for hospital mortality prediction(Figure 24).

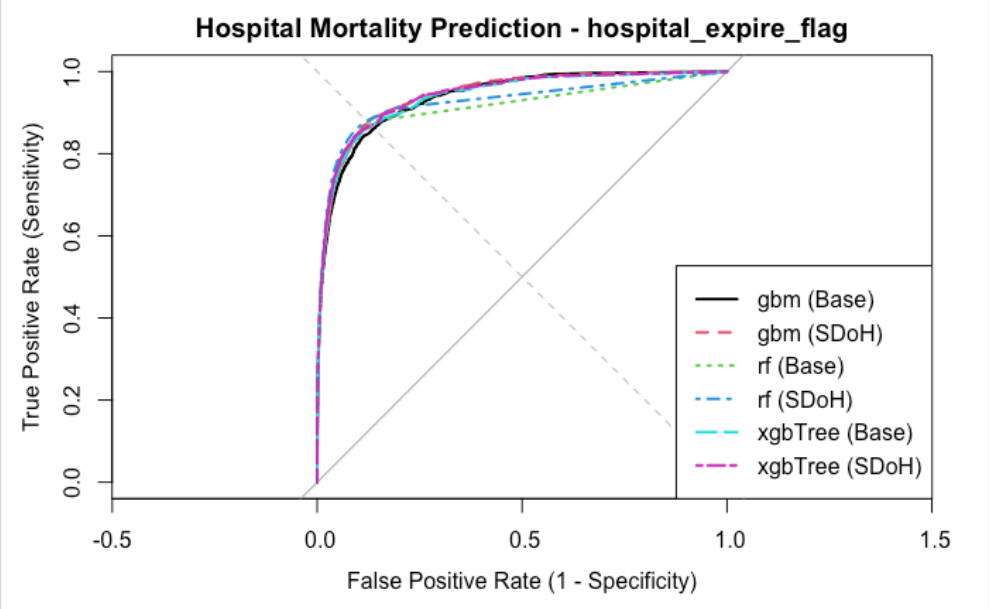


Figure 23 - ROC AUC - Hospital Mortality

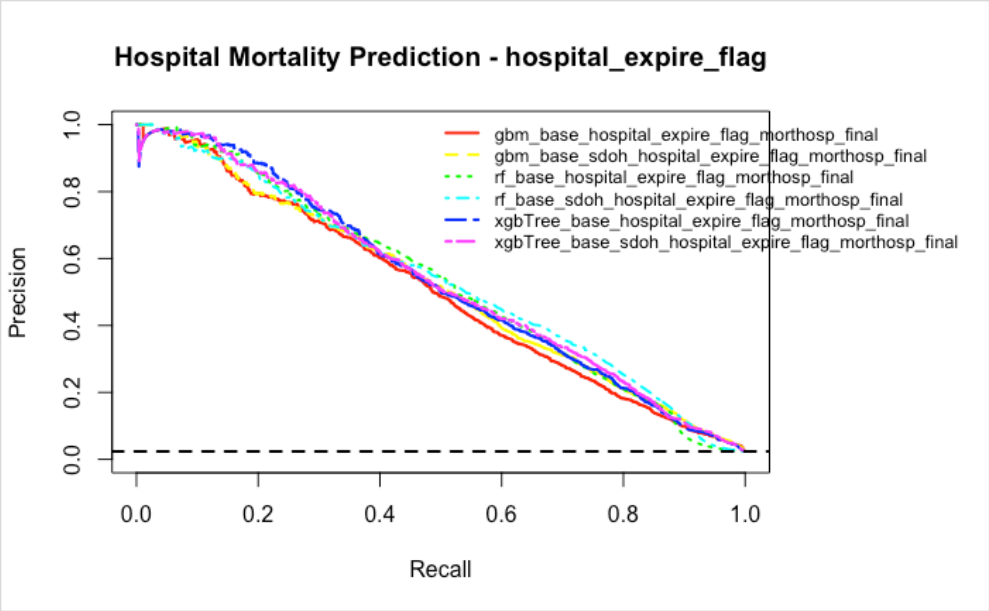


Figure 24 - PR AUC - Hospital Mortality

30 Day Readmission

Final model performance for 30-day readmission prediction was poor across all models, with F1 scores below 0.16 and low recall regardless of feature set. While neural networks and generalized linear models (GLM, bayesGLM) achieved slightly higher precision, the models struggled to capture meaningful signal in this noisy outcome. The relatively weak results were expected given the complexity of predicting readmissions, in addition to the previously identified cross-validation issue may have masked the relative performance of different models during early selection. It is possible that alternative models or feature sets would have emerged under a fully patient-grouped validation scheme. No further threshold optimization or model tuning was pursued for this target.

Target	Model	Feature_Set	Validation_Type	Accuracy	Kappa	ROC	PR_AUC	Sensitivity	Specificity	Precision	Recall	F1	AIC	BIC
readmitted	nnet	Base_SDOH	Final Model (No CV)	0.803	0.104	0.676	0.366	0.089	0.982	0.558	0.089	0.154	NA	NA
readmitted	nnet	Base	Final Model (No CV)	0.804	0.097	0.684	0.375	0.080	0.986	0.585	0.080	0.141	NA	NA
readmitted	glm	Base_SDOH	Final Model (No CV)	0.804	0.101	0.682	0.373	0.084	0.984	0.575	0.084	0.147	328812.152	329156.682
readmitted	glm	Base	Final Model (No CV)	0.804	0.099	0.678	0.372	0.083	0.985	0.573	0.083	0.145	329362.232	329631.396
readmitted	bayesglm	Base_SDOH	Final Model (No CV)	0.804	0.101	0.682	0.373	0.084	0.984	0.575	0.084	0.147	328812.156	329156.686
readmitted	bayesglm	Base	Final Model (No CV)	0.804	0.099	0.678	0.372	0.083	0.985	0.574	0.083	0.145	329362.237	329631.401

Table 25 - Readmission Final Results

Length of Stay

Among the three tree-based models evaluated for Length of Stay prediction (XGBoost, GBM, and Random Forest), XGBoost (xgbTree) achieved the best overall performance. It produced the lowest RMSE (4.21 days for SDOH and 4.21 days for Base feature sets) and the lowest MAE (2.07 days),

suggesting that on average, the model's predictions were within about 2 days of the actual length of stay (Table 26). While Random Forest explained variance in the target slightly better than the other models (R^2 0.6298 for Base compared to XGBoost's 0.6634), the differences were marginal.

Target	Model	Feature_Set	Validation_Type	Description	RMSE	Rsquared	MAE
length_of_stay	xgbTree	Base_SDOH	Final Model (No CV)	Model trained on Base_SDOH for length_of_stay	4.21069	0.663185	2.075493
length_of_stay	xgbTree	Base	Final Model (No CV)	Model trained on Base for length_of_stay	4.209163	0.663424	2.074382
length_of_stay	gbm	Base_SDOH	Final Model (No CV)	Model trained on Base_SDOH for length_of_stay	4.347972	0.641508	2.177525
length_of_stay	gbm	Base	Final Model (No CV)	Model trained on Base for length_of_stay	4.370915	0.637456	2.180746
length_of_stay	rf	Base_SDOH	Final Model (No CV)	Model trained on Base_SDOH for length_of_stay	4.540009	0.614099	2.173581
length_of_stay	rf	Base	Final Model (No CV)	Model trained on Base for length_of_stay	4.426778	0.629823	2.110011

Table 26 - Length of Stay Final Results

We also generated scatter plots comparing the actual versus predicted Length of Stay for each model. Across all models, there was a slight tendency to overestimate LOS, particularly for shorter stays. Among the models, XGBoost demonstrated the closest alignment to the 45-degree reference line, indicating better calibration between predictions and actual outcomes (Figure 25). Random Forest and GBM also showed reasonable agreement but exhibited more dispersion, particularly at the extremes of the LOS distribution (Figure 26 & 27).

To further assess model calibration and error patterns, we plotted actual versus predicted Length of Stay values for each final model across both the Base and SDOH feature sets. Across all models, there was a consistent tendency to slightly overestimate LOS, especially for shorter stays. However, the degree of bias was relatively minor overall. XGBoost models followed the 45-degree reference line most closely, showing better prediction alignment across the full range of stays (Figure 28).

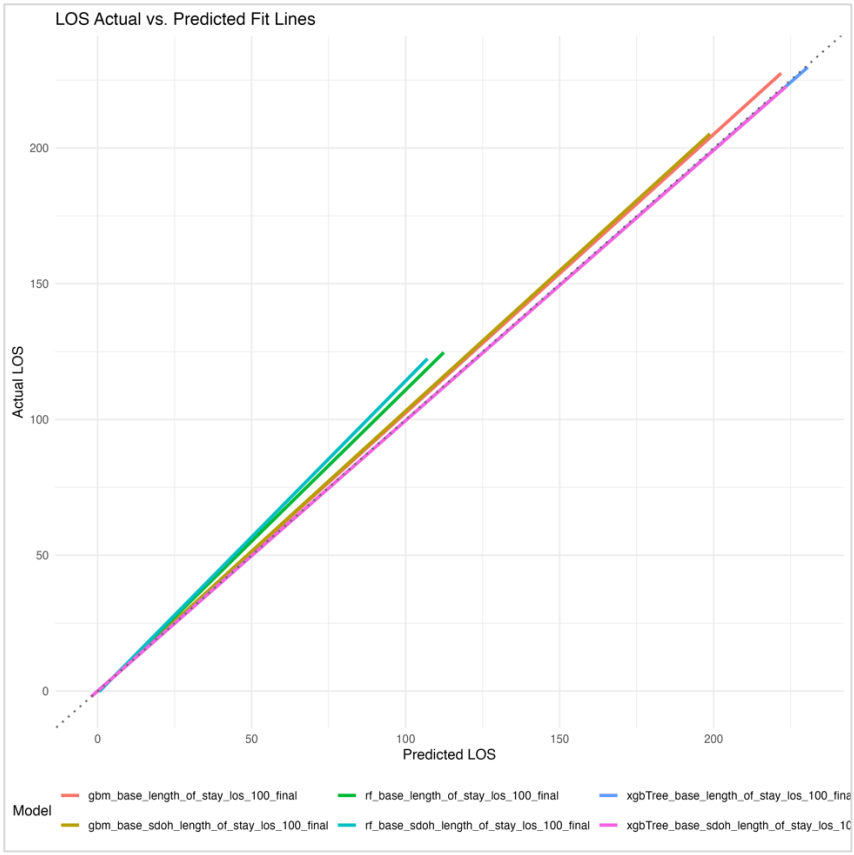


Figure 25 - Performance All Models - Length of Stay

The addition of SDOH variables had little effect on model performance. Predicted values and overestimation patterns looked nearly identical between the Base and SDOH models, confirming that SDOH features did not meaningfully improve LOS prediction in this dataset.

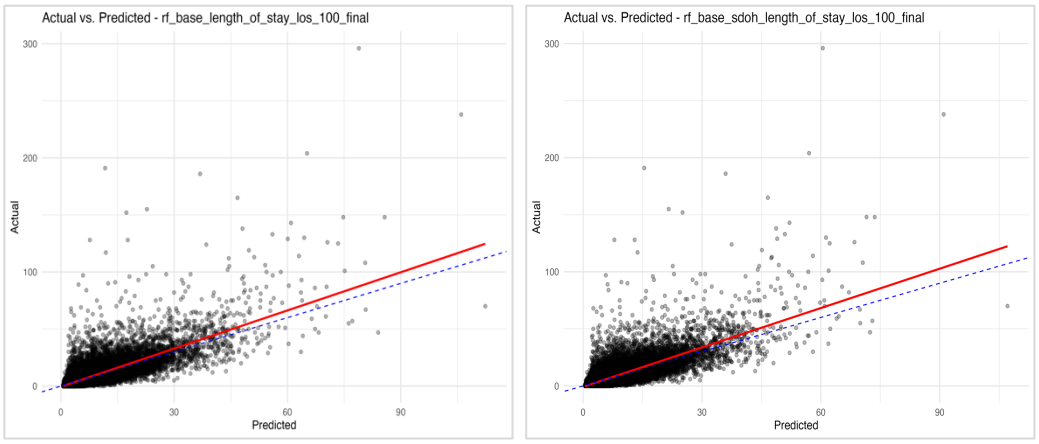


Figure 26 - Performance by Feature Set - GBM

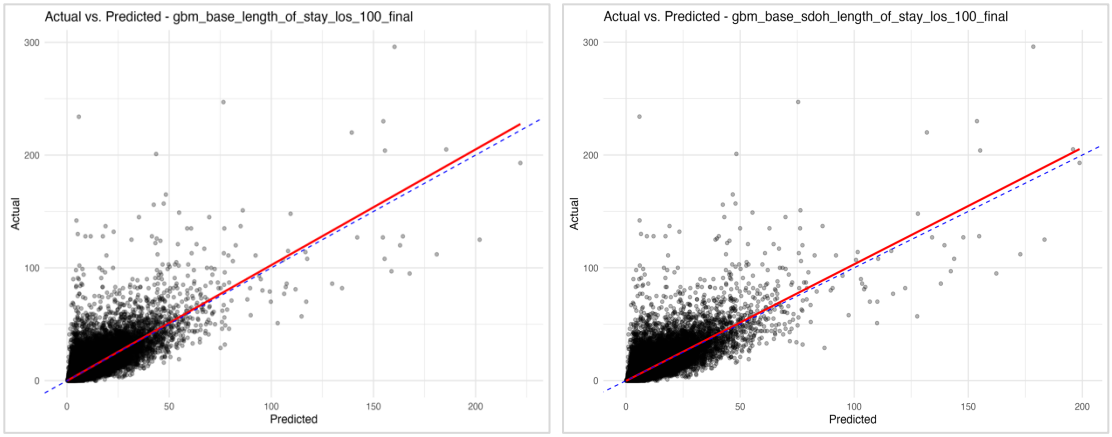


Figure 27 - Performance by Feature Set - RF

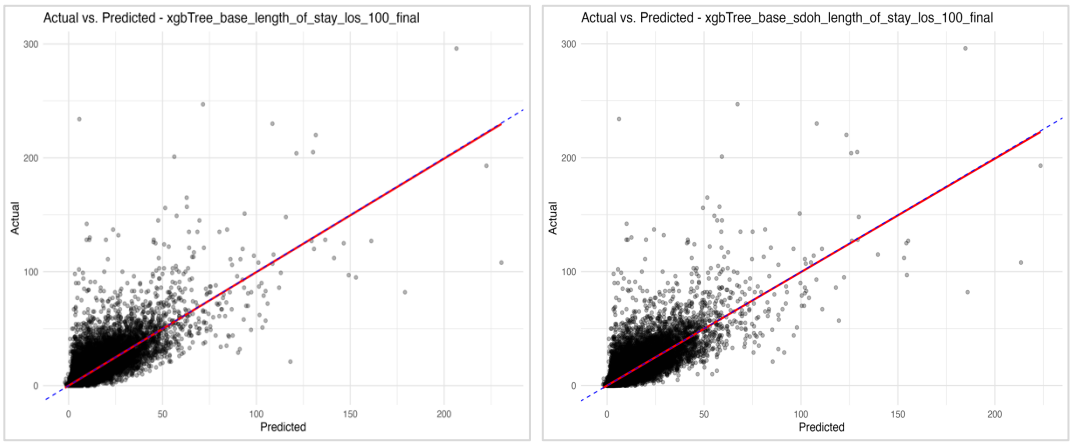


Figure 28 - Performance by Feature Set - xgbTree

5. Findings

Tree-based models (XGBoost, GBM, and Random Forest) consistently performed best across all targets. For Length of Stay (LOS) prediction, XGBoost achieved the lowest RMSE and most closely followed the actual LOS values across the patient population.

For binary outcomes, including 30-day mortality and 30-day readmission, GBM and XGBoost models provided the best balance between recall and precision at the default 0.5 probability threshold. Although differences between models were modest, GBM offered slightly higher F1 scores and better performance across key evaluation metrics.

Final model selection will depend on operational priorities, such as whether higher recall or higher precision is more critical for the clinical application.

The results of the analysis from these final models allow us to revisit the original research questions and hypotheses.

Research Question 1:

What patient-level and system-level features are most predictive of clinical outcomes including *30-day readmission, in-hospital mortality, and length of stay?*

Findings supported this inquiry. Clinical features such as the Charlson Comorbidity Index, patient age, ICU exposure, prior hospital utilization, and severity scores (e.g., SAPS II, OASIS, ED acuity log-transformed) consistently emerged as important predictors across models. In addition to these structural factors, key medications, including diuretics, sedatives, corticosteroids, broad-spectrum antibiotics, and vasopressor agents, were also highly predictive, particularly for mortality-related outcomes.

Notably, procedural variables were not consistently selected highest among the most predictive features. This may reflect either the limited frequency of certain high-risk procedures within the cohort, or the fact that their impact is captured indirectly through severity scores or medication administration patterns.

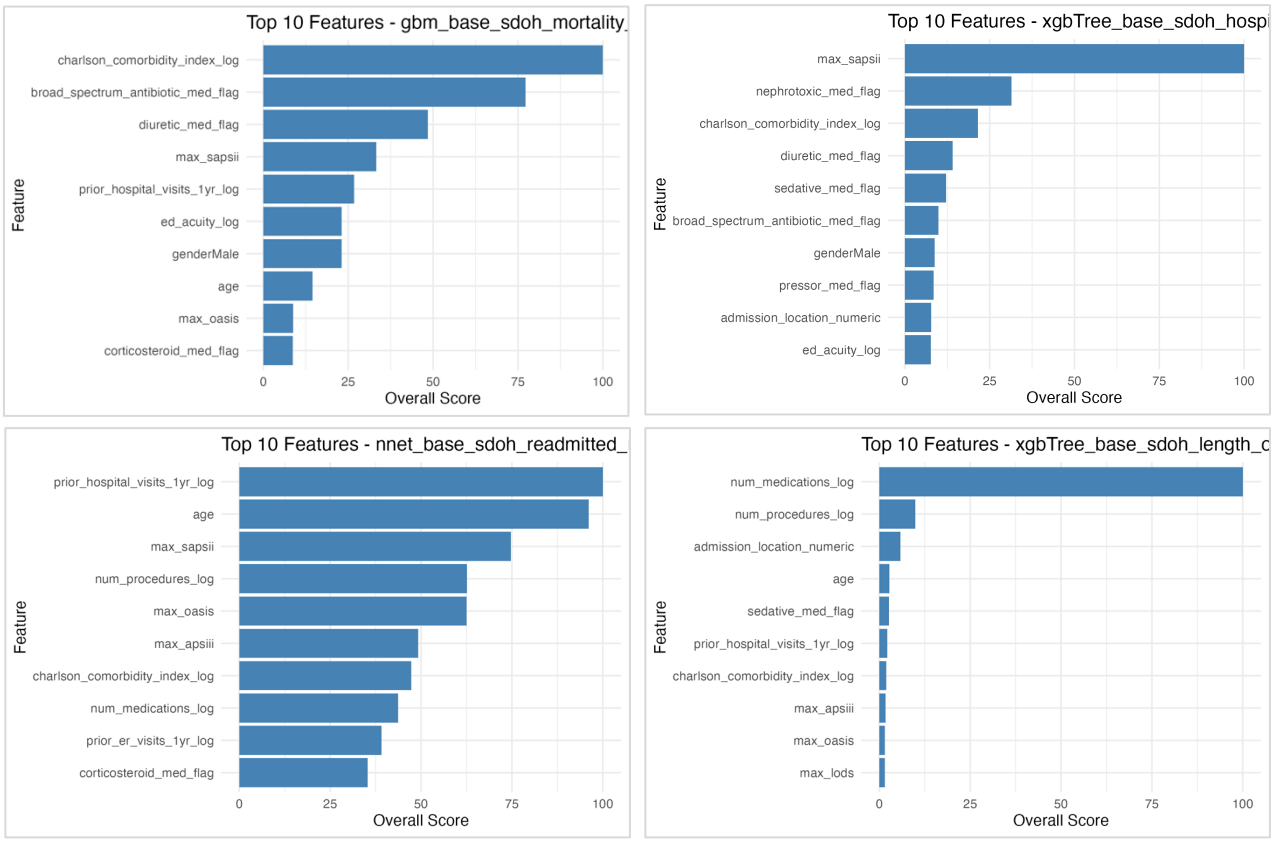


Figure 29 - Top Features for Top Models

Research Question 2:

To what extent do social determinants of health (SDoH), including race, insurance status, and marital status, influence these outcomes in the context of value-based care?

SDoH variables showed statistically significant associations with outcomes, particularly for insurance type, race group, and several socioeconomic indicators. However, the magnitude of these effects was modest, and their contribution to model performance remained limited.

Hypothesis 1 (H1):

Certain clinical variables (e.g., Charlson Comorbidity Index, age) are significant predictors of adverse outcomes.

Assessment: Supported.

Clinical features such as the Charlson Comorbidity Index, ICU exposure, select medications (diuretics, corticosteroids, sedatives, broad-spectrum antibiotics), and acuity measures (ED acuity score) consistently ranked among the most important predictors across models and outcomes.

Hypothesis 2 (H2):

SDoH features such as race and insurance type are statistically associated with differences in outcome predictions.

Assessment: Partially supported.

While some SDoH features (e.g., insurance status, race, weighted home ownership) showed modest statistical associations with outcomes, their overall impact on model performance was limited. Differences in predictive performance between models with and without SDoH features were small across all targets. Given the consistently minor changes in metrics, formal statistical comparison was not performed. Future work could focus on more specific patient subgroups—such as cohorts defined by certain diagnoses, procedures, or treatment patterns—to better understand where SDoH may have a stronger influence.

Hypothesis 3 (H3):

Models incorporating both clinical and socioeconomic variables will demonstrate improved predictive performance over those using clinical variables alone.

Assessment: Not supported.

Across all targets, models including SDoH variables did not demonstrate meaningful improvements in predictive metrics (e.g., RMSE, F1, ROC) compared to models using only clinical features. In several cases, models trained without SDoH features even outperformed their SDoH-augmented counterparts by small margins. The magnitude of these observed differences was consistently minor, suggesting that the addition of SDoH variables provided little incremental benefit in this cohort. Formal statistical tests were not performed because the differences were too small and inconsistent to be meaningful.

6. Threats To Validity

First, although patient-level grouping was enforced during final model evaluation, earlier feature selection and hyperparameter tuning stages were affected by data leakage. However, retrained models showed minimal differences (–0.03 to 0.01) in key metrics. Feature sets and optimal hyperparameters remained stable, supporting the robustness of final model choices despite early procedural issues.

					Corrected Split		
Target	Model	Feature_Set	Validation_Type	Train_Count	Δ Precision	Δ Recall	Δ F1
hospital_expire_flag	xgbTree	Base	4-Fold Cross-Validation	34285	-0.05	-0.04	-0.04
hospital_expire_flag	xgbTree	Base_SDOH	4-Fold Cross-Validation	34285	-0.05	-0.03	-0.04
hospital_expire_flag	gbm	Base	4-Fold Cross-Validation	34285	0.02	-0.01	0.01
hospital_expire_flag	gbm	Base_SDOH	4-Fold Cross-Validation	34285	0.02	-0.02	0.01
hospital_expire_flag	rf	Base	4-Fold Cross-Validation	34285	0.00	-0.03	-0.01
hospital_expire_flag	rf	Base_SDOH	4-Fold Cross-Validation	34285	0.01	-0.03	0.00
hospital_expire_flag	nnet	Base	4-Fold Cross-Validation	34285	-0.01	0.00	-0.02
hospital_expire_flag	nnet	Base_SDOH	4-Fold Cross-Validation	34285	0.01	-0.06	0.01
hospital_expire_flag	naive_bayes	Base	4-Fold Cross-Validation	34285	-0.02	0.03	-0.04
hospital_expire_flag	naive_bayes	Base	4-Fold Cross-Validation	34285	0.11	-0.20	-0.05
hospital_expire_flag	naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	-0.02	0.04	-0.04
hospital_expire_flag	naive_bayes	Base_SDOH	4-Fold Cross-Validation	34285	0.12	-0.19	-0.11

Table 27 - Mitigated Baseline Performance - Hospital Mortality

			Corrected Split		
Target	Model	Tuning_Params	Δ Precision	Δ Recall	Δ F1
hospital_expire_flag	xgbTree	nrounds = 100; max_	0.00	-0.03	0.00
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.02	0.00	-0.01
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.04	-0.01	-0.02
hospital_expire_flag	xgbTree	nrounds = 100; max_	0.00	-0.02	0.00
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.04	0.00	-0.01
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.05	-0.01	-0.03
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.04	-0.02	-0.03
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.04	-0.02	-0.02
hospital_expire_flag	xgbTree	nrounds = 100; max_	-0.03	-0.02	-0.02
hospital_expire_flag	xgbTree	nrounds = 200; max_	0.01	-0.03	0.00
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.01	-0.01	-0.01
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.05	0.00	-0.02
hospital_expire_flag	xgbTree	nrounds = 200; max_	0.00	-0.01	0.00
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.04	-0.01	-0.02
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.03	-0.02	-0.02
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.04	-0.02	-0.03
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.03	-0.02	-0.02
hospital_expire_flag	xgbTree	nrounds = 200; max_	-0.03	-0.02	-0.02
hospital_expire_flag	gbm	interaction.depth =	-0.01	-0.01	-0.01
hospital_expire_flag	gbm	interaction.depth =	-0.01	-0.01	-0.01
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.03	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.03	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.02	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.01	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.03	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.03	0.01
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	0.00	0.00
hospital_expire_flag	gbm	interaction.depth =	0.00	-0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.01	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	-0.01	0.01
hospital_expire_flag	gbm	interaction.depth =	0.01	0.00	0.01
hospital_expire_flag	gbm	interaction.depth =	0.00	0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	0.01	0.00	0.01
hospital_expire_flag	gbm	interaction.depth =	0.00	0.00	0.00
hospital_expire_flag	gbm	interaction.depth =	-0.01	0.01	0.00
hospital_expire_flag	gbm	interaction.depth =	-0.01	0.01	0.00
hospital_expire_flag	rf	mtry = 2	0.06	-0.07	0.03

Table 28 - Mitigated Model Tuning - Hospital Mortality

Second, the available features from MIMIC-IV and subsequently joined datasets do not fully capture the complexity of social determinants of health. The observed minimal impact of SDoH features could reflect data limitations rather than true lack of effect.

Third, outcome prevalence, particularly for mortality and readmission targets, was highly imbalanced. Although class imbalance techniques (e.g., SMOTE) were explored, future efforts could refine balancing approaches to further reduce bias.

Fourth, the use of a broad, heterogeneous hospital population may have obscured condition-specific patterns in outcomes or SDoH influence, limiting the ability to detect subgroup-specific effects.

Finally, all models were trained and evaluated using a single sample from the MIMIC-IV database, which may limit external validity. Results may differ in other hospital systems or more diverse patient populations.

7. Future Work

Re-evaluate Readmission Models with Corrected Validation Procedures

The readmission models suffered from poor final performance, likely due in part to initial data leakage during cross-validation. A full retraining and evaluation using patient-level grouping and corrected validation would help determine whether stronger predictive performance can be achieved.

Threshold Tuning and Calibration

We evaluated all models using a standard 0.5 probability threshold. Future work should explore adjusting thresholds to prioritize operational goals, such as favoring recall for mortality prediction or precision for resource-intensive outcomes like readmission. Additionally, calibration techniques such as Platt scaling or isotonic regression could be applied to better align predicted probabilities with observed event rates.

Subgroup Modeling, Fairness Assessment, and SDoH Exploration

Aggregating across a heterogeneous hospital population may obscure important clinical and social patterns. Future work could stratify patients by diagnosis category, service line, or admission source to identify group-specific predictors, improve predictive performance, and assess fairness across demographic groups. Additionally, within clinically homogeneous cohorts, the impact of social determinants of health (e.g., insurance status, race, neighborhood income) could be more meaningfully isolated, offering a clearer view of potential disparities in outcomes and model performance.

Failure Clustering and Model Robustness

Analyzing cases with high prediction error, false negatives, or misclassifications through clustering methods could reveal systematic vulnerabilities. Failure clustering could reveal subpopulations affected by feature gaps, label noise, or hidden confounders. Insights from these analyses could guide improved feature engineering or the design of hybrid models tailored to high-risk groups. It could also support fairness assessments by detecting performance disparities not visible through global metrics.

8. Reflections

This project helped me grow a lot technically, especially in working with large, real-world clinical datasets. I am proud of the technical growth I achieved by building an end-to-end predictive modeling workflow, from data extraction and cleaning to feature engineering, model training, and evaluation. Looking back, I probably tried to tackle too many outcomes at once, but that pushed me to figure out how to scale my experiments, stay organized, and prioritize what mattered most.

Dealing with noisy targets, imbalanced classes, and complex feature sets gave me a much better understanding of the real challenges in healthcare modeling. I also had to build and maintain a large, modular codebase to manage SQL queries, feature engineering in R, exploratory analysis, and modeling. This experience reinforced the importance of a structured process, reproducibility, and steady progress even when the project felt overwhelming.

References

Braveman, P., & Gottlieb, L. (2014). The social determinants of health: It's time to consider the causes of the causes. Public Health Reports, 129(Suppl 2), 19–31. National Center for Biotechnology Information (NCBI). https://pmc.ncbi.nlm.nih.gov/articles/PMC3863696/

Bhavsar, N. A., Gao, A., Phelan, M., Pagidipati, N. J., & Goldstein, B. A. (2018). Value of Neighborhood Socioeconomic Status in Predicting Risk of Outcomes in Studies That Use Electronic Health Record Data. JAMA Network Open, 1(5), e182716. https://doi.org/10.1001/jamanetworkopen.2018.2716

Cantor, M. N., & Thorpe, L. (2018). Integrating Data On Social Determinants Of Health Into Electronic Health Records. Health Affairs, 37(4), 585–590. https://doi.org/10.1377/hlthaff.2017.1252

Centers for Medicare & Medicaid Services. (n.d.). Value-based programs. Retrieved from https://www.cms.gov/medicare/quality/value-based-programs

Centers for Medicare & Medicaid Services. (2022). The CMS Framework for Health Equity 2022–2032. Retrieved from https://www.cms.gov/files/document/cms-framework-health-equity.pdf

Chen, M., Tan, X., & Padman, R. (2020). Social determinants of health in electronic health records and their impact on analysis and risk prediction: A systematic review. Journal of the American Medical Informatics Association, 27(11), 1764–1773. https://doi.org/10.1093/jamia/ocaa143

Cutler, D., & Lleras-Muney, A. (2006). Education and Health: Evaluating Theories and Evidence. National Bureau of Economic Research. https://doi.org/10.3386/w12352

Ghassemi, M., Celi, L. A., Stone, D. J., & Pollard, T. J. (2023). Integrating social determinants of health into critical care outcome prediction: Challenges and opportunities. ACM Transactions on Computing for Healthcare, 4(3), 1-27. https://doi.org/10.1145/3600211.3604719

Johnson, A., Bulgarelli, L., Pollard, T., Celi, L. A., Mark, R., & Horng, S. (2023). MIMIC-IV (version 3.1). PhysioNet. https://physionet.org/content/mimiciv/3.1/

Johnson, A., Bulgarelli, L., Pollard, T., Celi, L. A., Mark, R., & Horng, S. (2023). MIMIC-IV-ED (version 2.2). PhysioNet. https://physionet.org/content/mimic-iv-ed/2.2/

Pollard, T., Johnson, A., Raffa, J., Celi, L. A., Mark, R., & Badawi, O. (2018). eICU Collaborative Research Database (version 2.0). PhysioNet. https://physionet.org/content/eicu-crd/2.0/

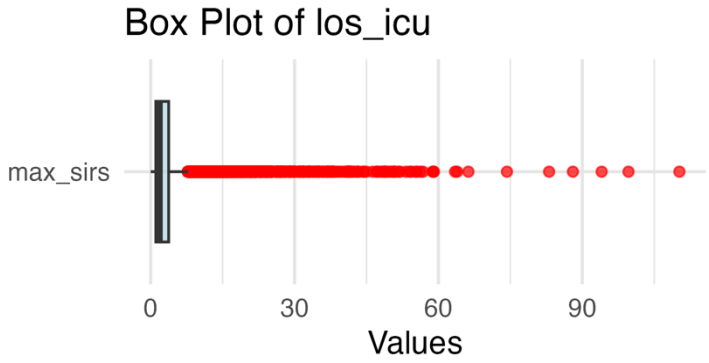
Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online], 101 (23), e215–e220.

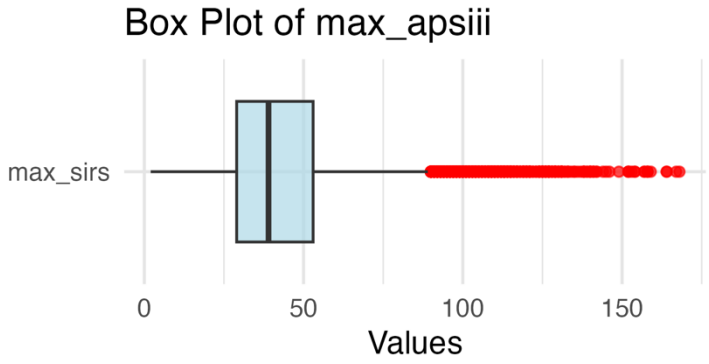
IPUMS USA. (n.d.). IPUMS USA: Version 13.0 [Database]. IPUMS. Retrieved from https://usa.ipums.org/usa-action/variables/

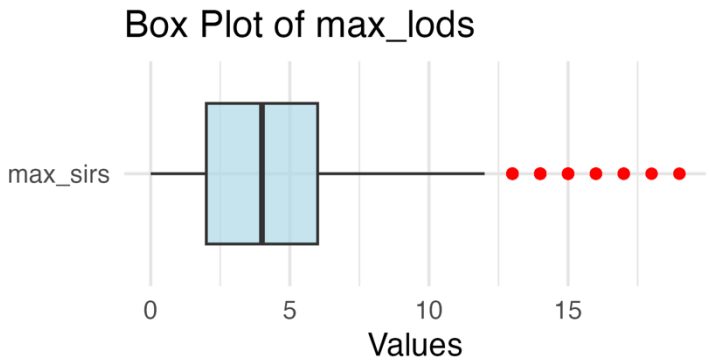
Centers for Medicare & Medicaid Services. (2019, November 6). Hospital Readmissions Reduction Program (HRRP) Archives. Retrieved from https://www.hhs.gov/guidance/document/hospital-readmissions-reduction-program-archives-0

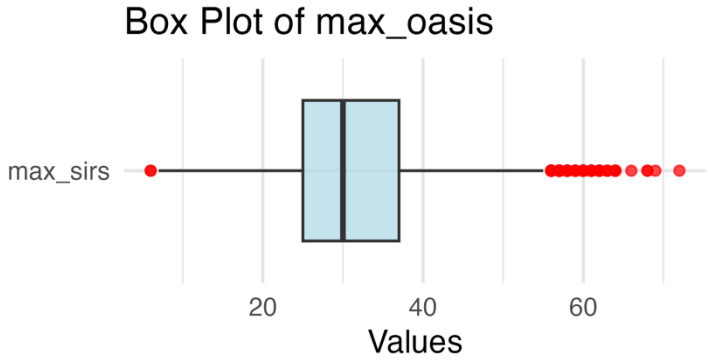
APPENDIX

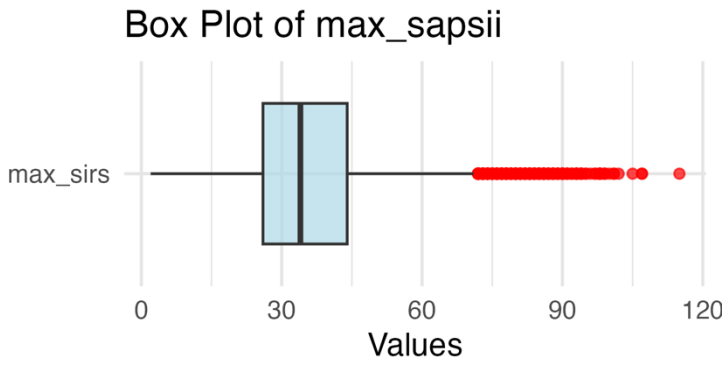
Section 1: Supplemental Graphs and Tables

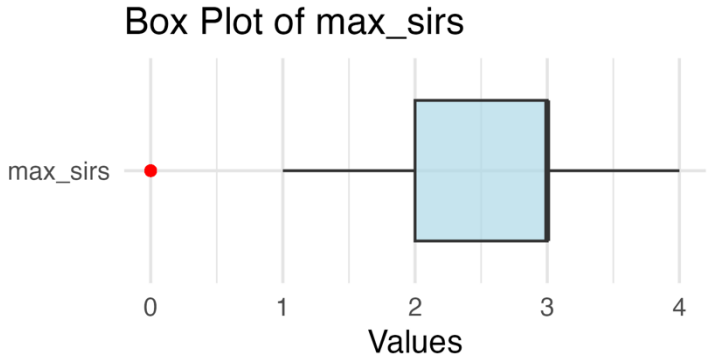


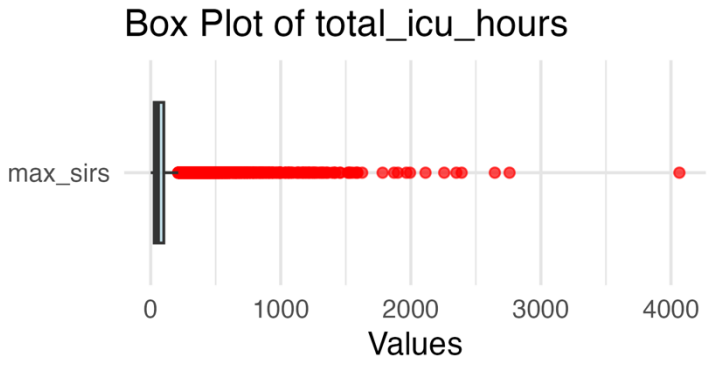


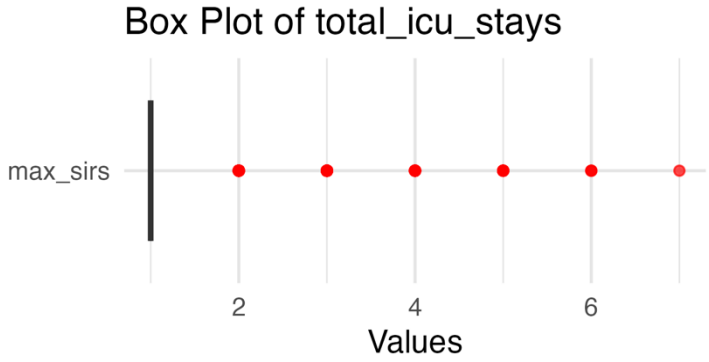


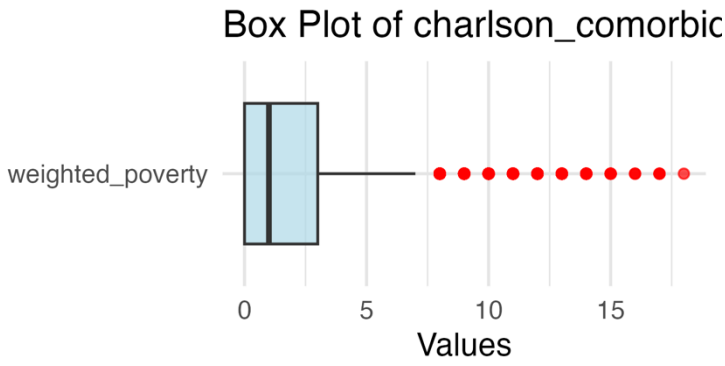


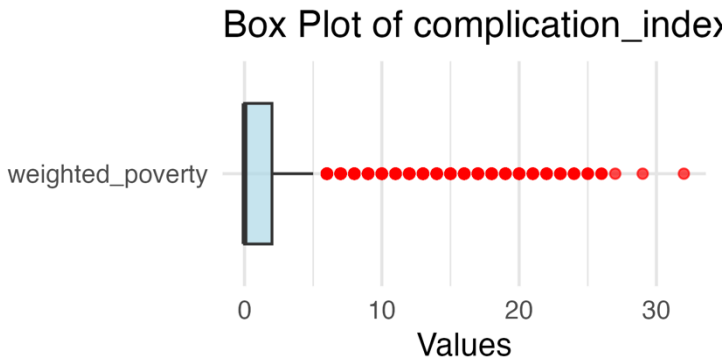

Box Plot of max_sapsii
max_sirs
0
30
60
90
120
Values

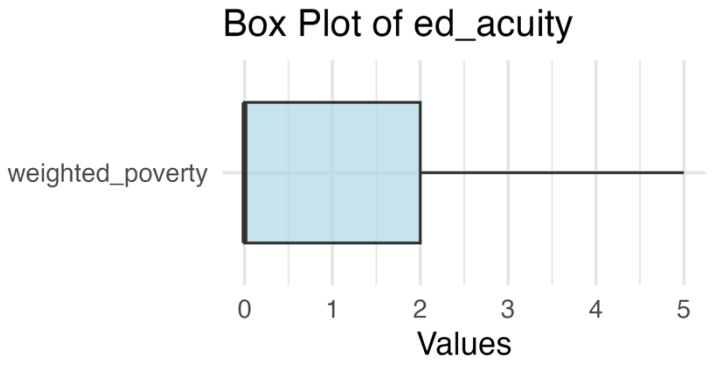

Box Plot of max_sirs
max_sirs
0
1
2
3
4
Values

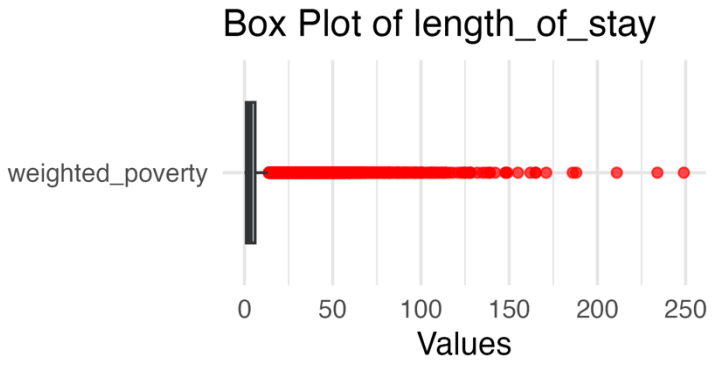

Box Plot of total_icu_hours
max_sirs
0
1000
2000
3000
4000
Values

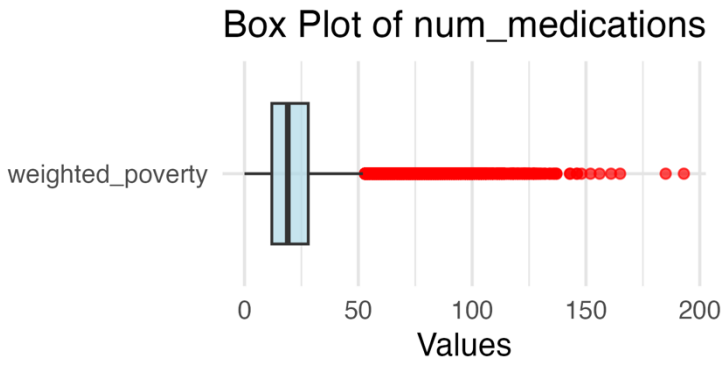

Box Plot of total_icu_stays
max_sirs
2
4
6
Values

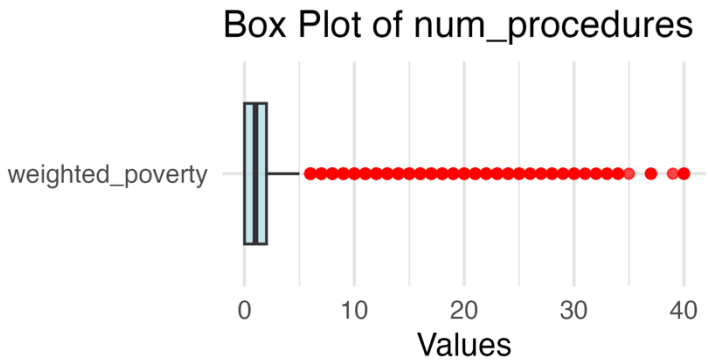

Box Plot of charlson_comorbic
weighted_poverty
0
5
10
15
Values

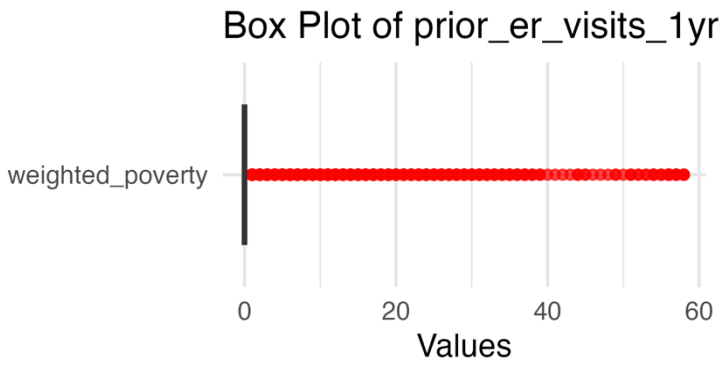

Box Plot of complication_index
weighted_poverty
0
10
20
30
Values

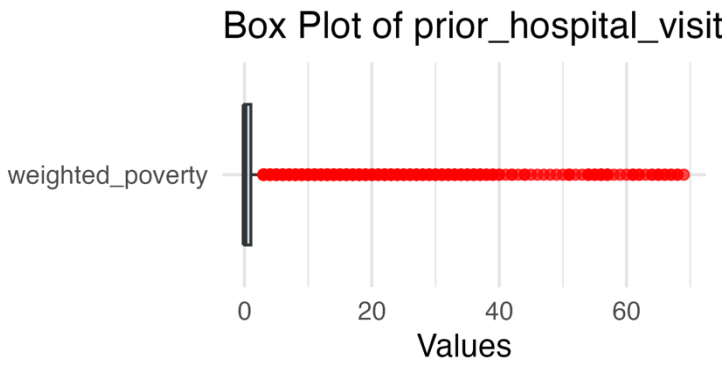

Box Plot of ed_acuity
weighted_poverty
0
1
2
3
4
5
Values

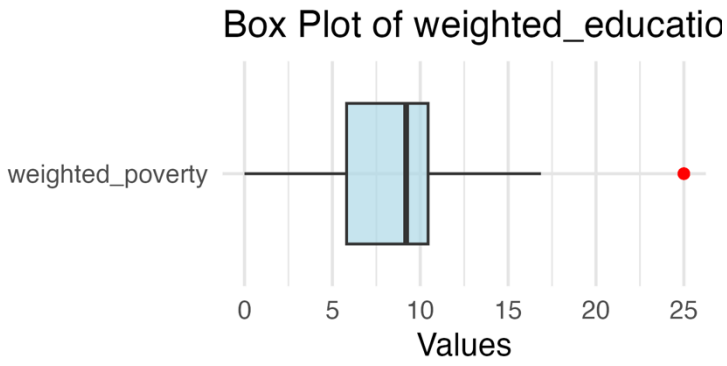

Box Plot of length_of_stay
weighted_poverty
0
50
100
150
200
250
Values

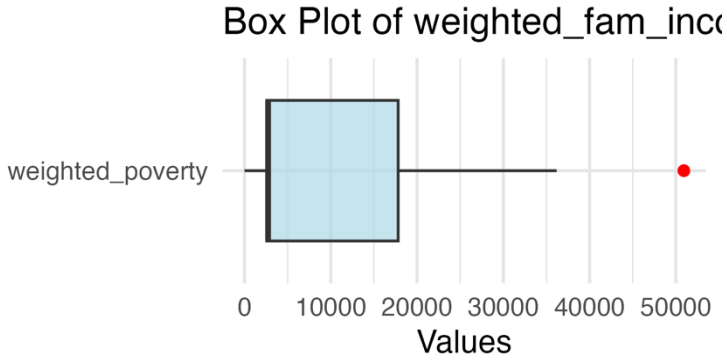

Box Plot of num_medications
weighted_poverty
0
50
100
150
200
Values

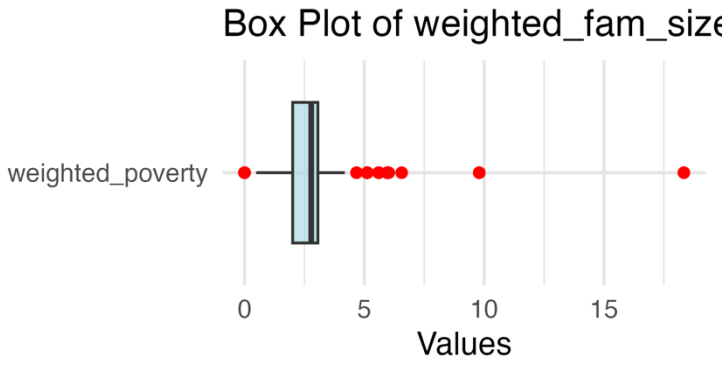

Box Plot of num_procedures
weighted_poverty
0
10
20
30
40
Values

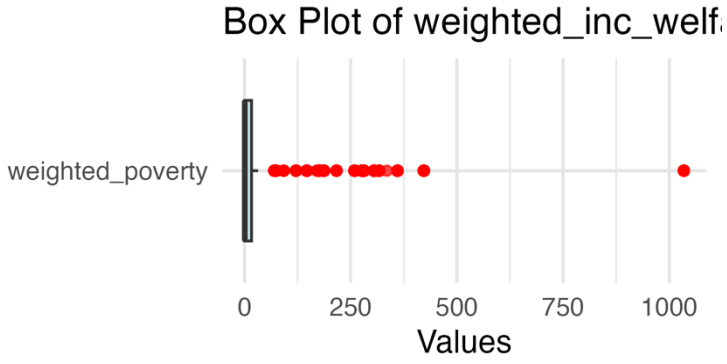

Box Plot of prior_er_visits_1yr
weighted_poverty
0
20
40
60
Values

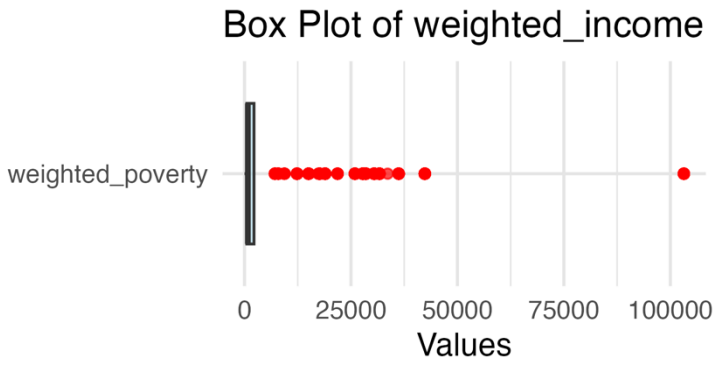

Box Plot of prior_hospital_visit
weighted_poverty
0
20
40
60
Values

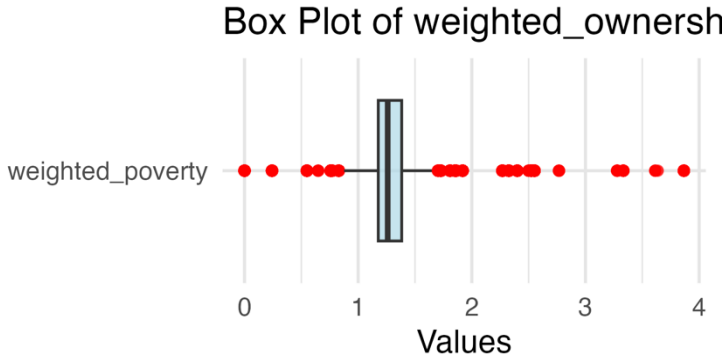

Box Plot of weighted_educatio
weighted_poverty
0
5
10
15
20
25
Values

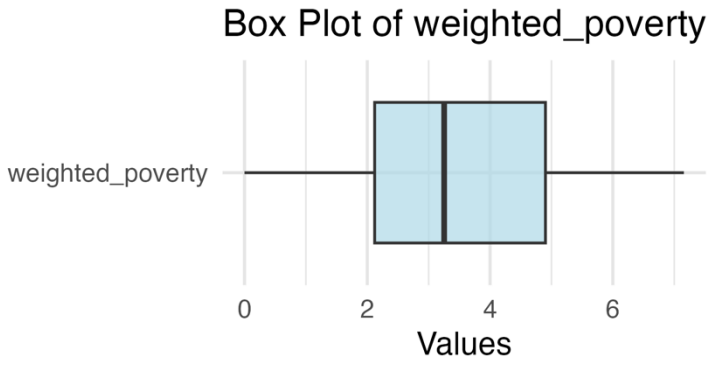

Box Plot of weighted_fam_inc
weighted_poverty
0
10000
20000
30000
40000
50000
Values

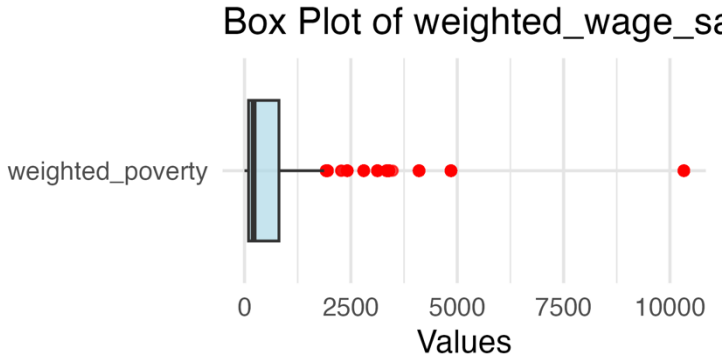

Box Plot of weighted_fam_size
weighted_poverty
0
5
10
15
Values

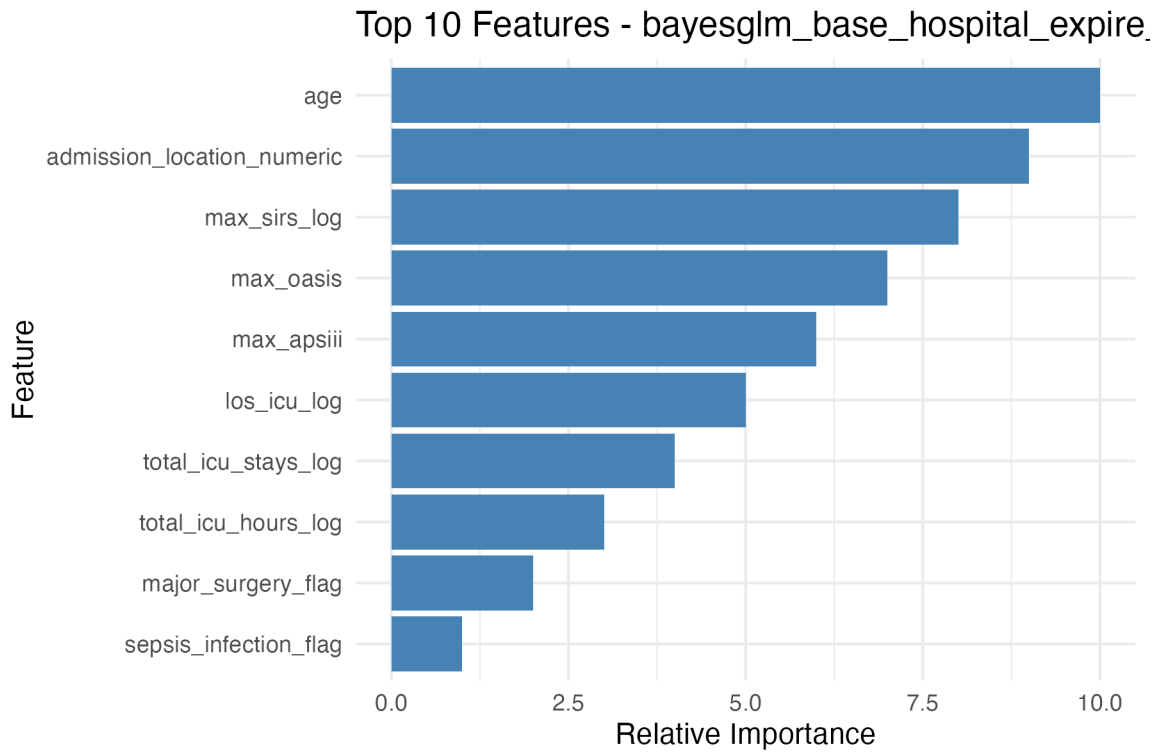

Box Plot of weighted_inc_welf
weighted_poverty
0
250
500
750
1000
Values

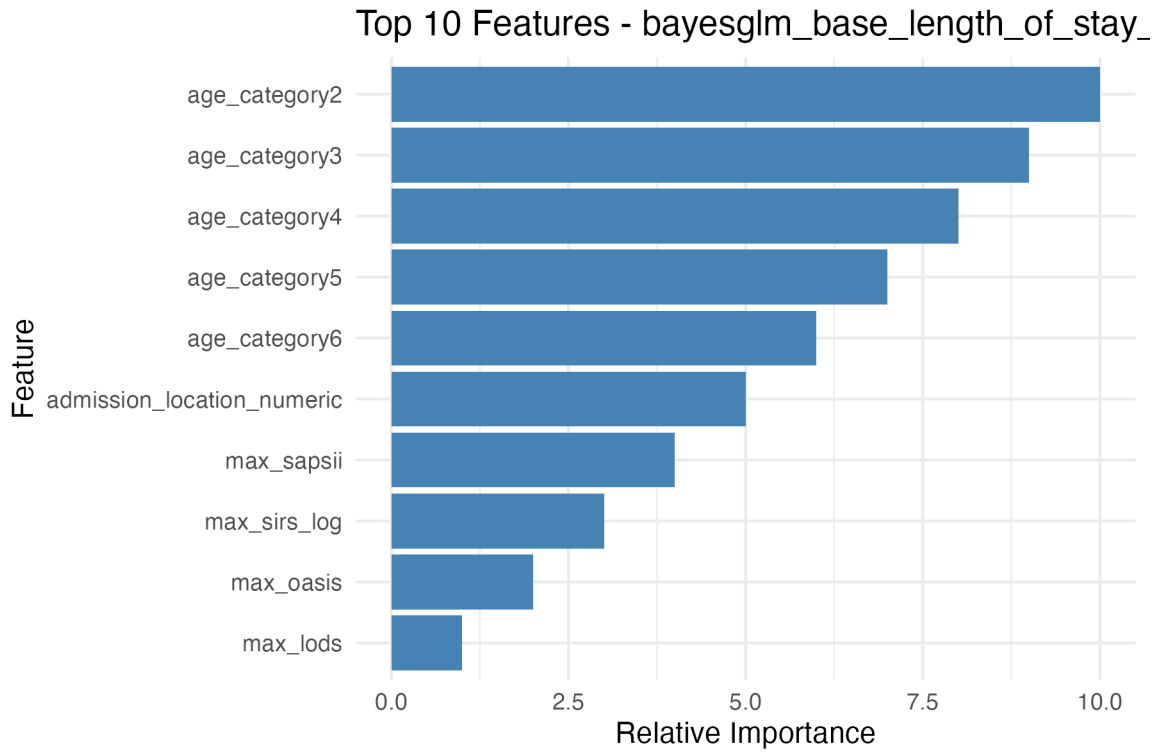

Box Plot of weighted_income
weighted_poverty
0
25000
50000
75000
100000
Values

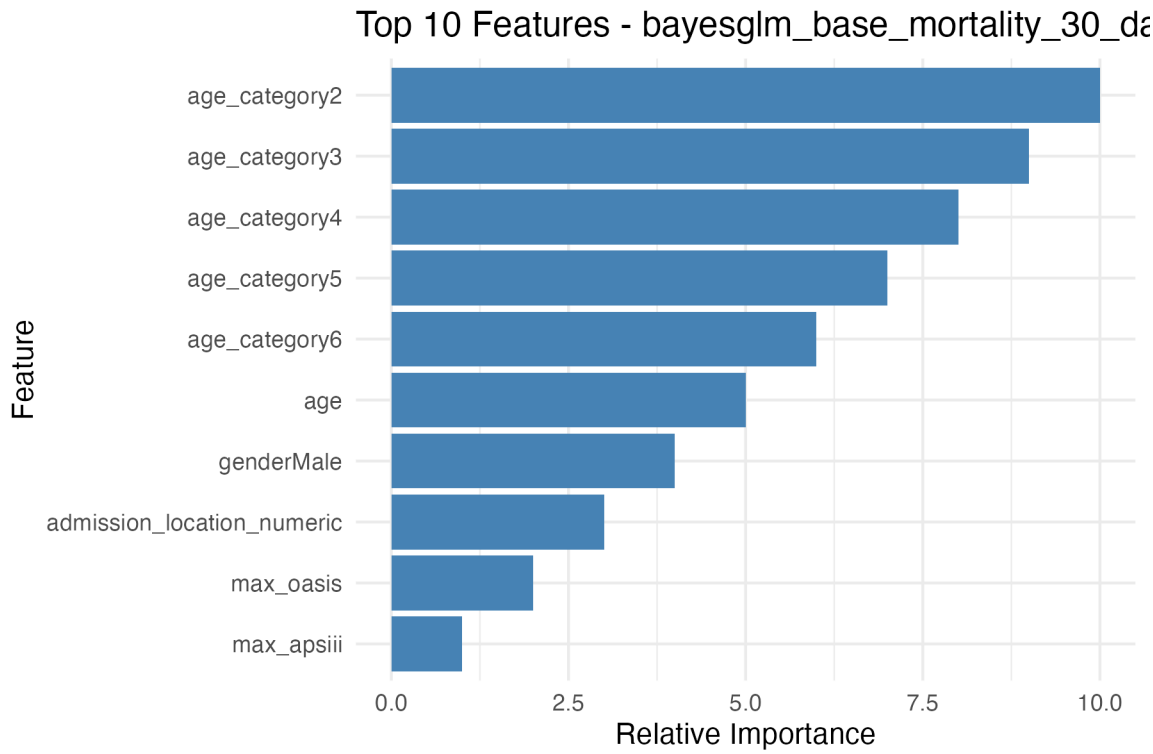

Box Plot of weighted_ownersh
weighted_poverty
0
1
2
3
4
Values

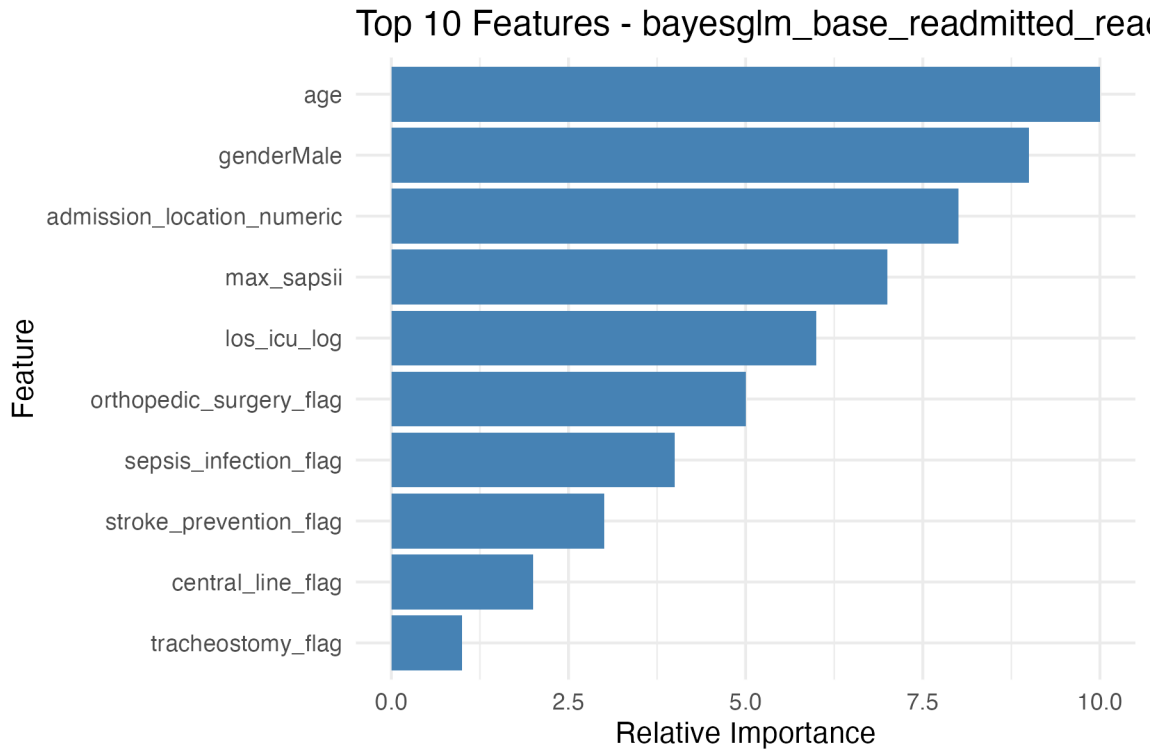

Box Plot of weighted_poverty
weighted_poverty
0
2
4
6
Values

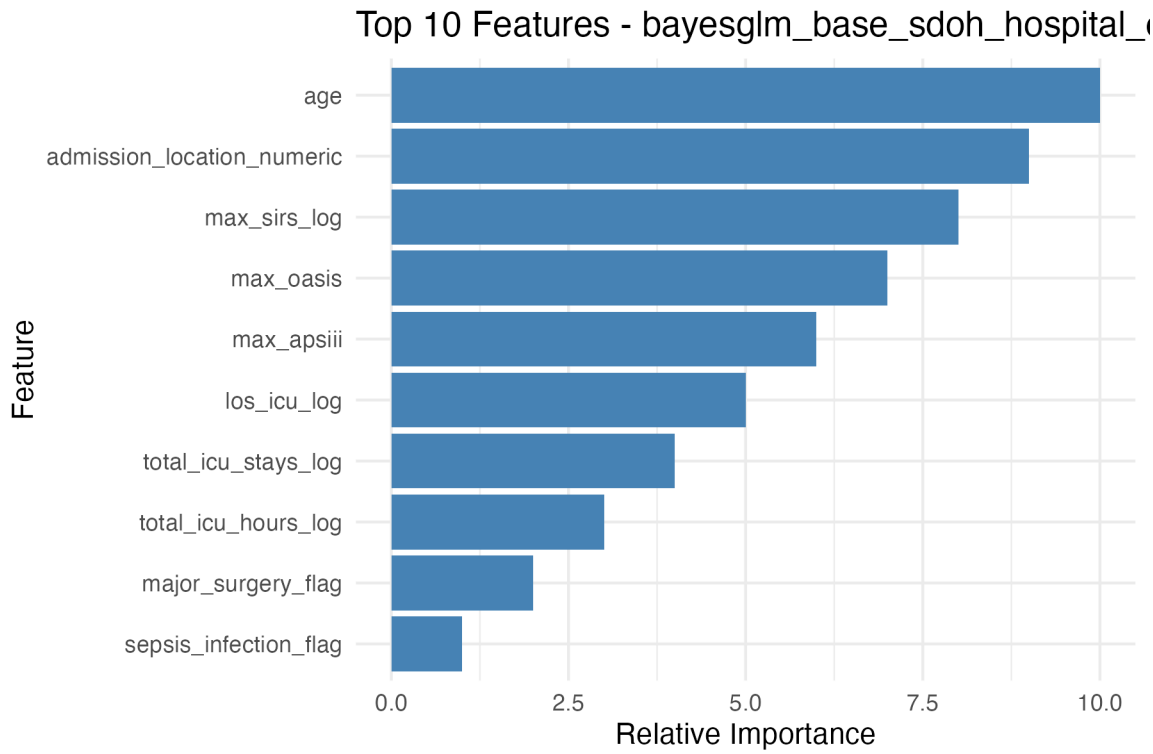

Box Plot of weighted_wage_sa
weighted_poverty
0
2500
5000
7500
10000
Values

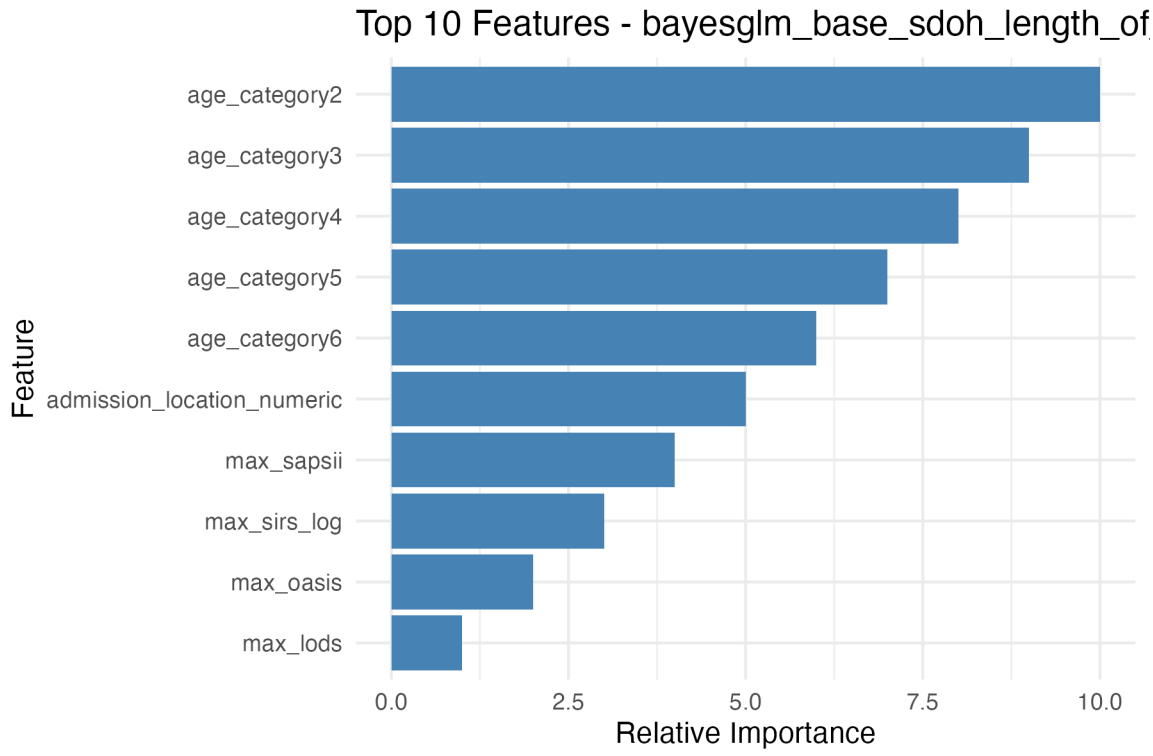

Top 10 Features - bayesglm_base_hospital_expire_
Feature
age
admission_location_numeric
max_sirs_log
max_oasis
max_apsiii
los_icu_log
total_icu_stays_log
total_icu_hours_log
major_surgery_flag
sepsis_infection_flag
0.0
2.5
5.0
7.5
10.0
Relative Importance

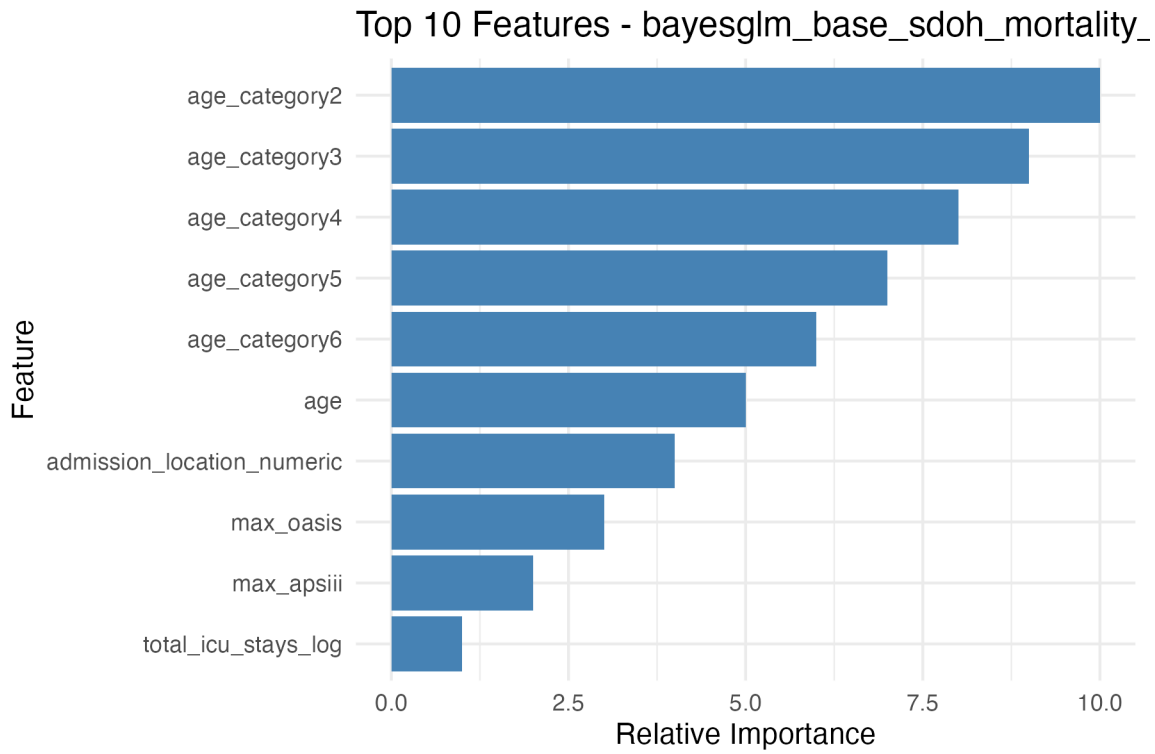

Top 10 Features - bayesglm_base_length_of_stay_
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
admission_location_numeric
max_sapsii
max_sirs_log
max_oasis
max_lods
0.0
2.5
5.0
7.5
10.0
Relative Importance

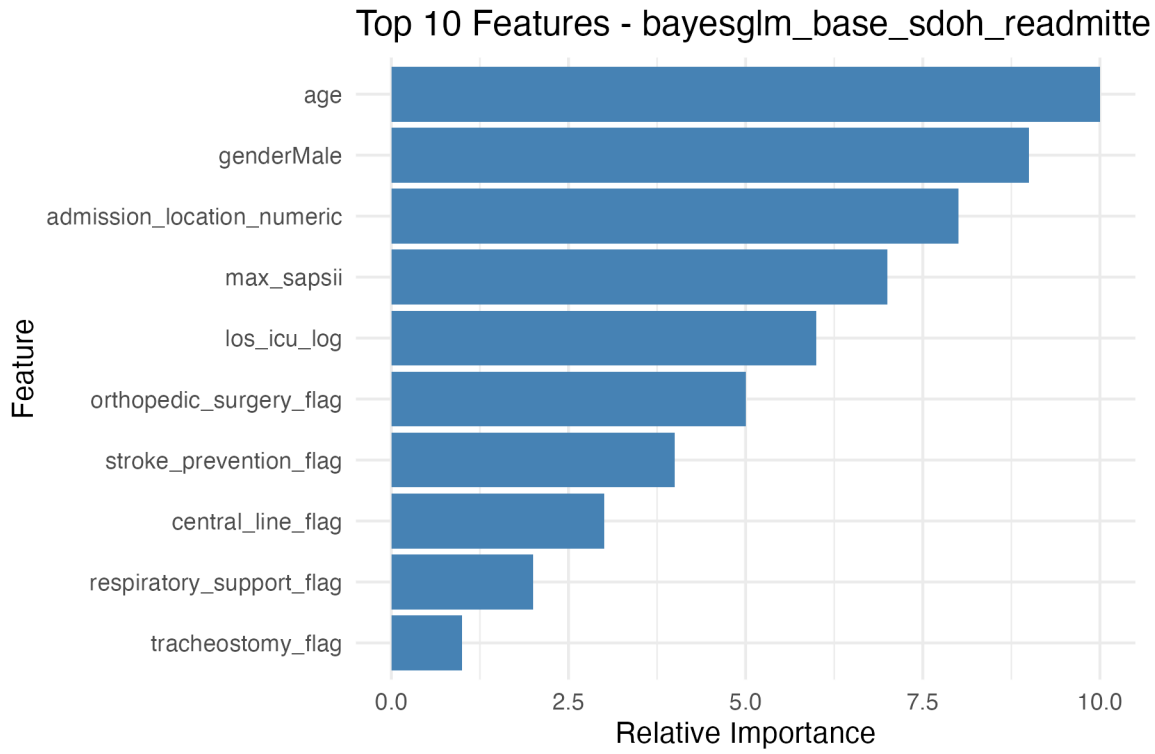

Top 10 Features - bayesglm_base_mortality_30_da
age_category2
age_category3
age_category4
age_category5
age_category6
age
genderMale
admission_location_numeric
max_oasis
max_apsiii
Feature
0.0
2.5
5.0
7.5
10.0
Relative Importance

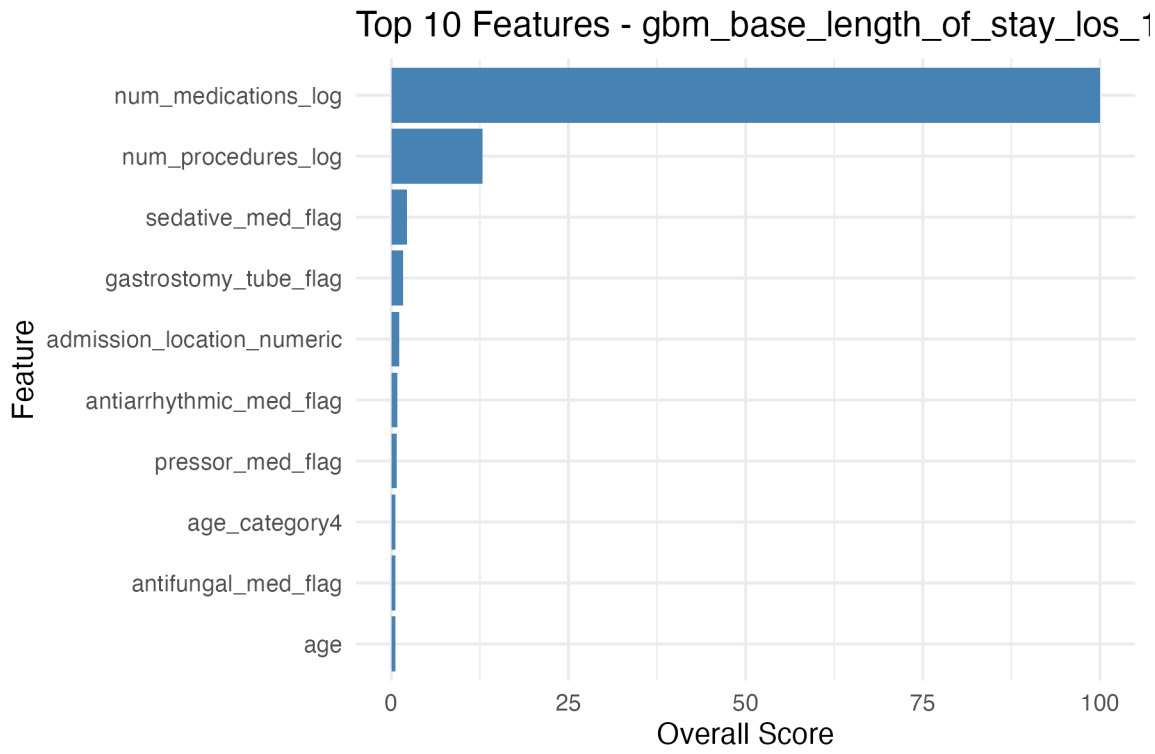

Top 10 Features - bayesglm_base_readmitted_rea
age
genderMale
admission_location_numeric
max_sapsii
los_icu_log
orthopedic_surgery_flag
sepsis_infection_flag
stroke_prevention_flag
central_line_flag
tracheostomy_flag
Feature
0.0
2.5
5.0
7.5
10.0
Relative Importance

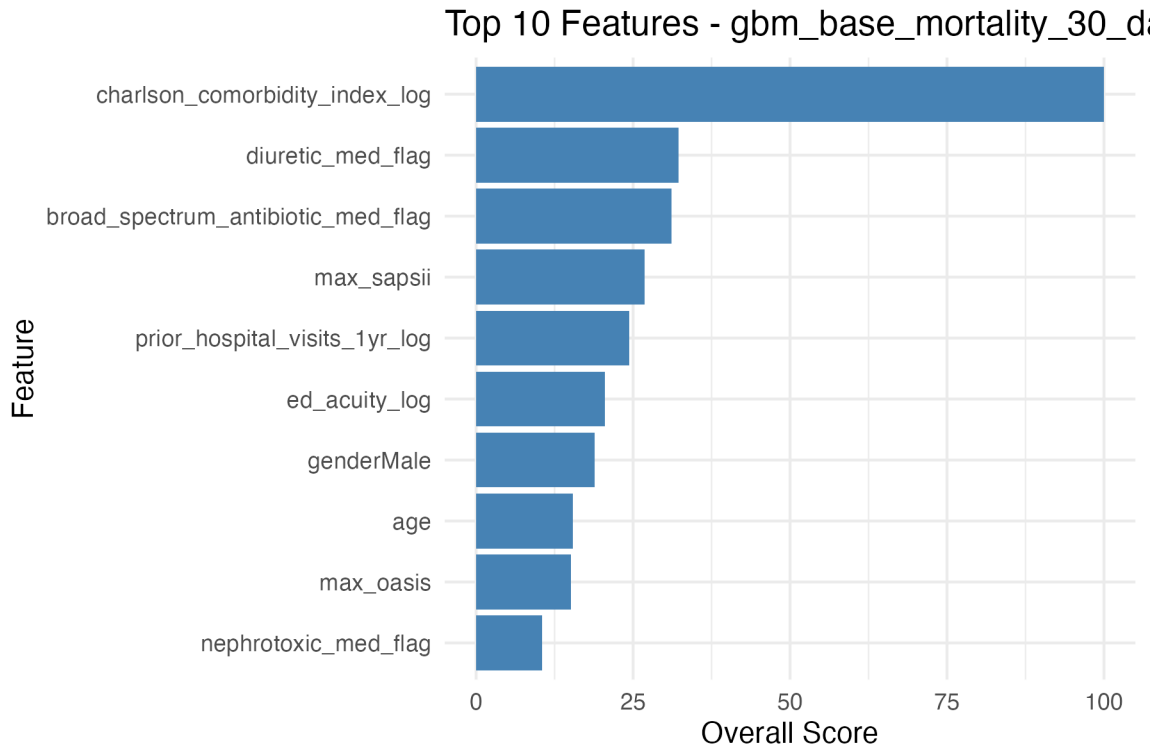

Top 10 Features - bayesglm_base_sdoh_hospital_
Feature
age
admission_location_numeric
max_sirs_log
max_oasis
max_apsiii
los_icu_log
total_icu_stays_log
total_icu_hours_log
major_surgery_flag
sepsis_infection_flag
0.0
2.5
5.0
7.5
10.0
Relative Importance

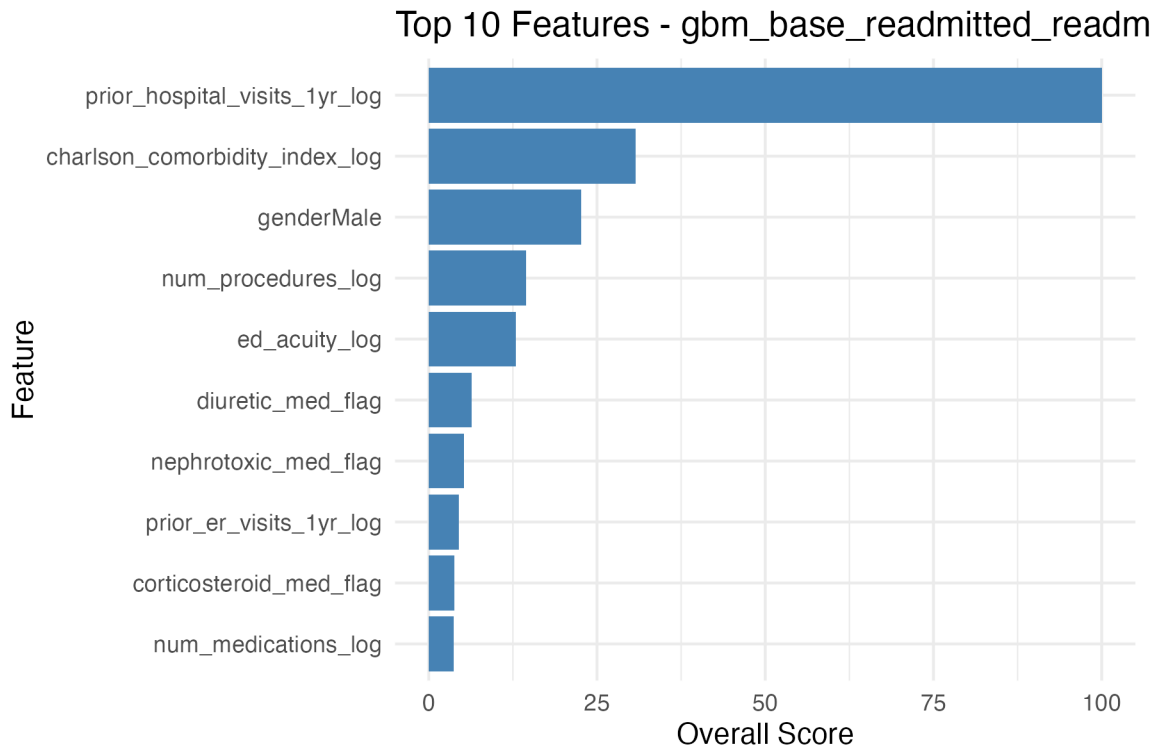

Top 10 Features - bayesglm_base_sdoh_length_of
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
admission_location_numeric
max_sapsii
max_sirs_log
max_oasis
max_lods
0.0
2.5
5.0
7.5
10.0
Relative Importance

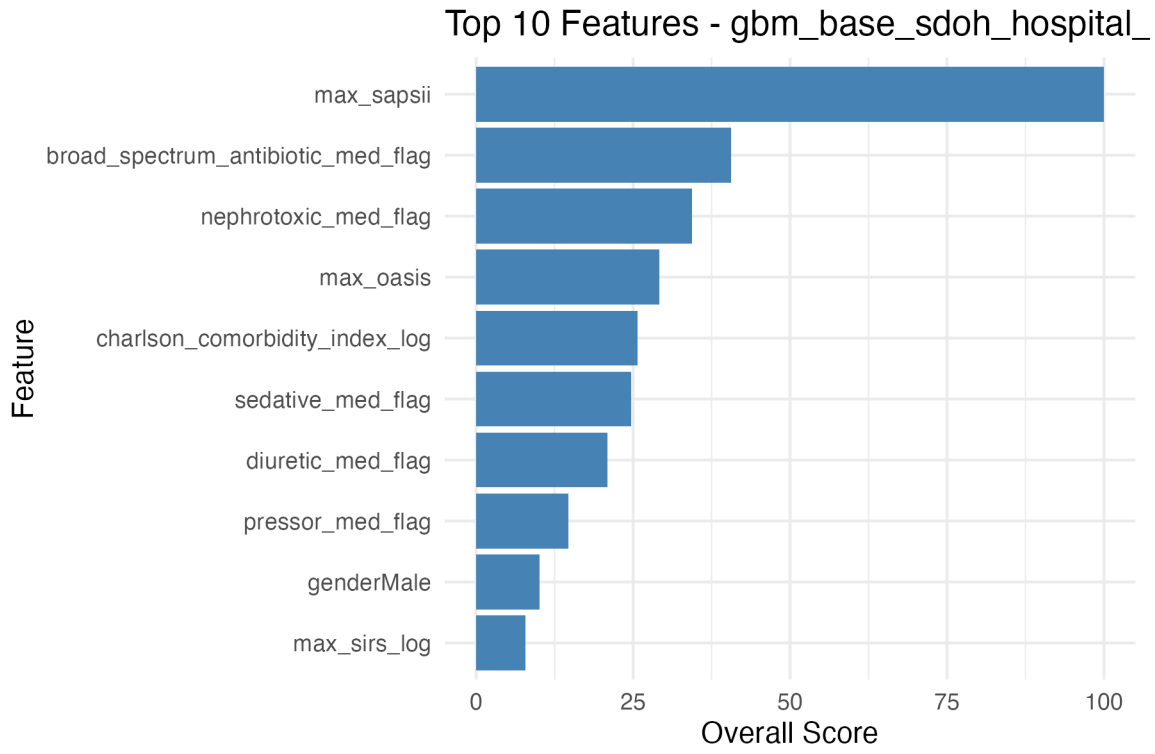

Top 10 Features - bayesglm_base_sdoh_mortality_
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
age
admission_location_numeric
max_oasis
max_apsiii
total_icu_stays_log
0.0
2.5
5.0
7.5
10.0
Relative Importance

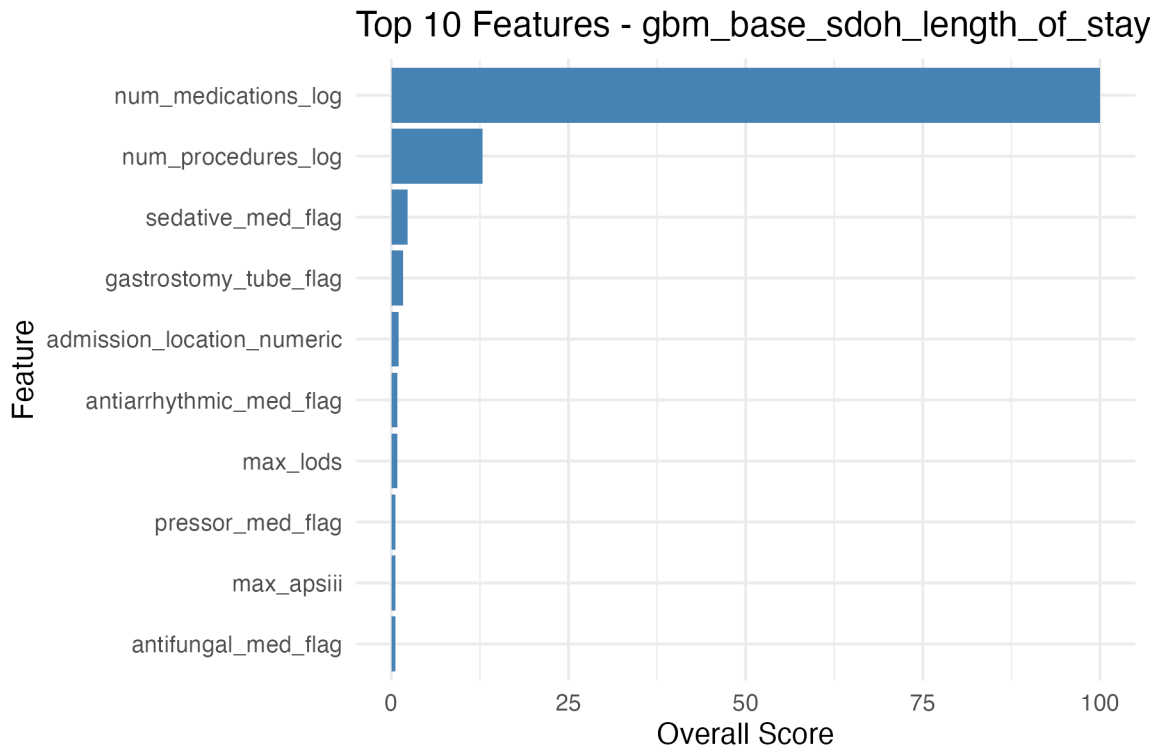

Top 10 Features - bayesglm_base_sdoh_readmitte
Feature
age
genderMale
admission_location_numeric
max_sapsii
los_icu_log
orthopedic_surgery_flag
stroke_prevention_flag
central_line_flag
respiratory_support_flag
tracheostomy_flag
0.0
2.5
5.0
7.5
10.0
Relative Importance

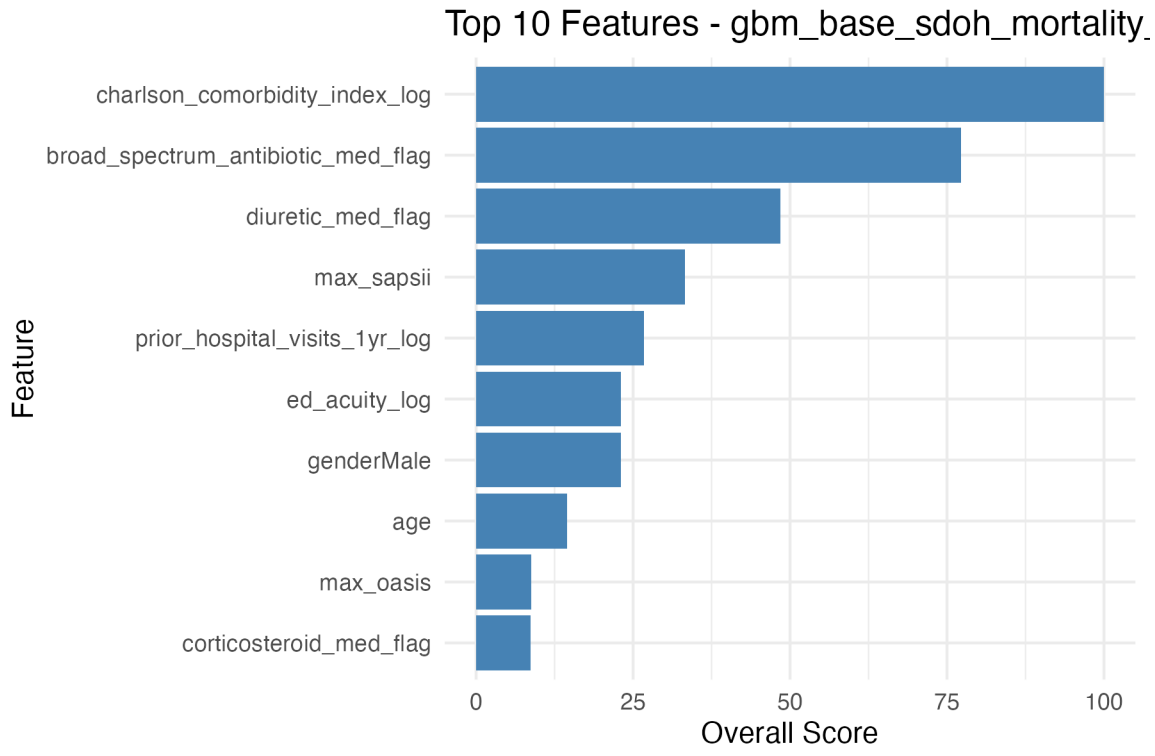

Top 10 Features - gbm_base_length_of_stay_los_1
num_medications_log
num_procedures_log
sedative_med_flag
gastrostomy_tube_flag
admission_location_numeric
antiarrhythmic_med_flag
pressor_med_flag
age_category4
antifungal_med_flag
age
Feature
0
25
50
75
100
Overall Score

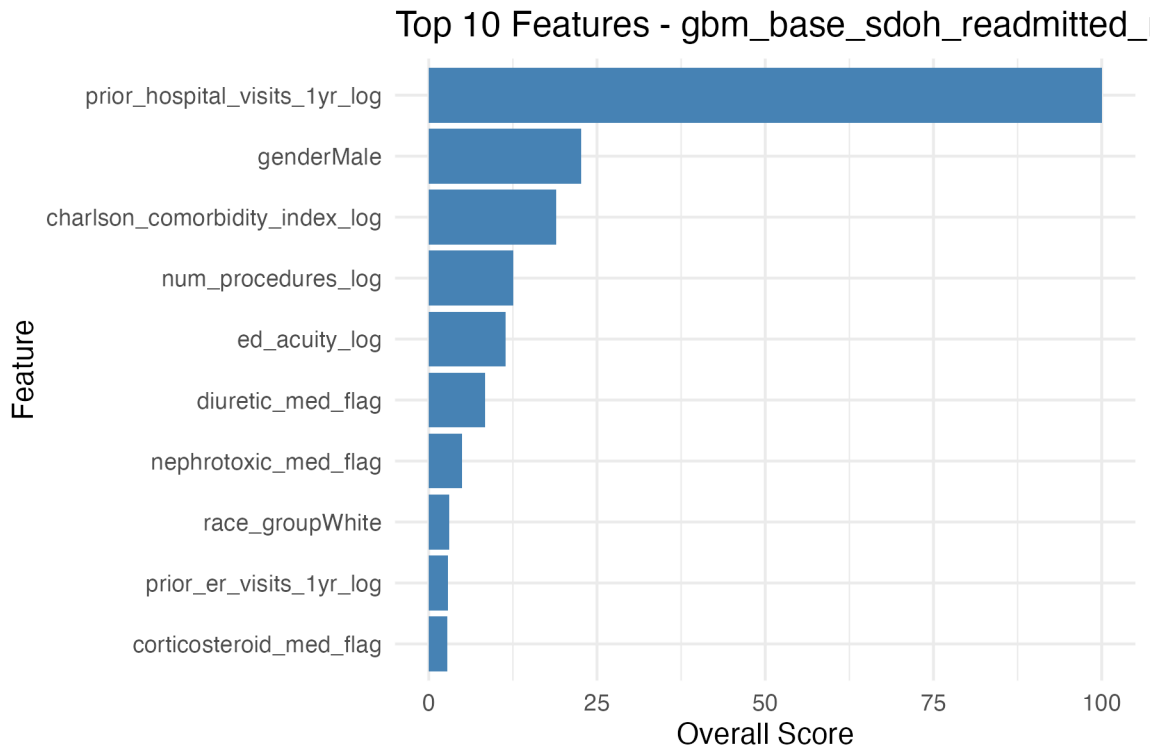

Top 10 Features - gbm_base_mortality_30_d
charlson_comorbidity_index_log
diuretic_med_flag
broad_spectrum_antibiotic_med_flag
max_sapsii
prior_hospital_visits_1yr_log
ed_acuity_log
genderMale
age
max_oasis
nephrotoxic_med_flag
Feature
0
25
50
75
100
Overall Score

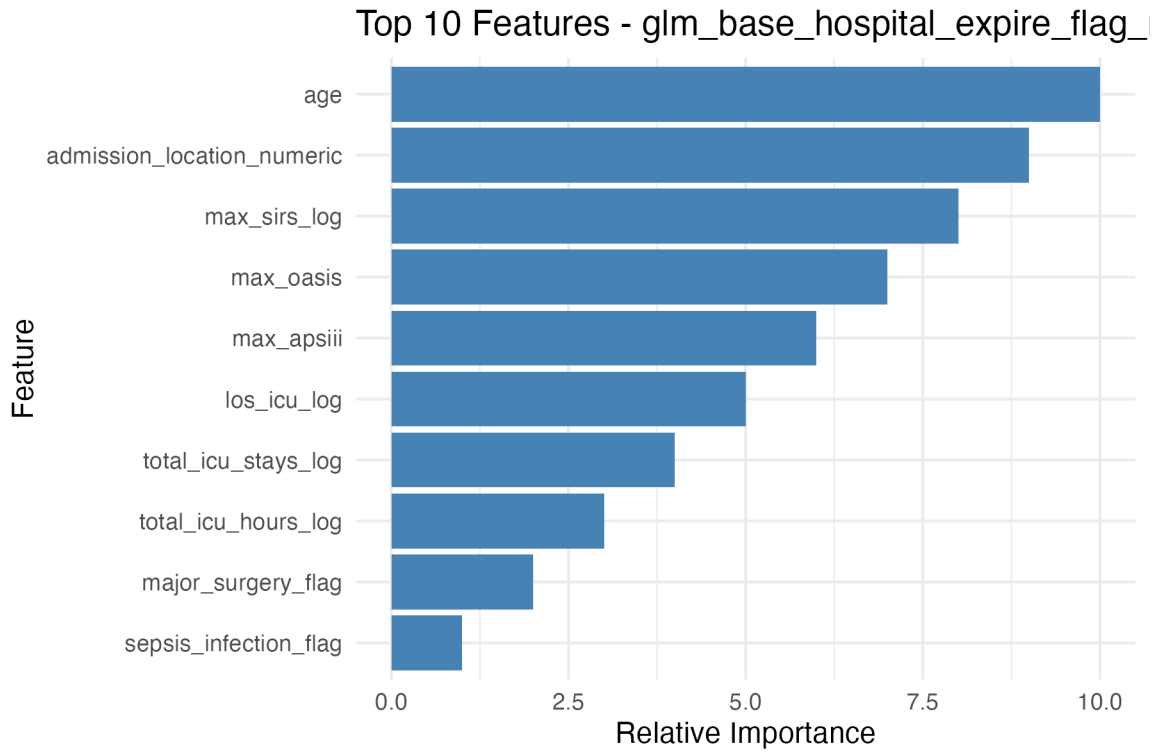

Top 10 Features - gbm_base_readmitted_readm
prior_hospital_visits_1yr_log
charlson_comorbidity_index_log
genderMale
num_procedures_log
ed_acuity_log
diuretic_med_flag
nephrotoxic_med_flag
prior_er_visits_1yr_log
corticosteroid_med_flag
num_medications_log
Feature
0
25
50
75
100
Overall Score

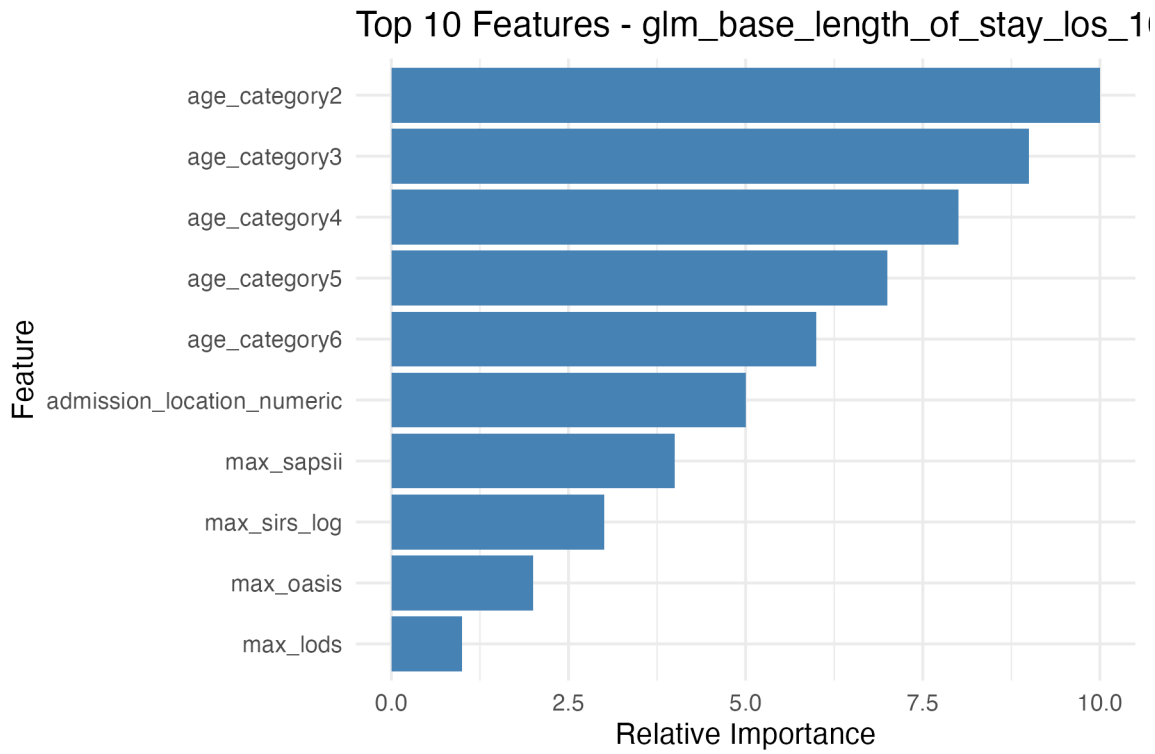

Top 10 Features - gbm_base_sdoh_hospital_
max_sapsii
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
max_oasis
charlson_comorbidity_index_log
sedative_med_flag
diuretic_med_flag
pressor_med_flag
genderMale
max_sirs_log
Feature
0
25
50
75
100
Overall Score

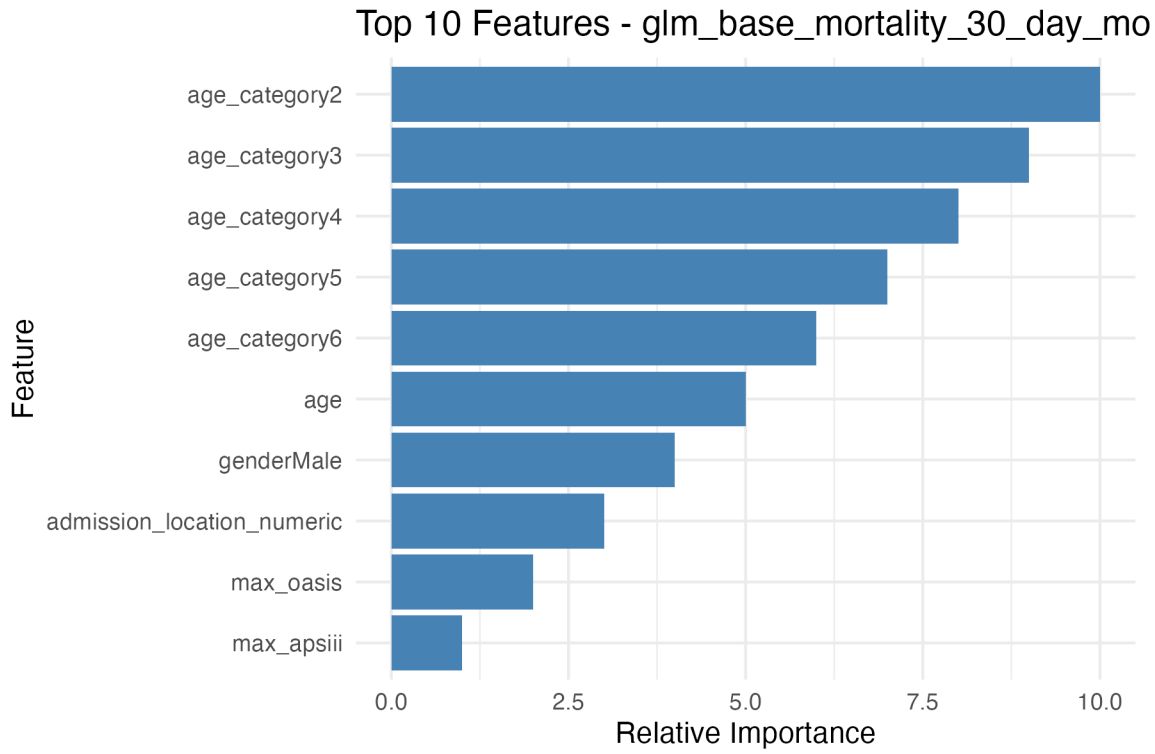

Top 10 Features - gbm_base_sdoh_length_of_stay
num_medications_log
num_procedures_log
sedative_med_flag
gastrostomy_tube_flag
admission_location_numeric
antiarrhythmic_med_flag
max_lods
pressor_med_flag
max_apsiii
antifungal_med_flag
Feature
0
25
50
75
100
Overall Score

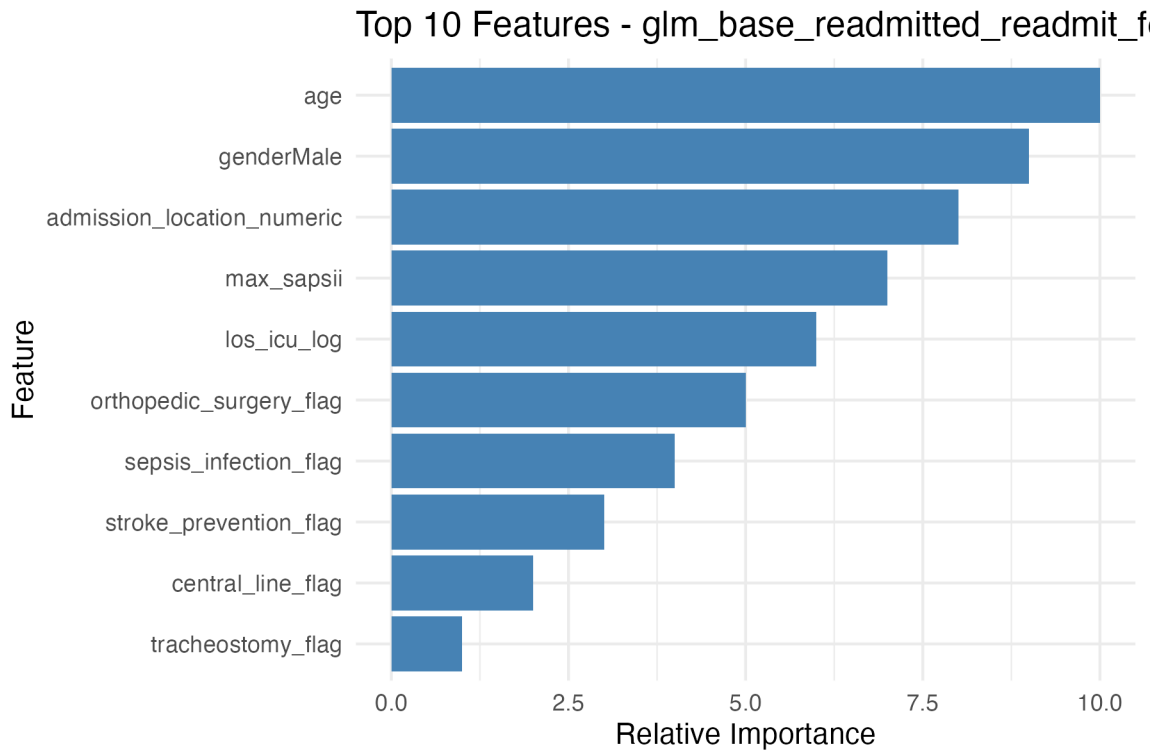

Top 10 Features - gbm_base_sdoh_mortality_
charlson_comorbidity_index_log
broad_spectrum_antibiotic_med_flag
diuretic_med_flag
max_sapsii
prior_hospital_visits_1yr_log
ed_acuity_log
genderMale
age
max_oasis
corticosteroid_med_flag
Feature
0
25
50
75
100
Overall Score

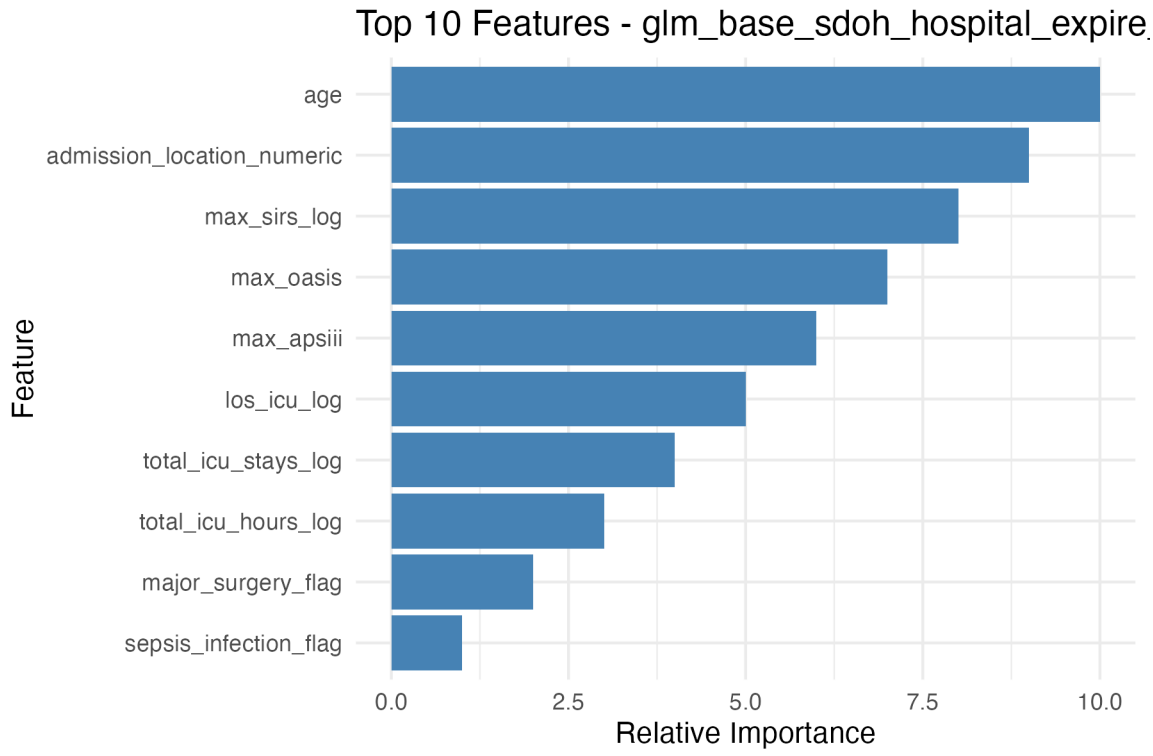

Top 10 Features - gbm_base_sdoh_readmitted_
prior_hospital_visits_1yr_log
genderMale
charlson_comorbidity_index_log
num_procedures_log
ed_acuity_log
diuretic_med_flag
nephrotoxic_med_flag
race_groupWhite
prior_er_visits_1yr_log
corticosteroid_med_flag
Feature
0
25
50
75
100
Overall Score

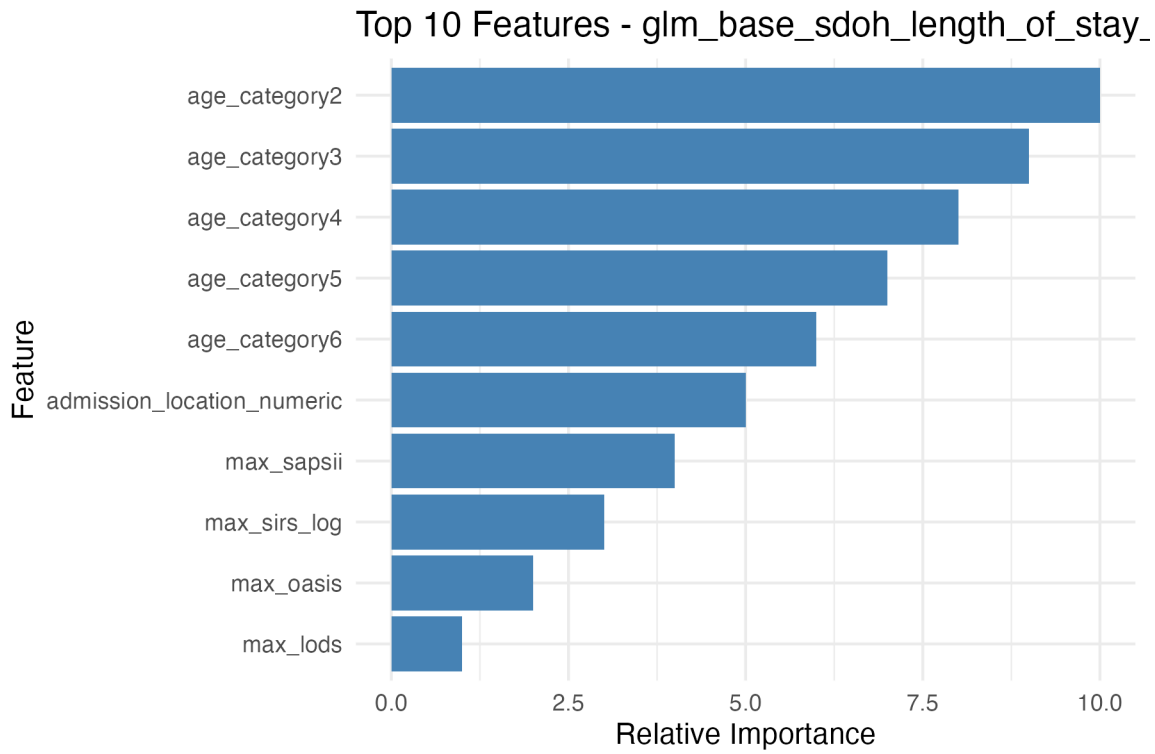

Top 10 Features - glm_base_hospital_expire_flag_
age
admission_location_numeric
max_sirs_log
max_oasis
max_apsiii
los_icu_log
total_icu_stays_log
total_icu_hours_log
major_surgery_flag
sepsis_infection_flag
Feature
0.0
2.5
5.0
7.5
10.0
Relative Importance

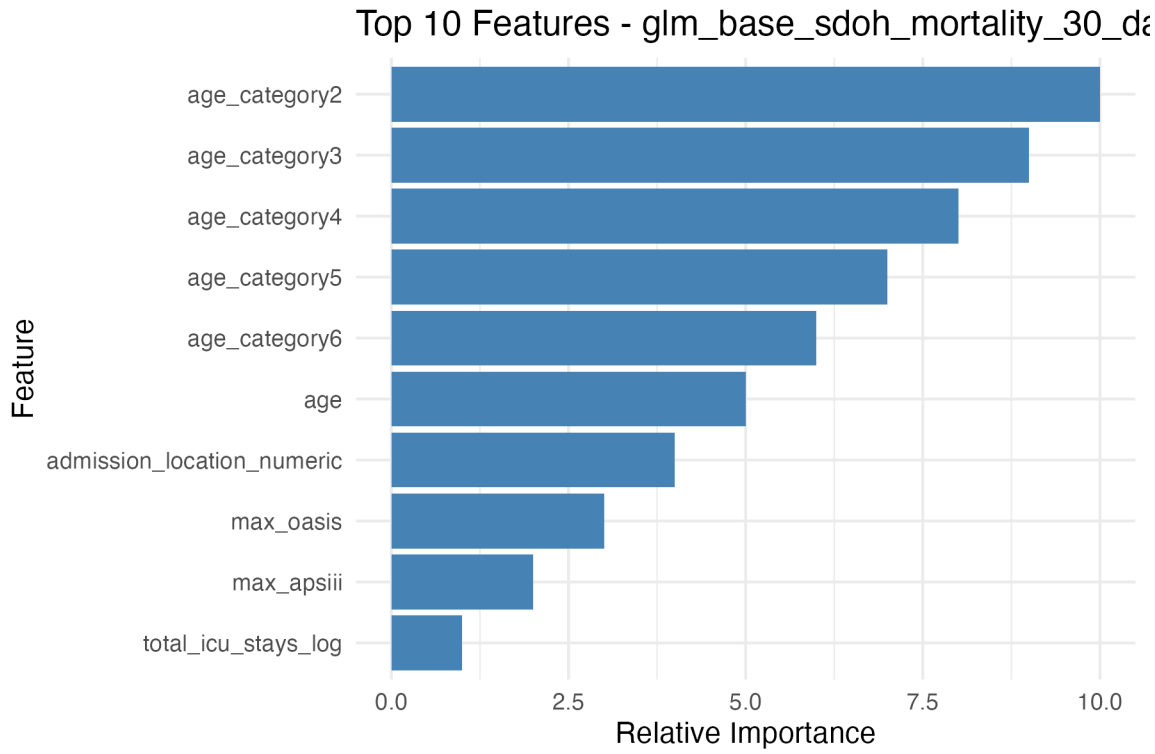

Top 10 Features - glm_base_length_of_stay_los_1
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
admission_location_numeric
max_sapsii
max_sirs_log
max_oasis
max_lods
0.0
2.5
5.0
7.5
10.0
Relative Importance

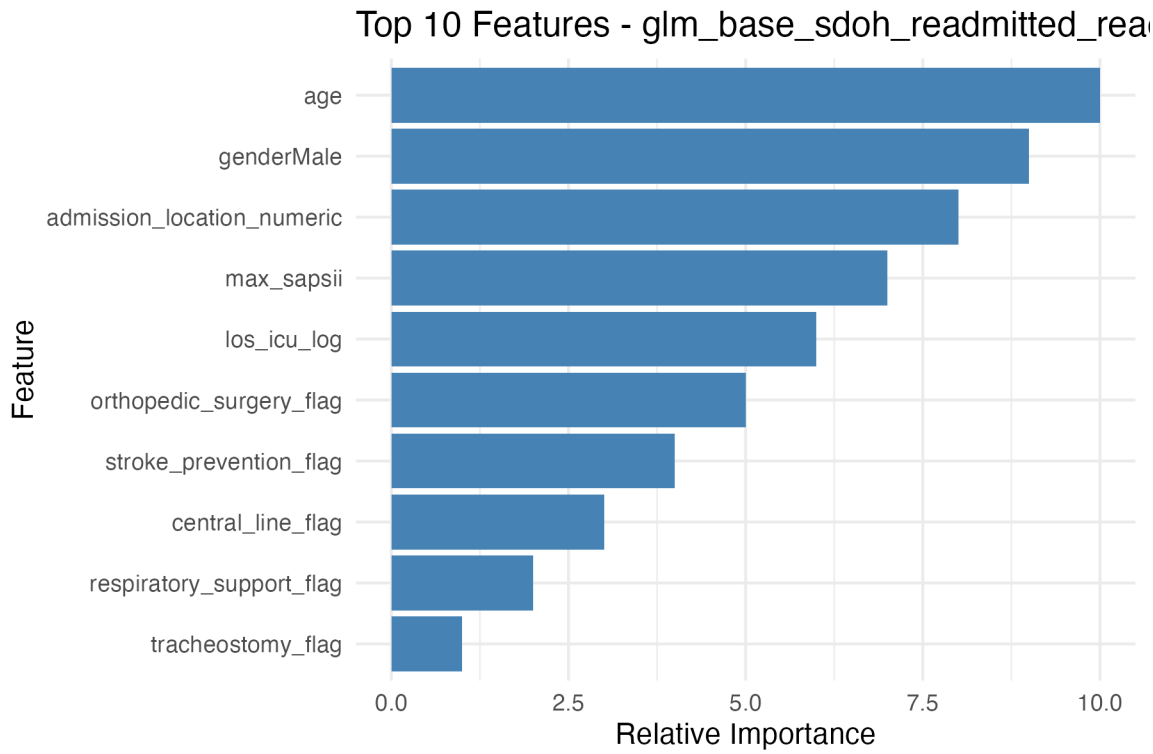

Top 10 Features - glm_base_mortality_30_day_mo
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
age
genderMale
admission_location_numeric
max_oasis
max_apsiii
0.0
2.5
5.0
7.5
10.0
Relative Importance

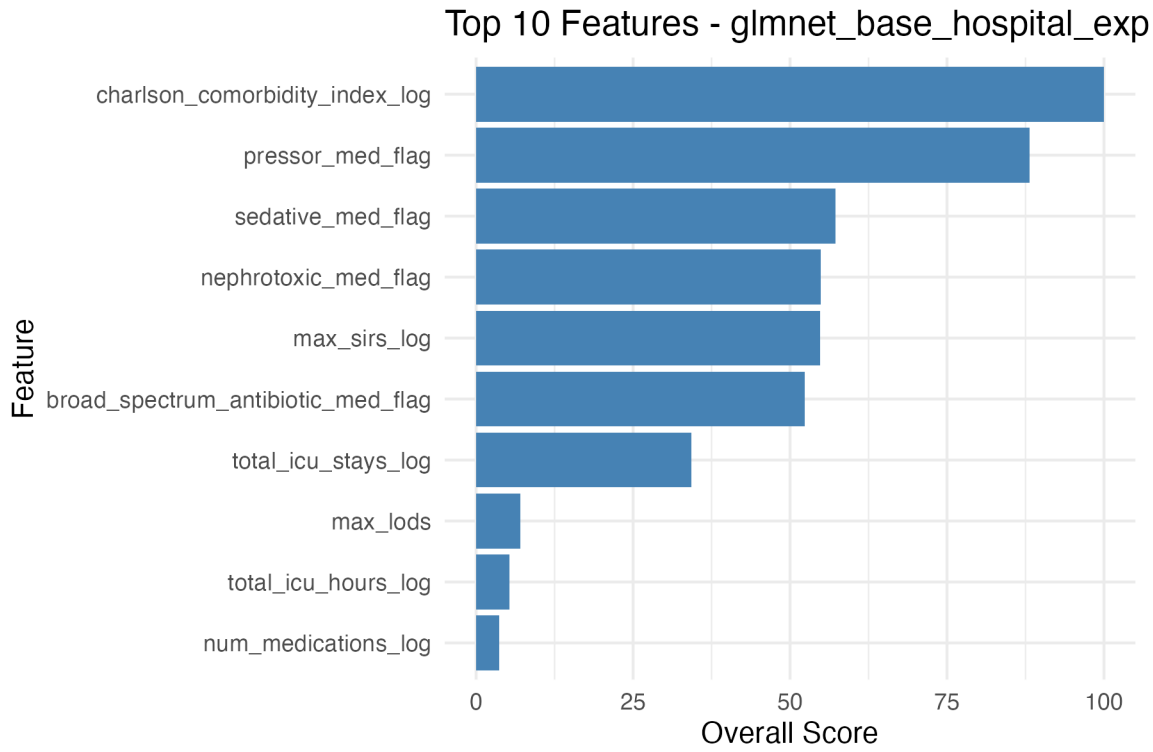

Top 10 Features - glm_base_readmitted_readmit_f
Feature
age
genderMale
admission_location_numeric
max_sapsii
los_icu_log
orthopedic_surgery_flag
sepsis_infection_flag
stroke_prevention_flag
central_line_flag
tracheostomy_flag
0.0
2.5
5.0
7.5
10.0
Relative Importance

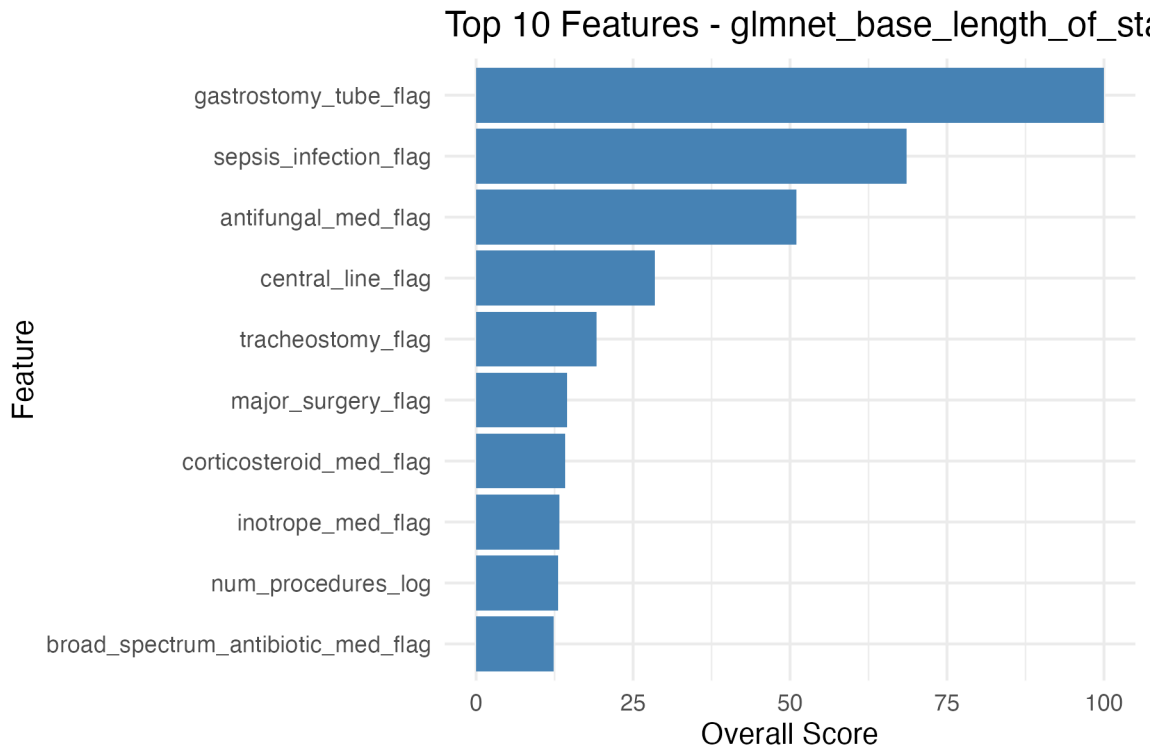

Top 10 Features - glm_base_sdoh_hospital_expire
Feature
age
admission_location_numeric
max_sirs_log
max_oasis
max_apsiii
los_icu_log
total_icu_stays_log
total_icu_hours_log
major_surgery_flag
sepsis_infection_flag
0.0
2.5
5.0
7.5
10.0
Relative Importance

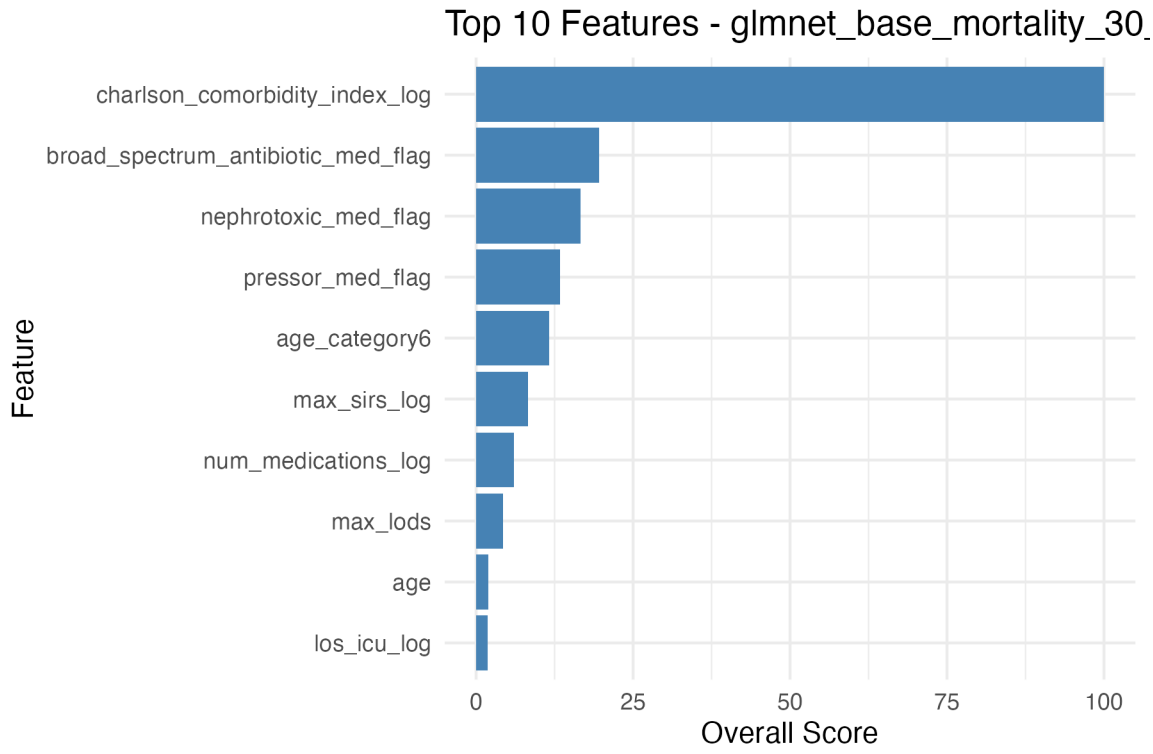

Top 10 Features - glm_base_sdoh_length_of_stay_
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
admission_location_numeric
max_sapsii
max_sirs_log
max_oasis
max_lods
0.0
2.5
5.0
7.5
10.0
Relative Importance

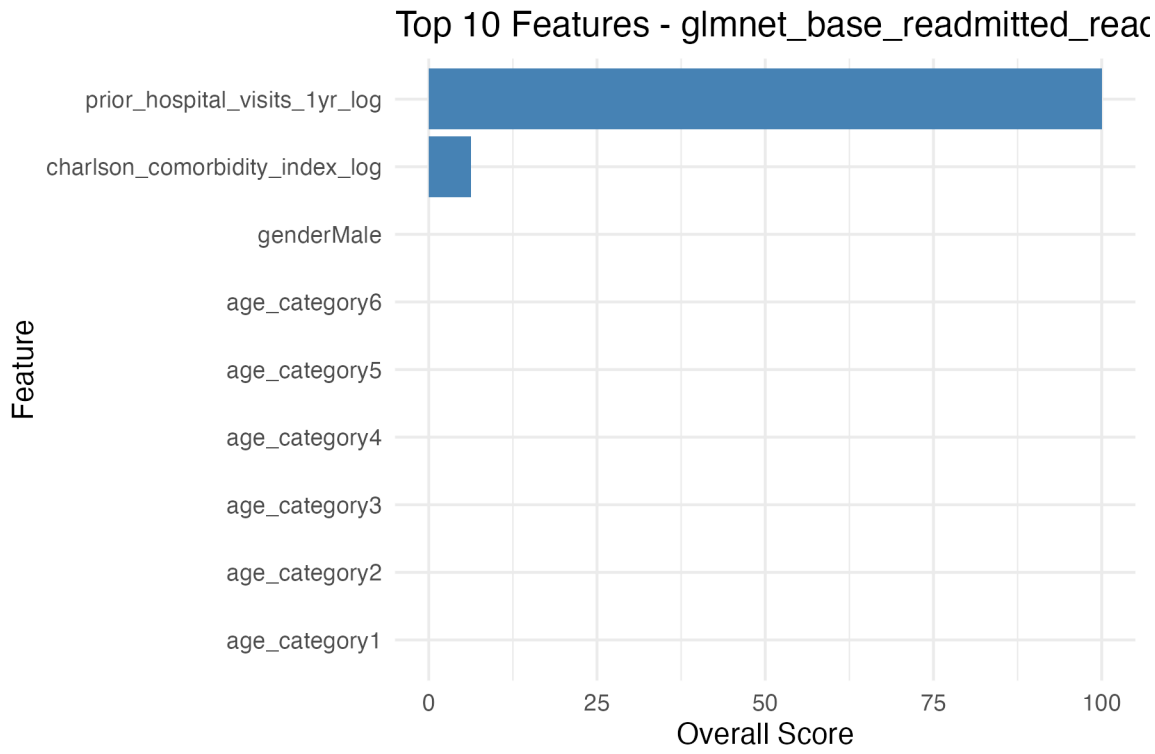

Top 10 Features - glm_base_sdoh_mortality_30_da
Feature
age_category2
age_category3
age_category4
age_category5
age_category6
age
admission_location_numeric
max_oasis
max_apsiii
total_icu_stays_log
0.0
2.5
5.0
7.5
10.0
Relative Importance

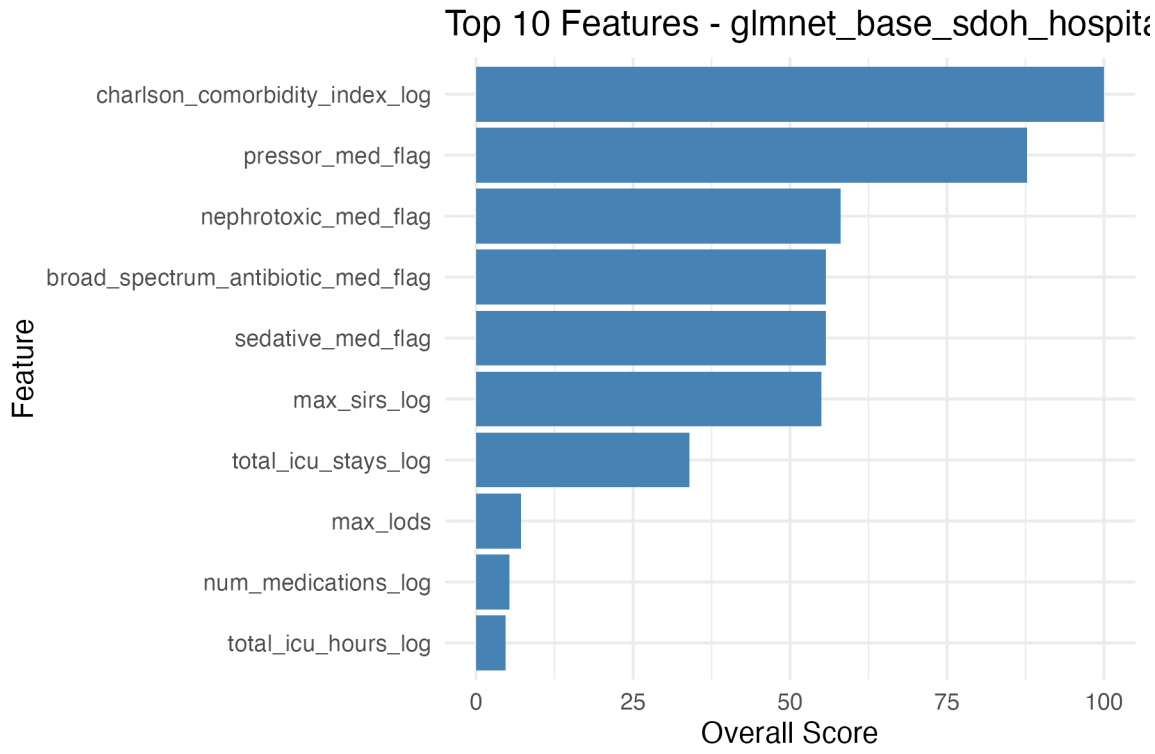

Top 10 Features - glm_base_sdoh_readmitted_rea
age
genderMale
admission_location_numeric
max_sapsii
los_icu_log
orthopedic_surgery_flag
stroke_prevention_flag
central_line_flag
respiratory_support_flag
tracheostomy_flag
Feature
0.0
2.5
5.0
7.5
10.0
Relative Importance

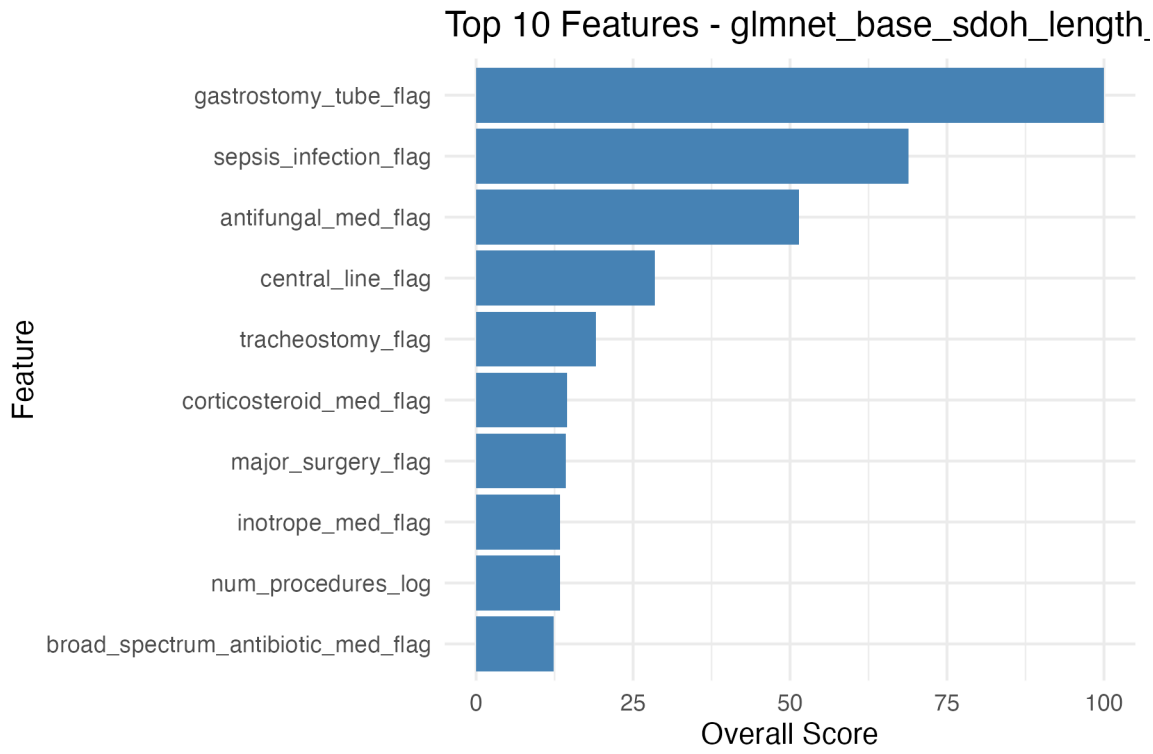

Top 10 Features - glmnet_base_hospital_exp
charlson_comorbidity_index_log
pressor_med_flag
sedative_med_flag
nephrotoxic_med_flag
max_sirs_log
broad_spectrum_antibiotic_med_flag
total_icu_stays_log
max_lods
total_icu_hours_log
num_medications_log
Feature
0
25
50
75
100
Overall Score

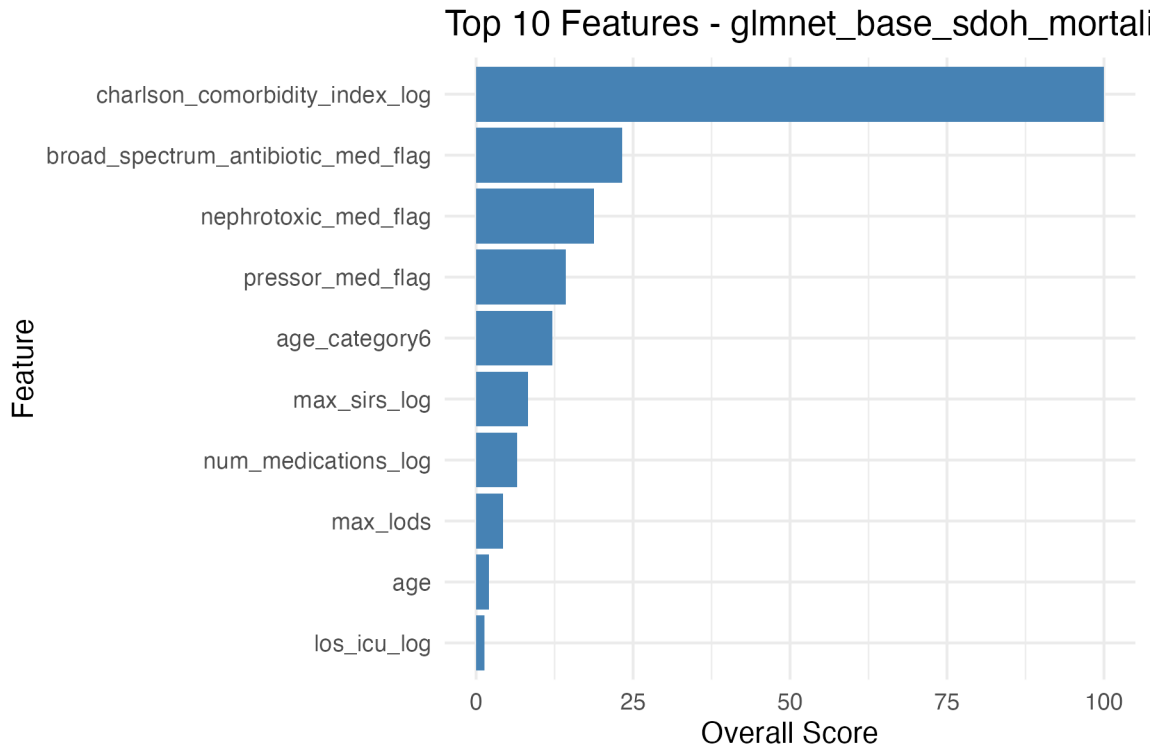

Top 10 Features - glmnet_base_length_of_sta
gastrostomy_tube_flag
sepsis_infection_flag
antifungal_med_flag
central_line_flag
tracheostomy_flag
major_surgery_flag
corticosteroid_med_flag
inotrope_med_flag
num_procedures_log
broad_spectrum_antibiotic_med_flag
Feature
0
25
50
75
100
Overall Score

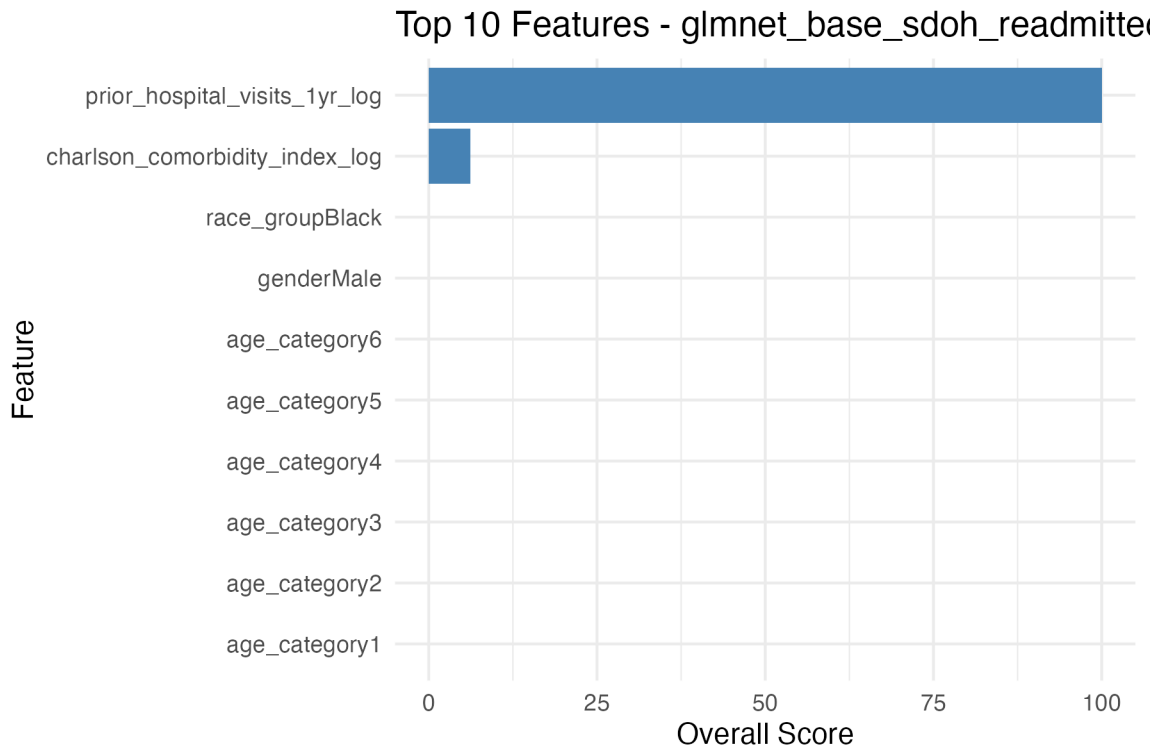

Top 10 Features - glmnet_base_mortality_30_
charlson_comorbidity_index_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
pressor_med_flag
age_category6
max_sirs_log
num_medications_log
max_lods
age
los_icu_log
Feature
0
25
50
75
100
Overall Score

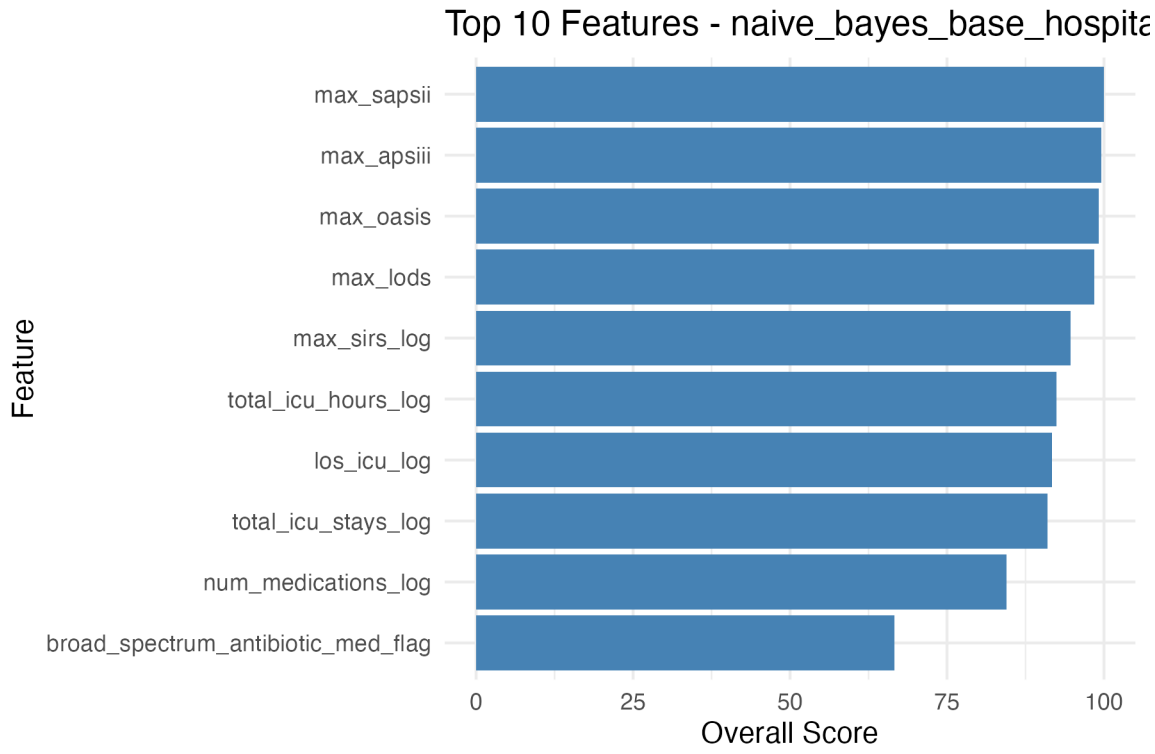

Top 10 Features - glmnet_base_readmitted_reac
prior_hospital_visits_1yr_log
charlson_comorbidity_index_log
genderMale
age_category6
age_category5
age_category4
age_category3
age_category2
age_category1
Feature
0
25
50
75
100
Overall Score

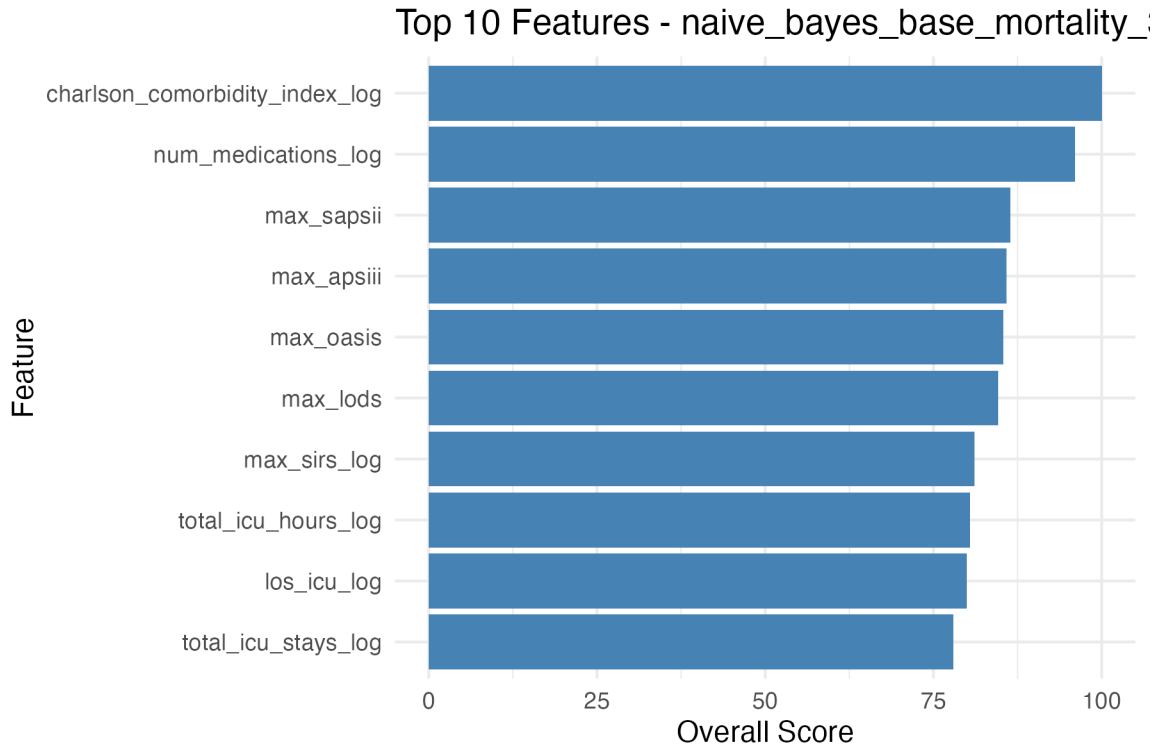

Top 10 Features - glmnet_base_sdoh_hospita
charlson_comorbidity_index_log
pressor_med_flag
nephrotoxic_med_flag
broad_spectrum_antibiotic_med_flag
sedative_med_flag
max_sirs_log
total_icu_stays_log
max_lods
num_medications_log
total_icu_hours_log
Feature
0
25
50
75
100
Overall Score

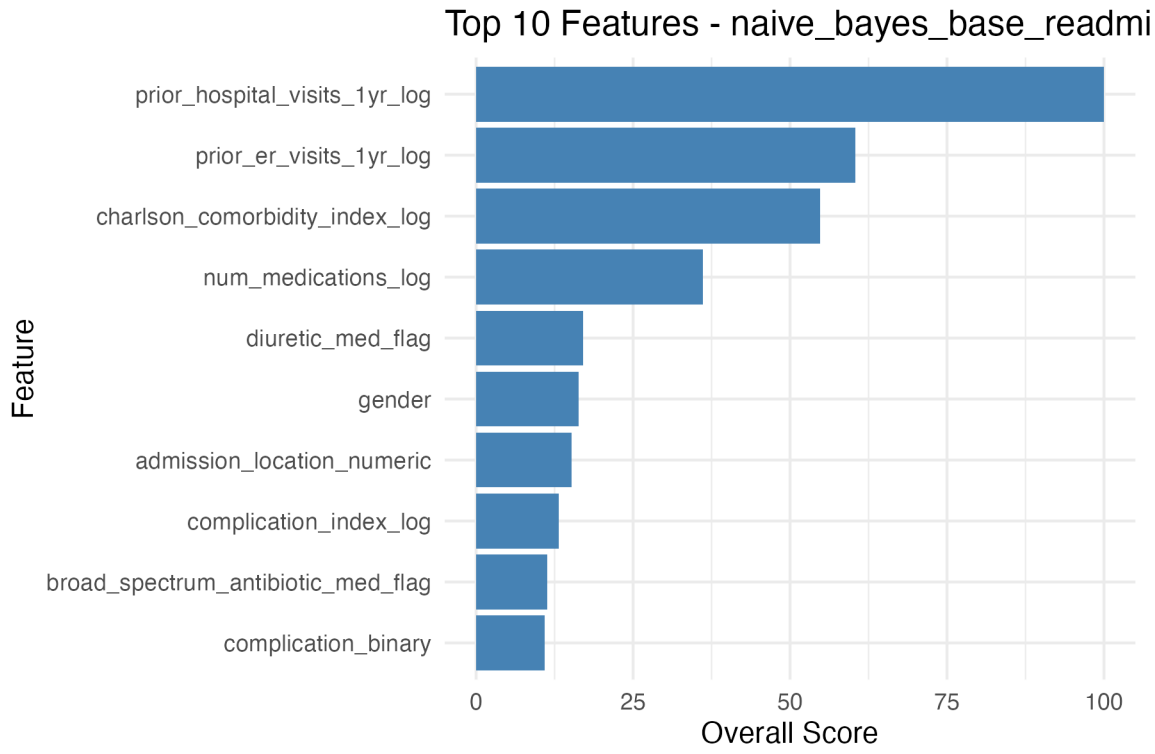

Top 10 Features - glmnet_base_sdoh_length_
gastrostomy_tube_flag
sepsis_infection_flag
antifungal_med_flag
central_line_flag
tracheostomy_flag
corticosteroid_med_flag
major_surgery_flag
inotrope_med_flag
num_procedures_log
broad_spectrum_antibiotic_med_flag
Feature
0
25
50
75
100
Overall Score

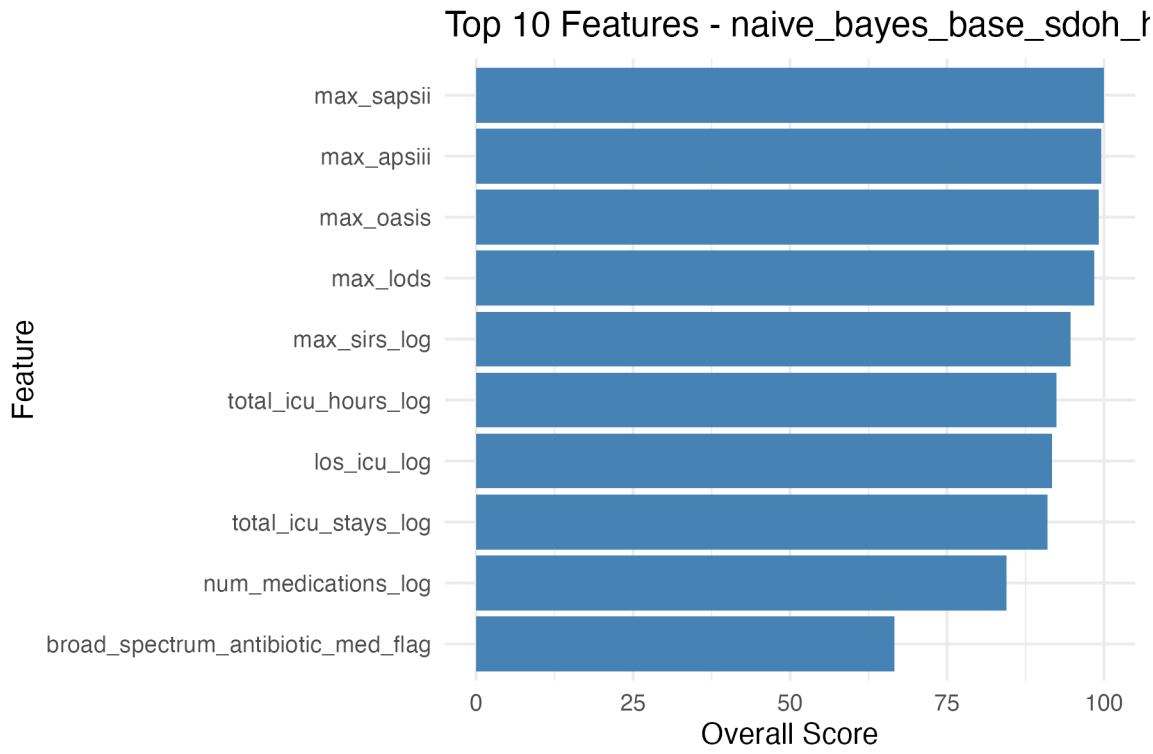

Top 10 Features - glmnet_base_sdoh_mortali
charlson_comorbidity_index_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
pressor_med_flag
age_category6
max_sirs_log
num_medications_log
max_lods
age
los_icu_log
Feature
0
25
50
75
100
Overall Score

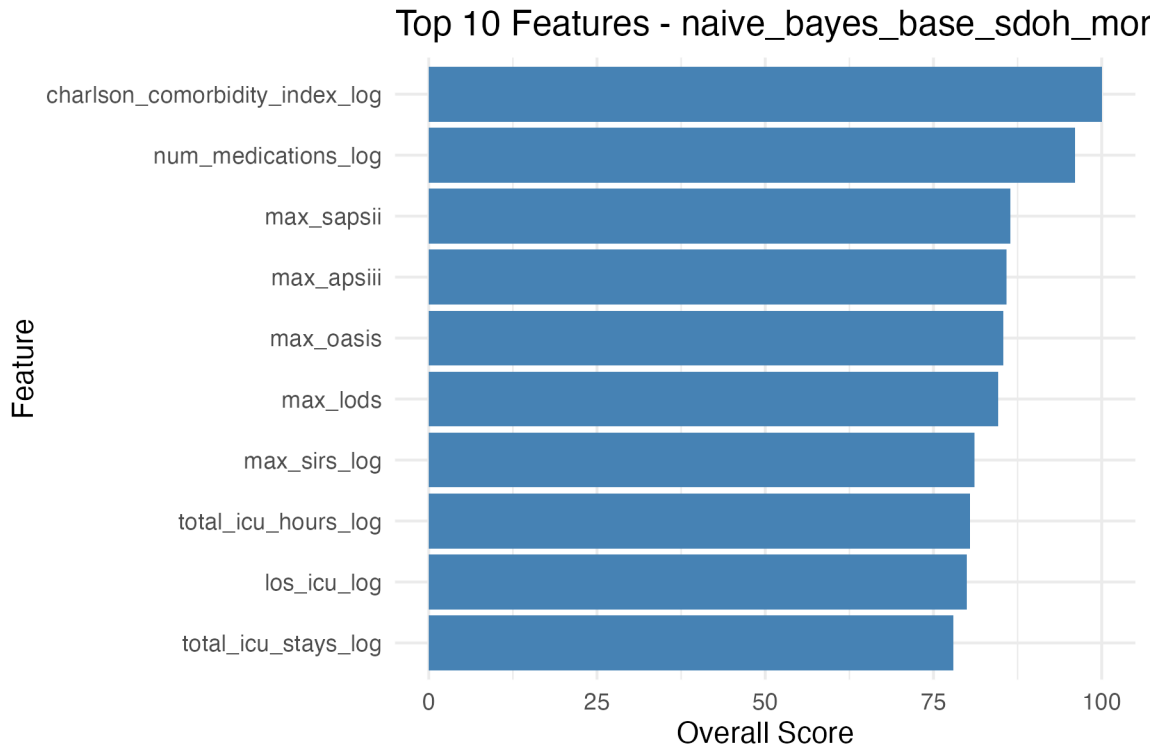

Top 10 Features - glmnet_base_sdoh_readmitte
prior_hospital_visits_1yr_log
charlson_comorbidity_index_log
race_groupBlack
genderMale
age_category6
age_category5
age_category4
age_category3
age_category2
age_category1
Feature
0
25
50
75
100
Overall Score

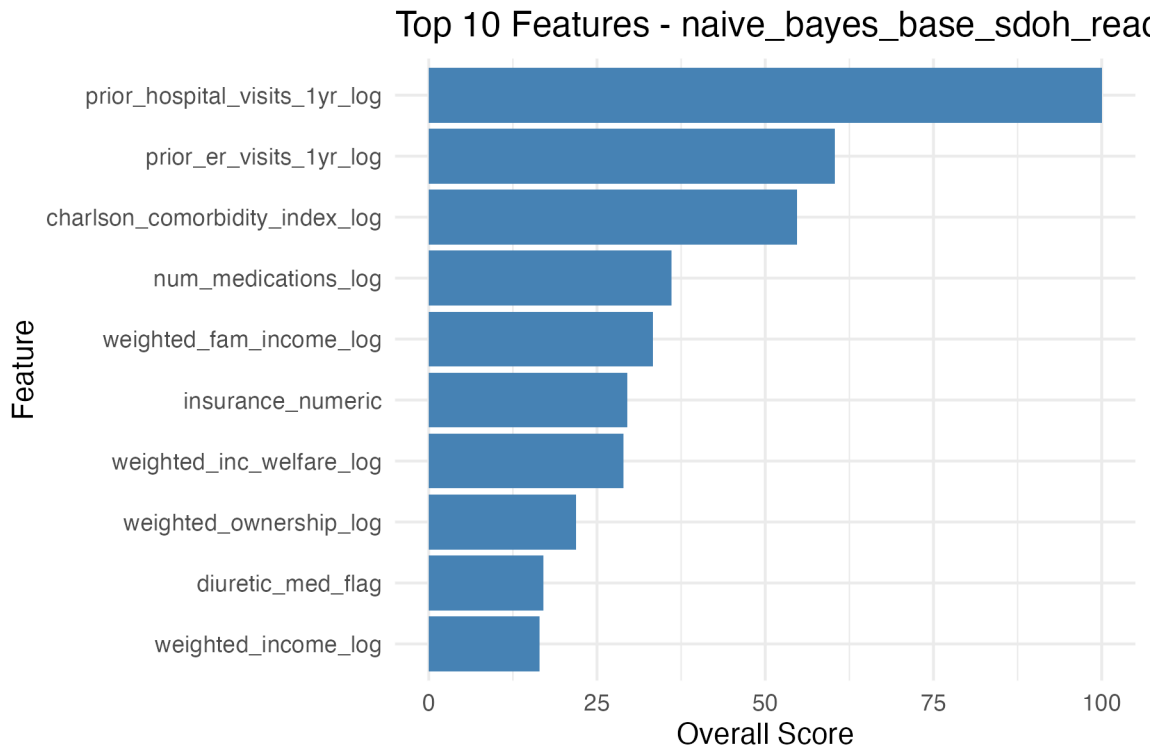

Top 10 Features - naive_bayes_base_hospita
max_sapsii
max_apsiii
max_oasis
max_lods
max_sirs_log
total_icu_hours_log
los_icu_log
total_icu_stays_log
num_medications_log
broad_spectrum_antibiotic_med_flag
Feature
0
25
50
75
100
Overall Score

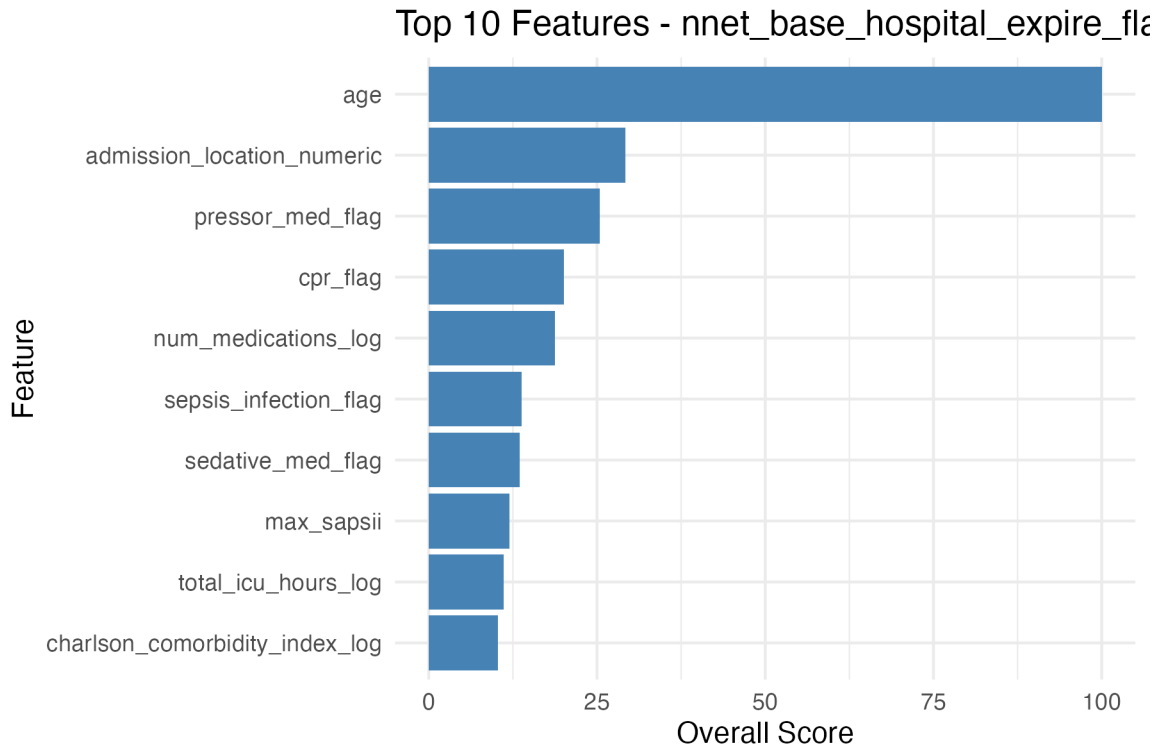

Top 10 Features - naive_bayes_base_mortality_
charlson_comorbidity_index_log
num_medications_log
max_sapsii
max_apsiii
max_oasis
max_lods
max_sirs_log
total_icu_hours_log
los_icu_log
total_icu_stays_log
Feature
0
25
50
75
100
Overall Score

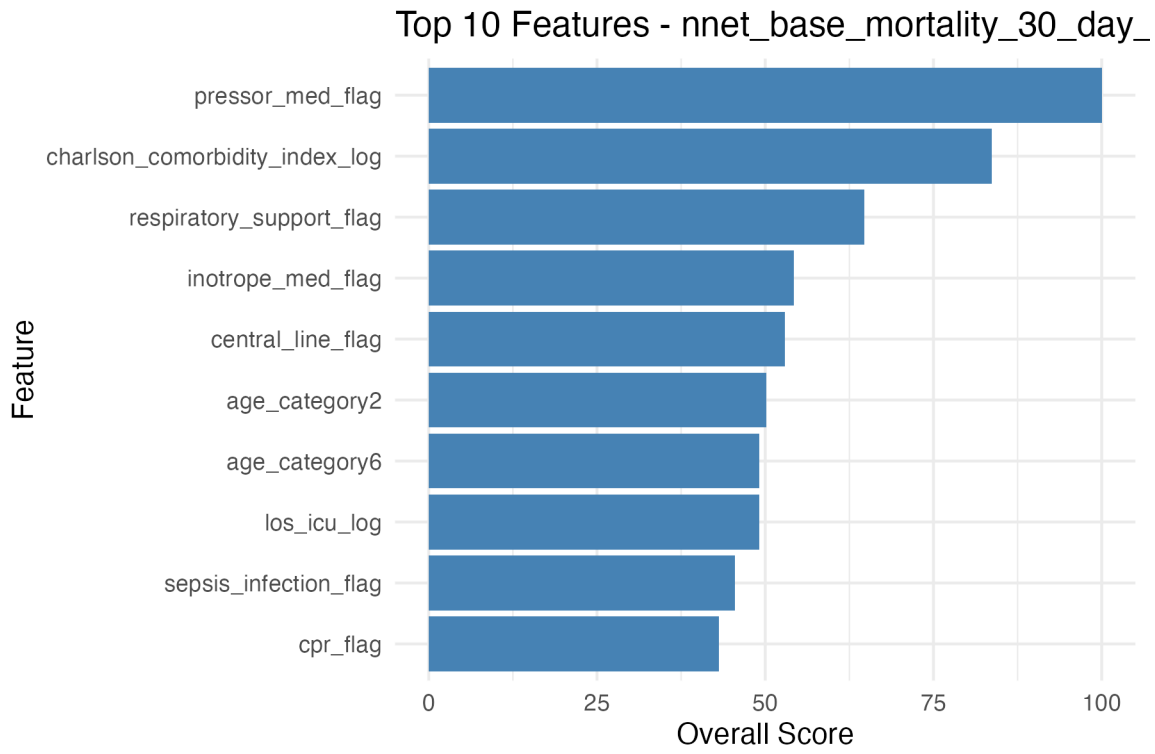

Top 10 Features - naive_bayes_base_readmi
prior_hospital_visits_1yr_log
prior_er_visits_1yr_log
charlson_comorbidity_index_log
num_medications_log
diuretic_med_flag
gender
admission_location_numeric
complication_index_log
broad_spectrum_antibiotic_med_flag
complication_binary
Feature
0
25
50
75
100
Overall Score

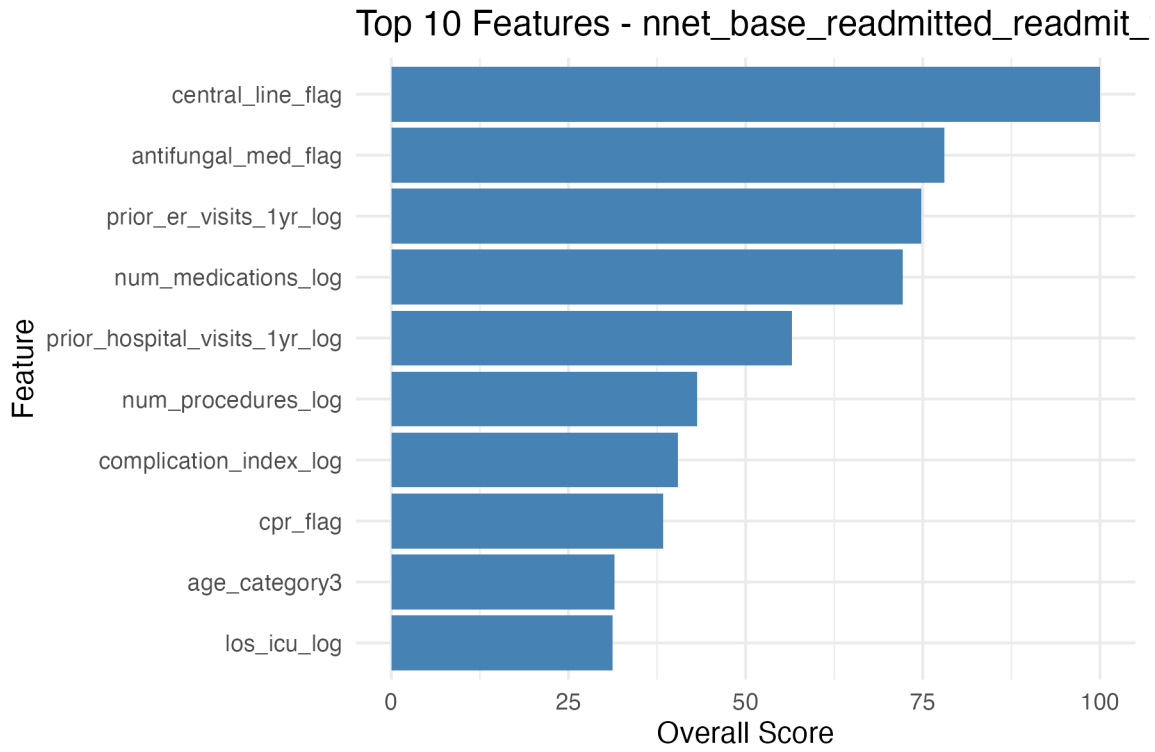

Top 10 Features - naive_bayes_base_sdoh_h
max_sapsii
max_apsiii
max_oasis
max_lods
max_sirs_log
total_icu_hours_log
los_icu_log
total_icu_stays_log
num_medications_log
broad_spectrum_antibiotic_med_flag
Feature
0
25
50
75
100
Overall Score

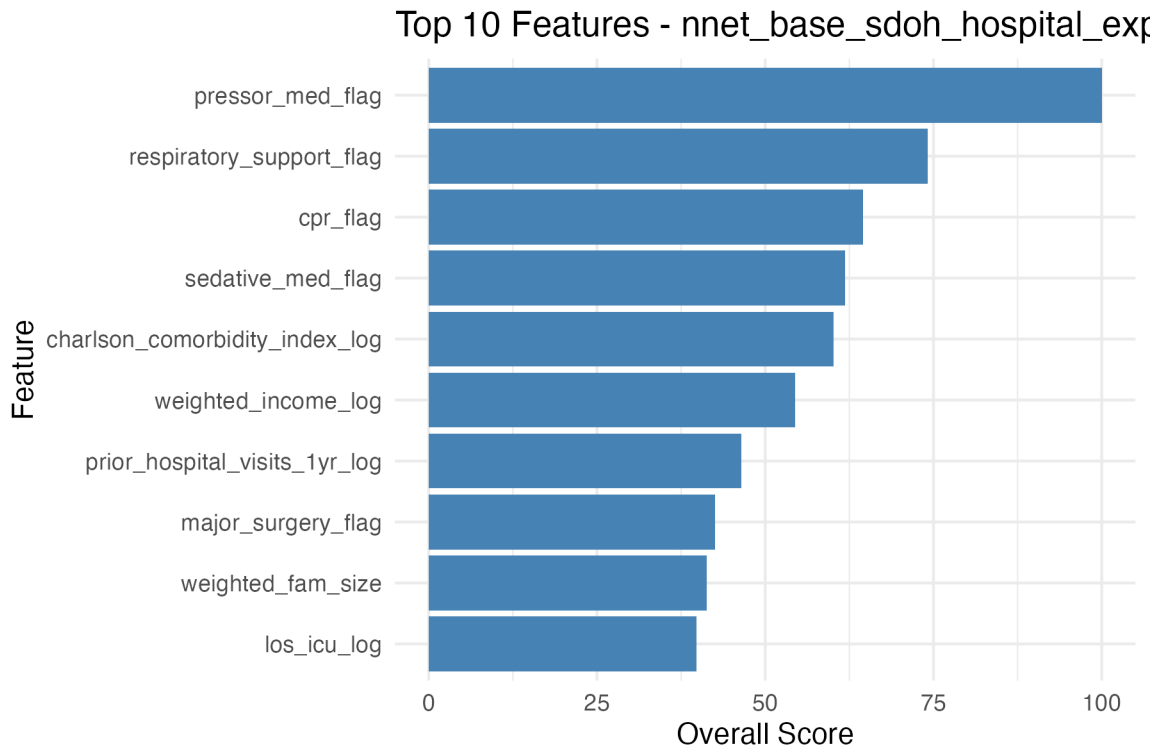

Top 10 Features - naive_bayes_base_sdoh_mor
charlson_comorbidity_index_log
num_medications_log
max_sapsii
max_apsiii
max_oasis
max_lods
max_sirs_log
total_icu_hours_log
los_icu_log
total_icu_stays_log
Feature
0
25
50
75
100
Overall Score

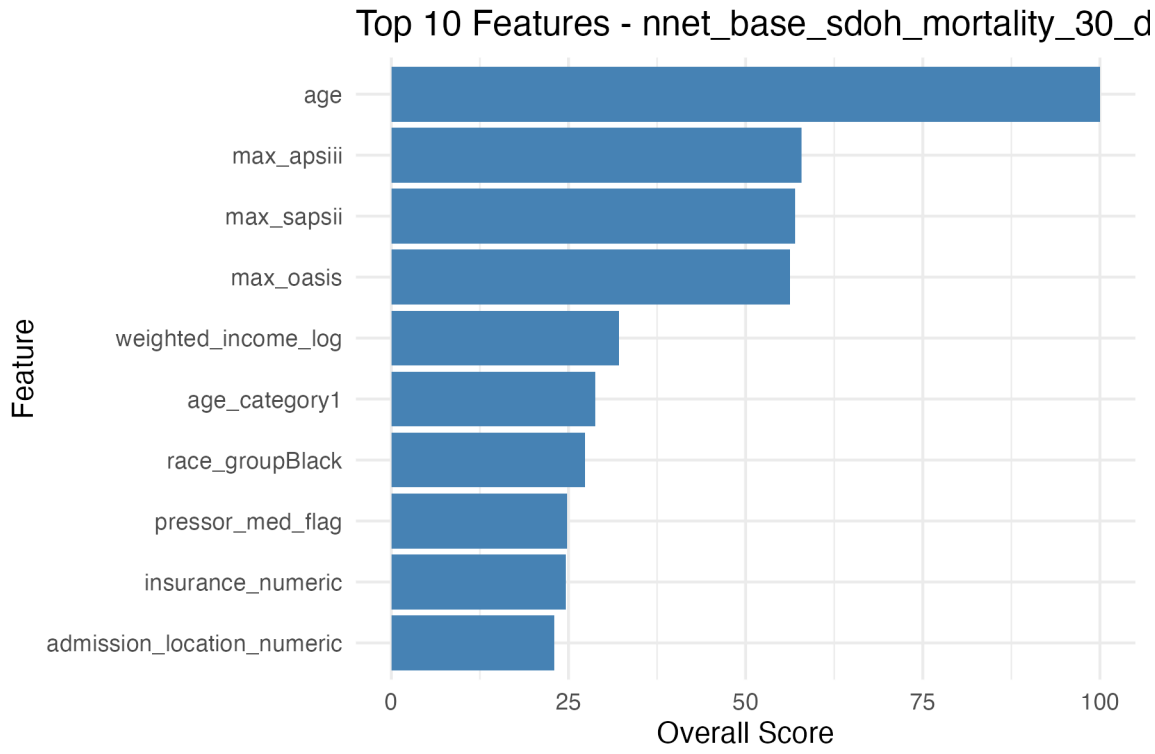

Top 10 Features - naive_bayes_base_sdoh_reac
prior_hospital_visits_1yr_log
prior_er_visits_1yr_log
charlson_comorbidity_index_log
num_medications_log
weighted_fam_income_log
insurance_numeric
weighted_inc_welfare_log
weighted_ownership_log
diuretic_med_flag
weighted_income_log
Feature
0
25
50
75
100
Overall Score

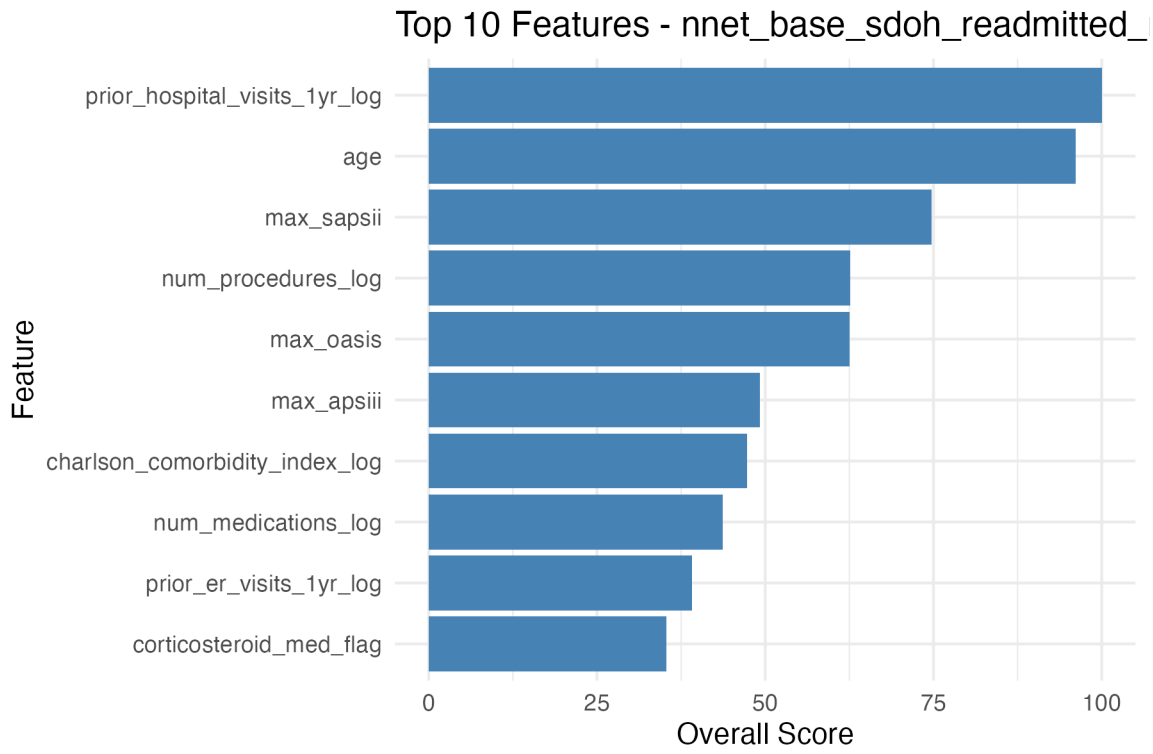

Top 10 Features - nnet_base_hospital_expire_fla
age
admission_location_numeric
pressor_med_flag
cpr_flag
num_medications_log
sepsis_infection_flag
sedative_med_flag
max_sapsii
total_icu_hours_log
charlson_comorbidity_index_log
Feature
0
25
50
75
100
Overall Score

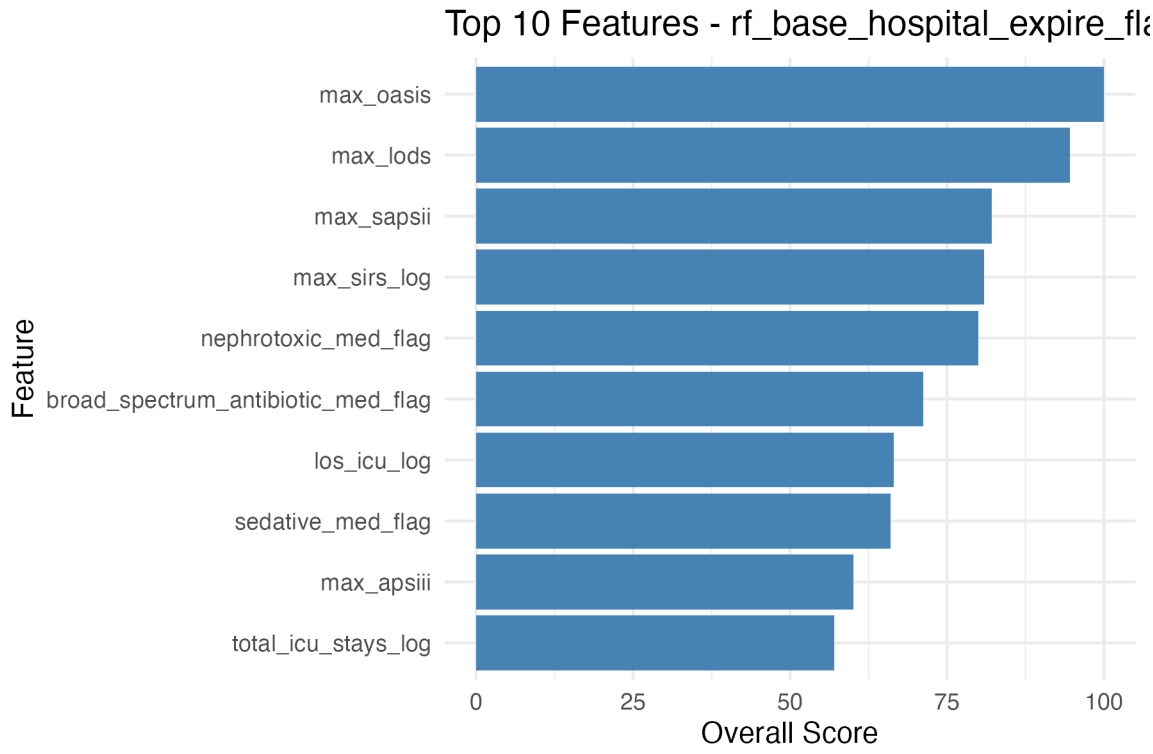

Top 10 Features - nnet_base_mortality_30_day_
pressor_med_flag
charlson_comorbidity_index_log
respiratory_support_flag
inotrope_med_flag
central_line_flag
age_category2
age_category6
los_icu_log
sepsis_infection_flag
cpr_flag
Feature
0
25
50
75
100
Overall Score

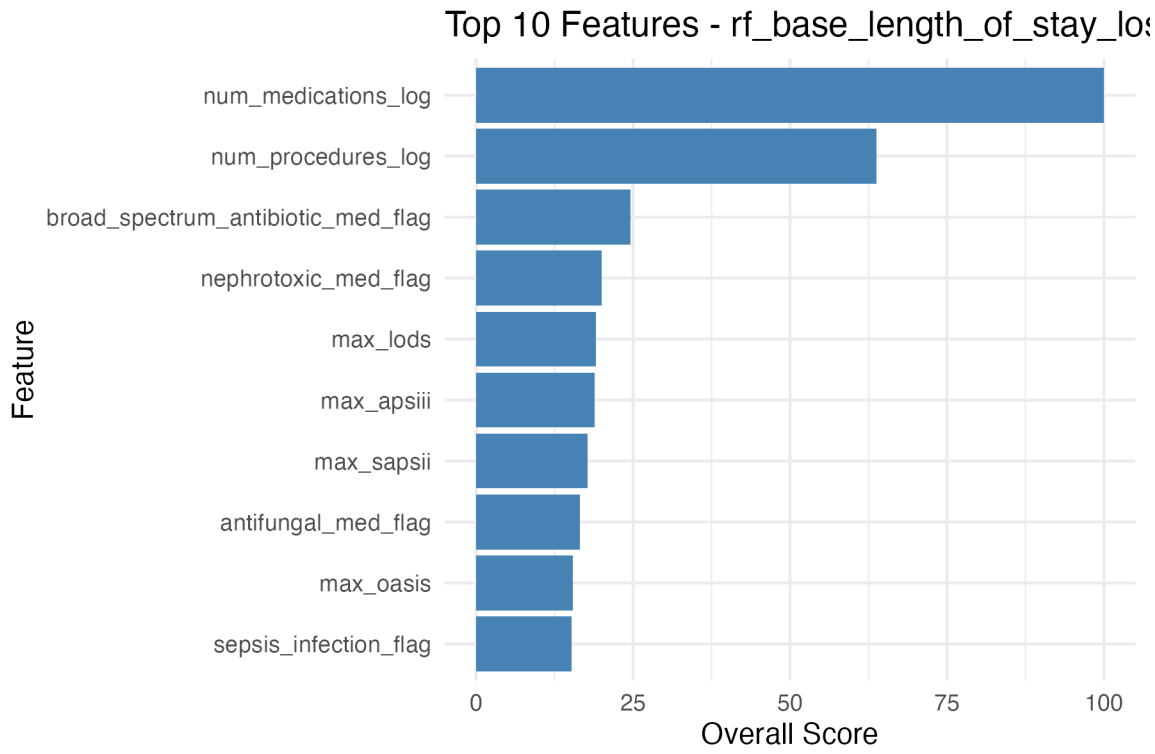

Top 10 Features - nnet_base_readmitted_readmit_
central_line_flag
antifungal_med_flag
prior_er_visits_1yr_log
num_medications_log
prior_hospital_visits_1yr_log
num_procedures_log
complication_index_log
cpr_flag
age_category3
los_icu_log
Feature
0
25
50
75
100
Overall Score

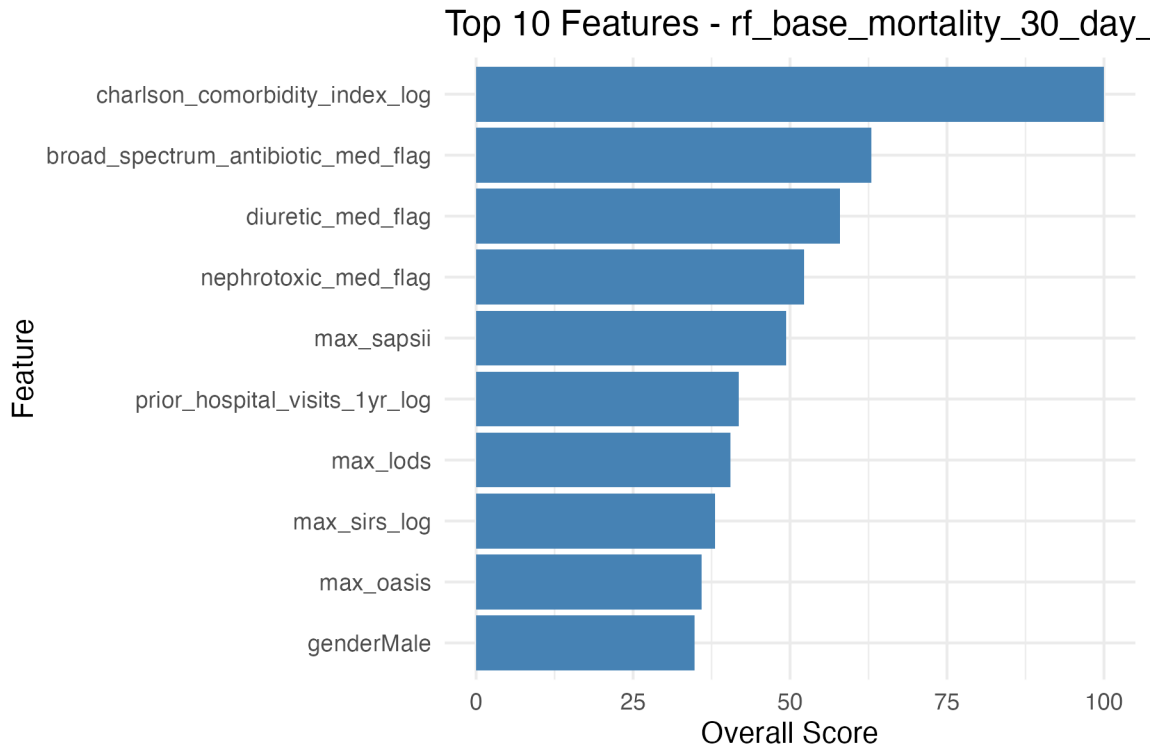

Top 10 Features - nnet_base_sdoh_hospital_exp
pressor_med_flag
respiratory_support_flag
cpr_flag
sedative_med_flag
charlson_comorbidity_index_log
weighted_income_log
prior_hospital_visits_1yr_log
major_surgery_flag
weighted_fam_size
los_icu_log
Feature
0
25
50
75
100
Overall Score

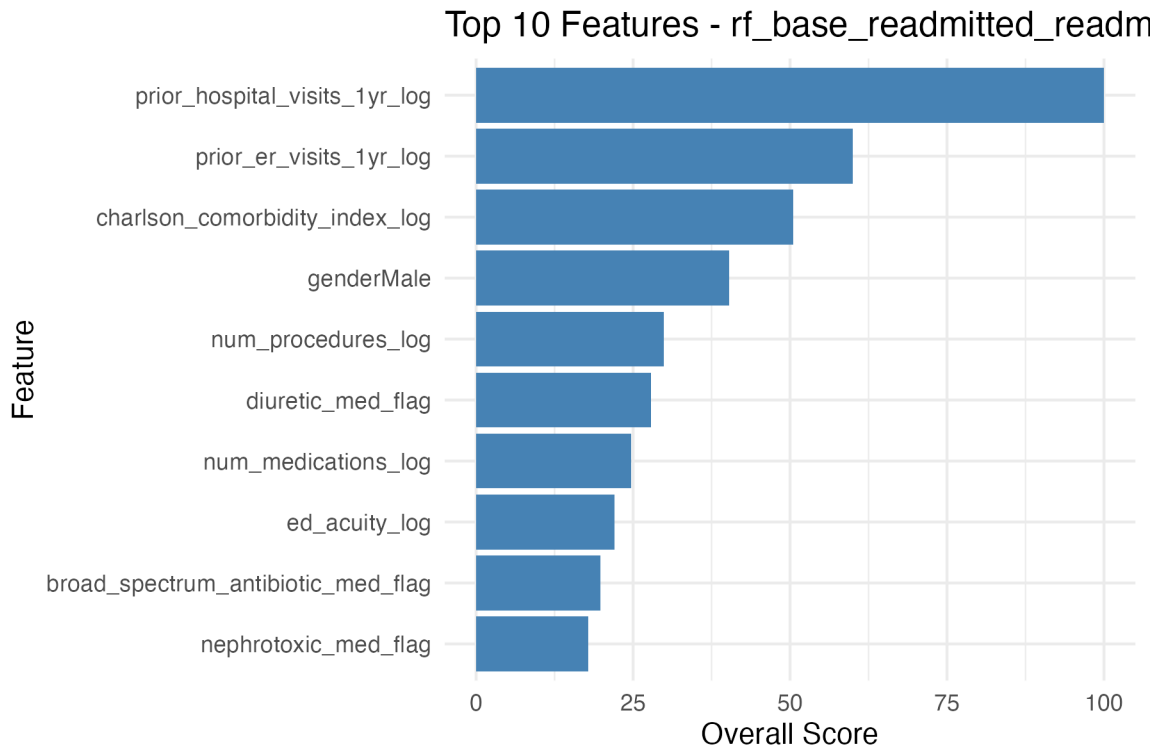

Top 10 Features - nnet_base_sdoh_mortality_30_d
age
max_apsiii
max_sapsii
max_oasis
weighted_income_log
age_category1
race_groupBlack
pressor_med_flag
insurance_numeric
admission_location_numeric
Feature
0
25
50
75
100
Overall Score

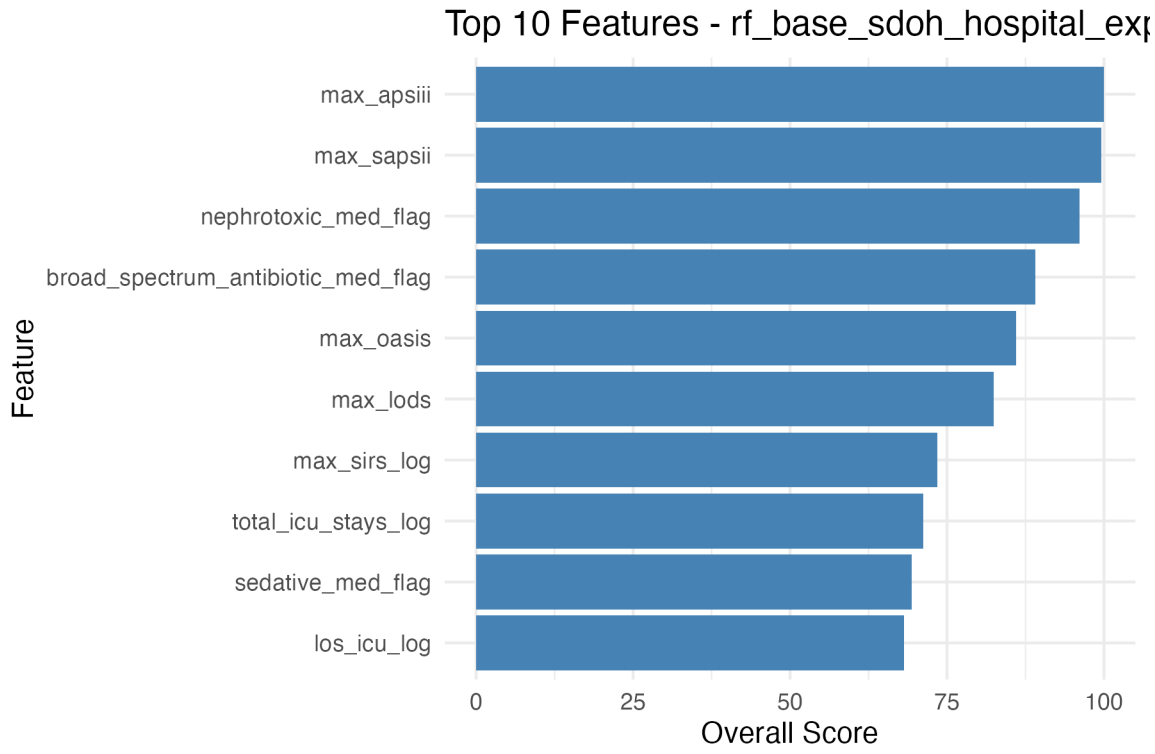

Top 10 Features - nnet_base_sdoh_readmitted_
prior_hospital_visits_1yr_log
age
max_sapsii
num_procedures_log
max_oasis
max_apsiii
charlson_comorbidity_index_log
num_medications_log
prior_er_visits_1yr_log
corticosteroid_med_flag
Feature
0
25
50
75
100
Overall Score

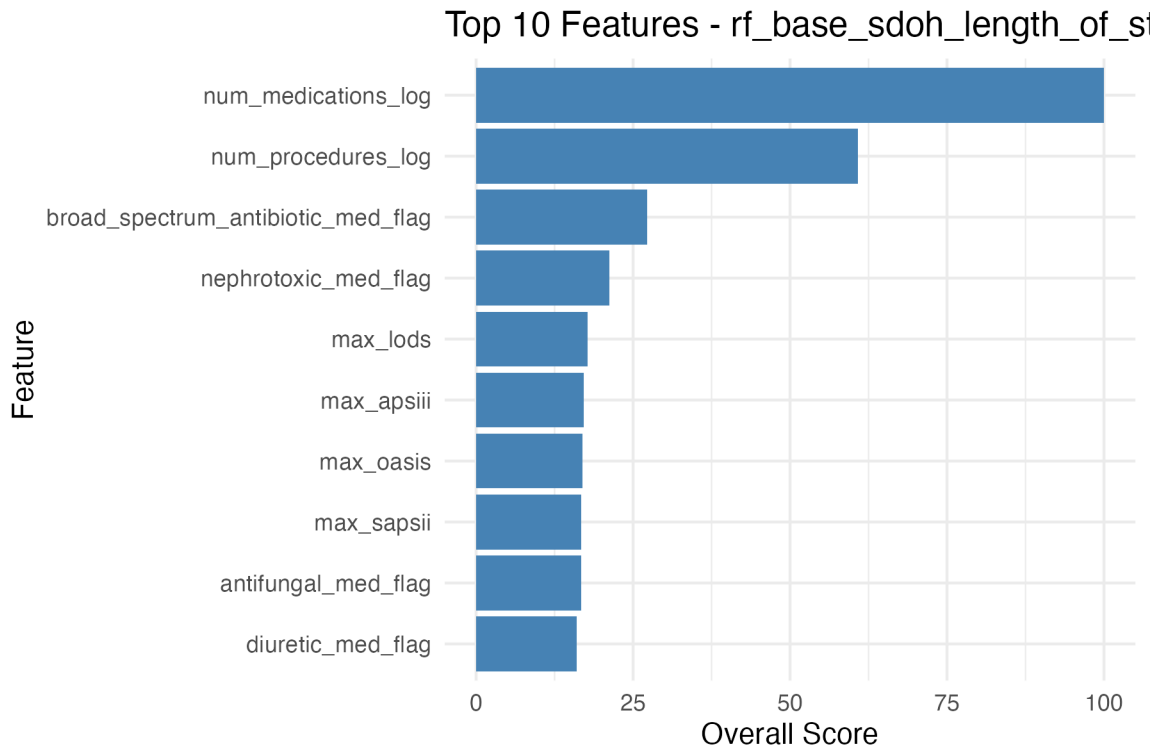

Top 10 Features - rf_base_hospital_expire_fl
max_oasis
max_lods
max_sapsii
max_sirs_log
nephrotoxic_med_flag
broad_spectrum_antibiotic_med_flag
los_icu_log
sedative_med_flag
max_apsiii
total_icu_stays_log
Feature
0
25
50
75
100
Overall Score

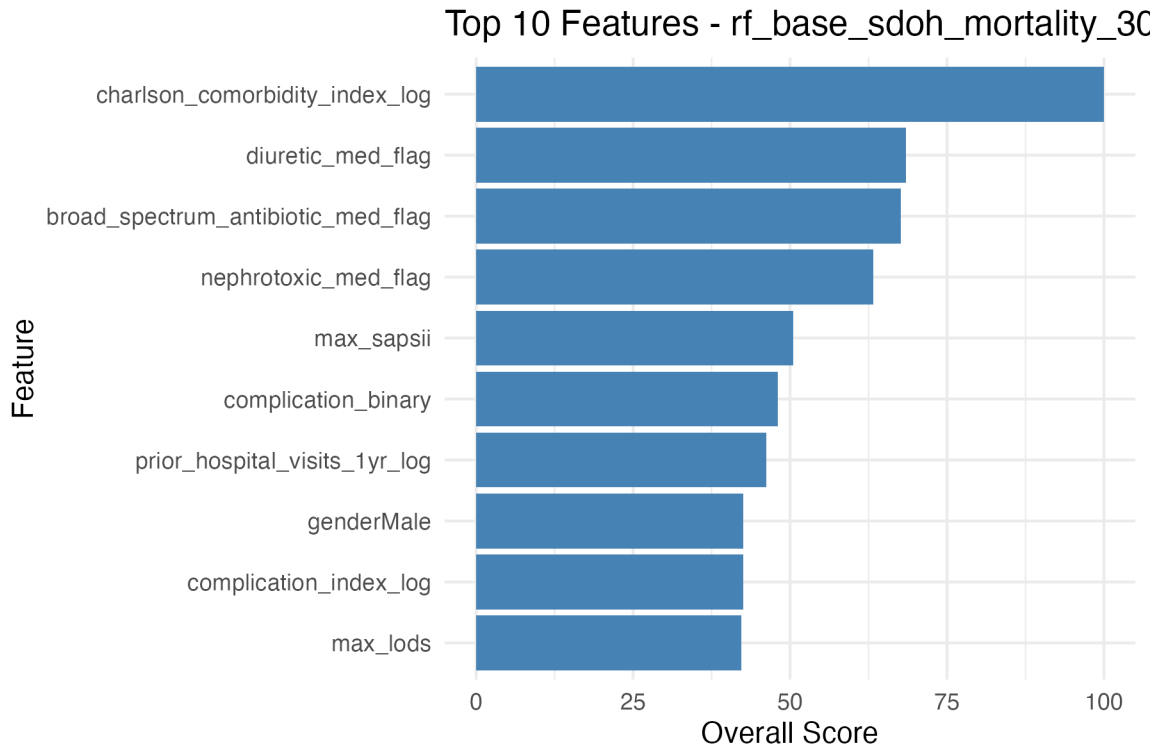

Top 10 Features - rf_base_length_of_stay_los
num_medications_log
num_procedures_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
max_lods
max_apsiii
max_sapsii
antifungal_med_flag
max_oasis
sepsis_infection_flag
Feature
0
25
50
75
100
Overall Score

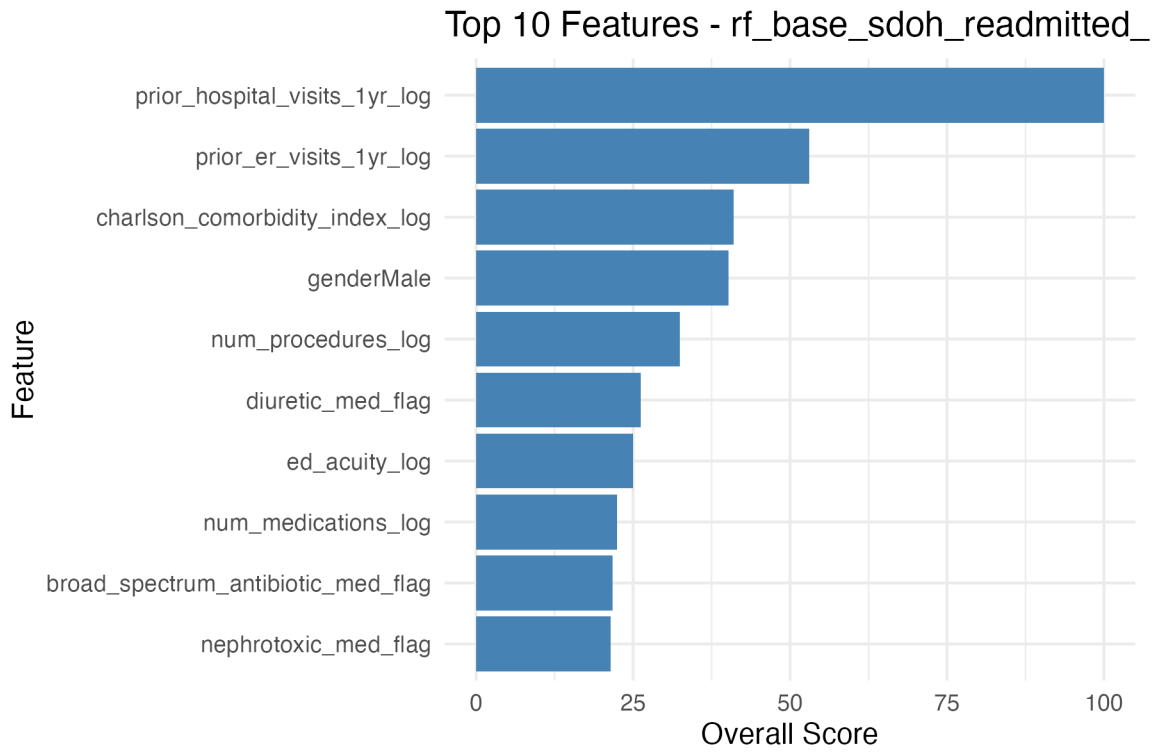

Top 10 Features - rf_base_mortality_30_day_
charlson_comorbidity_index_log
broad_spectrum_antibiotic_med_flag
diuretic_med_flag
nephrotoxic_med_flag
max_sapsii
prior_hospital_visits_1yr_log
max_lods
max_sirs_log
max_oasis
genderMale
Feature
0
25
50
75
100
Overall Score

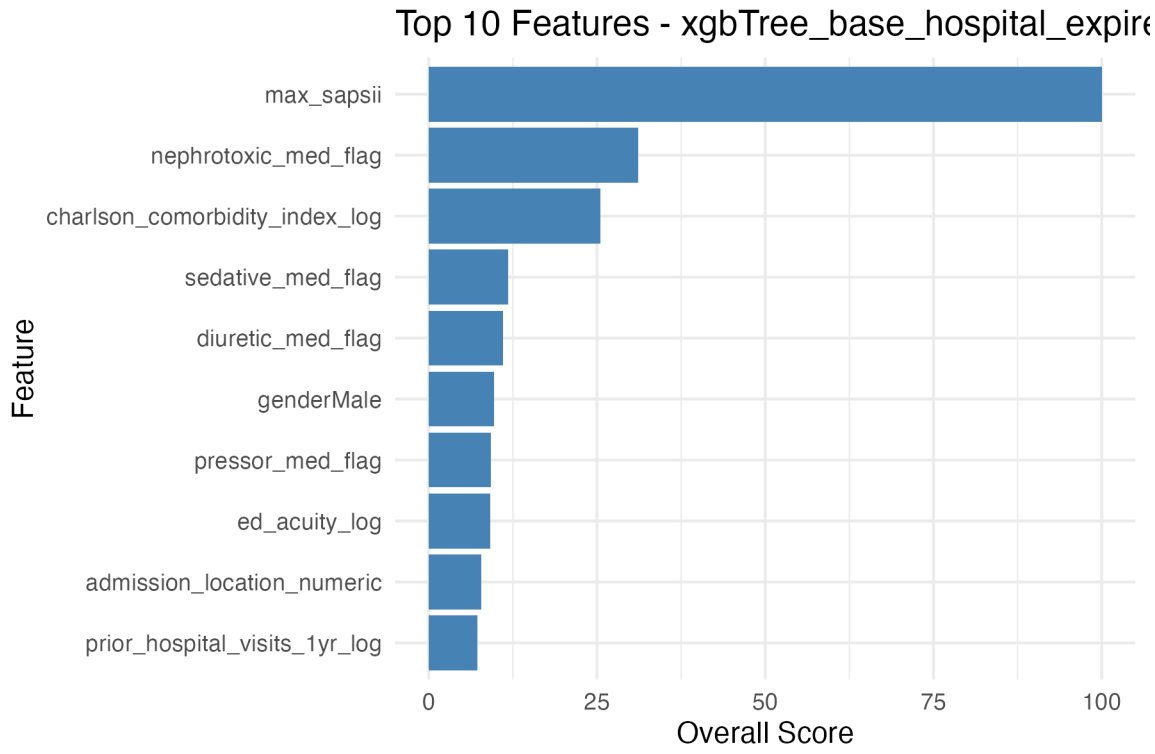

Top 10 Features - rf_base_readmitted_readm
prior_hospital_visits_1yr_log
prior_er_visits_1yr_log
charlson_comorbidity_index_log
genderMale
num_procedures_log
diuretic_med_flag
num_medications_log
ed_acuity_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
Feature
0
25
50
75
100
Overall Score

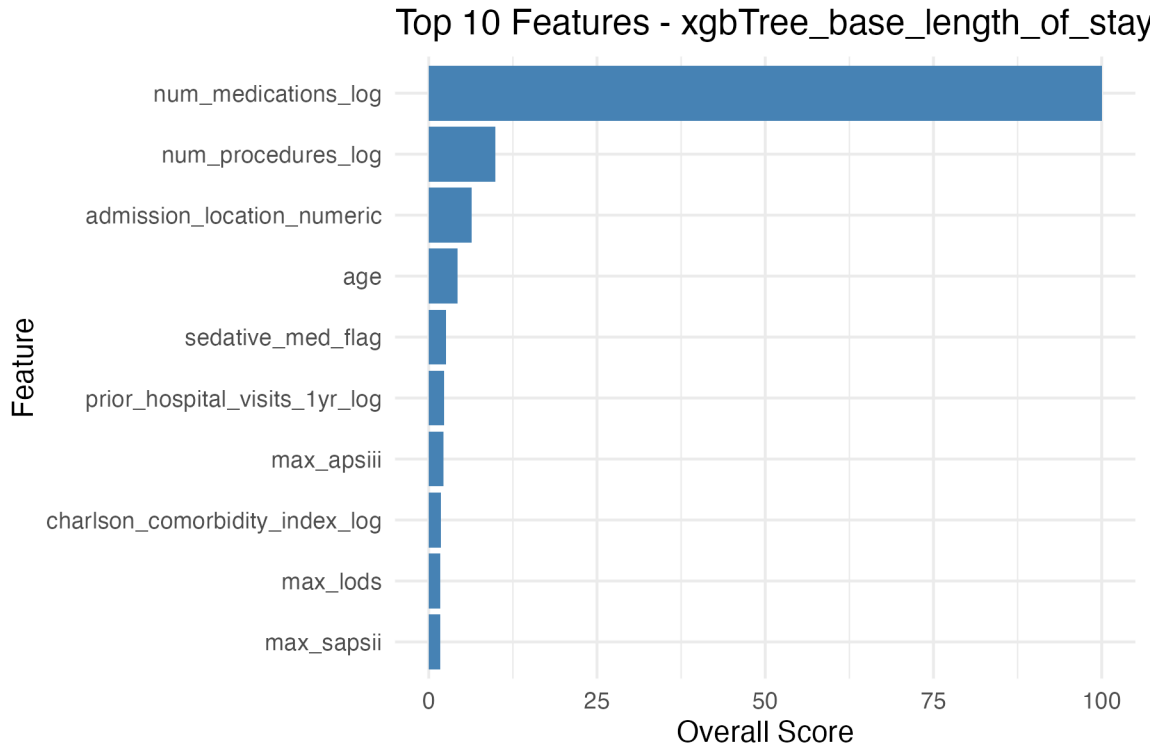

Top 10 Features - rf_base_sdoh_hospital_exp
max_apsiii
max_sapsii
nephrotoxic_med_flag
broad_spectrum_antibiotic_med_flag
max_oasis
max_lods
max_sirs_log
total_icu_stays_log
sedative_med_flag
los_icu_log
Feature
0
25
50
75
100
Overall Score

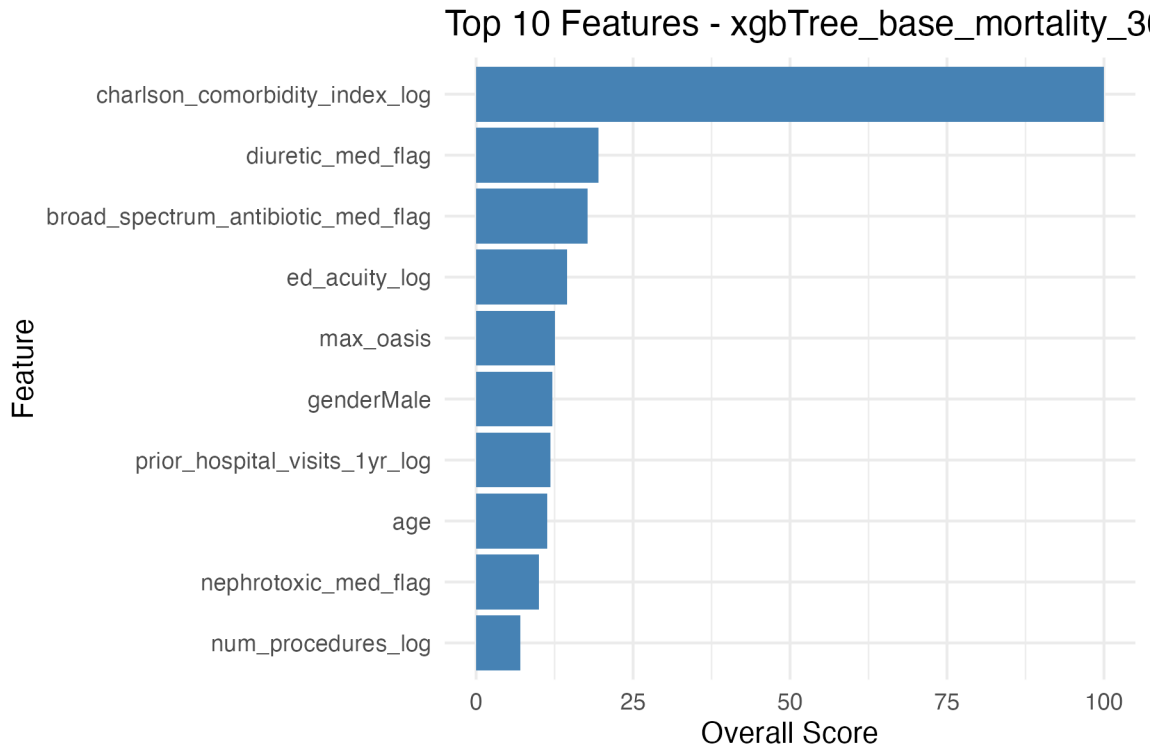

Top 10 Features - rf_base_sdoh_length_of_st
num_medications_log
num_procedures_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
max_lods
max_apsiii
max_oasis
max_sapsii
antifungal_med_flag
diuretic_med_flag
Feature
0
25
50
75
100
Overall Score

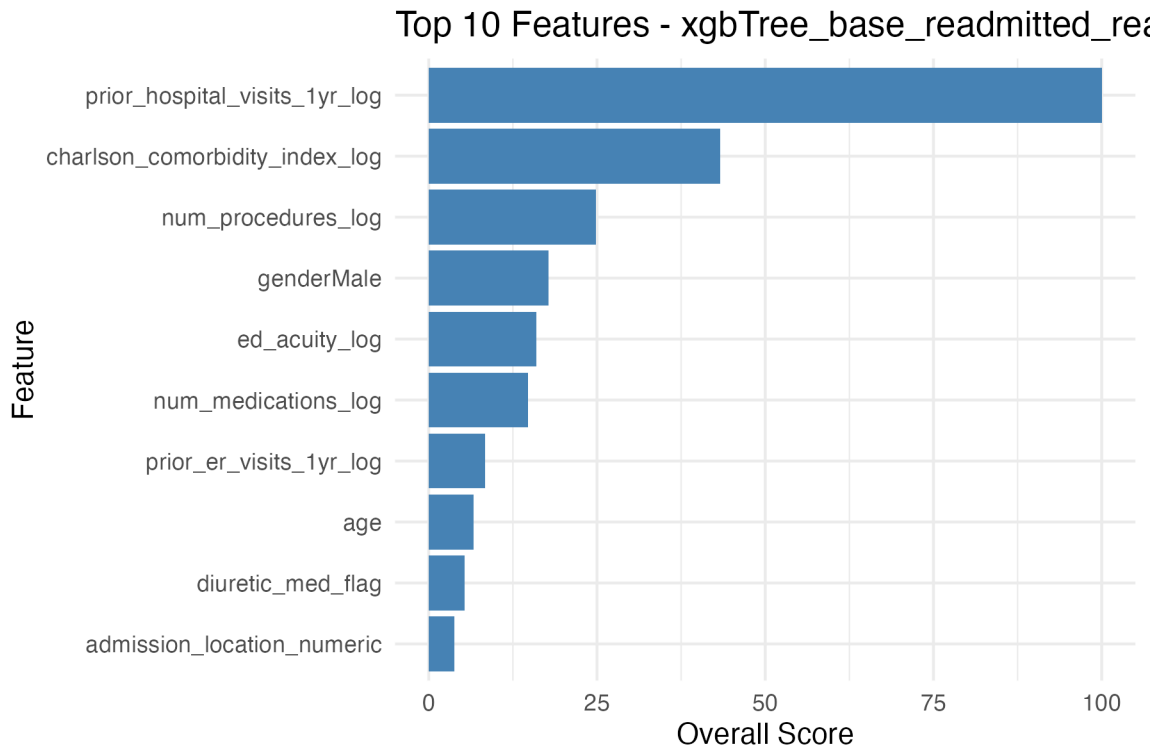

Top 10 Features - rf_base_sdoh_mortality_30
charlson_comorbidity_index_log
diuretic_med_flag
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
max_sapsii
complication_binary
prior_hospital_visits_1yr_log
genderMale
complication_index_log
max_lods
Feature
0
25
50
75
100
Overall Score

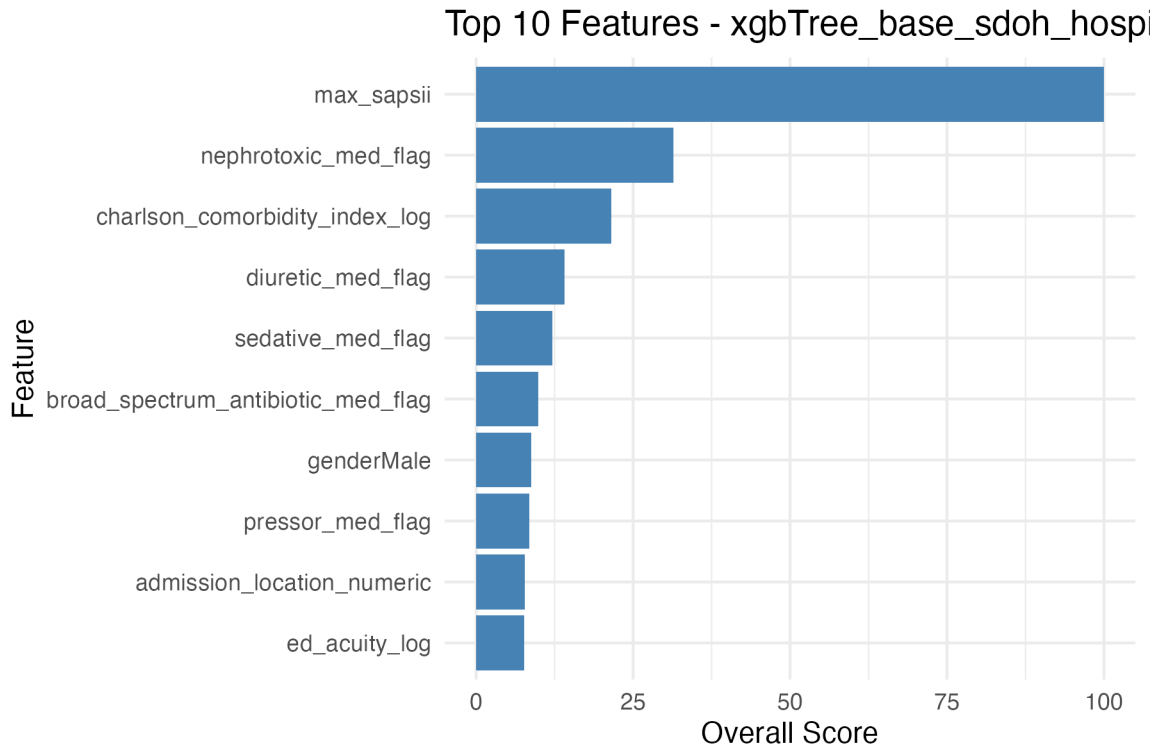

Top 10 Features - rf_base_sdoh_readmitted_
prior_hospital_visits_1yr_log
prior_er_visits_1yr_log
charlson_comorbidity_index_log
genderMale
num_procedures_log
diuretic_med_flag
ed_acuity_log
num_medications_log
broad_spectrum_antibiotic_med_flag
nephrotoxic_med_flag
Feature
0
25
50
75
100
Overall Score

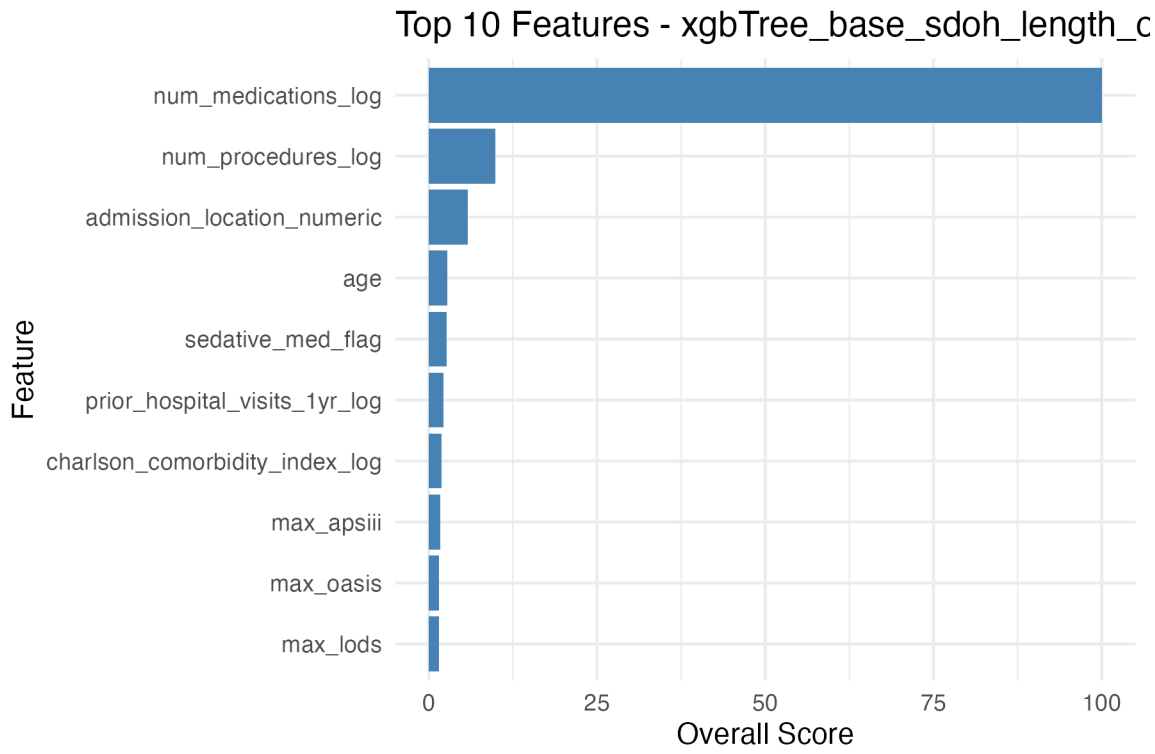

Top 10 Features - xgbTree_base_hospital_expire
max_sapsii
nephrotoxic_med_flag
charlson_comorbidity_index_log
sedative_med_flag
diuretic_med_flag
genderMale
pressor_med_flag
ed_acuity_log
admission_location_numeric
prior_hospital_visits_1yr_log
Feature
0
25
50
75
100
Overall Score

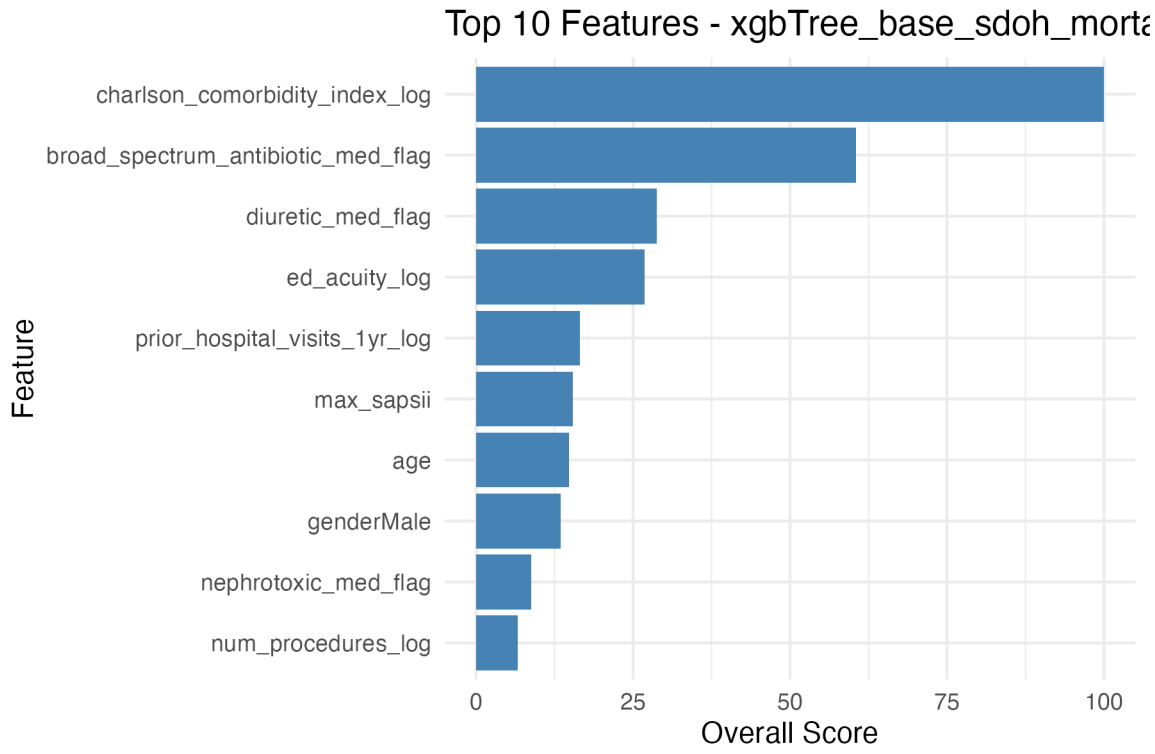

Top 10 Features - xgbTree_base_length_of_stay
num_medications_log
num_procedures_log
admission_location_numeric
age
sedative_med_flag
prior_hospital_visits_1yr_log
max_apsiii
charlson_comorbidity_index_log
max_lods
max_sapsii
Feature
0
25
50
75
100
Overall Score

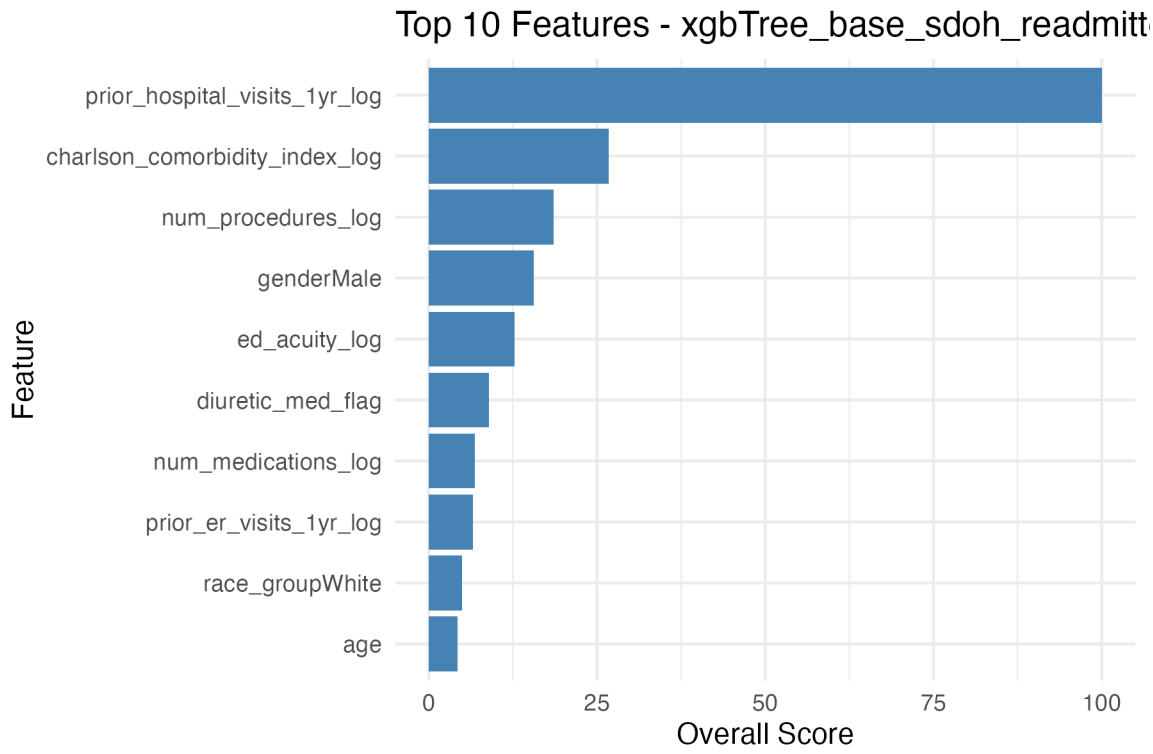

Top 10 Features - xgbTree_base_mortality_3
charlson_comorbidity_index_log
diuretic_med_flag
broad_spectrum_antibiotic_med_flag
ed_acuity_log
max_oasis
genderMale
prior_hospital_visits_1yr_log
age
nephrotoxic_med_flag
num_procedures_log
Feature
0
25
50
75
100
Overall Score


Top 10 Features - xgbTree_base_readmitted_rea
prior_hospital_visits_1yr_log
charlson_comorbidity_index_log
num_procedures_log
genderMale
ed_acuity_log
num_medications_log
prior_er_visits_1yr_log
age
diuretic_med_flag
admission_location_numeric
Feature
0
25
50
75
100
Overall Score


Top 10 Features - xgbTree_base_sdoh_hospi
max_sapsii
nephrotoxic_med_flag
charlson_comorbidity_index_log
diuretic_med_flag
sedative_med_flag
broad_spectrum_antibiotic_med_flag
genderMale
pressor_med_flag
admission_location_numeric
ed_acuity_log
Feature
0
25
50
75
100
Overall Score


Top 10 Features - xgbTree_base_sdoh_length_c
num_medications_log
num_procedures_log
admission_location_numeric
age
sedative_med_flag
prior_hospital_visits_1yr_log
charlson_comorbidity_index_log
max_apsiii
max_oasis
max_lods
Feature
0
25
50
75
100
Overall Score


Top 10 Features - xgbTree_base_sdoh_morta
charlson_comorbidity_index_log
broad_spectrum_antibiotic_med_flag
diuretic_med_flag
ed_acuity_log
prior_hospital_visits_1yr_log
max_sapsii
age
genderMale
nephrotoxic_med_flag
num_procedures_log
Feature
0
25
50
75
100
Overall Score



Section 2: Code Snippets

Bigquery_connect.R:

```
# Load required libraries
library(bigrquery)
library(DBI)
library(here)

# Set authentication options
options(httr_oauth_cache = TRUE)

# Define the path to the service account JSON file
json_path <- here::here("credentials", "msds-795-3b7180d11ca3.json")

# Authenticate with Google BigQuery
if (file.exists(json_path)) {
  bq_auth(path = json_path)  # Use service account if available
  message("Successfully authenticated using service account.")
} else {
  message("Service account JSON not found. Switching to manual authentication.")
  bq_auth()  # Prompt user login if service account is missing
}

# Function to connect to a BigQuery dataset
connect_to_bigquery <- function(project, dataset, billing_project) {
  tryCatch({
    con <- dbConnect(
```

```
      bigrquery::bigquery(),
      project = project,
      dataset = dataset,
      billing = billing_project
    )
    message("Connected to BigQuery dataset: ", dataset)
    return(con)
  }, error = function(e) {
    message("Connection failed: ", e$message)
    return(NULL)
  })
}

# Connect to the public MIMIC-IV hospital dataset
con_hosp <- connect_to_bigquery(
  project = "physionet-data",
  dataset = "mimiciv_3_1_hosp",
  billing_project = "msds-795"
)

# Connect to the private MIMIC-IV derived dataset
con_derived <- connect_to_bigquery(
  project = "msds-795",
  dataset = "mimiciv_derived",
  billing_project = "msds-795"
)

# Verify active connections
if (!is.null(con_hosp)) message("Connection to `mimiciv_3_1_hosp` is active.")
if (!is.null(con_derived)) message("Connection to `mimiciv_derived` is active.")
```

utils.R

```
# save a dataset
save_dataset <- function(data, filename, subdir = "data/processed") {
  # Ensure the directory exists
  dir.create(here(subdir), recursive = TRUE, showWarnings = FALSE)

  # full file path
  file_path <- here(subdir, filename)

  #Save dataset
  saveRDS(data, file = file_path)

  message("Dataset saved to: ", file_path)
}

# load a dataset
load_dataset <- function(filename, subdir = "data/processed") {
  # Construct full file path
  file_path <- here(subdir, filename)

  if (!file.exists(file_path)) {
    stop("Error: File does not exist - ", file_path)
  }
```

```
  data <- readRDS(file_path)
  message("Dataset loaded from: ", file_path)
  return(data)
}

# Clear Data

clear_data_objects <- function() {
  suppressWarnings({
    rm(data_hosp, data_ICU, data_cms, envir = .GlobalEnv)
  })
  gc(verbose = FALSE)
  message("Cleared data_hosp, data_ICU, data_cms from memory.")
}

######### Figures ############


save_named_plot <- function(plot_expr, filename, folder = "../presentations/figures", width = 8, height = 6, dpi =
300) {
  full_path <- file.path(folder, paste0(filename, ".png"))
  ggsave(full_path, plot = plot_expr, device = "png", width = width, height = height, dpi = dpi)
  message("Saved: ", full_path)
}



########  Models  ############

save_models <- function(models, subfolder = NULL, clear = FALSE) {
  dir_path <- here::here("models", subfolder)
  dir.create(dir_path, showWarnings = FALSE, recursive = TRUE)

  for (name in names(models)) {
    file_path <- file.path(dir_path, paste0(name, ".rds"))
    saveRDS(models[[name]], file = file_path)

    if (clear) {
      rm(list = name, envir = .GlobalEnv)
    }
  }

  message("Saved models in: ", dir_path)
}

# Load models from a subfolder inside /models/
load_models <- function(subfolder = NULL) {
  dir_path <- here::here("models", subfolder)
  files <- list.files(dir_path, pattern = "\\.rds$", full.names = TRUE)
  models <- lapply(files, readRDS)
  names(models) <- sub("\\.rds$", "", basename(files))
  models
}

load_models_to_list <- function(subfolder, pattern = NULL, strip_train = TRUE) {
```

```
  dir_path <- here::here("models", subfolder)
  files <- list.files(dir_path, pattern = "\\.rds$", full.names = TRUE)

  if (!is.null(pattern)) {
    files <- files[grepl(pattern, basename(files))]
  }

  if (length(files) == 0) {
    message("No matching .rds files found in: ", dir_path)
    return(list())
  }
  models <- list()
  for (file in files) {
    model <- readRDS(file)
    if (strip_train && !is.null(model$train_data)) model$train_data <- NULL
    name <- sub("\\.rds$", "", basename(file))
    models[[name]] <- model
    message("Loaded model: ", name)
  }
    return(models)
}
```

Process_data.R

```
# Load required libraries
library(bigrquery)
library(DBI)
library(here)
library(glue)

sql_query_admin_samp <- glue("

WITH all_admissions AS (
   -- Full admissions table for readmission calculation
   SELECT *
   FROM `physionet-data.mimiciv_3_1_hosp.admissions`
),

sampled_admissions AS (
   -- Randomly sample 50% of patients but keep all their admissions
   SELECT a.*
   FROM all_admissions AS a
   WHERE subject_id IN (
      -- Select 10% of patients but retain all their admissions
      SELECT DISTINCT subject_id
      FROM all_admissions
      WHERE MOD(ABS(FARM_FINGERPRINT(CAST(subject_id AS STRING))), 100) <
{sample_percent}
   )
),

## Complications:
```

```
# WITH surgery_patients AS (
#    SELECT
#       proc.subject_id, proc.hadm_id,
#       MIN(proc.charttime) AS first_surgery_time,
#       1 AS surgery_flag
#    FROM `physionet-data.mimiciv_3_1_hosp.procedures_icd` AS proc
#    WHERE proc.icd_code LIKE '0J%'  -- Example: Identify major surgical ICD-10 codes
#    GROUP BY proc.subject_id, proc.hadm_id
# )


diag AS (
   -- Extract diagnosis codes with ICD version for sampled admissions
   SELECT
      d.hadm_id,
      CASE WHEN d.icd_version = 9 THEN d.icd_code ELSE NULL END AS icd9_code,
      CASE WHEN d.icd_version = 10 THEN d.icd_code ELSE NULL END AS icd10_code
   FROM `physionet-data.mimiciv_3_1_hosp.diagnoses_icd` AS d
   WHERE d.hadm_id IN (SELECT hadm_id FROM sampled_admissions)
),

diagnoses_features AS (
   SELECT
      d.hadm_id

      -- Myocardial infarction
      , MAX(CASE WHEN
         SUBSTR(d.icd9_code, 1, 3) IN ('410', '412')
         OR
         SUBSTR(d.icd10_code, 1, 3) IN ('I21', 'I22')
         OR
         SUBSTR(d.icd10_code, 1, 4) = 'I252'
         THEN 1
         ELSE 0 END) AS myocardial_infarct

      -- Congestive heart failure
      , MAX(CASE WHEN
         SUBSTR(d.icd9_code, 1, 3) = '428'
         OR
         SUBSTR(
            icd9_code, 1, 5
         ) IN ('39891', '40201', '40211', '40291', '40401', '40403'
            , '40411', '40413', '40491', '40493')
         OR
         SUBSTR(d.icd9_code, 1, 4) BETWEEN '4254' AND '4259'
         OR
         SUBSTR(d.icd10_code, 1, 3) IN ('I43', 'I50')
         OR
         SUBSTR(
            icd10_code, 1, 4
         ) IN ('I099', 'I110', 'I130', 'I132', 'I255', 'I420'
             , 'I425', 'I426', 'I427', 'I428', 'I429', 'P290'
         )
         THEN 1
         ELSE 0 END) AS congestive_heart_failure
```

```
-- Peripheral vascular disease
, MAX(CASE WHEN
   SUBSTR(d.icd9_code, 1, 3) IN ('440', '441')
   OR
   SUBSTR(
      icd9_code, 1, 4
   ) IN ('0930', '4373', '4471', '5571', '5579', 'V434')
   OR
   SUBSTR(d.icd9_code, 1, 4) BETWEEN '4431' AND '4439'
   OR
   SUBSTR(d.icd10_code, 1, 3) IN ('I70', 'I71')
   OR
   SUBSTR(d.icd10_code, 1, 4) IN ('I731', 'I738', 'I739', 'I771', 'I790'
                                , 'I792'
                                , 'K551'
                                , 'K558'
                                , 'K559'
                                , 'Z958'
                                , 'Z959'
   )
   THEN 1
   ELSE 0 END) AS peripheral_vascular_disease

-- Cerebrovascular disease
, MAX(CASE WHEN
   SUBSTR(d.icd9_code, 1, 3) BETWEEN '430' AND '438'
   OR
   SUBSTR(d.icd9_code, 1, 5) = '36234'
   OR
   SUBSTR(d.icd10_code, 1, 3) IN ('G45', 'G46')
   OR
   SUBSTR(d.icd10_code, 1, 3) BETWEEN 'I60' AND 'I69'
   OR
   SUBSTR(d.icd10_code, 1, 4) = 'H340'
   THEN 1
   ELSE 0 END) AS cerebrovascular_disease

-- Dementia
, MAX(CASE WHEN
   SUBSTR(d.icd9_code, 1, 3) = '290'
   OR
   SUBSTR(d.icd9_code, 1, 4) IN ('2941', '3312')
   OR
   SUBSTR(d.icd10_code, 1, 3) IN ('F00', 'F01', 'F02', 'F03', 'G30')
   OR
   SUBSTR(d.icd10_code, 1, 4) IN ('F051', 'G311')
   THEN 1
   ELSE 0 END) AS dementia

-- Chronic pulmonary disease
, MAX(CASE WHEN
   SUBSTR(d.icd9_code, 1, 3) BETWEEN '490' AND '505'
   OR
   SUBSTR(d.icd9_code, 1, 4) IN ('4168', '4169', '5064', '5081', '5088')
```

```
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'J40' AND 'J47'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'J60' AND 'J67'
    OR
    SUBSTR(d.icd10_code, 1, 4) IN ('I278', 'I279', 'J684', 'J701', 'J703')
    THEN 1
    ELSE 0 END) AS chronic_pulmonary_disease

-- Rheumatic disease
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) = '725'
    OR
    SUBSTR(d.icd9_code, 1, 4) IN ('4465', '7100', '7101', '7102', '7103'
                            , '7104', '7140', '7141', '7142', '7148'
    )
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('M05', 'M06', 'M32', 'M33', 'M34')
    OR
    SUBSTR(d.icd10_code, 1, 4) IN ('M315', 'M351', 'M353', 'M360')
    THEN 1
    ELSE 0 END) AS rheumatic_disease

-- Peptic ulcer disease
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) IN ('531', '532', '533', '534')
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('K25', 'K26', 'K27', 'K28')
    THEN 1
    ELSE 0 END) AS peptic_ulcer_disease

-- Mild liver disease
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) IN ('570', '571')
    OR
    SUBSTR(
        icd9_code, 1, 4
    ) IN ('0706', '0709', '5733', '5734', '5738', '5739', 'V427')
    OR
    SUBSTR(
        icd9_code, 1, 5
    ) IN ('07022', '07023', '07032', '07033', '07044', '07054')
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('B18', 'K73', 'K74')
    OR
    SUBSTR(
        icd10_code, 1, 4
    ) IN ('K700', 'K701', 'K702', 'K703', 'K709', 'K713'
          , 'K714', 'K715', 'K717', 'K760', 'K762'
          , 'K763', 'K764', 'K768', 'K769', 'Z944')
    THEN 1
    ELSE 0 END) AS mild_liver_disease

-- Diabetes without chronic complication
, MAX(CASE WHEN
```

```
    SUBSTR(
      icd9_code, 1, 4
    ) IN ('2500', '2501', '2502', '2503', '2508', '2509')
    OR
    SUBSTR(
      icd10_code, 1, 4
    ) IN ('E100', 'E101', 'E106', 'E108', 'E109', 'E110', 'E111'
        , 'E116'
        , 'E118'
        , 'E119'
        , 'E120'
        , 'E121'
        , 'E126'
        , 'E128'
        , 'E129'
        , 'E130'
        , 'E131'
        , 'E136'
        , 'E138'
        , 'E139'
        , 'E140'
        , 'E141', 'E146', 'E148', 'E149')
    THEN 1
    ELSE 0 END) AS diabetes_without_cc

-- Diabetes with chronic complication
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 4) IN ('2504', '2505', '2506', '2507')
    OR
    SUBSTR(
      icd10_code, 1, 4
    ) IN ('E102', 'E103', 'E104', 'E105', 'E107', 'E112', 'E113'
        , 'E114'
        , 'E115'
        , 'E117'
        , 'E122'
        , 'E123'
        , 'E124'
        , 'E125'
        , 'E127'
        , 'E132'
        , 'E133'
        , 'E134'
        , 'E135'
        , 'E137'
        , 'E142'
        , 'E143', 'E144', 'E145', 'E147')
    THEN 1
    ELSE 0 END) AS diabetes_with_cc

-- Hemiplegia or paraplegia
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) IN ('342', '343')
    OR
    SUBSTR(d.icd9_code, 1, 4) IN ('3341', '3440', '3441', '3442'
```

```
                        , '3443', '3444', '3445', '3446', '3449'
    )
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('G81', 'G82')
    OR
    SUBSTR(d.icd10_code, 1, 4) IN ('G041', 'G114', 'G801', 'G802', 'G830'
                        , 'G831'
                        , 'G832'
                        , 'G833'
                        , 'G834'
                        , 'G839'
    )
    THEN 1
    ELSE 0 END) AS paraplegia

-- Renal disease
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) IN ('582', '585', '586', 'V56')
    OR
    SUBSTR(d.icd9_code, 1, 4) IN ('5880', 'V420', 'V451')
    OR
    SUBSTR(d.icd9_code, 1, 4) BETWEEN '5830' AND '5837'
    OR
    SUBSTR(
       icd9_code, 1, 5
    ) IN (
       '40301'
       , '40311'
       , '40391'
       , '40402'
       , '40403'
       , '40412'
       , '40413'
       , '40492'
       , '40493'
    )
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('N18', 'N19')
    OR
    SUBSTR(d.icd10_code, 1, 4) IN ('I120', 'I131', 'N032', 'N033', 'N034'
                        , 'N035'
                        , 'N036'
                        , 'N037'
                        , 'N052'
                        , 'N053'
                        , 'N054'
                        , 'N055'
                        , 'N056'
                        , 'N057'
                        , 'N250'
                        , 'Z490'
                        , 'Z491'
                        , 'Z492'
                        , 'Z940'
                        , 'Z992'
```

```
    )
    THEN 1 ELSE 0 END) AS renal_disease

-- Any malignancy, including lymphoma and leukemia,
-- except malignant neoplasm of skin.
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) BETWEEN '140' AND '172'
    OR
    SUBSTR(d.icd9_code, 1, 4) BETWEEN '1740' AND '1958'
    OR
    SUBSTR(d.icd9_code, 1, 3) BETWEEN '200' AND '208'
    OR
    SUBSTR(d.icd9_code, 1, 4) = '2386'
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('C43', 'C88')
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C00' AND 'C26'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C30' AND 'C34'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C37' AND 'C41'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C45' AND 'C58'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C60' AND 'C76'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C81' AND 'C85'
    OR
    SUBSTR(d.icd10_code, 1, 3) BETWEEN 'C90' AND 'C97'
    THEN 1
    ELSE 0 END) AS malignant_cancer

-- Moderate or severe liver disease
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 4) IN ('4560', '4561', '4562')
    OR
    SUBSTR(d.icd9_code, 1, 4) BETWEEN '5722' AND '5728'
    OR
    SUBSTR(
       icd10_code, 1, 4
    ) IN ('I850', 'I859', 'I864', 'I982', 'K704', 'K711'
         , 'K721', 'K729', 'K765', 'K766', 'K767')
    THEN 1
    ELSE 0 END) AS severe_liver_disease

-- Metastatic solid tumor
, MAX(CASE WHEN
    SUBSTR(d.icd9_code, 1, 3) IN ('196', '197', '198', '199')
    OR
    SUBSTR(d.icd10_code, 1, 3) IN ('C77', 'C78', 'C79', 'C80')
    THEN 1
    ELSE 0 END) AS metastatic_solid_tumor

-- AIDS/HIV
, MAX(CASE WHEN
```

```
        SUBSTR(d.icd9_code, 1, 3) IN ('042', '043', '044')
        OR
        SUBSTR(d.icd10_code, 1, 3) IN ('B20', 'B21', 'B22', 'B24')
        THEN 1
        ELSE 0 END) AS aids

    -- Binary complication flag (1 if any matched diagnosis)
    , MAX(CASE
        WHEN SUBSTR(d.icd9_code, 1, 3) IN ('410', '428', '440', '441', '430', '490', '290', '531', '250')
        OR SUBSTR(d.icd10_code, 1, 3) IN ('I21', 'I50', 'I70', 'I71', 'G45', 'J18', 'F00', 'K25', 'E10')
        THEN 1 ELSE 0 END) AS complication_binary

    -- Complication Severity Index (Weighted Score)
    , SUM(CASE
        WHEN
          SUBSTR(d.icd9_code, 1, 3) IN ('410', '428')
          OR SUBSTR(d.icd10_code, 1, 3) IN ('I21', 'I50') THEN 5  -- Severe: Heart Attack, CHF
        WHEN
          SUBSTR(d.icd9_code, 1, 3) IN ('440', '441', '430', '490')
          OR SUBSTR(d.icd10_code, 1, 3) IN ('I70', 'I71', 'G45', 'J18') THEN 3  -- Moderate: Stroke,
Pneumonia
        WHEN
          SUBSTR(d.icd9_code, 1, 3) IN ('290', '531', '250')
          OR SUBSTR(d.icd10_code, 1, 3) IN ('F00', 'K25', 'E10') THEN 2  -- Mild: Dementia, Ulcers,
Diabetes
        ELSE 0 END) AS complication_index

    -- CMS-Tracked Conditions (AMI, HF, Pneumonia, COPD, Stroke, Sepsis)
    , MAX(CASE WHEN
        -- Acute Myocardial Infarction (AMI)
        SUBSTR(d.icd9_code, 1, 3) IN ('410') OR SUBSTR(d.icd10_code, 1, 3) IN ('I21', 'I22')

        -- Heart Failure (HF)
        OR SUBSTR(d.icd9_code, 1, 3) IN ('428') OR SUBSTR(d.icd10_code, 1, 3) IN ('I50')

        -- Pneumonia (PN)
        OR SUBSTR(d.icd9_code, 1, 3) IN ('486') OR SUBSTR(d.icd10_code, 1, 3) IN ('J18')

        -- Chronic Obstructive Pulmonary Disease (COPD)
        OR SUBSTR(d.icd9_code, 1, 3) IN ('491', '492', '496') OR SUBSTR(d.icd10_code, 1, 3) IN ('J44')

        -- Stroke
        OR SUBSTR(d.icd9_code, 1, 3) IN ('434', '436') OR SUBSTR(d.icd10_code, 1, 3) IN ('I63', 'I64')

        -- Sepsis (NEW in 2024 CMS Measure)
        OR SUBSTR(d.icd9_code, 1, 3) IN ('995', '785') OR SUBSTR(d.icd10_code, 1, 3) IN ('A41',
'R65')
      THEN 1 ELSE 0 END) AS cms_flag
    FROM diag AS d
  GROUP BY d.hadm_id
),


admissions_features AS (
  SELECT
```

```
    a.subject_id,
    a.hadm_id,
    a.admittime,
    a.dischtime,
    a.race,
    a.admission_location,

    -- Calculate LOS
    TIMESTAMP_DIFF(a.dischtime, a.admittime, DAY) AS length_of_stay, -- Calculate LOS

    -- Capture next admission time for readmission calculation
      -- for 30-day readmission because a patient can be admitted multiple times partitioning ensures
      -- that each readmission is checked against the prior discharge event to determine if it happened
within 30 days

    LEAD(a.admittime) OVER (PARTITION BY a.subject_id ORDER BY a.admittime ASC) AS
next_admittime,

    -- Define 30-day readmission based on next_admittime
    CASE
      WHEN TIMESTAMP_DIFF(
        LEAD(a.admittime) OVER (PARTITION BY a.subject_id ORDER BY a.admittime ASC),
        a.dischtime,
        DAY
      ) <= 30 THEN 1
      ELSE 0
    END AS readmitted,

      --## Readmission:
      -- https://www.cms.gov/medicare/quality/value-based-programs/hospital-readmissions
      -- https://www.cms.gov/Medicare/Medicare-Fee-for-Service-
Payment/PhysicianFeedbackProgram/Downloads/2016-ACR-MIF.pdf
      -- https://www.sqlservercentral.com/forums/topic/readmission-within-30-days-from-a-discharge

    -- Race Binning (Moved from patient_features to admission_features)
    CASE
      WHEN LOWER(a.race) LIKE 'white%' THEN 'White'
      WHEN LOWER(a.race) LIKE 'black%' THEN 'Black'
      WHEN LOWER(a.race) LIKE 'asian%' THEN 'Asian'
      WHEN LOWER(a.race) LIKE '%hispanic%' OR LOWER(a.race) LIKE '%latino%' THEN
'Hispanic/Latino'
      WHEN LOWER(a.race) LIKE 'american indian%' THEN 'Native American'
      WHEN LOWER(a.race) LIKE 'native hawaiian%' THEN 'Pacific Islander'
      WHEN LOWER(a.race) IN ('unknown', 'patient declined to answer', 'unable to obtain') THEN
'Unknown/Declined'
      ELSE 'Other'
    END AS race_group,

    -- Binary column for race
    CASE
      WHEN LOWER(a.race) LIKE '%white%' THEN 1
      ELSE 0
    END AS is_white,

  FROM all_admissions AS a
```

```
),

prior_visits AS (
  -- Compute prior visits for each subject in the last 1 year
  SELECT
    a.subject_id,
    a.hadm_id,
    COUNT(CASE
      WHEN TIMESTAMP_DIFF(a.admittime, prev.admittime, DAY) <= 365
      THEN 1 ELSE NULL
    END) AS prior_hospital_visits_1yr,
    COUNT(CASE
      WHEN TIMESTAMP_DIFF(a.admittime, prev.admittime, DAY) <= 365
      AND prev.admission_location = 'EMERGENCY ROOM'
      THEN 1 ELSE NULL
    END) AS prior_er_visits_1yr
  FROM all_admissions AS a
  LEFT JOIN all_admissions AS prev
    ON a.subject_id = prev.subject_id
    AND prev.admittime < a.admittime  -- Only count **prior** admissions
  GROUP BY a.subject_id, a.hadm_id
),

patient_features AS (
  SELECT
    p.subject_id,
    p.dod,
    p.anchor_age,
    p.anchor_year,
    CASE
      WHEN p.gender = 'M' THEN 'Male'
      WHEN p.gender = 'F' THEN 'Female'
      ELSE 'Unknown' -- In case of unexpected values
    END AS gender
  FROM `physionet-data.mimiciv_3_1_hosp.patients` AS p
),


icu_features AS (
  -- ICU Admission Features Aggregated per Hospital Admission
  SELECT
    i.subject_id,
    i.hadm_id,

    -- First ICU admission time within this hospital stay
    MIN(i.intime) AS first_icu_admit,

    -- Last ICU discharge time within this hospital stay
    MAX(i.outtime) AS last_icu_discharge,

    -- Total number of ICU stays per hospital admission
    COUNT(*) AS total_icu_stays,

    -- Total hours spent in ICU for this admission
    SUM(TIMESTAMP_DIFF(i.outtime, i.intime, HOUR)) AS total_icu_hours,
```

```
      -- ICU flag: 1 if the hospital admission includes ICU stay, 0 otherwise
      CASE WHEN COUNT(*) > 0 THEN 1 ELSE 0 END AS ICU_flag

   FROM `physionet-data.mimiciv_3_1_icu.icustays` AS i
   GROUP BY i.subject_id, i.hadm_id
),

icu_filtered AS (
   -- Keep only first ICU stay per hospital admission
   SELECT *
   FROM `msds-795.mimiciv_derived.icustay_detail`
   WHERE first_icu_stay = TRUE
),


sapsii_max AS (
   SELECT hadm_id, MAX(sapsii) AS max_sapsii
   FROM `msds-795.mimiciv_derived.sapsii`
   GROUP BY hadm_id
),

sirs_max AS (
   SELECT hadm_id, MAX(sirs) AS max_sirs
   FROM `msds-795.mimiciv_derived.sirs`
   GROUP BY hadm_id
),

--sofa_max AS (
--    SELECT hadm_id, MAX(sofa_24hours) AS max_sofa_24hours
--    FROM `msds-795.mimiciv_derived.sofa`
--    GROUP BY hadm_id
--),

oasis_max AS (
   SELECT hadm_id, MAX(oasis) AS max_oasis
   FROM `msds-795.mimiciv_derived.oasis`
   GROUP BY hadm_id
),

lods_max AS (
   SELECT hadm_id, MAX(lods) AS max_lods
   FROM `msds-795.mimiciv_derived.lods`
   GROUP BY hadm_id
),

apsiii_max AS (
   SELECT hadm_id, MAX(apsiii) AS max_apsiii
   FROM `msds-795.mimiciv_derived.apsiii`
   GROUP BY hadm_id
),

procedures_features AS (
   SELECT
      proc.subject_id,
```

```
    proc.hadm_id,

    ----- Predicitve Procedures

    -- Count total procedures per hospital admission
    COUNT(*) AS num_procedures,

    -- Major Surgery Flag
    MAX(CASE WHEN proc.icd_code LIKE '0J%' THEN 1 ELSE 0 END) AS major_surgery_flag,

    -- Cardiac Surgery Flag
    MAX(CASE WHEN proc.icd_code IN ('0210093', '027034Z', '02RF07Z') THEN 1 ELSE 0 END) AS
cardiac_surgery_flag,

    -- Orthopedic Surgery (Joint Replacements)
    MAX(CASE WHEN proc.icd_code LIKE '0SR%' THEN 1 ELSE 0 END) AS
orthopedic_surgery_flag,

    -- Neurosurgery (Brain/Spinal Surgery)
    MAX(CASE WHEN proc.icd_code IN ('0016070', '00H00MZ') THEN 1 ELSE 0 END) AS
neurosurgery_flag,

    -- Sepsis/Infection Procedure Flag
    MAX(CASE WHEN proc.icd_code IN ('6A550Z3', '5A1955Z') THEN 1 ELSE 0 END) AS
sepsis_infection_flag,

    -- Dialysis Flag (Renal Failure Indicator)
    MAX(CASE WHEN proc.icd_code IN ('5A1D00Z', '5A1D60Z') THEN 1 ELSE 0 END) AS
dialysis_flag,

    -- Blood Transfusion Flag (Hemorrhage Risk)
    MAX(CASE WHEN proc.icd_code IN ('30233N1', '30233P1') THEN 1 ELSE 0 END) AS
blood_transfusion_flag,

    -- Emergency Surgery (High Severity)
    MAX(CASE WHEN proc.icd_code LIKE '0WQF%' THEN 1 ELSE 0 END) AS
emergency_surgery_flag,

    -- Stroke Prevention (Carotid Endarterectomy)
    MAX(CASE WHEN proc.icd_code = '03CL0ZZ' THEN 1 ELSE 0 END) AS stroke_prevention_flag,

    -- Bowel Resection (Sepsis & Readmission Risk)
    MAX(CASE WHEN proc.icd_code IN ('0DBE0ZZ', '0DBF0ZZ') THEN 1 ELSE 0 END) AS
bowel_resection_flag,

    -- Colostomy/Ileostomy (GI Complications)
    MAX(CASE WHEN proc.icd_code IN ('0D1B0Z4', '0D1D0Z4') THEN 1 ELSE 0 END) AS
colostomy_ileostomy_flag,

    -- Gastrostomy Tube (PEG Tube, Malnutrition Risk)
    MAX(CASE WHEN proc.icd_code = '0DH63UZ' THEN 1 ELSE 0 END) AS gastrostomy_tube_flag,

    -- Central Line Placement (Sepsis Risk)
    MAX(CASE WHEN proc.icd_code IN ('02HV33Z', '02HV34Z') THEN 1 ELSE 0 END) AS
central_line_flag,
```

```
    -- Wound Debridement (Infection Risk)
    MAX(CASE WHEN proc.icd_code = '0JDJ0ZZ' THEN 1 ELSE 0 END) AS
wound_debridement_flag,

    ----- Late Stage features

    -- Respiratory Support (Ventilation, Intubation)
    MAX(CASE WHEN proc.icd_code IN ('5A09357', '0BH17EZ', '0BH18EZ') THEN 1 ELSE 0 END)
AS respiratory_support_flag,

    -- ECMO Use (Extracorporeal Life Support)
    MAX(CASE WHEN proc.icd_code IN ('5A1522F', '5A1522G') THEN 1 ELSE 0 END) AS
ecmo_flag,

    -- Prolonged Ventilation > 96 Hours
    MAX(CASE WHEN proc.icd_code = '5A0945Z' THEN 1 ELSE 0 END) AS
prolonged_ventilation_flag,

    -- Tracheostomy (Long-Term Ventilation)
    MAX(CASE WHEN proc.icd_code IN ('0B110F4', '0B110Z4') THEN 1 ELSE 0 END) AS
tracheostomy_flag,

    -- Shock Management (Vasopressors, Resuscitation)
    MAX(CASE WHEN proc.icd_code IN ('3E013GC', '3E013GQ') THEN 1 ELSE 0 END) AS
shock_management_flag,

    -- CPR / Cardiac Arrest (Too Late to Predict)
    MAX(CASE WHEN proc.icd_code IN ('5A12012', '5A12013') THEN 1 ELSE 0 END) AS cpr_flag


  FROM `physionet-data.mimiciv_3_1_hosp.procedures_icd` AS proc
  GROUP BY proc.subject_id, proc.hadm_id
),

medications_features AS (
  SELECT
    med.subject_id,
    med.hadm_id,

    ----- Predicitve features:

    -- Count distinct medications per admission
    COUNT(DISTINCT med.drug) AS num_medications,

    -- Broad-Spectrum Antibiotics (Sepsis Risk)
    MAX(CASE WHEN med.drug IN ('Meropenem', 'Piperacillin-Tazobactam', 'Vancomycin',
'Cefepime') THEN 1 ELSE 0 END) AS broad_spectrum_antibiotic_med_flag,

    -- Antifungal Medications (Immunocompromised, High Mortality)
    MAX(CASE WHEN med.drug IN ('Amphotericin B', 'Fluconazole') THEN 1 ELSE 0 END) AS
antifungal_med_flag,

    -- Inotropes (Heart Failure & Shock)
```

```
    MAX(CASE WHEN med.drug IN ('Dobutamine', 'Milrinone') THEN 1 ELSE 0 END) AS
inotrope_med_flag,

    -- Antiarrhythmics (Cardiac Arrest & Stroke Risk)
    MAX(CASE WHEN med.drug = 'Amiodarone' THEN 1 ELSE 0 END) AS antiarrhythmic_med_flag,

    -- Nephrotoxic Medications (Acute Kidney Injury & Dialysis Risk)
    MAX(CASE WHEN med.drug IN ('Vancomycin', 'Gentamicin', 'Tobramycin') THEN 1 ELSE 0 END)
AS nephrotoxic_med_flag,

    -- Diuretics (Heart & Kidney Failure Risk)
    MAX(CASE WHEN med.drug IN ('Furosemide', 'Bumetanide') THEN 1 ELSE 0 END) AS
diuretic_med_flag,

    -- Corticosteroids (Sepsis & Inflammation)
    MAX(CASE WHEN med.drug IN ('Methylprednisolone', 'Dexamethasone', 'Hydrocortisone') THEN 1
ELSE 0 END) AS corticosteroid_med_flag,

    -- Sedative Use (Ventilation, ICU Risk)
    MAX(CASE WHEN med.drug IN ('Propofol', 'Midazolam', 'Dexmedetomidine') THEN 1 ELSE 0
END) AS sedative_med_flag,

    -- Pressor Use (Shock/ICU Support)
    MAX(CASE WHEN med.drug IN ('Norepinephrine', 'Vasopressin') THEN 1 ELSE 0 END) AS
pressor_med_flag,

  FROM `physionet-data.mimiciv_3_1_hosp.prescriptions` AS med
  GROUP BY med.subject_id, med.hadm_id
),

ed_features AS (
  SELECT
    es.subject_id,
    es.hadm_id,
    MIN(es.intime) AS ed_admit_time,
    MAX(es.outtime) AS ed_discharge_time,

    -- Count total ED visits for the patient
    COUNT(*) AS total_ed_visits,

    -- Highest Triage Acuity for This Hospital Stay
    MAX(et.acuity) AS ed_acuity,

    -- Most Frequent Chief Complaint using ARRAY_AGG
    ARRAY_AGG(et.chiefcomplaint ORDER BY et.acuity DESC LIMIT 1)[OFFSET(0)] AS
ed_chief_complaint

  FROM `physionet-data.mimiciv_ed.edstays` AS es
  LEFT JOIN `physionet-data.mimiciv_ed.triage` AS et
    ON es.stay_id = et.stay_id
  GROUP BY es.subject_id, es.hadm_id
)

SELECT
  sa.subject_id
```

```
    , sa.hospital_expire_flag
    , sa.hadm_id
    , sa.marital_status AS marital_status_category
    , sa.insurance AS insurance_category
    , sa.admission_location AS admission_location_category
    , sa.discharge_location AS discharge_location_category
    , sa.admittime
    , sa.dischtime
    , sa.race

    -- Addmisions Features
    , add.length_of_stay
    , add.next_admittime
    , add.readmitted
    , add.is_white
    , add.race_group

    , prior.prior_hospital_visits_1yr
    , prior.prior_er_visits_1yr

    -- Patient Features
    , pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0, 0), YEAR)
AS age
    , CASE
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 18 THEN -1  -- Pediatric Group
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 30 THEN 0
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 40 THEN 1
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 50 THEN 2
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 60 THEN 3
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 70 THEN 4
        WHEN (pat.anchor_age + DATETIME_DIFF(sa.admittime, DATETIME(pat.anchor_year, 1, 1, 0, 0,
0), YEAR)) < 80 THEN 5
        ELSE 6
      END AS age_category
    , pat.gender
    , pat.dod

    -- ICU Features
    , icuf.first_icu_admit
    , icuf.last_icu_discharge
    , icuf.total_icu_stays
    , icuf.total_icu_hours
    , icuf.ICU_flag
    , icu.los_icu

    -- Procedures Features
    , proc.num_procedures
    , proc.major_surgery_flag
    , proc.cardiac_surgery_flag
```

```
, proc.orthopedic_surgery_flag
, proc.respiratory_support_flag
, proc.neurosurgery_flag
, proc.sepsis_infection_flag
, proc.dialysis_flag
, proc.blood_transfusion_flag
, proc.ecmo_flag
, proc.emergency_surgery_flag
, proc.prolonged_ventilation_flag
, proc.tracheostomy_flag
, proc.shock_management_flag
, proc.cpr_flag
, proc.stroke_prevention_flag
, proc.bowel_resection_flag
, proc.colostomy_ileostomy_flag
, proc.gastrostomy_tube_flag
, proc.central_line_flag
, proc.wound_debridement_flag

-- Medications Features (Predictive)
, med.num_medications
, med.pressor_med_flag
, med.broad_spectrum_antibiotic_med_flag
, med.antifungal_med_flag
, med.inotrope_med_flag
, med.antiarrhythmic_med_flag
, med.nephrotoxic_med_flag
, med.diuretic_med_flag
, med.corticosteroid_med_flag
, med.sedative_med_flag

-- ED Features
, ed.ed_acuity
, ed.ed_chief_complaint
, ed.ed_admit_time
, ed.ed_discharge_time

-- Comorbidities & Complications
, diag.complication_index
, diag.complication_binary
, diag.myocardial_infarct
, diag.congestive_heart_failure
, diag.peripheral_vascular_disease
, diag.cerebrovascular_disease
, diag.dementia
, diag.chronic_pulmonary_disease
, diag.rheumatic_disease
, diag.peptic_ulcer_disease
, diag.mild_liver_disease
, diag.diabetes_without_cc
, diag.diabetes_with_cc
, diag.paraplegia
, diag.renal_disease
, diag.malignant_cancer
, diag.severe_liver_disease
```

```
    , diag.metastatic_solid_tumor
    , diag.aids
    , diag.cms_flag

    -- Calculate the Charlson Comorbidity Score using the original
    -- weights from Charlson, 1987.
    , + diag.myocardial_infarct + diag.congestive_heart_failure
    + diag.peripheral_vascular_disease + diag.cerebrovascular_disease
    + diag.dementia + diag.chronic_pulmonary_disease
    + diag.rheumatic_disease + diag.peptic_ulcer_disease
    + GREATEST(diag.mild_liver_disease, 3 * diag.severe_liver_disease)
    + GREATEST(2 * diag.diabetes_with_cc, diag.diabetes_without_cc)
    + GREATEST(2 * diag.malignant_cancer, 6 * diag.metastatic_solid_tumor)
    + 2 * diag.paraplegia + 2 * diag.renal_disease
    + 6 * diag.aids AS charlson_comorbidity_index

    -- 30-Day Mortality Indicator
    , CASE
        WHEN pat.dod IS NOT NULL
            AND pat.dod BETWEEN sa.admittime AND DATE_ADD(sa.admittime, INTERVAL 30 DAY)
        THEN 1 ELSE 0
      END AS mortality_30_day

    , sapsii.max_sapsii
    , sirs.max_sirs
    --, sofa.max_sofa_24hours
    , oasis.max_oasis
    , lods.max_lods
    , apsiii.max_apsiii


FROM sampled_admissions AS sa
LEFT JOIN admissions_features AS add ON sa.subject_id = add.subject_id AND sa.hadm_id =
add.hadm_id
LEFT JOIN prior_visits AS prior ON sa.subject_id = prior.subject_id AND sa.hadm_id = prior.hadm_id
LEFT JOIN diagnoses_features AS diag ON sa.hadm_id = diag.hadm_id
LEFT JOIN patient_features AS pat ON sa.subject_id = pat.subject_id

LEFT JOIN procedures_features AS proc ON sa.hadm_id = proc.hadm_id
LEFT JOIN medications_features AS med ON sa.hadm_id = med.hadm_id
LEFT JOIN ed_features AS ed ON sa.hadm_id = ed.hadm_id

LEFT JOIN icu_features AS icuf ON sa.hadm_id = icuf.hadm_id
LEFT JOIN icu_filtered icu ON sa.hadm_id = icu.hadm_id
LEFT JOIN sapsii_max sapsii ON sa.hadm_id = sapsii.hadm_id
LEFT JOIN sirs_max sirs ON sa.hadm_id = sirs.hadm_id
-- LEFT JOIN sofa_max sofa ON sa.hadm_id = sofa.hadm_id
LEFT JOIN oasis_max oasis ON sa.hadm_id = oasis.hadm_id
LEFT JOIN lods_max lods ON sa.hadm_id = lods.hadm_id
LEFT JOIN apsiii_max apsiii ON sa.hadm_id = apsiii.hadm_id;
"
)

if (!exists("base_features")) {
  source(here("scripts", "feature_sets.R"))
```

```
}

# Run the query
data_hosp <- dbGetQuery(con_hosp, sql_query_admin_samp)

# Load IPUMS Data
library(here)
ipums_data <- read.csv(here("data", "processed", "IPUMS_data.csv"), stringsAsFactors = FALSE)

# Count rows before merge
message("Rows before merge: ", nrow(data_hosp))

# Perform left join on matching keys
data_hosp <- merge(data_hosp, ipums_data,
                   by = c("race_group", "insurance_category", "marital_status_category"),
                   all.x = TRUE)

# Verify count are the  same
message("Rows after merge: ", nrow(data_hosp))

### Data Cleaning

## Ensure column names are valid in R
names(data_hosp) <- make.names(names(data_hosp))

# Convert weighted features to numeric and replace NA with 0
data_hosp[weighted_features] <- lapply(data_hosp[weighted_features], function(x) {
  x <- as.numeric(x)  # Ensure numeric format
  x[is.na(x)] <- 0    # Replace missing values with 0
  return(x)
})

data_hosp[binary_features] <- lapply(data_hosp[binary_features], function(x) {
  x[is.na(x)] <- 0
  return(as.numeric(as.character(x)))  # Convert to numeric (0/1) for ML models
})

# # Convert Target Variables to Factors with Proper Labels
data_hosp[target_vars] <- lapply(data_hosp[target_vars], function(x) {
  factor(x, levels = c(0, 1), labels = c("No", "Yes"))  # Convert to "No"/"Yes"
})

data_hosp[continuous_features] <- lapply(data_hosp[continuous_features], function(x) {
  x <- as.numeric(x)  # Explicitly convert to numeric
  x[is.na(x)] <- 0    # Replace missing continuous values with 0
  return(x)
})

# # Categorical Variables - Replace NA with "Unknown" and Convert to Factor
data_hosp[categorical_vars] <- lapply(data_hosp[categorical_vars], function(x) {
  x <- as.character(x)
  x[is.na(x)] <- "Unknown"
  return(as.factor(x))  # Convert to factor
})
```

```
# Convert Categorical Variables to Numeric Encoding for Modeling
data_hosp$marital_status_numeric <- as.numeric(as.factor(data_hosp$marital_status_category))
data_hosp$insurance_numeric <- as.numeric(as.factor(data_hosp$insurance_category))
data_hosp$admission_location_numeric <- as.numeric(as.factor(data_hosp$admission_location_category))
data_hosp$discharge_location_numeric <- as.numeric(as.factor(data_hosp$discharge_location_category))

# ICU Features - Replace NA with Placeholder Dates
data_hosp[icu_date_features] <- lapply(data_hosp[icu_date_features], function(x) {
  x[is.na(x)] <- as.Date("1900-01-01")
  return(x)
})

# Emergency Department Features - Handle Missing Values
data_hosp$ed_chief_complaint[is.na(data_hosp$ed_chief_complaint)] <- "Unknown"
data_hosp$ed_admit_time[is.na(data_hosp$ed_admit_time)] <- as.Date("1900-01-01")
data_hosp$ed_discharge_time[is.na(data_hosp$ed_discharge_time)] <- as.Date("1900-01-01")

# Exclusion criteria: Remove patients with age_category == 0
data_hosp <- data_hosp[order(data_hosp$admittime), ]

# Remove all patients with age_category == 0 (ALl Targets)
data_hosp <- data_hosp[data_hosp$age_category != 0, ]
data_hosp <- data_hosp[order(data_hosp$admittime), ]

# # Remove fully correlated features for mortality target
# data_hosp_mort <- data_hosp[!(data_hosp$hospital_expire_flag == 1), ]
# data_hosp_mort <- data_hosp[!(data_hosp$discharge_location %in% c("HOSPICE", "DIED")), ]

data_ICU <- data_hosp[data_hosp$ICU_flag == 1 & data_hosp$total_icu_stays > 0, ]
data_cms <- data_hosp[data_hosp$cms_flag == 1, ]

suffix <- paste0("_", sample_percent)
assign(paste0("data_hosp", suffix), data_hosp, envir = .GlobalEnv)
assign(paste0("data_ICU", suffix), data_ICU, envir = .GlobalEnv)
assign(paste0("data_cms", suffix), data_cms, envir = .GlobalEnv)

# Save to RDS files with dynamic names
saveRDS(data_hosp, here("data", "processed", paste0("data_hosp", suffix, ".rds")))
saveRDS(data_ICU, here("data", "processed", paste0("data_ICU", suffix, ".rds")))
saveRDS(data_cms, here("data", "processed", paste0("data_cms", suffix, ".rds")))

message("Processed data saved as: data_hosp", suffix, ".rds, data_ICU", suffix, ".rds, data_cms", suffix,
".rds")
```

figs_funcs.R

```
################################ Global Vars
################################################

# ---- Plot Styling ----

library(ggplot2)
```

```
theme_set(
  theme_minimal(base_size = 12) +
    theme(
      plot.title = element_text(size = 14),
      plot.background = element_rect(fill = "white", color = NA),
      panel.background = element_rect(fill = "white", color = NA),
      panel.grid = element_line(color = "grey90")
    )
)


style_kable_default <- function(kable_obj, df = NULL) {
  # Guard clause: skip if df is missing or not a data frame
  if (is.null(df) || !is.data.frame(df)) {
    warning("No valid df provided; skipping row_spec.")
    return(
      kable_obj %>%
        kableExtra::kable_styling(
          full_width = FALSE,
          bootstrap_options = c("hover", "condensed", "responsive"),
          font_size = 13,
          position = "center"
        )
    )

  }
    kable_obj %>%
    kableExtra::kable_styling(
      full_width = FALSE,
      bootstrap_options = c("hover", "condensed", "responsive"),
      font_size = 13,
      position = "center"
    ) %>%
    kableExtra::row_spec(0:nrow(df), extra_css = "color: black;")
}

################################ FIGURES ###############################################

#---- Figures

library(ggplot2)
library(gridExtra)
library(dplyr)

numeric_targets <- c("length_of_stay")

# Generate histograms for each numeric target
hist_list <- lapply(numeric_targets, function(target) {
  ggplot(data_hosp, aes(x = .data[[target]])) +
    geom_histogram(color = "black", fill = "darkgreen", bins = 100, alpha = 0.7) +
    xlab(target) +
    ylab("Frequency") +
    theme_light() +
    ggtitle(paste("Distribution of", target)) +
    theme(plot.title = element_text(size = 8, face = "bold"),
```

```
        axis.title.x = element_blank())
})

binary_targets <- c("hospital_expire_flag", "readmitted", "mortality_30_day")

# Generate bar charts for each binary target
bar_list <- lapply(binary_targets, function(target) {
  ggplot(data_hosp, aes(x = factor(.data[[target]]))) +
    geom_bar(color = "black", fill = "darkgreen", alpha = 0.7) +
    xlab(target) +
    ylab("Count") +
    theme_light() +
    ggtitle(paste(target)) +
    theme(plot.title = element_text(size = 8, face = "bold"),
          axis.title.x = element_blank())
})

# Create summary table
binary_summary_table <- bind_rows(
  data.frame(Target = "hospital_expire_flag", table(data_hosp$hospital_expire_flag)) %>%
mutate(Proportion = Freq / sum(Freq)),
  data.frame(Target = "readmitted", table(data_hosp$readmitted)) %>% mutate(Proportion = Freq /
sum(Freq)),
  data.frame(Target = "mortality_30_day", table(data_hosp$mortality_30_day)) %>% mutate(Proportion =
Freq / sum(Freq))
)

caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  "Binary Target Summary Table",
  "</div>"
)

colnames(binary_summary_table) <- c("Target", "Category", "Count", "Proportion")
binary_df <- binary_summary_table

# Step 2: Render as styled kable object
binary_summary_table <- binary_df %>%
  kable(format = "html", caption = caption_html) %>%
  style_kable_default(binary_df)

# Save plots and summary table as a list to be called from Rmd
plots <- list(hist_list = hist_list, bar_list = bar_list, summary_table = binary_summary_table)

#------------------------------------------------------------------------

#---- Figure #
readmitted_patients <- data_hosp %>%
  filter(readmitted == "Yes") %>%  # Filter only patients with at least one readmission
  distinct(subject_id)  # Get unique patient IDs

set.seed(42)  # For reproducibility
example_patient <- sample(readmitted_patients$subject_id, 1)

patient_history <- data_hosp %>%
```

```
  filter(subject_id == example_patient) %>%  # Pull all records for this patient
  arrange(admittime)  # Order by admission time to check sequence


admission_counts <- data_hosp %>%
  group_by(subject_id) %>%
  summarize(num_admissions = n())

# Define bins for admissions
admission_counts <- admission_counts %>%
  mutate(binned_admissions = cut(num_admissions,
                                 breaks = c(1, 2, 4, 6, 10, 15, 20, Inf),  # Bins
                                 labels = c("1","2-3","4-5","6-9","10-15", "15-20","20+"),
                                 include.lowest = TRUE))

# Plot histogram of the number of admissions per patient with binning
figure_4_4_7 <- ggplot(admission_counts, aes(x = binned_admissions)) +
  geom_bar(color = "black", fill = "darkgreen", alpha = 0.8) +
  labs(title = "Number of Admissions Per Patient",
       x = "Admissions Count",
       y = "Patients Count")


#------------------------------------------------------------------------



################################ FUNCTIONS
#################################################


#################
###   EDA     ###
#################

library(ggplot2)
library(dplyr)
library(gridExtra)

# Step 1: Readmitted Patients
readmitted_flags <- data_hosp %>%
  group_by(subject_id) %>%
  summarize(any_readmission = ifelse(any(readmitted == "Yes"), "Yes", "No")) %>%
  mutate(any_readmission = factor(any_readmission, levels = c("No", "Yes")))  # Ensure factor levels
match scale_fill_manual()

# Step 2: One Admission Per Patient
cleaned_patient_data <- data_hosp %>%
  left_join(readmitted_flags, by = "subject_id") %>%  # Merge readmission flag
  group_by(subject_id) %>%
  filter(ifelse(any_readmission == "Yes", readmitted == "Yes", row_number() == 1)) %>%  # Convert
factor to logical
  ungroup()

plot_readmission_by_category <- function(dataset, category_var) {
```

```
  ggplot(dataset, aes(x = .data[[category_var]], fill = any_readmission)) +
    geom_bar(color = "black", alpha = 0.8,linewidth = 0.3) +
    labs(title = paste(category_var),
         x = NULL,
         y = "Patients") +  # Removed legend label
    scale_fill_manual(values = c("No" = "lightblue", "Yes" = "red"),
                      drop = FALSE) +  # Ensures all levels appear even if missing
    theme(
      plot.background = element_rect(fill = "white", color = NA),
      panel.background = element_rect(fill = "white", color = NA),
      plot.title = element_text(size = 14),
      axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
      axis.text.y = element_text(size = 10),
      legend.position = "none"
    )
}

#-------------------------------------------------------------------------

plot_feature_distribution <- function(df, column, target) {
  # Check if column exists
  if (!(column %in% colnames(df))) {
    stop(paste("Column", column, "not found in the dataset"))
  }

  # Ensure target variable exists
  if (!(target %in% colnames(df))) {
    stop(paste("Target variable", target, "not found in the dataset"))
  }

  # Convert target to factor if not already
  df[[target]] <- as.factor(df[[target]])

  # Convert binary features to factor with Yes/No labels
  unique_vals <- unique(df[[column]])
  if (length(unique_vals) == 2 && all(unique_vals %in% c(0, 1))) {
    df[[column]] <- factor(df[[column]], levels = c(0, 1), labels = c("No", "Yes"))
  }

  # Remove 0 values for visualization (but keep in dataset)
  filtered_data <- df[df[[column]] != 0, , drop = FALSE]

  if (is.numeric(df[[column]])) {
    # Stacked density plot based on target variable
    p <- ggplot(filtered_data, aes(x = .data[[column]], fill = .data[[target]])) +
      geom_density(alpha = 0.8, position = "stack", color = "black", linewidth = 0.3) +  # Stacked density
      labs(title = paste("Stacked Density of", column, "by", target),
           x = column, y = "Density", fill = target)

  } else {
    # Categorical features: Stacked bar chart
    p <- ggplot(df, aes(x = .data[[column]], fill = .data[[target]])) +
      geom_bar(position = "fill", alpha = 0.8, color = "black", linewidth = 0.3) +  # Add outlines
      labs(title = paste("Stacked Category Distribution of", column, "by", target),
           x = column, y = "Proportion", fill = target) +
```

```
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.background = element_rect(fill = "white", color = NA),
          panel.background = element_rect(fill = "white", color = NA))
  }

  return(p)
}

#-------------------------------------------------------------------------

plot_horizontal_box <- function(data, features) {
  plot_list <- list()

  for (feature in features) {
    p <- ggplot(data, aes(x = .data[[feature]], y = "")) +
      geom_boxplot(fill = "lightblue", outlier.colour = "red", alpha = 0.7) +
      labs(title = paste("Box Plot of", feature), x = "Values", y = "")

    plot_list[[feature]] <- p
  }

  return(plot_list)
}

#-------------------------------------------------------------------------

plot_qq_by_variable <- function(data, features, title_prefix = "Q-Q Plot for", drop_zeros = TRUE) {
  plot_list <- list()
    for (feature in features) {
    plot_data <- if (drop_zeros) data[data[[feature]] != 0 & !is.na(data[[feature]]), , drop = FALSE] else data

    p <- ggplot(plot_data, aes(sample = .data[[feature]])) +
      stat_qq() +
      stat_qq_line(color = "red") +
      labs(
        title = paste(title_prefix, feature),
        x = "Theoretical Quantiles", y = "Sample Quantiles"
      )

    plot_list[[feature]] <- p
  }
    return(plot_list)
}

#-------------------------------------------------------------------------

check_normality_shapiro <- function(data, vars, sample_size = 500, seed = 43, remove_zeros = FALSE) {
  set.seed(seed)
  data_sample <- data[sample(nrow(data), min(nrow(data), sample_size)), ]

  results <- lapply(vars, function(col) {
    col_data <- na.omit(data_sample[[col]])
    if (remove_zeros) col_data <- col_data[col_data != 0]

    if (length(col_data) > 3) {
```

```
    test <- shapiro.test(col_data)
    data.frame(Feature = col, statistic = test$statistic, p.value = test$p.value)
   } else {
    data.frame(Feature = col, statistic = NA, p.value = NA)
   }
  })

  dplyr::bind_rows(results) %>%
   dplyr::arrange(p.value)
}

render_normality_table <- function(results_df,
                                   title = "Shapiro-Wilk Normality Test Results",
                                   digits = 5,
                                   end = 0.6) {
  df <- results_df %>%
   arrange(statistic) %>%
   dplyr::select(Feature, statistic, p.value) %>%
  mutate(p.value = format(p.value, scientific = TRUE, digits = digits))

  caption_html <- paste0(
   "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
   title,
   "</div>"
  )

  df %>%
   kable("html", escape = FALSE, caption = caption_html) %>%
   style_kable_default(df) %>%
   column_spec(3, width = "120px", background = spec_color(df$statistic, end = end, direction = 1))
}

#-------------------------------------------------------------------------------

log_transform_vars <- function(data, vars, suffix = "_log", filter_expr = NULL) {
  for (v in vars) {
   if (!is.numeric(data[[v]])) next

   transformed <- if (!is.null(filter_expr)) {
    result <- rep(NA, nrow(data))
    rows <- eval(parse(text = filter_expr), envir = data)
    result[rows] <- log(data[[v]][rows] + 1)
    result
   } else {
    log(data[[v]] + 1)
   }

   data[[paste0(v, suffix)]] <- transformed
  }
  return(data)
}

#-------------------------------------------------------------------------------

replace_features_by_suffix <- function(base_list, all_features, suffix = "_log") {
```

```
  sapply(base_list, function(f) {
    transformed_name <- paste0(f, suffix)
    if (transformed_name %in% all_features) {
      transformed_name
    } else {
      f
    }
  }, USE.NAMES = FALSE)
}

#-------------------------------------------------------------------------

############################
### Statistical_tests  ###
############################

######## Parametric    ############################

## Logistic Regression (binary targets)
run_logistic_models <- function(data, binary_targets, predictors) {
  models <- list()
  for (target in binary_targets) {
    formula <- as.formula(paste(target, "~", paste(predictors, collapse = "+")))
    model <- glm(formula, data = data, family = "binomial")
    models[[target]] <- model
  }
  return(models)
}

#-------------------------------------------------------------------------

render_logit_summary_table <- function(model,
                                       target_name = "Target",
                                       p_filter = 1000,
                                       exclude_terms = NULL,
                                       digits = 4,
                                       begin = 0.3,
                                       end = 0.7,
                                       rank_by = "p.value") {
  # compute odds ratios
  tidy_output <- broom::tidy(model) %>%
    dplyr::mutate(OR = exp(estimate)) %>%
    dplyr::filter(p.value < p_filter)

  # optional filter
  if (!is.null(exclude_terms)) {
    tidy_output <- tidy_output %>%
      filter(!term %in% exclude_terms)
  }

  # rank and sort
  tidy_output <- tidy_output %>%
    dplyr::mutate(Rank = dplyr::dense_rank(!!rlang::sym(rank_by))) %>%
    dplyr::arrange(Rank) %>%
    dplyr::select(Rank, term, estimate, OR, statistic, p.value)
```

```
 title <- paste0("Predictors for: ", target_name)
 caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  title,
  "</div>"
 )

 # style table
 tidy_output %>%
  knitr::kable("html", digits = digits, escape = FALSE,
               caption = caption_html
  ) %>%
  style_kable_default(tidy_output) %>%
  kableExtra::column_spec(3, width = "120px", background = spec_color(tidy_output$estimate, begin =
begin, end = end)) %>%
  kableExtra::column_spec(4, width = "120px", background = spec_color(tidy_output$OR, begin = begin,
end = end)) %>%
  kableExtra::column_spec(5, width = "120px", background = spec_color(tidy_output$statistic, begin =
begin, end = end)) %>%
  kableExtra::column_spec(6, width = "120px", background = spec_color(tidy_output$p.value, direction =
-1, begin = begin, end = end))
}

#------------------------------------------------------------------------------

## Linear Regression (continuous targets i.e. length of stay)
run_linear_models <- function(data, continuous_targets, predictors) {
 models <- list()
 for (target in continuous_targets) {
  formula <- as.formula(paste(target, "~", paste(predictors, collapse = "+")))
  model <- lm(formula, data = data)
  models[[target]] <- model
 }
 return(models)
}

#------------------------------------------------------------------------------

render_lm_summary_table <- function(model,
                                    target_name = "Target",
                                    exclude_terms = NULL,
                                    digits = 4,
                                    begin = begin,
                                    end = 0.8,
                                    rank_by = "p.value") {
 # summary
 tidy_output <- broom::tidy(model)

 # filter
 if (!is.null(exclude_terms)) {
  tidy_output <- tidy_output %>%
   dplyr::filter(!term %in% exclude_terms)
 }
```

```
 # sort
 tidy_output <- tidy_output %>%
  dplyr::mutate(Rank = dplyr::dense_rank(!!rlang::sym(rank_by))) %>%
  dplyr::arrange(Rank) %>%
  dplyr::select(Rank, term, estimate, statistic, p.value)

 title <- paste0("Predictors for: ", target_name)
 caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  title,
  "</div>"
 )

 # Style table
 tidy_output %>%
  knitr::kable("html", digits = digits, escape = FALSE,
               caption = caption_html) %>%
  style_kable_default(tidy_output) %>%
  kableExtra::column_spec(3, width = "120px", background = spec_color(tidy_output$estimate, begin =
begin, end = end)) %>%
  kableExtra::column_spec(4, width = "120px", background = spec_color(tidy_output$statistic, begin =
begin, end = end)) %>%
  kableExtra::column_spec(5, width = "120px", background = spec_color(tidy_output$p.value, direction =
-1, begin = begin, end = end))
}

#-------------------------------------------------------------------------

render_vif_summary_table <- function(model,
                                     target_name = "Target",
                                     digits = 4,
                                     begin = 0.2,
                                     end = 0.8,
                                     rank_by = "GVIF^(1/(2*Df))") {

 #  VIF
 vif_df <- as.data.frame(car::vif(model))
 vif_df$term <- rownames(vif_df)
 rownames(vif_df) <- NULL

 # Rank
 vif_df <- vif_df %>%
  dplyr::mutate(Rank = dplyr::dense_rank(-!!rlang::sym(rank_by))) %>%
  dplyr::arrange(Rank) %>%
  dplyr::select(Rank, term, dplyr::everything())

 title <- paste0("VIF Summary for: ", target_name)
 caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  title,
  "</div>"
 )

 # style table
```

```
  vif_df %>%
    kable("html", digits = digits, escape = FALSE,
          caption = title) %>%
    style_kable_default(vif_df) %>%
    column_spec(
      which(colnames(vif_df) == rank_by),
      background = spec_color(vif_df[[rank_by]], direction = -1, begin= begin, end = end)
    )
}

#-------------------------------------------------------------------------

run_t_tests <- function(data, numeric_vars, target_var, target_label = "Outcome") {
  if (!is.numeric(data[[target_var]])) {
    target_numeric <- as.numeric(as.factor(data[[target_var]]))
  } else {
    target_numeric <- data[[target_var]]
  }

  results <- lapply(numeric_vars, function(var) {
    if (length(unique(na.omit(data[[var]]))) < 2) return(NULL)
    test <- t.test(data[[var]] ~ target_numeric, na.action = na.omit)
    data.frame(Feature = var, P_Value = test$p.value, Test = "t-test", stringsAsFactors = FALSE)
  })

  results_df <- do.call(rbind, results)
  results_df$Target <- target_label
  results_df <- results_df[order(results_df$P_Value), ]
  return(results_df)
}

#-------------------------------------------------------------------------


render_anova_summary_table <- function(data,
                                       target,
                                       predictors,
                                       title = NULL,
                                       digits = 5,
                                       begin = 0.2,
                                       end = 0.7) {
  # ANOVA for each predictor
  results <- lapply(predictors, function(pred) {
    formula <- as.formula(paste(target, "~", pred))
    test <- aov(formula, data = data)
    summary_stats <- summary(test)[[1]]
    p_val <- summary_stats[["Pr(>F)"]][1]

    data.frame(Predictor = pred, P_Value = p_val, stringsAsFactors = FALSE)
  })

  results_df <- do.call(rbind, results) %>%
    arrange(P_Value)

  # Caption title
```

```
 if (is.null(title)) {
  title <- paste0("ANOVA: Predictors of ", target)
 }

 caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  title,
  "</div>"
 )

 # tidy table
 results_df %>%
  kable("html", digits = digits, escape = FALSE, caption = caption_html) %>%
  style_kable_default(df = results_df) %>%
  kableExtra::column_spec(2, width = "120px", background = spec_color(results_df$P_Value, direction =
-1, begin= begin, end = end))
}

######## Non-Parametric    ############################

#-------------------------------------------------------------------------

run_wilcoxon_tests <- function(data, continuous_vars, binary_var, target_label = NULL) {
 results <- lapply(continuous_vars, function(cont_var) {
  test <- wilcox.test(data[[cont_var]] ~ data[[binary_var]], na.action = na.omit)
  data.frame(
   Feature = cont_var,
   Target = ifelse(is.null(target_label), binary_var, target_label),
   P_Value = test$p.value,
   stringsAsFactors = FALSE
  )
 })

 results_df <- do.call(rbind, results)
 results_df <- results_df[order(results_df$P_Value), ]
 return(results_df)
}

#-------------------------------------------------------------------------

render_wilcoxon_summary_table <- function(results_df,
                                          title = "Wilcoxon Test Results",
                                          digits = 4,
                                          begin = 0.2,
                                          end = 0.7) {
 df <- results_df %>%
  arrange(desc(P_Value)) %>%
  mutate(Rank = dense_rank(desc(P_Value))) %>%
  dplyr::select(Rank, Feature, Target, P_Value)

 caption_html <- paste0(
  "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
  title,
  "</div>"
 )
```

```
  df %>%
    kable("html", digits = digits, escape = FALSE,
          caption = caption_html) %>%
    style_kable_default(df) %>%
    column_spec(4, width = "120px", background = spec_color(df$P_Value, begin = begin, end = end,
direction = -1))
}


#-------------------------------------------------------------------------

run_kruskal_tests <- function(data, numeric_vars, target_var) {
  target_numeric <- as.numeric(as.factor(data[[target_var]]))
  data[[paste0(target_var, "_numeric")]] <- target_numeric

  results <- lapply(numeric_vars, function(col) {
    test <- kruskal.test(data[[col]] ~ data[[paste0(target_var, "_numeric")]], data = data)
    return(data.frame(Feature = col, P_Value = test$p.value))
  })

  df <- do.call(rbind, results)
  df$Target <- target_var
  df <- df[order(df$P_Value), ]
  return(df)
}


#-------------------------------------------------------------------------

plot_correlation_heatmap <- function(data, vars) {
  cor_matrix <- cor(data[, vars], use = "complete.obs", method = "pearson")
  cor_melt <- melt(cor_matrix)

  ggplot(cor_melt, aes(Var1, Var2, fill = value)) +
    geom_tile(color = "white") +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                         midpoint = 0, limit = c(-1,1), space = "Lab",
                         name = "Pearson\nCorrelation") +
    geom_text(aes(label = sprintf("%.2f", value)), color = "black", size = 3) +
    theme(
      plot.background = element_rect(fill = "white", color = NA),
      panel.background = element_rect(fill = "white", color = NA),
      panel.grid = element_line(color = "grey90"),
      axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1, size = 12),
      axis.text.y = element_text(size = 12),
      axis.title = element_blank(),
      plot.title = element_text(size = 14, face = "bold")
    ) +
    ggtitle("Correlation Heatmap")
}

#-------------------------------------------------------------------------

run_fishers_tests <- function(data, binary_vars, target_var, target_label = "Outcome") {
```

```
  results <- lapply(binary_vars, function(var) {
    if (length(unique(data[[var]])) != 2) return(NULL)
    if (length(unique(data[[target_var]])) != 2) return(NULL)

    tbl <- table(data[[var]], data[[target_var]])
    if (any(tbl == 0)) return(NULL)  # skip if any cell is zero

    test <- fisher.test(tbl)

    # Extract fields safely
    odds_ratio <- if ("estimate" %in% names(test)) unname(test$estimate) else NA
    ci_low <- if (!is.null(test$conf.int)) test$conf.int[1] else NA
    ci_high <- if (!is.null(test$conf.int)) test$conf.int[2] else NA

    data.frame(
      Feature = var,
      P_Value = test$p.value,
      Odds_Ratio = odds_ratio,
      CI_Lower = ci_low,
      CI_Upper = ci_high,
      Test = "Fisher",
      stringsAsFactors = FALSE
    )
  })

  results_df <- do.call(rbind, results)
  results_df$Target <- target_label
  results_df <- results_df[order(results_df$P_Value), ]
  return(results_df)
}

#------------------------------------------------------------------------------

render_fisher_summary_table <- function(results_df,
                                        title = "Fisher's Exact Test Results",
                                        digits = 4,
                                        begin = 0.2,
                                        end = 0.8) {

  caption_html <- paste0(
    "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
    title,
    "</div>"
  )

  # Sort by Target, then Odds Ratio (descending = stronger association)
  table_df <- results_df %>%
    dplyr::arrange(Target, desc(Odds_Ratio)) %>%
    dplyr::mutate(Rank = dplyr::row_number()) %>%
    dplyr::select(Rank, Feature, Target, Odds_Ratio, CI_Lower, CI_Upper, P_Value)

  # Render HTML table
  knitr::kable(table_df, "html", digits = digits, escape = FALSE,
               caption = caption_html,
```

```
            col.names = c("Rank", "Feature", "Target", "Odds Ratio", "CI Lower", "CI Upper", "P-
Value")) %>%
   style_kable_default(table_df) %>%
   kableExtra::column_spec(4, width = "100px", background = spec_color(table_df$Odds_Ratio, begin =
begin, end = end)) %>%
   kableExtra::column_spec(7, width = "100px", background = spec_color(table_df$P_Value, begin =
begin, end = end, direction = -1))
}


#------------------------------------------------------------------------

run_chisq_tests_df <- function(data, target_var, cat_vars = NULL) {
 if (is.null(cat_vars)) {
   cat_vars <- names(data)[sapply(data, is.factor) & names(data) != target_var]
 }

 results <- lapply(cat_vars, function(col) {
   tbl <- table(data[[col]], data[[target_var]])
   if (all(dim(tbl) > 1)) {
     test <- chisq.test(tbl)
     return(data.frame(
       Feature = col,
       P_Value = test$p.value,
       Statistic = test$statistic
     ))
   } else {
     return(data.frame(
       Feature = col,
       P_Value = NA,
       Statistic = NA
     ))
   }
 })

 result_df <- do.call(rbind, results)
 result_df <- result_df[order(result_df$P_Value), ]
 return(result_df)
}

run_chisq_for_targets <- function(data, target_vars, cat_vars = NULL) {
 results <- do.call(rbind, lapply(target_vars, function(target) {
   df <- run_chisq_tests_df(data, target_var = target, cat_vars = cat_vars)
   df$Target <- target
   return(df)
 }))
 return(results)
}

#------------------------------------------------------------------------

render_chisq_summary_table <- function(results_df,
                                       title = "Chi-Squared Test Results",
                                       digits = 4,
                                       begin = 0.2,
```

```
                              end = 0.8) {

  caption_html <- paste0(
    "<div style='text-align:center; font-weight:bold; color:black; font-size:16px;'>",
    title,
    "</div>"
  )

  # Sort, rank, prepare
  table_df <- results_df %>%
    dplyr::arrange(Target, Statistic) %>%
    dplyr::mutate(Rank = dplyr::row_number()) %>%
    dplyr::select(Rank, Feature, Target, Statistic, P_Value)

  # Render HTML table
  knitr::kable(table_df, "html", digits = digits, escape = FALSE,
               caption = caption_html,
               col.names = c("Rank", "Feature", "Target", "Chi-Squared Statistic", "P-Value")) %>%
    style_kable_default(table_df) %>%
    kableExtra::column_spec(5, width = "120px",
                            background = spec_color(table_df$Statistic, begin = begin, end = end))
}

#-------------------------------------------------------------------------

######################
### Model Training ###
######################

## Split & Train Single Model - With Performance to CSV export
## Function can split time based

customSummary <- function(data, lev = NULL, model = NULL) {
  obs <- data$obs
  prob_yes <- data$Yes
  pred_class <- ifelse(prob_yes >= 0.5, "Yes", "No")
  pred_class <- factor(pred_class, levels = lev)

  # Compute Metrics
  roc_auc <- yardstick::roc_auc_vec(truth = obs, estimate = prob_yes, event_level = "second")
  pr_auc <- yardstick::pr_auc_vec(truth = obs, estimate = prob_yes, event_level = "second")
  precision <- yardstick::precision_vec(truth = obs, estimate = pred_class, event_level = "second")
  recall <- as.numeric(yardstick::recall_vec(truth = obs, estimate = pred_class, event_level = "second"))
  f1_score <- yardstick::f_meas_vec(truth = obs, estimate = pred_class, event_level = "second")
  sens <- yardstick::sens_vec(truth = obs, estimate = pred_class, event_level = "second")
  spec <- yardstick::spec_vec(truth = obs, estimate = pred_class, event_level = "second")
  accuracy <- yardstick::accuracy_vec(truth = obs, estimate = pred_class)
  kappa <- yardstick::kap_vec(truth = obs, estimate = pred_class)  # Add Cohen's Kappa

  # Return all metrics
  c(Accuracy = accuracy, Kappa = kappa, ROC = roc_auc, PR_AUC = pr_auc,
    Precision = precision, Recall = recall, F1 = f1_score, Specificity = spec, Sensitivity = sens)
}

# Generalized Training Function with Time-Based Splitting & Correct Summary Function
```

```
train_single_model <- function(model_name, target, feature_set_name, selected_features, dataset,
dataset_name, csv_path, tuneGrid = NULL, time_slices = 5) {

 # Sort dataset by time (to ensure chronological training)
 dataset <- dataset[order(dataset$admittime), ]

 # Define Time-Based Cross-Validation with Correct Summary Function
 ctrl <- if (is.numeric(dataset[[target]])) {
  trainControl(
   method = "timeslice",
   initialWindow = round(0.7 * nrow(dataset)),  # Train on 70% of historical data
   #horizon = round(0.3 * nrow(dataset)),        # Test on 30% of future data
   horizon = round(0.3 * nrow(dataset)),  # Increase test set to 30% of data,
   fixedWindow = TRUE,
   #skip = round(nrow(dataset) / (time_slices + 1)),  # Number of skipped observations per window
   skip = max(1, round(nrow(dataset) / (time_slices + 1))),
   summaryFunction = defaultSummary,
   savePredictions = "final",
   verboseIter = TRUE
  )
 } else {
  trainControl(
   method = "timeslice",
   initialWindow = round(0.7 * nrow(dataset)),
   horizon = round(0.3 * nrow(dataset)),
   #horizon = min(100, round(0.2 * nrow(dataset))),
   fixedWindow = TRUE,
   #skip = round(nrow(dataset) / (time_slices + 1)),  # Number of skipped observations per window
   skip = max(1, round(nrow(dataset) / (time_slices + 1))),
   classProbs = TRUE,
   summaryFunction = customSummary,
   savePredictions = "final",
   sampling = "smote",
   verboseIter = TRUE
  )
 }

 # Construct formula
 formula <- as.formula(paste(target, "~", paste(selected_features, collapse = " + ")))

 # Train Model
 model <- train(formula, data = dataset, method = model_name, trControl = ctrl, tuneGrid = tuneGrid)

 results <- model$results

 if (nrow(results) == 0) {
  results <- data.frame(
   RMSE = NA, Rsquared = NA, MAE = NA,  # baseline regression metrics
   Accuracy = NA, Kappa = NA, ROC = NA, PR_AUC = NA,
   Sensitivity = NA, Specificity = NA, Precision = NA, Recall = NA, F1 = NA,
   stringsAsFactors = FALSE
  )
 }

 # Timestamp of Model Run
```

```
  model_timestamp <- format(Sys.time(), "%Y-%m-%d %H:%M:%S")  # Format as "YYYY-MM-DD
HH:MM:SS"

  # Extract AIC & BIC if applicable
  final_model <- model$finalModel
  results$AIC <- if ("glm" %in% class(final_model) || "lm" %in% class(final_model)) AIC(final_model)
else NA
  results$BIC <- if ("glm" %in% class(final_model) || "lm" %in% class(final_model)) BIC(final_model)
else NA


  all_metrics <- c(
    "Accuracy",    # Standard classification metric (useful for balanced datasets)
    "Kappa",       # Agreement metric useful for imbalanced datasets
    "ROC",         # Area under the ROC curve (classification)
    "PR_AUC",      # Precision-Recall AUC (better for highly imbalanced datasets)
    "Sensitivity", # Sensitivity (Recall) - True Positive Rate
    "Specificity", # Specificity - True Negative Rate
    "Precision",   # Precision (Positive Predictive Value)
    "Recall",      # Recall (Sensitivity)
    "F1",          # F1-Score (harmonic mean of Precision & Recall)
    "RMSE",        # Root Mean Squared Error (regression)
    "Rsquared",    # R² - Variance explained (regression)
    "MAE",         # Mean Absolute Error (regression)
    "AIC",         # Akaike Information Criterion
    "BIC"          # Bayesian Information Criterion
  )

  # Ensure all metrics exist, setting missing ones to NA
  for (metric in all_metrics) {
    if (!(metric %in% colnames(results))) {
      results[[metric]] <- NA  # Fill missing columns with NA for consistency
    }
  }

  # Ensure numerical values are stored with higher precision
  results[all_metrics] <- lapply(results[all_metrics], function(x) {
    round(as.numeric(x), 6)  # Increase to 6 decimal places
  })


  # Store tuning parameters for each row in results
  if (!is.null(tuneGrid)) {
    results$Tuning_Params <- apply(model$results[, names(tuneGrid), drop = FALSE], 1, function(row) {
      paste(names(row), "=", row, collapse = "; ")
    })
  } else {
    results$Tuning_Params <- "Default"
  }

  # Standardize Output Format
  results$Data_Set <- dataset_name
  results$Timestamp <- model_timestamp
  results$Target <- target
  results$Model <- model_name
```

```
  results$Feature_Set <- feature_set_name
  results$Features_Used <- paste(selected_features, collapse = ", ")
  results$Validation_Type <- "Time-Based CV"
  results$Time_Slices <- time_slices
  results$Description <- paste("Model trained on", feature_set_name, "for", target)


  final_results <- results[, c("Timestamp","Data_Set", "Target", "Model", "Feature_Set", "Features_Used",
"Validation_Type", "Time_Slices", "Tuning_Params", "Description", all_metrics), drop = FALSE]

  # Save to CSV (Append or Create New)
  if (file.exists(csv_path)) {
    existing_results <- read.csv(csv_path)  # Read existing results
    final_results <- rbind(existing_results, final_results)  # Append new results
  }

  write.csv(final_results, csv_path, row.names = FALSE)  # Write everything back

  cat("\n Model results saved to:", csv_path, "\n")

  print(final_results)

  return(model)

}

#------------------------------------------------------------------------

## Split & Train Single Model 2 - With Performance to CSV export
## Function can split time based or by patient, default CV, but 70/30 for final

train_single_model2 <- function(model_name, target, feature_set_name, selected_features, dataset,
dataset_name, csv_path,
                                tuneGrid = NULL, split_type = "patient", train_size = 0.7, time_slices = 5, cv_splits =
5, final_model = FALSE) {

  set.seed(42)  # Ensure reproducibility

  # Train-test split based on selected method
  if (split_type == "patient") {
    unique_patients <- unique(dataset$subject_id)
    train_patients <- sample(unique_patients, size = round(train_size * length(unique_patients)), replace =
FALSE)
    train_data <- dataset[dataset$subject_id %in% train_patients, ]
    test_data <- dataset[!dataset$subject_id %in% train_patients, ]
  } else if (split_type == "time") {
    dataset <- dataset[order(dataset$admittime), ]
    train_size_count <- round(train_size * nrow(dataset))
    train_data <- dataset[1:train_size_count, ]
    test_data <- dataset[(train_size_count + 1):nrow(dataset), ]
  } else {
    stop("Invalid split_type. Use 'time' or 'patient'.")
  }

  # Define train control based on whether this is the final model
```

```
 if (final_model) {
    ctrl <- if (is.numeric(dataset[[target]])) {
     trainControl(
       method = "none",
       summaryFunction = defaultSummary,
       savePredictions = "final",
       verboseIter = TRUE
     )
   } else {
    trainControl(
      method = "none",
      classProbs = TRUE,
      summaryFunction = customSummary,
      savePredictions = "final",
      verboseIter = TRUE
    )
   }
} else {
 ctrl <- if (is.numeric(dataset[[target]])) {
   trainControl(
     method = if (split_type == "time") "timeslice" else "cv",
     number = if (split_type == "patient") cv_splits else NA,  # number of splits
     initialWindow = if (split_type == "time") round(train_size * nrow(dataset)) else NA,
     horizon = if (split_type == "time") round((1 - train_size) * nrow(dataset)) else NA,
     fixedWindow = if (split_type == "time") TRUE else NA,
     skip = if (split_type == "time") max(1, round(nrow(dataset) / (time_slices + 1))) else NA,
     summaryFunction = defaultSummary,
     savePredictions = "final",
     verboseIter = TRUE
   )
 } else {
   trainControl(
     method = if (split_type == "time") "timeslice" else "cv",
     number = if (split_type == "patient") cv_splits else NA,  # number of splits
     initialWindow = if (split_type == "time") round(train_size * nrow(dataset)) else NA,
     horizon = if (split_type == "time") round((1 - train_size) * nrow(dataset)) else NA,
     fixedWindow = if (split_type == "time") TRUE else NA,
     skip = if (split_type == "time") max(1, round(nrow(dataset) / (time_slices + 1))) else NA,
     classProbs = TRUE,
     summaryFunction = customSummary,
     savePredictions = "final",
     sampling = "smote",  # Only apply SMOTE when using CV, not final model
     verboseIter = TRUE
   )
 }
}

# Construct formula
formula <- as.formula(paste(target, "~", paste(selected_features, collapse = " + ")))

# Train Model
model <- train(
 formula,
 data = if (final_model) train_data else dataset,
 method = model_name,
```

```
  trControl = ctrl,
  tuneGrid = tuneGrid
 )

 results <- if (final_model) data.frame() else model$results

 if (final_model) {
  is_classification <- is.factor(dataset[[target]])

  if (is_classification) {
   prob_yes <- predict(model, newdata = test_data, type = "prob")[, "Yes"]
   obs <- test_data[[target]]
   eval_data <- data.frame(obs = obs, Yes = prob_yes)
   summary_vals <- customSummary(eval_data, lev = levels(obs))
   results <- as.data.frame(as.list(summary_vals))  # convert to 1-row df
  } else {
   preds <- predict(model, newdata = test_data)
   actual <- test_data[[target]]
   summary_vals <- list(
    RMSE = caret::RMSE(preds, actual),
    MAE  = caret::MAE(preds, actual),
    Rsquared = caret::R2(preds, actual),
    MSE  = mean((preds - actual)^2)
   )
   results <- as.data.frame(summary_vals)
  }
 }

 # Timestamp of Model Run
 model_timestamp <- format(Sys.time(), "%Y-%m-%d %H:%M:%S")  # Format as "YYYY-MM-DD
HH:MM:SS"

 # Extract AIC & BIC if applicable
 model_final <- model$finalModel  # Rename to avoid conflict

 results$AIC <- if ("glm" %in% class(model_final) || "lm" %in% class(model_final)) AIC(model_final)
else NA
 results$BIC <- if ("glm" %in% class(model_final) || "lm" %in% class(model_final)) BIC(model_final)
else NA


 all_metrics <- c(
  "Accuracy",    # Standard classification metric (useful for balanced datasets)
  "Kappa",       # Agreement metric useful for imbalanced datasets
  "ROC",         # Area under the ROC curve (classification)
  "PR_AUC",      # Precision-Recall AUC (better for highly imbalanced datasets)
  "Sensitivity", # Sensitivity (Recall) - True Positive Rate
  "Specificity", # Specificity - True Negative Rate
  "Precision",   # Precision (Positive Predictive Value)
  "Recall",      # Recall (Sensitivity)
  "F1",          # F1-Score (harmonic mean of Precision & Recall)
  "RMSE",        # Root Mean Squared Error (regression)
  "Rsquared",    # R² - Variance explained (regression)
  "MAE",         # Mean Absolute Error (regression)
  "AIC",         # Akaike Information Criterion
```

```
  "BIC"          # Bayesian Information Criterion
)

# Set missing metrics to NA
for (metric in all_metrics) {
  if (!(metric %in% colnames(results))) {
    results[[metric]] <- NA  # Fill missing columns with NA for consistency
  }
}

# stored numerical values with higher precision
results[all_metrics] <- lapply(results[all_metrics], function(x) {
  round(as.numeric(x), 6)  # Increase to 6 decimal places
})


# Store tuning parameters for each row in results
if (!is.null(tuneGrid)) {
  if (final_model) {
    # For final models, just log the single grid row you passed
    results$Tuning_Params <- paste(names(tuneGrid), "=", unlist(tuneGrid[1, ]), collapse = "; ")
  } else {
    # For CV models with multiple rows in results
    results$Tuning_Params <- apply(model$results[, names(tuneGrid), drop = FALSE], 1, function(row) {
      paste(names(row), "=", row, collapse = "; ")
    })
  }
} else {
  results$Tuning_Params <- "Default"
}

# Validation Type
if (final_model) {
  validation_type <- "Final Model (No CV)"
} else if (split_type == "time") {
  validation_type <- "Time-Based CV"
} else {
  validation_type <- paste0(cv_splits, "-Fold Cross-Validation")
}

# Split Type
split_method <- if (split_type == "time") {
  paste0("Time-Split (", round(train_size * 100), "% Train / ", round((1 - train_size) * 100), "% Test)")
} else {
  paste0("Patient-Split (", round(train_size * 100), "% Train / ", round((1 - train_size) * 100), "% Test)")
}

split_feature <- ifelse(split_type == "time", "Time", "Patient")

# Compute Train/Test Row Counts
train_count <- nrow(train_data)
test_count <- nrow(test_data)

# Standardize Output Format
results$Data_Set <- dataset_name
```

```
  results$Timestamp <- model_timestamp
  results$Target <- target
  results$Model <- model_name
  results$Feature_Set <- feature_set_name
  results$Features_Used <- paste(selected_features, collapse = ", ")
  results$Validation_Type <- validation_type
  results$Split_Type <- split_method
  results$Split_Feature <- split_feature  # Now simply "Time" or "Patient"
  results$Splits <- ifelse(final_model, NA, ifelse(split_type == "time", time_slices, cv_splits))
  results$Train_Percentage <- paste0(round(train_size * 100), "%")
  results$Description <- paste("Model trained on", feature_set_name, "for", target)
  results$Train_Count <- nrow(train_data)
  results$Test_Count <- nrow(test_data)


  final_results <- results[, c("Timestamp","Data_Set", "Target", "Model", "Feature_Set", "Features_Used",
"Validation_Type", "Split_Type", "Split_Feature" ,"Splits", "Train_Percentage" ,"Tuning_Params",
"Description", "Train_Count","Test_Count", all_metrics), drop = FALSE]

  # model$train_data <- train_data
  model$test_data  <- test_data

  # Save to CSV (Append or Create New)
  if (file.exists(csv_path)) {
    existing_results <- read.csv(csv_path)  # Read existing results
    final_results <- rbind(existing_results, final_results)  # Append new results
  }

  write.csv(final_results, csv_path, row.names = FALSE)  # Write everything back

  cat("\n Model results saved to:", csv_path, "\n")

  print(final_results)

  return(model)

}

#------------------------------------------------------------------------------


## Split & Train Single Model 3 - With Performance to CSV export
## Function can split time based or by patient, default CV, but 70/30 for final
## Fixed CV split with leakage :C

train_single_model3 <- function(model_name, target, feature_set_name, selected_features, dataset,
dataset_name, csv_path,
                                tuneGrid = NULL, split_type = "patient", train_size = 0.7, time_slices = 5, cv_splits =
5, final_model = FALSE) {

  set.seed(42)  # Ensure reproducibility

  # --- Train/Test Split (only used for final_model = TRUE)
  if (split_type == "patient") {
    unique_patients <- unique(dataset$subject_id)
```

```
   train_patients <- sample(unique_patients, size = round(train_size * length(unique_patients)), replace =
FALSE)
   train_data <- dataset[dataset$subject_id %in% train_patients, ]
   test_data <- dataset[!dataset$subject_id %in% train_patients, ]
  } else {
   stop("Currently only 'patient' split_type is supported.")
  }

  # --- Train Control Setup
  if (final_model) {
   ctrl <- if (is.numeric(dataset[[target]])) {
    trainControl(
     method = "none",
     summaryFunction = defaultSummary,
     savePredictions = "final",
     verboseIter = TRUE
    )
   } else {
    trainControl(
     method = "none",
     classProbs = TRUE,
     summaryFunction = customSummary,
     savePredictions = "final",
     verboseIter = TRUE
    )
   }
  } else {
   # Cross-validation folds by unique patient
   set.seed(42)
   all_patients <- unique(dataset$subject_id)
   patient_folds <- createFolds(all_patients, k = cv_splits)

   # Translate patient IDs to row indices in dataset
   index <- lapply(patient_folds, function(p_ids) {
    which(dataset$subject_id %in% all_patients[p_ids])
   })

   ctrl <- if (is.numeric(dataset[[target]])) {
    trainControl(
     method = "cv",
     number = cv_splits,
     index = index,
     summaryFunction = defaultSummary,
     savePredictions = "final",
     verboseIter = TRUE
    )
   } else {
    trainControl(
     method = "cv",
     number = cv_splits,
     index = index,
     classProbs = TRUE,
     summaryFunction = customSummary,
     savePredictions = "final",
     sampling = "smote",
```

```
      verboseIter = TRUE
    )
  }
}

# Construct formula
formula <- as.formula(paste(target, "~", paste(selected_features, collapse = " + ")))

# Train Model
model <- train(
  formula,
  data = if (final_model) train_data else dataset,
  method = model_name,
  trControl = ctrl,
  tuneGrid = tuneGrid
)

results <- if (final_model) data.frame() else model$results

if (final_model) {
  is_classification <- is.factor(dataset[[target]])

  if (is_classification) {
    prob_yes <- predict(model, newdata = test_data, type = "prob")[, "Yes"]
    obs <- test_data[[target]]
    eval_data <- data.frame(obs = obs, Yes = prob_yes)
    summary_vals <- customSummary(eval_data, lev = levels(obs))
    results <- as.data.frame(as.list(summary_vals))  # convert to 1-row df
  } else {
    preds <- predict(model, newdata = test_data)
    actual <- test_data[[target]]
    summary_vals <- list(
      RMSE = caret::RMSE(preds, actual),
      MAE  = caret::MAE(preds, actual),
      Rsquared = caret::R2(preds, actual),
      MSE  = mean((preds - actual)^2)
    )
    results <- as.data.frame(summary_vals)
  }
}

# Timestamp of Model Run
model_timestamp <- format(Sys.time(), "%Y-%m-%d %H:%M:%S")  # Format as "YYYY-MM-DD
HH:MM:SS"

# Extract AIC & BIC if applicable
model_final <- model$finalModel  # Rename to avoid conflict

results$AIC <- if ("glm" %in% class(model_final) || "lm" %in% class(model_final)) AIC(model_final)
else NA
results$BIC <- if ("glm" %in% class(model_final) || "lm" %in% class(model_final)) BIC(model_final)
else NA


all_metrics <- c(
```

```
  "Accuracy",    # Standard classification metric (useful for balanced datasets)
  "Kappa",       # Agreement metric useful for imbalanced datasets
  "ROC",         # Area under the ROC curve (classification)
  "PR_AUC",      # Precision-Recall AUC (better for highly imbalanced datasets)
  "Sensitivity", # Sensitivity (Recall) - True Positive Rate
  "Specificity", # Specificity - True Negative Rate
  "Precision",   # Precision (Positive Predictive Value)
  "Recall",      # Recall (Sensitivity)
  "F1",          # F1-Score (harmonic mean of Precision & Recall)
  "RMSE",        # Root Mean Squared Error (regression)
  "Rsquared",    # R² - Variance explained (regression)
  "MAE",         # Mean Absolute Error (regression)
  "AIC",         # Akaike Information Criterion
  "BIC"          # Bayesian Information Criterion
)

# Set missing metrics to NA
for (metric in all_metrics) {
  if (!(metric %in% colnames(results))) {
    results[[metric]] <- NA  # Fill missing columns with NA for consistency
  }
}

# stored numerical values with higher precision
results[all_metrics] <- lapply(results[all_metrics], function(x) {
  round(as.numeric(x), 6)  # Increase to 6 decimal places
})


# Store tuning parameters for each row in results
if (!is.null(tuneGrid)) {
  if (final_model) {
    # For final models, just log the single grid row you passed
    results$Tuning_Params <- paste(names(tuneGrid), "=", unlist(tuneGrid[1, ]), collapse = "; ")
  } else {
    # For CV models with multiple rows in results
    results$Tuning_Params <- apply(model$results[, names(tuneGrid), drop = FALSE], 1, function(row) {
      paste(names(row), "=", row, collapse = "; ")
    })
  }
} else {
  results$Tuning_Params <- "Default"
}

# Validation Type
if (final_model) {
  validation_type <- "Final Model (No CV)"
} else if (split_type == "time") {
  validation_type <- "Time-Based CV"
} else {
  validation_type <- paste0(cv_splits, "-Fold Cross-Validation")
}

# Split Type
split_method <- if (split_type == "time") {
```

```
  paste0("Time-Split (", round(train_size * 100), "% Train / ", round((1 - train_size) * 100), "% Test)")
 } else {
  paste0("Patient-Split (", round(train_size * 100), "% Train / ", round((1 - train_size) * 100), "% Test)")
 }

 split_feature <- ifelse(split_type == "time", "Time", "Patient")

 # Compute Train/Test Row Counts
 train_count <- nrow(train_data)
 test_count <- nrow(test_data)

 # Standardize Output Format
 results$Data_Set <- dataset_name
 results$Timestamp <- model_timestamp
 results$Target <- target
 results$Model <- model_name
 results$Feature_Set <- feature_set_name
 results$Features_Used <- paste(selected_features, collapse = ", ")
 results$Validation_Type <- validation_type
 results$Split_Type <- split_method
 results$Split_Feature <- split_feature  # Now simply "Time" or "Patient"
 results$Splits <- ifelse(final_model, NA, ifelse(split_type == "time", time_slices, cv_splits))
 results$Train_Percentage <- paste0(round(train_size * 100), "%")
 results$Description <- paste("Model trained on", feature_set_name, "for", target)
 results$Train_Count <- nrow(train_data)
 results$Test_Count <- nrow(test_data)


 final_results <- results[, c("Timestamp","Data_Set", "Target", "Model", "Feature_Set", "Features_Used",
"Validation_Type", "Split_Type", "Split_Feature" ,"Splits", "Train_Percentage" ,"Tuning_Params",
"Description", "Train_Count","Test_Count", all_metrics), drop = FALSE]

 # model$train_data <- train_data
 model$test_data  <- test_data

 # Save to CSV (Append or Create New)
 if (file.exists(csv_path)) {
  existing_results <- read.csv(csv_path)  # Read existing results
  final_results <- rbind(existing_results, final_results)  # Append new results
 }

 write.csv(final_results, csv_path, row.names = FALSE)  # Write everything back

 cat("\n Model results saved to:", csv_path, "\n")

 print(final_results)

 return(model)

}


#------------------------------------------------------------------------------

## Call Model Train Fucntion
```

```
train_and_assign <- function(
    train_function = get("train_function", envir = .GlobalEnv),
    model_type, feature_set_name, selected_features, tuneGrid = NULL,
    dataset = get("data", envir = .GlobalEnv),
    dataset_name = get("dataset_name", envir = .GlobalEnv),
    target = get("target", envir = .GlobalEnv),
    file_name = get("file_name", envir = .GlobalEnv),
    slices = get("slices", envir = .GlobalEnv),
    model_run = get("model_run", envir = .GlobalEnv),
    split_type = get("split_type", envir = .GlobalEnv),
    train_size = get("train_size", envir = .GlobalEnv),
    cv_splits = get("cv_splits", envir = .GlobalEnv),
    final_model = get("final_model", envir = .GlobalEnv)
) {

  # Define dynamic model name
  model_name <- paste0(model_type, "_", gsub(" ", "_", tolower(feature_set_name)), "_", target, "_",
model_run)

  # Call the selected training function dynamically
  trained_model <- do.call(train_function, list(
    model_name = model_type,
    target = target,
    feature_set_name = feature_set_name,
    selected_features = selected_features,
    dataset = dataset,
    dataset_name = dataset_name,
    csv_path = here("results", file_name),
    time_slices = slices,
    tuneGrid = tuneGrid,
    split_type = split_type,
    train_size = train_size,
    cv_splits = cv_splits,
    final_model = final_model
  ))

  # Assign the trained model dynamically
  assign(model_name, trained_model, envir = .GlobalEnv)

  return(model_name)
}

#------------------------------------------------------------------------------

feature_selection <- function(
    model,
    target,
    features,
    dataset,
    fast_mode = TRUE,
    return_stats = FALSE
) {
  model_type <- model$method
  selected_features <- NULL
```

```
is_classification <- is.factor(dataset[[target]])

if (model_type %in% c("rf", "gbm", "xgbTree", "svmRadial", "svmLinear", "nnet", "naive_bayes")) {

 imp <- varImp(model, scale = TRUE)
 imp_df <- imp$importance

 if (!"Overall" %in% colnames(imp_df)) {
  overall_col <- colnames(imp_df)[1]
  imp_df$Overall <- imp_df[[overall_col]]
 }

 imp_df <- imp_df %>%
  tibble::rownames_to_column("feature") %>%
  dplyr::mutate(rank = rank(-Overall)) %>%
  dplyr::arrange(rank)

 selected_features <- imp_df$feature[imp_df$Overall >= 1]

 if (return_stats) return(imp_df)

} else if (model_type %in% c("glm", "bayesglm")) {

 formula <- as.formula(paste(target, "~", paste(features, collapse = "+")))
 family_used <- if (is_classification) binomial() else gaussian()
 full_model <- glm(formula, data = dataset, family = family_used)
 step_model <- stepAIC(full_model, direction = "backward", trace = FALSE)
 selected_features <- names(coef(step_model))[-1]

 if (return_stats) {
  return(tibble::tibble(
   feature = selected_features,
   rank = seq_along(selected_features)
  ))
 }

} else if (!is.null(varImp(model, scale = FALSE))) {

 imp <- varImp(model, scale = TRUE)
 imp_df <- imp$importance

 if (!"Overall" %in% colnames(imp_df)) {
  stop(paste("No 'Overall' column found in importance for model:", model_type))
 }

 imp_df <- imp_df %>%
  tibble::rownames_to_column("feature") %>%
  dplyr::mutate(rank = rank(-Overall)) %>%
  dplyr::arrange(rank)

 selected_features <- imp_df$feature[imp_df$Overall >= 1]

 if (return_stats) return(imp_df)

} else {
```

```
    stop(paste("Feature selection not implemented for model type:", model_type))
  }

  return(selected_features)
}


#-----------------------------------------------------

get_model_data_and_features <- function(model_obj, target) {
  if (!is.null(model_obj$trainingData)) {
    data <- model_obj$trainingData
    data[[target]] <- data$.outcome
    features <- setdiff(names(data), c(".outcome", target))
    return(list(data = data, features = features))
  }
}


#-----------------------------------------------------

get_training_ready_features <- function(model_type, feature_set, target, model_run,
feature_selection_results, original_feature_list) {
  key <- paste(model_type, feature_set, target, model_run, sep = "_")
  model_result <- feature_selection_results[[key]]

  if (is.null(model_result)) {
    warning("Missing model result for: ", key)
    return(character(0))
  }

  removed_feats <- model_result$removed
  usable_feats <- setdiff(original_feature_list, removed_feats)

  return(usable_feats)
}


########################
### Model Evaluation ###
########################


add_predictions_to_models <- function(models, target_col = "hospital_expire_flag") {
  for (i in seq_along(models)) {
    model <- models[[i]]

    test_data <- model$test_data
    probs <- predict(model, newdata = test_data, type = "prob")[, "Yes"]

    pred_df <- data.frame(
      rowIndex = 1:nrow(test_data),
      obs = test_data[[target_col]],
      Yes = probs
    )
```

```
    model$pred <- pred_df

    models[[i]] <- model
  }
  return(models)
}

### ROC AUC Plot

plot_roc_curves <- function(models, model_names = names(models), target, plot_title = "ROC Curves") {
  roc_curves <- list()
  legend_labels <- character()

  for (i in seq_along(models)) {
    model <- models[[i]]
    model_name <- model_names[i]

    # Extract model type (first part before "_")
    model_type <- strsplit(model_name, "_")[[1]][1]

    # Detect if SDoH or Base
    base_or_sdoh <- if (grepl("sdoh", tolower(model_name))) "(SDoH)" else "(Base)"

    # Create clean legend label
    label <- paste(model_type, base_or_sdoh)

    test_preds <- model$pred
    test_preds_filtered <- test_preds[test_preds$rowIndex %in% unique(test_preds$rowIndex), ]

    actual_values <- test_preds_filtered$obs
    pred_probs <- if ("Yes" %in% names(test_preds_filtered)) {
      test_preds_filtered$Yes
    } else if ("pred_prob" %in% names(test_preds_filtered)) {
      test_preds_filtered$pred_prob
    } else {
      stop("Prediction probability column not found.")
    }

    roc_obj <- pROC::roc(response = actual_values, predictor = pred_probs)
    # Store curve with unique internal key
    internal_label <- paste0(model_name, "_", i)
    roc_curves[[internal_label]] <- roc_obj
    legend_labels <- c(legend_labels, label)
  }

  # Plot the first ROC curve
  plot.roc(roc_curves[[1]], col = 1, main = paste(plot_title, "-", target),
           xlab = "False Positive Rate (1 - Specificity)",
           ylab = "True Positive Rate (Sensitivity)", lwd = 2, legacy.axes = TRUE)

  # Overlay others
  for (i in 2:length(roc_curves)) {
    plot.roc(roc_curves[[i]], col = i, add = TRUE, lwd = 2, lty = i)
  }
```

```
  abline(0, 1, col = "gray", lty = 2)

  # Legend with short names
  legend("bottomright", legend = legend_labels, col = 1:length(roc_curves), lwd = 2, lty =
1:length(roc_curves))
}

#------------------------------------------------------------------------

### PR AUC Plot

library(PRROC)

plot_pr_curves <- function(models, model_names, target, plot_title = "PR AUC Curves") {
  pr_curves <- list()

  for (i in seq_along(models)) {
    model <- models[[i]]
    model_name <- model_names[i]

    # Extract test predictions stored in `model$pred`
    test_preds <- model$pred

    # Filter to only test set predictions (based on rowIndex)
    test_preds_filtered <- test_preds[test_preds$rowIndex %in% unique(test_preds$rowIndex), ]

    # Extract actual test labels and predicted probabilities
    actual_values <- test_preds_filtered$obs  # True labels
    pred_probs <- test_preds_filtered$Yes  # Predicted probabilities for "Yes"

    # Convert actual values to numeric (1 = "Yes", 0 = "No")
    actual_values_numeric <- ifelse(actual_values == "Yes", 1, 0)

    # Compute Precision-Recall Curve
    pr_obj <- pr.curve(scores.class0 = pred_probs, weights.class0 = actual_values_numeric, curve = TRUE)

    # Store PR Curve for later comparison
    pr_curves[[model_name]] <- pr_obj
  }

  # Set up plot margins to leave space for legend
  par(mar = c(5, 4, 4, 8) + 0.1)  # Increase right margin for legend space

  # Define colors and line types
  colors <- rainbow(length(pr_curves))
  line_types <- rep(1:6, length.out = length(pr_curves))  # Cycle through different line types

  # Plot PR Curves
  plot(pr_curves[[1]]$curve, type = "l", col = colors[1], lwd = 2, lty = line_types[1],
       xlab = "Recall", ylab = "Precision",
       main = paste(plot_title, "-", target),
       xlim = c(0, 1), ylim = c(0, 1))

  for (i in 2:length(pr_curves)) {
```

```
    lines(pr_curves[[i]]$curve, col = colors[i], lwd = 2, lty = line_types[i])
  }

  # random guess line
  p <- mean(actual_values_numeric)  # Proportion of positive cases
  abline(h = p, col = "black", lwd = 2, lty = 2)

  # Move the legend **outside** the plot area (to the right)
  legend("topright", inset = c(-0.3, 0), legend = model_names, col = colors, lwd = 2, lty = line_types,
         bty = "n", cex = 0.8, xpd = TRUE)
}

#-------------------------------------------------------------------------

## Prediciton Probabilites Plot

extract_predicted_probs <- function(models) {
  if (!is.list(models)) models <- list(models)
  results_list <- list()

  for (model_name in names(models)) {
    model <- models[[model_name]]
    if (!"pred" %in% names(model) || nrow(model$pred) == 0) next

    test_preds <- model$pred
    test_preds_filtered <- test_preds[test_preds$rowIndex %in% unique(test_preds$rowIndex), ]
    if (nrow(test_preds_filtered) == 0) next

    results_list[[model_name]] <- data.frame(
      predicted_prob = test_preds_filtered$Yes,
      actual = test_preds_filtered$obs,
      model = model_name
    )
  }
  if (length(results_list) > 0) return(do.call(rbind, results_list))
  return(NULL)
}

library(ggplot2)

plot_prediction_histograms <- function(results_df) {
  if (is.null(results_df)) return(warning("No valid predictions found!"))

  for (model_name in unique(results_df$model)) {
    print(
      ggplot(subset(results_df, model == model_name), aes(x = predicted_prob, fill = actual)) +
        geom_histogram(binwidth = 0.05, position = "dodge", alpha = 0.7) +
        scale_fill_manual(values = c("No" = "lightblue", "Yes" = "red")) +
        labs(title = paste("Prediction Probability Distribution -", model_name),
             x = "Predicted Probability", y = "Count")
    )
  }
}
```

```
#------------------------------------------------------------------------

## Numeric Plots

evaluate_model <- function(model, target, model_name) {
  test_data <- model$test_data
  predictions <- predict(model, newdata = test_data)
  actuals <- test_data[[target]]

  df <- data.frame(Predicted = predictions, Actual = actuals, Residuals = actuals - predictions)

  p1 <- ggplot(df, aes(x = Predicted, y = Actual)) +
    geom_point(alpha = 0.3) +
    geom_smooth(method = "lm", color = "red") +
    geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "blue") +
    ggtitle(paste("Actual vs. Predicted -", model_name)) +
    xlab("Predicted") + ylab("Actual")

  p2 <- ggplot(df, aes(x = Residuals)) +
    geom_histogram(binwidth = 1, fill = "steelblue", alpha = 0.6) +
    ggtitle(paste("Residuals -", model_name)) +
    xlab("Residuals") + ylab("Count")

  return(list(actual_vs_pred = p1, residual_hist = p2))
}

#------------------------------------------------------------------------
```

Capstone_presentation.Rmd

```
---
title: "DS 795 - Capstone Project"
author: "Jeff Jay"
date: "2025-02-13"
output:
  # pdf_document:
  html_document:
    theme: cerulean
    toc: true
    toc_float:
      collapsed: true
      smooth_scroll: false

---

```{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

# install.packages(c("arm","gbm","class","elmNN","kernlab")

# data mgmt & analysis
library(here)
```
```

```
library(bigrquery)
library(DBI)
library(tidyverse)
library(GGally)
library(lattice)
library(MASS)
library(dplyr)
library(car)
library(corrplot)
library(broom)
library(scales)
library(glue)
library(doParallel)

# Training, validation, tuning, evalutaion
library(caret)
library(yardstick)
library(pROC)
library(PRROC)
library(smotefamily)
library(SHAPforxgboost)
library(iml)
library(shapper)

# Model
library(xgboost)      # Extreme Gradient Boosting
library(gbm)          # Gradient Boosting Machines Log Reg
library(randomForest) # Random Forest Tree
library(glmnet)       # Logistic Regression
library(arm)          # Bayesian Logistic Regression
library(nnet)         # Neural Network
library(class)        # K-Nearest Neighbors
library(kernlab)      # SVM's
library(e1071)        # SVM's
library(naivebayes)   # Naive Bayes

# visualization
library(ggplot2)
library(ggcorrplot)
library(gridExtra)
library(reshape2)

#tables
library(formattable)
library(knitr)
library(kableExtra)

# CPU? Set Cores
num_cores <- parallel::detectCores() - 1
registerDoParallel(cores = num_cores)
cat("Registered", num_cores, "cores for parallel training.\n")

```

Load Function and Figures

```{r load.support.scripts}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

```

Connections

1. MIMIC-IV (Hospital. ICU, & ER)
2. IPUMS: Census/ACS - Weighted Averages of Cross-tabulations
   - Suffolk County Massachusetts grouped by Marital Status, Insurance type, and race.
   > IPUMS provided individualized Harmonizes data Suffolk County Massachusetts grouped by Marital Status, Insurance type, and race.

```{r dataset.build}

# Connect to BigQuery
source(here("scripts", "bigquery_connect.R"))

# Run SQL and process and save data by size
sample_sizes <- c(5, 10, 25, 50, 100)

# Loop each sample size
for (sp in sample_sizes) {
  sample_percent <- sp
  source(here("scripts", "process_data.R"))
  # clear
  rm(list = ls(pattern = "^data"), envir = .GlobalEnv)
  gc()
  message("Dataset saved, memory cleared")
}

```

```{r dataset.load.eda}

# Load 25% Data Sample for EDA
data_hosp <- load_dataset("data_hosp_25.rds")
data_cms  <- load_dataset("data_cms_25.rds")
data_ICU  <- load_dataset("data_ICU_25.rds")

# Load Functions
source(here("scripts", "figs_funcs.R"))

```

EDA

```{r eda.4.4.0}

## Summaries
```

```
summary(data_hosp)
str(data_hosp)
head(data_hosp)
unique_counts <- sapply(data_hosp, function(x) length(unique(x)))
hosp_val_counts <- data.frame(Feature = names(unique_counts), Unique_Count = unique_counts)
hosp_val_counts <- hosp_val_counts[order(-hosp_val_counts$Unique_Count), ]; hosp_val_counts

cat("CMS Diagnosis only DF size:", dim(data_cms), "\n")
cat("ICU patients only DF size:", dim(data_ICU), "\n")
```

Given ICU is a subset of hospital dataset, we will review ICU features using onyl ICU patients, otherwise 0/null would dominate the distributions. Conversly we will only look at hospitalwide variables in the full hospital dataset.

```{r r eda.4.4.1-3}

# numeric target histograms
f_4_4_1_EDA_target_hist <- grid.arrange(grobs = plots$hist_list, ncol = 1)

# binary target bar charts
f_4_4_2_EDA_target_bar <- grid.arrange(grobs = plots$bar_list, ncol = 3)

# summary table
f_4_4_3_EDA_target_percent <- binary_summary_table; f_4_4_3_EDA_target_percent

ggsave("../presentations/figures/f_4_4_1_EDA_target_hist.png", plot = f_4_4_1_EDA_target_hist, width = 5, height = 2, dpi = 300)
ggsave("../presentations/figures/f_4_4_2_EDA_target_bar.png", plot = f_4_4_2_EDA_target_bar, width = 5, height = 2, dpi = 300)
save_kable(f_4_4_3_EDA_target_percent, file = "../presentations/figures/f_4_4_3_EDA_target_percent.png")
```

```{r eda.4.4.4.4-5}

f_4_4_4_EDA_icu_boxplot <- plot_horizontal_box(data_ICU, icu_features)
for (name in names(f_4_4_4_EDA_icu_boxplot)) {
  save_named_plot(f_4_4_4_EDA_icu_boxplot[[name]],
                  paste0("f_4_4_4_boxplot_", name),
                  width = 4, height = 2, dpi = 400)
}

continuous_nonICU <- setdiff(continuous_features, icu_features)
f_4_4_5_EDA_hosp_boxplot <-plot_horizontal_box(data_hosp, continuous_nonICU)
for (name in names(f_4_4_5_EDA_hosp_boxplot)) {
  save_named_plot(f_4_4_5_EDA_hosp_boxplot[[name]],
                  paste0("f_4_4_5_boxplot_", name),
                  width = 4, height = 2, dpi = 400)
}
```

```{r eda.4.4.6.1}
```

```
# categorical features to visualize
categorical_features_1 <- c("age_category", "insurance_category",
                            "marital_status_category", "gender")

# plots (looped by features)
readmission_plots <- lapply(categorical_features_1, function(var) {
  plot_readmission_by_category(cleaned_patient_data, var)
})

grid.arrange(grobs = readmission_plots, ncol = 2)

f_4_4_6_1_EDA_read_bar <- grid.arrange(grobs = readmission_plots, ncol = 2)
ggsave("../presentations/figures/f_4_4_6_1_EDA_read_bar.png", plot = f_4_4_6_1_EDA_read_bar, width
= 6, height = 4, dpi = 400)

```

```{r eda.4.4.6.2}

# categorical features to visualize
categorical_features_2 <- c("is_white", "ICU_flag","race_group")

# plots (looped by features)
readmission_plots2 <- lapply(categorical_features_2, function(var) {
  plot_readmission_by_category(cleaned_patient_data, var)
})

grid.arrange(grobs = readmission_plots2, ncol = 2)

f_4_4_6_2_EDA_read_bar <- grid.arrange(grobs = readmission_plots2, ncol = 2)
ggsave("../presentations/figures/f_4_4_6_2_EDA_read_bar.png", plot = f_4_4_6_2_EDA_read_bar, width
= 6, height = 4, dpi = 400)

```

```{r eda.4.4.7}

figure_4_4_7
ggsave("../presentations/figures/f_4_4_7_readm_patient_cnt.png", plot = figure_4_4_7, width = 6, height =
4, dpi = 400)

```

```{r eda.4.4.8}

target_var <- "readmitted"

# Continuous / Numerical
num_plots <- list()
for (feature in continuous_features) {
  num_plots[[feature]] <- plot_feature_distribution(data_hosp, feature, target_var)
  save_named_plot(num_plots[[feature]], paste0("f_4_4_8_num_", feature), width = 5, height = 3, dpi =
300)
}
```

```
# Binary
binary_plots <- list()
for (feature in binary_features) {
  binary_plots[[feature]] <- plot_feature_distribution(data_hosp, feature, target_var)
  save_named_plot(binary_plots[[feature]], paste0("f_4_4_8_bin_", feature), width = 5, height = 3, dpi =
300)
}

# Categorical
cat_plots <- list()
for (feature in categorical_vars) {
  cat_plots[[feature]] <- plot_feature_distribution(data_hosp, feature, target_var)
  save_named_plot(cat_plots[[feature]], paste0("f_4_4_8_cat_", feature), width = 5, height = 3, dpi = 300)
}


```

```{r eda.4.4.9}

target_var <- "hospital_expire_flag"

# Continuous / Numerical
num_plots <- list()
for (feature in continuous_features) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  num_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_9_num_", feature))
}

# Binary
binary_plots <- list()
for (feature in binary_features) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  binary_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_9_bin_", feature))
}

# Categorical
cat_plots <- list()
for (feature in categorical_vars) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  cat_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_9_cat_", feature))
}

```

```{r eda.4.4.10}

target_var <- "mortality_30_day"

# Continuous / Numerical Features
num_plots <- list()
```

```
for (feature in continuous_features) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  num_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_10_num_", feature))
}

# Binary Features
binary_plots <- list()
for (feature in binary_features) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  binary_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_10_bin_", feature))
}

# Categorical Features
cat_plots <- list()
for (feature in categorical_vars) {
  p <- plot_feature_distribution(data_hosp, feature, target_var)
  cat_plots[[feature]] <- p
  save_named_plot(p, paste0("f_4_4_10_cat_", feature))
}
```

Clear workspace

```{r rm.1}

rm(list = ls(pattern = "f_4_4"))
rm(list = ls(pattern = "plots"))
rm(list = ls(pattern = "_list"))
rm(list = ls(pattern = "figure_4_4"))
rm(list = ls(pattern = "readmitted_"))
rm(list = ls(pattern = "_counts"))
rm(list = ls(pattern = "patient_"))
rm(list = ls(pattern = "^p$"))
rm(list = ls(pattern = "ipums_data"), envir = .GlobalEnv)
rm(list = ls(pattern = "^data"), envir = .GlobalEnv)
gc()

```

Normality and Trasnformaiton

```{r transform.4.5.0}

data_hosp <- load_dataset("data_hosp_100.rds")

normality_results_full <- check_normality_shapiro(
  data_hosp,
  continuous_features,
  sample_size = 500,  # internal sample of 500 is fine
  remove_zeros = TRUE
)

non_normal_features <- normality_results_full %>%
```

```
  filter(statistic < 0.9) %>%
  arrange(statistic) %>%
  pull(Feature)

saveRDS(non_normal_features, here("data", "metadata", "non_normal_features.rds"))

data_hosp <- log_transform_vars(data_hosp, non_normal_features, suffix = "_log")

# Update feature sets
transformed_features <- names(data_hosp)[grepl("_log$", names(data_hosp))]
transformed_base <- gsub("_log$", "", transformed_features)
intersecting_base <- intersect(continuous_features, transformed_base)

updated_continuous_features <- union(
  setdiff(continuous_features, intersecting_base),
  transformed_features
)

features_sdoh_log           <- replace_features_by_suffix(features_sdoh, names(data_hosp))
features_no_sdoh_log        <- replace_features_by_suffix(features_no_sdoh, names(data_hosp))
features_hosp_sdoh_log      <- replace_features_by_suffix(features_hosp_sdoh, names(data_hosp))
features_hosp_icu_log       <- replace_features_by_suffix(features_hosp_icu, names(data_hosp))
features_hosp_icu_sdoh_log  <- replace_features_by_suffix(features_hosp_icu_sdoh, names(data_hosp))
features_non_late_log       <- replace_features_by_suffix(features_non_late, names(data_hosp))
features_non_late_sdoh_log  <- replace_features_by_suffix(features_non_late_sdoh, names(data_hosp))

# Save
saveRDS(data_hosp, here("data", "processed", "data_hosp_100_log.rds"))
saveRDS(updated_continuous_features, here("data", "metadata", "updated_continuous_features.rds"))
saveRDS(features_sdoh_log, here("data", "metadata", "features_sdoh_log.rds"))
saveRDS(features_no_sdoh_log, here("data", "metadata", "features_no_sdoh_log.rds"))
saveRDS(features_non_late_log, here("data", "metadata", "features_non_late_log.rds"))
saveRDS(features_non_late_sdoh_log, here("data", "metadata", "features_non_late_sdoh_log.rds"))

rm(list = ls(pattern = "^data"), envir = .GlobalEnv)
gc()

```

```{r transform.4.5.1}

f_4_5_1_shapiro <- render_normality_table(normality_results_full, end = 0.8)
save_kable(f_4_5_1_shapiro, file = "../presentations/figures/f_4_5_1_shapiro.png")

```

```{r transform.4.5.2}

## Load 10% and transform. Ruun QQ plots (100% to much data)
data_hosp <- load_dataset("data_hosp_10.rds")

qq_plots <- plot_qq_by_variable(data_hosp, continuous_features)

for (feature in names(qq_plots)) {
  save_named_plot(qq_plots[[feature]], paste0("f_4_5_2_qq_", feature), width = 4, height = 3, dpi = 300)
```

```
}

data_hosp <- log_transform_vars(data_hosp, non_normal_features, suffix = "_log")

# Save
saveRDS(data_hosp, here("data", "processed", "data_hosp_10_log.rds"))

# ----------------------------------------------------------------

data_hosp_25 <- load_dataset("data_hosp_25.rds")
data_hosp_25 <- log_transform_vars(data_hosp_25, non_normal_features, suffix = "_log")

# Save
saveRDS(data_hosp_25, here("data", "processed", "data_hosp_25_log.rds"))

# ----------------------------------------------------------------

data_hosp_50 <- load_dataset("data_hosp_50.rds")
data_hosp_50 <- log_transform_vars(data_hosp_50, non_normal_features, suffix = "_log")

# Save
saveRDS(data_hosp_50, here("data", "processed", "data_hosp_50_log.rds"))

```

```{r transform.4.5.3}

## Load transformed full dataset and render results
data_hosp_100 <- load_dataset("data_hosp_100_log.rds")

normality_trans_results <- check_normality_shapiro(data_hosp_100, transformed_features, sample_size =
500, remove_zeros = TRUE)
f_4_5_3_shapiro <- render_normality_table(normality_trans_results, end = 0.8)
save_kable(f_4_5_3_shapiro, file = "../presentations/figures/f_4_5_3_shapiro.png")

```

```{r transform.4.5.4}

#replot 10% sample after transformation
qq_plots_2 <- plot_qq_by_variable(data_hosp, transformed_features)

for (feature in names(qq_plots_2)) {
  save_named_plot(qq_plots_2[[feature]], paste0("f_4_5_4_qq_", feature), width = 4, height = 3, dpi = 300)
}

```

Clear workspace

```{r rm.2}

rm(list = ls(pattern = "f_4_5"))
rm(list = ls(pattern = "plots"))
rm(list = ls(pattern = "_list"))
```

```
rm(list = ls(pattern = "results_"))
rm(list = ls(pattern = "^data"), envir = .GlobalEnv)
gc()
```

Statistical Testing LOS

```{r tests.4.6.1.LOS.Parametric}

data_hosp <- load_dataset("data_hosp_25_log.rds")

#LM
# Collinearity Check
clean_predictors_los <- setdiff(c(categorical_vars, updated_continuous_features, binary_features),
c("is_white"))
LOS_LM_Model <- run_linear_models(data_hosp, "length_of_stay", clean_predictors_los)
f_4_6_1_los_lm_vif <- render_vif_summary_table(LOS_LM_Model[["length_of_stay"]], target_name =
"length_of_stay")
save_kable(f_4_6_1_los_lm_vif, file = "../presentations/figures/f_4_6_1_los_lm_vif.png")

```

```{r tests.4.6.2.LOS.Parametric}

# Remove illogical, collinear, data leagae features
filter_los = c("total_icu_stays_log", "length_of_stay_log","los_icu_log","total_icu_hours_log","is_white",
"ICU_flag", "mortality_30_day")

clean_predictors_los_2 <- setdiff(c(categorical_vars, updated_continuous_features, binary_features),
filter_los)
LOS_LM_Model_2 <- run_linear_models(data_hosp, "length_of_stay", clean_predictors_los_2)

f_4_6_2_los_lm <- render_lm_summary_table(LOS_LM_Model_2[["length_of_stay"]], target_name =
"length_of_stay", exclude_terms = filter_los, begin= .2, end = 1)
save_kable(f_4_6_2_los_lm, file = "../presentations/figures/f_4_6_2_los_lm.png")

```

```{r tests.4.6.3.LOS.Parametric}

# ANOVA for categorical features
f_4_6_3_los_anova <- render_anova_summary_table(data_hosp,
                                    target = "length_of_stay",
                                    predictors = categorical_vars)

save_kable(f_4_6_3_los_anova, file = "../presentations/figures/f_4_6_3_los_anova.png")

```

```{r tests.4.6.4.LOS.Non-Parametric}

# Loop over binary features and run test vs. LOS
wilcox_binary_results <- lapply(binary_features, function(bin_var) {
  run_wilcoxon_tests(
```

```
    data = data_hosp,
    continuous_vars = "length_of_stay",
    binary_var = bin_var,
    target_label = bin_var
  )
})

# Combine and render
wilcox_binary_df <- do.call(rbind, wilcox_binary_results)
f_4_6_4_los_wilcox <- render_wilcoxon_summary_table(wilcox_binary_df, begin = .2, end = .9)
save_kable(f_4_6_4_los_wilcox, file = "../presentations/figures/f_4_6_4_los_wilcox.png")

wilcox_ns <- c("wound_debridement_flag", "stroke_prevention_flag",
               "shock_management_flag", "neurosurgery_flag")
```

> Wilcox Not significant: wound_debridement_flag, stroke_prevention_flag, shock_management_flag, neurosurgery_flag

```{r tests.4.6.5.LOS.Non-Parametric}

kruskal_los_df <- run_kruskal_tests(
  data = data_hosp,
  numeric_vars = categorical_vars,
  target_var = "length_of_stay"
)

f_4_6_5_los_kruskal <- render_wilcoxon_summary_table(kruskal_los_df, title = "Kruskal-Wallis Test: Categorical Features vs LOS")
save_kable(f_4_6_5_los_kruskal, file = "../presentations/figures/f_4_6_5_los_kruskal.png")
```

```{r tests.4.6.6.LOS.Non-Parametric}

f_4_6_6_los_cor <- plot_correlation_heatmap(data_hosp, updated_continuous_features)
ggsave("../presentations/figures/f_4_6_6_los_cor.png", plot = f_4_6_6_los_cor, width = 11, height = 10, dpi = 300)
```

```{r tests.4.6.los.summary}

filter_los = c("total_icu_stays_log",
"length_of_stay_log","los_icu_log","total_icu_hours_log","is_white","ICU_flag", "mortality_30_day")
wilcox_ns <- c("wound_debridement_flag", "stroke_prevention_flag",
               "shock_management_flag", "neurosurgery_flag")
features_to_remove_los <- c(filter_los, wilcox_ns)
saveRDS(features_to_remove_los, here("data", "metadata", "features_to_remove_los.rds"))
features_no_sdoh_log_los <- setdiff(features_no_sdoh_log, features_to_remove_los)
features_sdoh_log_los <- setdiff(features_sdoh_log, features_to_remove_los)
```

Reviewing the output from the testing we see the features below either show little to no effect, do not fit the anakysis, or represent data leakage and are too predictive.

Statistical Testing Binary
> non-transformed continuous features

```{r tests.4.6.7.binary.Parametric}

clean_predictors_binary_1 <- setdiff(c(categorical_vars, continuous_features, binary_features), c("is_white","hospital_expire_flag","mortality_30_day","readmitted"))
GLM_binary_models <- run_logistic_models(data_hosp, target_vars, clean_predictors_binary_1)

f_4_6_7_readmit_log <- render_vif_summary_table(GLM_binary_models[["readmitted"]], target_name = "readmitted", begin = .4, end = 1)
f_4_6_7_mort30_log <- render_vif_summary_table(GLM_binary_models[["mortality_30_day"]], target_name = "mortality_30_day", begin = .4, end = 1)
f_4_6_7_morthosp_log <-render_vif_summary_table(GLM_binary_models[["hospital_expire_flag"]], target_name = "hospital_expire_flag", begin = .4, end = 1)

save_kable(f_4_6_7_readmit_log, file = "../presentations/figures/f_4_6_7_readmit_binary.png")
save_kable(f_4_6_7_mort30_log, file = "../presentations/figures/f_4_6_7_mort30_binary.png")
save_kable(f_4_6_7_morthosp_log, file = "../presentations/figures/f_4_6_7_morthosp_binary.png")

```

> Based on high VIF, Drop 1 or 2 of the income features

```{r tests.4.6.8.binary.Parametric}

# Remove illogical, collinear, data leagae features
filter_binary_1 = c("weighted_income","weighted_inc_welfare") # , "weighted_inc_welfare"

clean_predictors_binary_2 <- setdiff(clean_predictors_binary_1, filter_binary_1)
GLM_binary_models_2 <- run_logistic_models(data_hosp, target_vars, clean_predictors_binary_2)

f_4_6_8_read_log <- render_logit_summary_table(GLM_binary_models_2[["readmitted"]], target_name = "readmitted", exclude_terms = filter_binary_1, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05
f_4_6_8_mort30_log <- render_logit_summary_table(GLM_binary_models_2[["mortality_30_day"]], target_name = "mortality_30_day", exclude_terms = filter_binary_1, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05
f_4_6_8_morthosp_log <- render_logit_summary_table(GLM_binary_models_2[["hospital_expire_flag"]], target_name = "hospital_expire_flag", exclude_terms = filter_binary_1, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05

save_kable(f_4_6_8_read_log, file = "../presentations/figures/f_4_6_8_read_binary.png")
save_kable(f_4_6_8_mort30_log, file = "../presentations/figures/f_4_6_8_mort30_binary.png")
save_kable(f_4_6_8_morthosp_log, file = "../presentations/figures/f_4_6_8_morthosp_binary.png")

```

> Reviewing the T stat and odd ratio

Bases on the Z-stat and odds ratio we can see features that are heavily influencing the the model which shuold not be included:
> discharge_location_categoryDIED, discharge_location_categoryHOSPICE. This makes sence as they are highly related, or wholy inclusive or exclusive of the targets, so we will remove the whole feature.

```{r tests.4.6.9.binary.Parametric}

filter_binary_2 = c("weighted_income", "weighted_inc_welfare","discharge_location_categoryDIED", "discharge_location_categoryHOSPICE")

f_4_6_9_read_glm <- render_logit_summary_table(GLM_binary_models_2[["readmitted"]], target_name = "Readmitted", exclude_terms = filter_binary_2, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05
f_4_6_9_mort30_glm <- render_logit_summary_table(GLM_binary_models_2[["mortality_30_day"]], target_name = "mortality_30_day", exclude_terms = filter_binary_2, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05
f_4_6_9_morthosp_glm <- render_logit_summary_table(GLM_binary_models_2[["hospital_expire_flag"]], target_name = "hospital_expire_flag", exclude_terms = filter_binary_2, rank_by = "p.value", begin = .3, end = 1) # p_filter = .05


save_kable(f_4_6_9_read_glm, file = "../presentations/figures/f_4_6_9_read_glm.png")
save_kable(f_4_6_9_mort30_glm, file = "../presentations/figures/f_4_6_9_mort30_glm.png")
save_kable(f_4_6_9_morthosp_glm, file = "../presentations/figures/f_4_6_9_morthosp_glm.png")

```
The final list looks suitable to include for feature selection

```{r tests.4.6.10.binary.Non-Parametric}

# Run tests
wilcox_readmit_df <- run_wilcoxon_tests(
  data = data_hosp,
  continuous_vars = continuous_features,
  binary_var = "readmitted",
  target_label = "Readmission"
)

wilcox_mortality_df <- run_wilcoxon_tests(
  data = data_hosp,
  continuous_vars = continuous_features,
  binary_var = "mortality_30_day",
  target_label = "30-Day Mortality"
)

wilcox_hosp_mort_df <- run_wilcoxon_tests(
  data = data_hosp,
  continuous_vars = continuous_features,
  binary_var = "hospital_expire_flag",
  target_label = "Hospital Mortality"
)

# Render results
f_4_6_10_read_wilcox <- render_wilcoxon_summary_table(wilcox_readmit_df, title = "Wilcoxon Test: Readmission",begin = .3, end = 1)
```

```
f_4_6_10_mort30_wilcox <- render_wilcoxon_summary_table(wilcox_mortality_df, title = "Wilcoxon Test: 30-Day Mortality",begin = .3, end = 1)
f_4_6_10_morthosp_wilcox <- render_wilcoxon_summary_table(wilcox_hosp_mort_df, title = "Wilcoxon Test: Hospital Mortality",begin = .3, end = 1)


save_kable(f_4_6_10_read_wilcox, file = "../presentations/figures/f_4_6_10_read_wilcox.png")
save_kable(f_4_6_10_mort30_wilcox, file = "../presentations/figures/f_4_6_10_mort30_wilcox.png")
save_kable(f_4_6_10_morthosp_wilcox, file = "../presentations/figures/f_4_6_10_morthosp_wilcox.png")
```

all feautres we significant for 30-Day mort and Readmission, and "prior_hospital_visits_1yr" (.9285) was not for in hospital mortality
> we will remove these given the high p stat

```{r tests.4.6.11.binary.Non-Parametric}

all_chisq_results_1 <- run_chisq_for_targets(data_hosp, c("readmitted", "mortality_30_day","hospital_expire_flag"))
all_chisq_results_clean <- all_chisq_results_1 %>% filter(!is.na(P_Value))
f_4_6_11_a_chisq <- render_chisq_summary_table(all_chisq_results_clean, begin = 0.4, end = 1, title = "Chi-Squared Results: All Binary Targets")

save_kable(f_4_6_11_a_chisq, file = "../presentations/figures/f_4_6_11_a_chisq.png")

filter_cat = c("mortality_30_day","discharge_location_category")
clean_cat <- setdiff(categorical_vars, filter_cat)

all_chisq_results_2 <- run_chisq_for_targets(data = data_hosp, target_var = c( "readmitted", "mortality_30_day","hospital_expire_flag"), cat_vars = clean_cat)
all_chisq_results_clean_2 <- all_chisq_results_2 %>% filter(!is.na(P_Value))
f_4_6_11_b_chisq <- render_chisq_summary_table(all_chisq_results_clean_2, begin = 0.4, end = 1, title = "Chi-Squared Results: All Binary Targets")

save_kable(f_4_6_11_b_chisq, file = "../presentations/figures/f_4_6_11_b_chisq.png")

```

for Mortality targets discharge location is too significant and represents some data leakage in the case of "DIED" "HOSPICE" and for the entire Hopsital MORT target. This feature may be rmeoived entiredly or kept in part.

```{r tests.4.6.12.binary.Non-Parametric}

clean_features_binary <- setdiff((binary_features), c("hospital_expire_flag","mortality_30_day","readmitted"))
fishers_df_read <- run_fishers_tests(data = data_hosp, binary_var = clean_features_binary, target_var = "readmitted", target_label = "Readmitted")
fishers_df_30mort <- run_fishers_tests(data = data_hosp, binary_var = clean_features_binary, target_var = "mortality_30_day", target_label = "mortality_30_day")
fishers_df_hospmort <- run_fishers_tests(data = data_hosp, binary_var = clean_features_binary, target_var = "hospital_expire_flag", target_label = "hospital_expire_flag")

f_4_6_12_readmit_fishers <- render_fisher_summary_table(fishers_df_read, begin = .4, end = 1)
```

```
f_4_6_12_30mort_fishers <- render_fisher_summary_table(fishers_df_30mort, begin = .4, end = 1)
f_4_6_12_hospmort_fishers <- render_fisher_summary_table(fishers_df_hospmort, begin = .4, end = 1)

save_kable(f_4_6_12_readmit_fishers, file = "../presentations/figures/f_4_6_12_readmit_fishers.png")
save_kable(f_4_6_12_30mort_fishers, file = "../presentations/figures/f_4_6_12_30mort_fishers.png")
save_kable(f_4_6_12_hospmort_fishers, file = "../presentations/figures/f_4_6_12_hospmort_fishers.png")
```

Late stage features such as ecmo, cpr, prolonged ventilation, Pressor Meds all may be too late to be supportive in terms of intervention, but in the context of CMS retrospective analysis, hospital quality models, and outcome interpretation that could be useful, espcially to evalute clusters on final models. Given that we will keep them fpor now but perhaps train a second model type that removed the "late stage" features that could represent data leakage or be too late to be useful in realtime.

> In-hospital CPR survival rates are low (~15-25%), often occures minuntws before death
> ECMO is used in extreme cases of cardiac or respiratory failure, usually hours to days before before end of life
> Pressor Meds are used to maintain blood pressure in shock (e.g., sepsis, cardiac), and can increase ovedr time especially during severe instability
> Prolonged ventilation is only known after 96h, so cannot be used as any early indicator

```{r tests.4.6.binary.summary}

filter_h_mort = c("mortality_30_day", "hospital_expire_flag", "discharge_location_category", "readmitted", "prior_hospital_visits_1yr")
filter_30_mort = c("mortality_30_day", "hospital_expire_flag", "discharge_location_category", "readmitted")
filter_readmit = c("mortality_30_day", "hospital_expire_flag", "discharge_location_category", "readmitted")

features_no_sdoh_log              <- readRDS(here("data", "metadata", "features_no_sdoh_log.rds"))
features_sdoh_log                 <- readRDS(here("data", "metadata", "features_sdoh_log.rds"))

features_no_sdoh_log_h_mort       <- setdiff(features_no_sdoh_log, filter_h_mort)
features_sdoh_log_h_mort          <- setdiff(features_sdoh_log, filter_h_mort)
saveRDS(features_no_sdoh_log_h_mort, here("data", "metadata", "features_no_sdoh_log_h_mort.rds"))
saveRDS(features_sdoh_log_h_mort, here("data", "metadata", "features_sdoh_log_h_mort.rds"))

features_no_sdoh_log_30_mort      <- setdiff(features_no_sdoh_log, filter_30_mort)
features_sdoh_log_30_mort         <- setdiff(features_sdoh_log, filter_30_mort)
saveRDS(features_no_sdoh_log_30_mort, here("data", "metadata", "features_no_sdoh_log_30_mort.rds"))
saveRDS(features_sdoh_log_30_mort, here("data", "metadata", "features_sdoh_log_30_mort.rds"))

features_no_sdoh_log_readmit      <- setdiff(features_no_sdoh_log, filter_readmit)
features_sdoh_log_readmit         <- setdiff(features_sdoh_log, filter_readmit)
saveRDS(features_no_sdoh_log_readmit, here("data", "metadata", "features_no_sdoh_log_readmit.rds"))
saveRDS(features_sdoh_log_readmit, here("data", "metadata", "features_sdoh_log_readmit.rds"))
```

Clear workspace

```{r rm.3}

rm(list = ls(pattern = "f_4_6"))
rm(list = ls(pattern = "plots"))
rm(list = ls(pattern = "_list"))
rm(list = ls(pattern = "fishers"))
rm(list = ls(pattern = "wilcox"))
rm(list = ls(pattern = "chisq"))
rm(list = ls(pattern = "LM_Model"))
rm(list = ls(pattern = "binary_model"))
rm(list = ls(pattern = "^data"), envir = .GlobalEnv)
gc()

```

Model Training & Feature Selection - Continuous

```{r feat.sel.4.7.0.los}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 10% Data Sample for EDA
data_hosp <- load_dataset("data_hosp_10_log.rds")
# Load Functions
source(here("scripts", "figs_funcs.R"))

features_sdoh_log           <- readRDS(here("data", "metadata", "features_sdoh_log.rds"))
features_no_sdoh_log        <- readRDS(here("data", "metadata", "features_no_sdoh_log.rds"))
features_non_late_log       <- readRDS(here("data", "metadata", "features_non_late_log.rds"))
features_non_late_sdoh_log  <- readRDS(here("data", "metadata", "features_non_late_sdoh_log.rds"))

features_to_remove_los <- readRDS(here("data", "metadata", "features_to_remove_los.rds"))
features_no_sdoh_log_los <- setdiff(features_no_sdoh_log, features_to_remove_los)
features_sdoh_log_los <- setdiff(features_sdoh_log, features_to_remove_los)


```

```{r feat.sel.4.7.los.inputs}

data_hosp <- load_dataset("data_hosp_10_log.rds")
### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_10"
target <- "length_of_stay"
file_name <- "los_10_feat_sel"
slices <- 4
model_run <- "los_10_feat_sel"
split_type <- "patient"
```

```
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r feat.sel.4.7.los.train}

set.seed(42)

##############################
### Logistic Regression
##############################

train_and_assign(
  model_type = "glm",
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = NULL
)

train_and_assign(
  model_type = "glm",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = NULL
)

##############################
### GLMNet (Ridge, Lasso, Elastic Net)
##############################

tuneGrid_glmnet <- expand.grid(
  alpha =  1,  # Ridge = 0, Elastic = 0.5, Lasso = 1
  lambda = 0.1
  )

train_and_assign(
  model_type = "glmnet",
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = tuneGrid_glmnet
)

train_and_assign(
  model_type = "glmnet",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = tuneGrid_glmnet
)

##############################
### Bayesian Logistic Regression
##############################
```

```
train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = NULL
)

train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = NULL
)

################################
### Gradient Boosted Trees (XGBoost)
################################

tuneGrid_xgb <- expand.grid(
  nrounds = 200,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = tuneGrid_xgb
)

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = tuneGrid_xgb
)

################################
### Gradient Boosting Machines (GBM)
################################

tuneGrid_gbm <- expand.grid(
  interaction.depth = 3,
  n.trees = 100,
  shrinkage = 0.1,
  n.minobsinnode = 10
)

train_and_assign(
  model_type = "gbm",
```

```
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = tuneGrid_gbm
)

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = tuneGrid_gbm
)

##############################
### Random Forest
##############################

tuneGrid_rf <- expand.grid(mtry = 2)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = features_sdoh_log_los,
  tuneGrid = tuneGrid_rf
)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = features_no_sdoh_log_los,
  tuneGrid = tuneGrid_rf
)

##############################
### Final Model Verification
##############################

# print(ls(pattern = model_run))

```

```{r feat.sel.4.7.los.save}

matching_models_los_fs <- mget(
  ls(pattern = model_run, envir = .GlobalEnv),
  envir = .GlobalEnv
)

save_models(matching_models_los_fs, subfolder = "length_of_stay_10")

```

```{r feat.sel.4.7.los}

# matching_models_los_fs <- mget(
#   ls(pattern = "^gbm", envir = .GlobalEnv),
```

```
#   envir = .GlobalEnv
# )

feature_selection_results_los <- list()

for (model_name in names(matching_models_los_fs)) {
  model_obj <- matching_models_los_fs[[model_name]]
  model_type <- model_obj$method

  message("Feature selection for model: ", model_name)

  model_input <- get_model_data_and_features(model_obj, target = "length_of_stay")

selected_feats <- feature_selection(
  model = model_obj,
  target = "length_of_stay",
  features = model_input$features,
  dataset = model_input$data,
  return_stats = TRUE
)

  if ("Overall" %in% colnames(selected_feats)) {
    selected_names <- selected_feats$feature[selected_feats$Overall > 0]
  } else {
    selected_names <- selected_feats$feature
  }
  removed_feats <- setdiff(model_input$features, selected_names)

  # results
  feature_selection_results_los[[model_name]] <- list(
    selected = selected_feats,
    removed = removed_feats
  )

  saveRDS(
  feature_selection_results_los,
  here::here("data", "metadata", "feature_selection_results_los.rds")
  )

}

```

```{r feat.sel.4.7.los.review}

feature_selection_results_los <- readRDS(here::here("data", "metadata",
"feature_selection_results_los.rds"))

# Get all unique removed features across all models
all_removed <- unique(unlist(lapply(feature_selection_results_los, function(x) x$removed)))

# Create a matrix: rows = features, cols = models
removed_matrix <- sapply(names(feature_selection_results_los), function(model_name) {
  removed <- feature_selection_results_los[[model_name]]$removed
  all_removed %in% removed
```

```
})

# Convert to data frame
removed_df <- as.data.frame(removed_matrix)
rownames(removed_df) <- all_removed

removed_df_pretty <- removed_df
removed_df_pretty[] <- lapply(removed_df_pretty, function(col) ifelse(col, "X", "-"))

t_4_7_1_removed_los <- removed_df_pretty
write.csv(t_4_7_1_removed_los, file = "../presentations/figures/t_4_7_1_removed_los.csv", row.names =
TRUE, fileEncoding = "UTF-8")

```

```{r feat.sel.4.7.los.cleanup}

rm(list = ls(pattern = "los_10_feat_sel"), envir = .GlobalEnv)
gc()

```

Final Model Training and Tuning - Length of Stay

```{r tunning.4.7.los.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_25_log.rds")

source(here("scripts", "figs_funcs.R"))

features_sdoh_log           <- readRDS(here("data", "metadata", "features_sdoh_log.rds"))
features_no_sdoh_log        <- readRDS(here("data", "metadata", "features_no_sdoh_log.rds"))
features_non_late_log       <- readRDS(here("data", "metadata", "features_non_late_log.rds"))
features_non_late_sdoh_log  <- readRDS(here("data", "metadata", "features_non_late_sdoh_log.rds"))

features_to_remove_los <- readRDS(here("data", "metadata", "features_to_remove_los.rds"))
features_no_sdoh_log_los <- setdiff(features_no_sdoh_log, features_to_remove_los)
features_sdoh_log_los <- setdiff(features_sdoh_log, features_to_remove_los)

feature_selection_results_los <- readRDS(here("data", "metadata", "feature_selection_results_los.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_25"
target <- "length_of_stay"
file_name <- "los_25_tuning.csv"
slices <- 4
model_run <- "los_25_tuning"
```

```
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r tunning.4.7.los.train}

################################
### Tunning Grids
################################

# XGBoost
tuneGrid_xgb <- expand.grid(
  nrounds = c(100, 200),
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.1, 0.3),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

# GBM
tuneGrid_gbm <- expand.grid(
  interaction.depth = c(1, 3, 5),
  n.trees = c(50, 100),
  shrinkage = c(0.05, 0.1),
  n.minobsinnode = c(5, 10)
)

# Random Forest
tuneGrid_rf <- expand.grid(
  mtry = c(10) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

################################
### XGBoost (Gradient Boosted Trees)
################################

nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_sdoh_log_los),
  tuneGrid = tuneGrid_xgb
)

train_and_assign(
```

```
  model_type = "xgbTree",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "xgbTree", "base", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_no_sdoh_log_los),
  tuneGrid = tuneGrid_xgb
)

################################
### GBM (Gradient Boosting Machines)
################################

interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_sdoh_log_los),
  tuneGrid = tuneGrid_gbm
)

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "gbm", "base", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_no_sdoh_log_los),
  tuneGrid = tuneGrid_gbm
)

################################
### Random Forest
################################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_sdoh_log_los),
  tuneGrid = tuneGrid_rf
)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "rf", "base", "length_of_stay", "los_10_feat_sel", feature_selection_results_los,
features_no_sdoh_log_los),
  tuneGrid = tuneGrid_rf
)

```

```{r tunning.4.7.los.save}

matching_models_los_fs <- mget(
  ls(pattern = model_run, envir = .GlobalEnv),
  envir = .GlobalEnv
)

save_models(matching_models_los_fs, subfolder = "length_of_stay_25")

rm(list = ls(pattern = "los_25"))
gc()

```

```{r Final.4.7.1.los.inputs}

### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 100% Data  for Finak Model
data_hosp <- load_dataset("data_hosp_50_log.rds")

source(here("scripts", "figs_funcs.R"))

features_sdoh_log           <- readRDS(here("data", "metadata", "features_sdoh_log.rds"))
features_no_sdoh_log        <- readRDS(here("data", "metadata", "features_no_sdoh_log.rds"))
features_non_late_log       <- readRDS(here("data", "metadata", "features_non_late_log.rds"))
features_non_late_sdoh_log  <- readRDS(here("data", "metadata", "features_non_late_sdoh_log.rds"))

features_to_remove_los <- readRDS(here("data", "metadata", "features_to_remove_los.rds"))
features_no_sdoh_log_los <- setdiff(features_no_sdoh_log, features_to_remove_los)
features_sdoh_log_los <- setdiff(features_sdoh_log, features_to_remove_los)

feature_selection_results_los <- readRDS(here("data", "metadata", "feature_selection_results_los.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Final_100"
target <- "length_of_stay"
file_name <- "los_100_final.csv"
slices <- 1
model_run <- "los_100_final"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 1
final_model <- TRUE

```

```{r Final.4.7.1.los.train}

# ####################################
# ###        XGBoost            ###
# ####################################
#
# best_grid_xgb <- expand.grid(
#   nrounds = 200,
#   max_depth = 6,
#   eta = 0.1,
#   gamma = 0,
#   colsample_bytree = 1,
#   min_child_weight = 1,
#   subsample = 1
# )
#
# train_and_assign(
#   model_type = "xgbTree",
#   feature_set_name = "Base_SDOH",
#   selected_features = get_training_ready_features("xgbTree", "base_sdoh", "length_of_stay",
"los_10_feat_sel", feature_selection_results_los, features_sdoh_log_los),
#   tuneGrid = best_grid_xgb,
#   final = TRUE
# )
#
# train_and_assign(
#   model_type = "xgbTree",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features("xgbTree", "base", "length_of_stay", "los_10_feat_sel",
feature_selection_results_los, features_no_sdoh_log_los),
#   tuneGrid = best_grid_xgb,
#   final = TRUE
# )
#
# ####################################
# ###        GBM                ###
# ####################################
#
# best_grid_gbm <- expand.grid(
#   interaction.depth = 5,
#   n.trees = 100,
#   shrinkage = 0.1,
#   n.minobsinnode = 5
# )
#
# train_and_assign(
#   model_type = "gbm",
#   feature_set_name = "Base_SDOH",
#   selected_features = get_training_ready_features("gbm", "base_sdoh", "length_of_stay",
"los_10_feat_sel", feature_selection_results_los, features_sdoh_log_los),
#   tuneGrid = best_grid_gbm,
#   final = TRUE
# )
#
# train_and_assign(
```

```
#   model_type = "gbm",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features("gbm", "base", "length_of_stay", "los_10_feat_sel",
feature_selection_results_los, features_no_sdoh_log_los),
#   tuneGrid = best_grid_gbm,
#   final = TRUE
# )

###################################
###      Random Forest         ###
###################################


# Random Forest
tuneGrid_rf <- expand.grid(
  mtry = c(6) # mtry = c(2, 4, 6, 8)
)

# train_and_assign(
#   model_type = "rf",
#   feature_set_name = "Base_SDOH",
#   selected_features = get_training_ready_features("rf", "base_sdoh", "length_of_stay", "los_10_feat_sel",
feature_selection_results_los, features_sdoh_log_los),
#   tuneGrid = tuneGrid_rf,
#   final = TRUE
# )

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = get_training_ready_features("rf", "base", "length_of_stay", "los_10_feat_sel",
feature_selection_results_los, features_no_sdoh_log_los),
  tuneGrid = tuneGrid_rf,
  final = TRUE
)

```

```{r final.4.7.los.save}

matching_models_los_fs <- mget(
  ls(pattern = model_run, envir = .GlobalEnv),
  envir = .GlobalEnv
)

save_models(matching_models_los_fs, subfolder = "length_of_stay_final")

rm(list = ls(pattern = model_run), envir = .GlobalEnv)
gc()

```

```{r final.4.7.los.load}

load_models("models/length_of_stay_final")
```

```

### Model Training & Feature Selection - Mort 30

```{r feat.sel.4.7.0.binary}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 10% Data Sample for EDA
data_hosp <- load_dataset("data_hosp_10_log.rds")
# Load Functions
source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort         <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort            <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort           <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit           <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

```

```{r feat.sel.4.7.mort30.inputs}

### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"
data_hosp <- load_dataset("data_hosp_10_log.rds")

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_10"
target <- "mortality_30_day"
file_name <- "mort30_feat_sel.csv"
slices <- 4
model_run <- "mort30_feat_sel"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r feat.sel.4.7.mort30.train}

set.seed(42)
```
```

```
# === Feature Sets
base_feats  <- features_no_sdoh_log_30_mort
sdoh_feats  <- features_sdoh_log_30_mort

############################
#### XGBoost
############################

tuneGrid_xgb <- expand.grid(
  nrounds = 200,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### GBM
############################

tuneGrid_gbm <- expand.grid(
  interaction.depth = 3,
  n.trees = 100,
  shrinkage = 0.1,
  n.minobsinnode = 10
)

train_and_assign(model_type = "gbm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "gbm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### Random Forest
############################

tuneGrid_rf <- expand.grid(mtry = 2)

train_and_assign(model_type = "rf", feature_set_name = "Base", selected_features = base_feats, tuneGrid
= tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "30_mort_10", clear = TRUE)
```

```
train_and_assign(model_type = "rf", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### Neural Network
############################

tuneGrid_nnet <- expand.grid(size = 5, decay = 0.1)

train_and_assign(model_type = "nnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "nnet", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### Logistic Regression
############################

train_and_assign(model_type = "glm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "glm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### GLMNet (Ridge, Lasso, Elastic Net)
############################

tuneGrid_glmnet <- expand.grid(
  alpha = .5,
  lambda = 0.1
)

train_and_assign(model_type = "glmnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "glmnet", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### Bayesian Logistic Regression
############################

train_and_assign(model_type = "bayesglm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "30_mort_10", clear = TRUE)
```

```
train_and_assign(model_type = "bayesglm", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### SVM (Radial)
############################

# tuneGrid_svm <- expand.grid(C = 1, sigma = 0.05)
#
# train_and_assign(model_type = "svmRadial", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_svm)
# save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_mort_10", clear = TRUE)
#
# train_and_assign(model_type = "svmRadial", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_svm)
# save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### Naive Bayes
############################

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "30_mort_10", clear = TRUE)

############################
#### K-Nearest Neighbors
############################

tuneGrid_knn <- expand.grid(k = 10)

train_and_assign(model_type = "knn", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_mort_10", clear = TRUE)

train_and_assign(model_type = "knn", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_mort_10", clear = TRUE)


```

```{r feat.sel.4.7.mort30.save}

matching_models_los_fs <- mget(
  ls(pattern = model_run, envir = .GlobalEnv),
  envir = .GlobalEnv
)

save_models(matching_models_los_fs, subfolder = "mort30_10")
```

```

```{r feat.sel.4.7.mort30}

feature_selection_results_30mort <- list()

# Load from saved models
files <- list.files(here("models", "30_mort_10"), pattern = "\\.rds$", full.names = TRUE)

for (file in files) {
  model_name <- sub("\\.rds$", "", basename(file))
  model_obj <- readRDS(file)

  if (grepl("^knn", model_name)) {
    message("Skipping KNN model: ", model_name)
    next
  }


  message("Feature selection for model: ", model_name)

  model_input <- get_model_data_and_features(model_obj, target = "mortality_30_day")

  selected_feats <- feature_selection(
    model = model_obj,
    target = "mortality_30_day",
    features = model_input$features,
    dataset = model_input$data,
    return_stats = TRUE
  )

  if ("Overall" %in% colnames(selected_feats)) {
    selected_names <- selected_feats$feature[selected_feats$Overall > 0]
  } else {
    selected_names <- selected_feats$feature
  }

  removed_feats <- setdiff(model_input$features, selected_names)

  feature_selection_results_30mort[[model_name]] <- list(
    selected = selected_feats[selected_feats$feature %in% selected_names, , drop = FALSE],
    removed = removed_feats
  )

  saveRDS(
    feature_selection_results_30mort,
    here::here("data", "metadata", "feature_selection_results_30mort.rds")
  )

  rm(model_obj); gc()  # Clear memory
}


```
```

```{r feat.sel.4.7.30mort.review}

mort30_results <- readRDS(here::here("data", "metadata", "feature_selection_results_30mort.rds"))

# Get all unique removed features across all models
all_removed <- unique(unlist(lapply(feature_selection_results_30mort, function(x) x$removed)))

# Create a matrix: rows = features, cols = models
removed_matrix <- sapply(names(feature_selection_results_30mort), function(model_name) {
  removed <- feature_selection_results_30mort[[model_name]]$removed
  all_removed %in% removed
})

# Convert to data frame
removed_df <- as.data.frame(removed_matrix)
rownames(removed_df) <- all_removed

removed_df_pretty <- removed_df
removed_df_pretty[] <- lapply(removed_df_pretty, function(col) ifelse(col, "X", "-"))

t_4_7_1_removed_30mort <- removed_df_pretty
write.csv(t_4_7_1_removed_30mort, file = "../presentations/figures/t_4_7_1_removed_30mort.csv",
row.names = TRUE, fileEncoding = "UTF-8")

```

```{r tunning.4.7.30mort.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_25_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort         <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort            <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort           <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit           <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_30mort <- readRDS(here("data", "metadata",
"feature_selection_results_30mort.rds"))

set.seed(42)
train_function = "train_single_model2"
```

```
data <- data_hosp
dataset_name <- "Samp_25"
target <- "mortality_30_day"
file_name <- "mort30_tuning.csv"
slices <- 4
model_run <- "Mort30_tuning"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r tunning.4.7.30mort.train}

################################
### Tunning Grids
################################

# XGBoost
tuneGrid_xgb <- expand.grid(
  nrounds = c(100, 200),
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.1, 0.3),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

# GBM
tuneGrid_gbm <- expand.grid(
  interaction.depth = c(1, 3, 5),
  n.trees = c(50, 100),
  shrinkage = c(0.05, 0.1),
  n.minobsinnode = c(5, 10)
)

# Random Forest
tuneGrid_rf <- expand.grid(
  mtry = c(2,4) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

################################
### XGBoost (Gradient Boosted Trees)
################################

# nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
```

```
    "xgbTree", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_xgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Mort30_25", clear = TRUE)

# train_and_assign(
#   model_type = "xgbTree",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "xgbTree", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
#   tuneGrid = tuneGrid_xgb
# )

################################
### GBM (Gradient Boosting Machines)
################################

# interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_gbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Mort30_25", clear = TRUE)

# train_and_assign(
#   model_type = "gbm",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "gbm", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
#   tuneGrid = tuneGrid_gbm
# )

################################
### Random Forest
################################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_rf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Mort30_25", clear = TRUE)
```

```
# train_and_assign(
#   model_type = "rf",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "rf", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
#   tuneGrid = tuneGrid_rf
# )


```

```{r final.4.7.30mort.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_50_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort          <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit          <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_30mort <- readRDS(here("data", "metadata",
"feature_selection_results_30mort.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Final"
target <- "mortality_30_day"
file_name <- "mort30_final.csv"
slices <- 1
model_run <- "Mort30_final"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 1
final_model <- TRUE

```

```{r Final.4.7.1.30mort.train}


```

```
##############################
### Tunning Grids
##############################

# XGBoost
tuneGrid_bestxgb <- expand.grid(
  nrounds = c(200),
  max_depth = c(3),
  eta = c(0.01),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

# GBM
tuneGrid_bestgbm <- expand.grid(
  interaction.depth = c(3),
  n.trees = c(50),
  shrinkage = c(0.1),
  n.minobsinnode = c(10)
)

# Random Forest
tuneGrid_bestrf <- expand.grid(
  mtry = c(2)
)

set.seed(42)

##############################
### XGBoost (Gradient Boosted Trees)
##############################

# nrounds = 200; max_depth = 3; eta = 0.01; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Mort30_final", clear = TRUE)

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "xgbTree", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
```

```
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Mort30_final", clear = TRUE)

##############################
### GBM (Gradient Boosting Machines)
##############################

#interaction.depth = 3; n.trees = 50; shrinkage = 0.1; n.minobsinnode = 10

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Mort30_final", clear = TRUE)

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "gbm", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Mort30_final", clear = TRUE)

##############################
### Random Forest
##############################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_sdoh_log_30_mort),
  tuneGrid = tuneGrid_bestrf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Mort30_final", clear = TRUE)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "rf", "base", "mortality_30_day", "mort30_feat_sel", feature_selection_results_30mort,
features_no_sdoh_log_30_mort),
  tuneGrid = tuneGrid_bestrf
```

```
)

save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Mort30_final", clear = TRUE)

```

Model Training & Feature Selection - Mort Hospital


```{r feat.sel.4.7.morthosp.inputs}

### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"
data_hosp <- load_dataset("data_hosp_10_log.rds")

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_10"
target <- "hospital_expire_flag"
file_name <- "morthosp_feat_sel.csv"
slices <- 4
model_run <- "morthosp_feat_sel"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r feat.sel.4.7.morthosp.train}

set.seed(42)

# === Feature Sets
base_feats  <- features_no_sdoh_log_h_mort
sdoh_feats  <- features_sdoh_log_h_mort

############################
#### XGBoost
############################

tuneGrid_xgb <- expand.grid(
  nrounds = 200,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "30_hosp_10", clear = TRUE)
```

```
train_and_assign(model_type = "xgbTree", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

############################
#### GBM
############################

tuneGrid_gbm <- expand.grid(
  interaction.depth = 3,
  n.trees = 100,
  shrinkage = 0.1,
  n.minobsinnode = 10
)

train_and_assign(model_type = "gbm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "gbm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

############################
#### Random Forest
############################

tuneGrid_rf <- expand.grid(mtry = 2)

train_and_assign(model_type = "rf", feature_set_name = "Base", selected_features = base_feats, tuneGrid
= tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "rf", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

# ############################
# #### Neural Network
# ############################

tuneGrid_nnet <- expand.grid(size = 5, decay = 0.1)

train_and_assign(model_type = "nnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "nnet", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

# ############################
# #### Logistic Regression
# ############################
```

```
train_and_assign(model_type = "glm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "glm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

# ############################
# #### GLMNet (Ridge, Lasso, Elastic Net)
# ############################

tuneGrid_glmnet <- expand.grid(
  alpha = .5,
  lambda = 0.1
)

train_and_assign(model_type = "glmnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "glmnet", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

# ############################
# #### Bayesian Logistic Regression
# ############################

train_and_assign(model_type = "bayesglm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "bayesglm", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

# ############################
# #### SVM (Radial)
# ############################
#
# # tuneGrid_svm <- expand.grid(C = 1, sigma = 0.05)
# #
# # train_and_assign(model_type = "svmRadial", feature_set_name = "Base", selected_features =
base_feats, tuneGrid = tuneGrid_svm)
# # save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_hosp_10", clear = TRUE)
# #
# # train_and_assign(model_type = "svmRadial", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_svm)
# # save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

############################
#### Naive Bayes
############################
```

```
train_and_assign(model_type = "naive_bayes", feature_set_name = "Base", selected_features = base_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

############################
#### K-Nearest Neighbors
############################

tuneGrid_knn <- expand.grid(k = 10)

train_and_assign(model_type = "knn", feature_set_name = "Base", selected_features = base_feats, tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

train_and_assign(model_type = "knn", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

```

```{r feat.sel.4.7.morthosp}

feature_selection_results_hospmort <- list()

# Load from saved models
files <- list.files(here("models", "Hosp_10"), pattern = "\\.rds$", full.names = TRUE)

for (file in files) {
  model_name <- sub("\\.rds$", "", basename(file))
  model_obj <- readRDS(file)

  if (!grepl("^gbm_base_h", model_name)) {
    message("Skipping model: ", model_name)
    next
  }

    if (grepl("^knn", model_name)) {
    message("Skipping KNN model: ", model_name)
    next
  }

  message("Feature selection for model: ", model_name)

  model_input <- get_model_data_and_features(model_obj, target = "hospital_expire_flag")

  selected_feats <- feature_selection(
    model = model_obj,
    target = "hospital_expire_flag",
    features = model_input$features,
    dataset = model_input$data,
    return_stats = TRUE
```

```
  )

  if ("Overall" %in% colnames(selected_feats)) {
    selected_names <- selected_feats$feature[selected_feats$Overall > 0]
  } else {
    selected_names <- selected_feats$feature
  }

  removed_feats <- setdiff(model_input$features, selected_names)

  feature_selection_results_hospmort[[model_name]] <- list(
    selected = selected_feats,
    removed = removed_feats
  )

  saveRDS(
    feature_selection_results_hospmort,
    here::here("data", "metadata", "feature_selection_results_hospmort_gbm_base.rds")
  )

  rm(model_obj); gc()  # Clear memory
}


```

```{r feat.sel.4.7.morthosp.review}

feature_selection_results_hospmort <- readRDS(here::here("data", "metadata",
"feature_selection_results_hospmort.rds"))

# Get all unique removed features across all models
all_removed <- unique(unlist(lapply(feature_selection_results_hospmort, function(x) x$removed)))

# Create a matrix: rows = features, cols = models
removed_matrix <- sapply(names(feature_selection_results_hospmort), function(model_name) {
  removed <- feature_selection_results_hospmort[[model_name]]$removed
  all_removed %in% removed
})

# Convert to data frame
removed_df <- as.data.frame(removed_matrix)
rownames(removed_df) <- all_removed

removed_df_pretty <- removed_df
removed_df_pretty[] <- lapply(removed_df_pretty, function(col) ifelse(col, "X", "-"))

t_4_7_1_removed_hospmort <- removed_df_pretty
write.csv(t_4_7_1_removed_hospmort, file = "../presentations/figures/t_4_7_1_removed_hospmort_2.csv",
row.names = TRUE, fileEncoding = "UTF-8")

```

```{r tunning.4.7.morthosp.inputs}
```

```
# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_25_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort          <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit          <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_hospmort <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort.rds"))
feature_selection_results_hospmort_gbm_base <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort_gbm_base.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_25"
target <- "hospital_expire_flag"
file_name <- "morthosp_tuning.csv"
slices <- 4
model_run <- "Morthosp_tuning"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r tunning.4.7.morthosp.train}

##############################
### Tunning Grids
##############################

# XGBoost
tuneGrid_xgb <- expand.grid(
  nrounds = c(100, 200),
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.1, 0.3),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
```

```
  subsample = 1
)

# GBM
tuneGrid_gbm <- expand.grid(
  interaction.depth = c(1, 3, 5),
  n.trees = c(50, 100),
  shrinkage = c(0.05, 0.1),
  n.minobsinnode = c(5, 10)
)

# Random Forest
tuneGrid_rf <- expand.grid(
  mtry = c(2,4) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

################################
### XGBoost (Gradient Boosted Trees)
################################

# nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_xgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "xgbTree",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "xgbTree", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_xgb
# )

################################
### GBM (Gradient Boosting Machines)
################################

# interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
```

```
  tuneGrid = tuneGrid_gbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "gbm",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "gbm", "base", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort_gbm_base, features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_gbm
# )

################################
### Random Forest
################################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_rf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "rf",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "rf", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_rf
# )

```

```{r final.4.7.morthosp.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_50_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
```

```
features_no_sdoh_log_30_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort           <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit           <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_hospmort <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort.rds"))
feature_selection_results_hospmort_gbm_base <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort_gbm_base.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Final"
target <- "hospital_expire_flag"
file_name <- "morthosp_final.csv"
slices <- 1
model_run <- "morthosp_final"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 1
final_model <- TRUE

```

```{r final.4.7.morthosp.train}

##############################
### Tunning Grids
##############################

# XGBoost
tuneGrid_bestxgb <- expand.grid(
  nrounds = c(100),
  max_depth = c(9),
  eta = c(0.01),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

# GBM
tuneGrid_bestgbm <- expand.grid(
  interaction.depth = c(5),
  n.trees = c(50),
  shrinkage = c(0.1),
  n.minobsinnode = c(5)
)

# Random Forest
tuneGrid_bestrf <- expand.grid(
```

```
  mtry = c(2) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

##############################
### XGBoost (Gradient Boosted Trees)
##############################

# nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "xgbTree", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Morthosp_final", clear = TRUE)

##############################
### GBM (Gradient Boosting Machines)
##############################

# interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "gbm", "base", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort_gbm_base, features_no_sdoh_log_h_mort),
```

```
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Morthosp_final", clear = TRUE)

##############################
### Random Forest
##############################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestrf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "rf", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestrf
)

save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Morthosp_final", clear = TRUE)

```

Model Training & Feature Selection - Readmitted

```{r feat.sel.4.7.readmit.inputs}

### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"
data_hosp <- load_dataset("data_hosp_10_log.rds")

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "Samp_10"
target <- "readmitted"
file_name <- "readmit_feat_sel.csv"
slices <- 4
model_run <- "readmit_feat_sel"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r feat.sel.4.7.readmit.train}

set.seed(42)

# === Feature Sets
base_feats  <- features_no_sdoh_log_readmit
sdoh_feats  <- features_sdoh_log_readmit

############################
#### XGBoost
############################

tuneGrid_xgb <- expand.grid(
  nrounds = 200,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### GBM
############################

tuneGrid_gbm <- expand.grid(
  interaction.depth = 3,
  n.trees = 100,
  shrinkage = 0.1,
  n.minobsinnode = 10
)

train_and_assign(model_type = "gbm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "gbm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### Random Forest
############################

tuneGrid_rf <- expand.grid(mtry = 2)
```

```
train_and_assign(model_type = "rf", feature_set_name = "Base", selected_features = base_feats, tuneGrid
= tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "rf", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### Neural Network
############################

tuneGrid_nnet <- expand.grid(size = 5, decay = 0.1)

train_and_assign(model_type = "nnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "nnet", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### Logistic Regression
############################

train_and_assign(model_type = "glm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "glm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### GLMNet (Ridge, Lasso, Elastic Net)
############################

tuneGrid_glmnet <- expand.grid(
  alpha = .5,
  lambda = 0.1
)

train_and_assign(model_type = "glmnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "glmnet", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_glmnet)
save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### Bayesian Logistic Regression
############################
```

```
train_and_assign(model_type = "bayesglm", feature_set_name = "Base", selected_features = base_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "bayesglm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### SVM (Radial)
############################

# tuneGrid_svm <- expand.grid(C = 1, sigma = 0.05)
#
# train_and_assign(model_type = "svmRadial", feature_set_name = "Base", selected_features = base_feats, tuneGrid = tuneGrid_svm)
# save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_hosp_10", clear = TRUE)
#
# train_and_assign(model_type = "svmRadial", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = tuneGrid_svm)
# save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

############################
#### Naive Bayes
############################

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base", selected_features = base_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "Readmit_10", clear = TRUE)

############################
#### K-Nearest Neighbors
############################

tuneGrid_knn <- expand.grid(k = 10)

train_and_assign(model_type = "knn", feature_set_name = "Base", selected_features = base_feats, tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "Readmit_10", clear = TRUE)

train_and_assign(model_type = "knn", feature_set_name = "Base_SDOH", selected_features = sdoh_feats, tuneGrid = tuneGrid_knn)
save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "Readmit_10", clear = TRUE)

```

```{r feat.sel.4.7.readmit}

feature_selection_results_readmit <- list()

# Load from saved models
```

```
files <- list.files(here("models", "Readmit_10"), pattern = "\\.rds$", full.names = TRUE)

for (file in files) {
  model_name <- sub("\\.rds$", "", basename(file))
  model_obj <- readRDS(file)

  if (grepl("^knn", model_name)) {
    message("Skipping KNN model: ", model_name)
    next
  }

  message("Feature selection for model: ", model_name)

  model_input <- get_model_data_and_features(model_obj, target = "readmitted")

  selected_feats <- feature_selection(
    model = model_obj,
    target = "readmitted",
    features = model_input$features,
    dataset = model_input$data,
    return_stats = TRUE
  )

  if ("Overall" %in% colnames(selected_feats)) {
    selected_names <- selected_feats$feature[selected_feats$Overall > 0]
  } else {
    selected_names <- selected_feats$feature
  }

  removed_feats <- setdiff(model_input$features, selected_names)

  feature_selection_results_readmit[[model_name]] <- list(
    selected = selected_feats,
    removed = removed_feats
  )

  saveRDS(
    feature_selection_results_readmit,
    here::here("data", "metadata", "feature_selection_results_readmit.rds")
  )

  rm(model_obj); gc()  # Clear memory
}

```

```{r feat.sel.4.7.readmit.review}

feature_selection_results_readmit <- readRDS(here::here("data", "metadata",
"feature_selection_results_readmit.rds"))

# Get all unique removed features across all models
all_removed <- unique(unlist(lapply(feature_selection_results_readmit, function(x) x$removed)))

# Create a matrix: rows = features, cols = models
```

```
removed_matrix <- sapply(names(feature_selection_results_readmit), function(model_name) {
  removed <- feature_selection_results_readmit[[model_name]]$removed
  all_removed %in% removed
})

# Convert to data frame
removed_df <- as.data.frame(removed_matrix)
rownames(removed_df) <- all_removed

removed_df_pretty <- removed_df
removed_df_pretty[] <- lapply(removed_df_pretty, function(col) ifelse(col, "X", "-"))

t_4_7_1_removed_readmit <- removed_df_pretty
write.csv(t_4_7_1_removed_readmit, file = "../presentations/figures/t_4_7_1_removed_readmit.csv",
row.names = TRUE, fileEncoding = "UTF-8")

```

```{r tunning.4.7.readmit.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 10% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_10_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort         <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort            <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort           <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit           <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_readmit <- readRDS(here("data", "metadata",
"feature_selection_results_readmit.rds"))

set.seed(42)
train_function = "train_single_model3"
data <- data_hosp
dataset_name <- "Samp_25"
target <- "readmitted"
file_name <- "readmit_tuning_2.csv"
slices <- 4
model_run <- "readmit_tuning_2"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
```

```
final_model <- FALSE

```

```{r tunning.4.7.readmit.train}

################################
### Tunning Grids
################################

# Neural Network
tuneGrid_nnet <- expand.grid(
  size = c(3, 5, 7),     # Number of hidden units
  decay = c(0.01, 0.1, 0.5)  # Weight decay (regularization)
)

set.seed(42)

################################
### Neural Network
################################

train_and_assign(
  model_type = "nnet",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "nnet", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit),
  tuneGrid = tuneGrid_nnet
)

save_models(mget(ls(pattern = "^nnet"), inherits = TRUE), "Readmit_10_2", clear = TRUE)

# train_and_assign(
#   model_type = "nnet",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "nnet", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit),
#   tuneGrid = tuneGrid_nnet
# )

################################
### Logistic Regression
################################

train_and_assign(
  model_type = "glm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "glm", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit)
  )

save_models(mget(ls(pattern = "^glm"), inherits = TRUE), "Readmit_25", clear = TRUE)
```

```
train_and_assign(
  model_type = "glm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "glm", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit)
)

##############################
### Bayesian Logistic Regression
##############################

train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "bayesglm", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit),
)
save_models(mget(ls(pattern = "^bayesglm"), inherits = TRUE), "Readmit_25", clear = TRUE)

train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "bayesglm", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit),
  tuneGrid = tuneGrid_bayesglm
)

```

```{r final.4.7.readmit.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 50% Data Sample for final
data_hosp <- load_dataset("data_hosp_100_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort          <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit          <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))
```

```
feature_selection_results_readmit <- readRDS(here("data", "metadata",
"feature_selection_results_readmit.rds"))

set.seed(42)
train_function = "train_single_model2"
data <- data_hosp
dataset_name <- "final"
target <- "readmitted"
file_name <- "readmit_final.csv"
slices <- 1
model_run <- "readmit_final"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 1
final_model <- TRUE

```

```{r final.4.7.readmit.train}

################################
### Tunning Grids
################################

# Neural Network
tuneGrid_bestnnet <- expand.grid(
  size = c(5),     # Number of hidden units
  decay = c(0.5)  # Weight decay (regularization)
)

set.seed(42)

################################
### Neural Network
################################

train_and_assign(
  model_type = "nnet",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "nnet", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit),
  tuneGrid = tuneGrid_bestnnet
)

save_models(mget(ls(pattern = "^nnet"), inherits = TRUE), "Readmit_final", clear = TRUE)

train_and_assign(
  model_type = "nnet",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "nnet", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit),
  tuneGrid = tuneGrid_bestnnet
```

```
)

save_models(mget(ls(pattern = "^nnet"), inherits = TRUE), "Readmit_final", clear = TRUE)

##############################
### Logistic Regression
##############################

train_and_assign(
  model_type = "glm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "glm", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit)
  )

save_models(mget(ls(pattern = "^glm"), inherits = TRUE), "Readmit_final", clear = TRUE)

train_and_assign(
  model_type = "glm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "glm", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit)
)

save_models(mget(ls(pattern = "^glm"), inherits = TRUE), "Readmit_final", clear = TRUE)

##############################
### Bayesian Logistic Regression
##############################

train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "bayesglm", "base_sdoh", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_sdoh_log_readmit),
)
save_models(mget(ls(pattern = "^bayesglm"), inherits = TRUE), "Readmit_final", clear = TRUE)

train_and_assign(
  model_type = "bayesglm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "bayesglm", "base", "readmitted", "readmit_feat_sel", feature_selection_results_readmit,
features_no_sdoh_log_readmit),
)

save_models(mget(ls(pattern = "^bayesglm"), inherits = TRUE), "Readmit_final", clear = TRUE)

```

Mitigation

```{r feat.sel.4.7.morthosp.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 10% Data Sample for EDA
data_hosp <- load_dataset("data_hosp_10_log.rds")
# Load Functions
source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort          <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit          <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

### "mortality_30_day", "readmitted", "hospital_expire_flag", "length_of_stay"

set.seed(42)
train_function = "train_single_model3"
data <- data_hosp
dataset_name <- "Samp_10"
target <- "hospital_expire_flag"
file_name <- "mit_morthosp_feat_sel.csv"
slices <- 4
model_run <- "mit_morthosp_feat_sel"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r feat.sel.4.7.morthosp.train}

set.seed(42)

# === Feature Sets
base_feats  <- features_no_sdoh_log_h_mort
sdoh_feats  <- features_sdoh_log_h_mort

############################
#### XGBoost
############################

tuneGrid_xgb <- expand.grid(
```

```
  nrounds = 200,
  max_depth = 6,
  eta = 0.3,
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

train_and_assign(model_type = "xgbTree", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_xgb)
save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

############################
#### GBM
############################

tuneGrid_gbm <- expand.grid(
  interaction.depth = 3,
  n.trees = 100,
  shrinkage = 0.1,
  n.minobsinnode = 10
)

train_and_assign(model_type = "gbm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

train_and_assign(model_type = "gbm", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_gbm)
save_models(mget(ls(pattern = "^gbm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

############################
#### Random Forest
############################

tuneGrid_rf <- expand.grid(mtry = 2)

train_and_assign(model_type = "rf", feature_set_name = "Base", selected_features = base_feats, tuneGrid
= tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

train_and_assign(model_type = "rf", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_rf)
save_models(mget(ls(pattern = "^rf_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

# ############################
# #### Neural Network
# ############################

tuneGrid_nnet <- expand.grid(size = 5, decay = 0.1)
```

```
train_and_assign(model_type = "nnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

train_and_assign(model_type = "nnet", feature_set_name = "Base_SDOH", selected_features = sdoh_feats,
tuneGrid = tuneGrid_nnet)
save_models(mget(ls(pattern = "^nnet_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

# ############################
# #### Logistic Regression
# ############################

# train_and_assign(model_type = "glm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
# save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)
#
# train_and_assign(model_type = "glm", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
# save_models(mget(ls(pattern = "^glm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

# ############################
# #### GLMNet (Ridge, Lasso, Elastic Net)
# ############################

# tuneGrid_glmnet <- expand.grid(
#   alpha = .5,
#   lambda = 0.1
# )
#
# train_and_assign(model_type = "glmnet", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_glmnet)
# save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)
#
# train_and_assign(model_type = "glmnet", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_glmnet)
# save_models(mget(ls(pattern = "^glmnet_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

# ############################
# #### Bayesian Logistic Regression
# ############################

# train_and_assign(model_type = "bayesglm", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
# save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)
#
# train_and_assign(model_type = "bayesglm", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
# save_models(mget(ls(pattern = "^bayesglm_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

# ############################
# #### SVM (Radial)
# ############################
#
# # tuneGrid_svm <- expand.grid(C = 1, sigma = 0.05)
```

```
# #
# # train_and_assign(model_type = "svmRadial", feature_set_name = "Base", selected_features =
base_feats, tuneGrid = tuneGrid_svm)
# # save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)
# #
# # train_and_assign(model_type = "svmRadial", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_svm)
# # save_models(mget(ls(pattern = "^svmRadial_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

############################
#### Naive Bayes
############################

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

train_and_assign(model_type = "naive_bayes", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = NULL)
save_models(mget(ls(pattern = "^naive_bayes_"), inherits = TRUE), "mit_hosp_10", clear = TRUE)

############################
#### K-Nearest Neighbors
############################

# tuneGrid_knn <- expand.grid(k = 10)
#
# train_and_assign(model_type = "knn", feature_set_name = "Base", selected_features = base_feats,
tuneGrid = tuneGrid_knn)
# save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_hosp_10", clear = TRUE)
#
# train_and_assign(model_type = "knn", feature_set_name = "Base_SDOH", selected_features =
sdoh_feats, tuneGrid = tuneGrid_knn)
# save_models(mget(ls(pattern = "^knn_"), inherits = TRUE), "30_hosp_10", clear = TRUE)

```

```{r feat.sel.4.7.morthosp}

mit_feature_selection_results_hospmort <- list()

# Load from saved models
files <- list.files(here("models", "mit_hosp_10"), pattern = "\\.rds$", full.names = TRUE)

for (file in files) {
  model_name <- sub("\\.rds$", "", basename(file))
  model_obj <- readRDS(file)

  # if (!grepl("^gbm_base_h", model_name)) {
  #   message("Skipping model: ", model_name)
  #   next
  # }

    if (grepl("^knn", model_name)) {
    message("Skipping KNN model: ", model_name)
```

```
  next
 }

 message("Feature selection for model: ", model_name)

 model_input <- get_model_data_and_features(model_obj, target = "hospital_expire_flag")

 selected_feats <- feature_selection(
  model = model_obj,
  target = "hospital_expire_flag",
  features = model_input$features,
  dataset = model_input$data,
  return_stats = TRUE
 )

 if ("Overall" %in% colnames(selected_feats)) {
  selected_names <- selected_feats$feature[selected_feats$Overall > 0]
 } else {
  selected_names <- selected_feats$feature
 }

 removed_feats <- setdiff(model_input$features, selected_names)

 mit_feature_selection_results_hospmort[[model_name]] <- list(
  selected = selected_feats,
  removed = removed_feats
 )

 saveRDS(
  mit_feature_selection_results_hospmort,
  here::here("data", "metadata", "mit_feature_selection_results_hospmort.rds")
 )

 rm(model_obj); gc()  # Clear memory
}


```

```{r feat.sel.4.7.morthosp.review}

mit_feature_selection_results_hospmort <- readRDS(here::here("data", "metadata",
"mit_feature_selection_results_hospmort.rds"))

# Get all unique removed features across all models
all_removed <- unique(unlist(lapply(mit_feature_selection_results_hospmort, function(x) x$removed)))

# Create a matrix: rows = features, cols = models
removed_matrix <- sapply(names(mit_feature_selection_results_hospmort), function(model_name) {
 removed <- mit_feature_selection_results_hospmort[[model_name]]$removed
 all_removed %in% removed
})

# Convert to data frame
removed_df <- as.data.frame(removed_matrix)
```

```
rownames(removed_df) <- all_removed

removed_df_pretty <- removed_df
removed_df_pretty[] <- lapply(removed_df_pretty, function(col) ifelse(col, "X", "-"))

t_4_7_1_removed_hospmort <- removed_df_pretty
write.csv(t_4_7_1_removed_hospmort, file =
"../presentations/figures/t_4_7_1_removed_hospmort_mit.csv", row.names = TRUE, fileEncoding = "UTF-
8")

```

```{r tunning.4.7.morthosp.inputs}

# Load Featurte Sets
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_25_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort           <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort          <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit       <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit          <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

mit_feature_selection_results_hospmort <- readRDS(here("data", "metadata",
"mit_feature_selection_results_hospmort.rds"))

set.seed(42)
train_function = "train_single_model3"
data <- data_hosp
dataset_name <- "Samp_25"
target <- "hospital_expire_flag"
file_name <- "mit_morthosp_tuning.csv"
slices <- 4
model_run <- "mit_Morthosp_tuning"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 4
final_model <- FALSE

```

```{r tunning.4.7.morthosp.train}
```

```
##############################
### Tunning Grids
##############################

# XGBoost
tuneGrid_xgb <- expand.grid(
  nrounds = c(100, 200),
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.1, 0.3),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

# GBM
tuneGrid_gbm <- expand.grid(
  interaction.depth = c(1, 3, 5),
  n.trees = c(50, 100),
  shrinkage = c(0.05, 0.1),
  n.minobsinnode = c(5, 10)
)

# Random Forest
tuneGrid_rf <- expand.grid(
  mtry = c(2) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

##############################
### XGBoost (Gradient Boosted Trees)
##############################

# nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "hospital_expire_flag", "mit_morthosp_feat_sel",
mit_feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_xgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "mit_Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "xgbTree",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "xgbTree", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_xgb
# )
```

```
##############################
### GBM (Gradient Boosting Machines)
##############################

# interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "gbm", "base_sdoh", "hospital_expire_flag", "mit_morthosp_feat_sel",
mit_feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_gbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "mit_Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "gbm",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "gbm", "base", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort_gbm_base, features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_gbm
# )

##############################
### Random Forest
##############################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "hospital_expire_flag", "mit_morthosp_feat_sel",
mit_feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_rf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "mit_Morthosp_25", clear = TRUE)

# train_and_assign(
#   model_type = "rf",
#   feature_set_name = "Base",
#   selected_features = get_training_ready_features(
#     "rf", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
#   tuneGrid = tuneGrid_rf
# )

```

```{r final.4.7.morthosp.inputs}

# Load Featurte Sets
```

```
source(here("scripts", "feature_sets.R"))
# Load utilities
source(here("scripts", "utils.R"))

# Load 20% Data Sample for tuning
data_hosp <- load_dataset("data_hosp_50_log.rds")

source(here("scripts", "figs_funcs.R"))

features_no_sdoh_log_h_mort         <- readRDS(here("data", "metadata",
"features_no_sdoh_log_h_mort.rds"))
features_sdoh_log_h_mort            <- readRDS(here("data", "metadata", "features_sdoh_log_h_mort.rds"))
features_no_sdoh_log_30_mort        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_30_mort.rds"))
features_sdoh_log_30_mort           <- readRDS(here("data", "metadata",
"features_sdoh_log_30_mort.rds"))
features_no_sdoh_log_readmit        <- readRDS(here("data", "metadata",
"features_no_sdoh_log_readmit.rds"))
features_sdoh_log_readmit           <- readRDS(here("data", "metadata", "features_sdoh_log_readmit.rds"))

feature_selection_results_hospmort <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort.rds"))
feature_selection_results_hospmort_gbm_base <- readRDS(here("data", "metadata",
"feature_selection_results_hospmort_gbm_base.rds"))

set.seed(42)
train_function = "train_single_model3"
data <- data_hosp
dataset_name <- "Final"
target <- "hospital_expire_flag"
file_name <- "morthosp_final.csv"
slices <- 1
model_run <- "morthosp_final"
split_type <- "patient"
train_size <- 0.7
cv_splits <- 1
final_model <- TRUE

```

```{r final.4.7.morthosp.train}

##############################
### Tunning Grids
##############################

# XGBoost
tuneGrid_bestxgb <- expand.grid(
  nrounds = c(100),
  max_depth = c(9),
  eta = c(0.01),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
```

```
)

# GBM
tuneGrid_bestgbm <- expand.grid(
  interaction.depth = c(5),
  n.trees = c(50),
  shrinkage = c(0.1),
  n.minobsinnode = c(5)
)

# Random Forest
tuneGrid_bestrf <- expand.grid(
  mtry = c(2) # mtry = c(2, 4, 6, 8)
)

set.seed(42)

################################
### XGBoost (Gradient Boosted Trees)
################################

# nrounds = 200; max_depth = 6; eta = 0.1; gamma = 0; colsample_bytree = 1; min_child_weight = 1;
subsample = 1
train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "xgbTree", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort, features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "xgbTree",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "xgbTree", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestxgb
)

save_models(mget(ls(pattern = "^xgbTree_"), inherits = TRUE), "Morthosp_final", clear = TRUE)

################################
### GBM (Gradient Boosting Machines)
################################

# interaction.depth = 5; n.trees = 100; shrinkage = 0.1; n.minobsinnode = 5
train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
```

```
    "gbm", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "gbm",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "gbm", "base", "hospital_expire_flag", "morthosp_feat_sel",
feature_selection_results_hospmort_gbm_base, features_no_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestgbm
)

save_models(mget(ls(pattern = "^gbm"), inherits = TRUE), "Morthosp_final", clear = TRUE)

################################
### Random Forest
################################


train_and_assign(
  model_type = "rf",
  feature_set_name = "Base_SDOH",
  selected_features = get_training_ready_features(
    "rf", "base_sdoh", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestrf
)
save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Morthosp_final", clear = TRUE)

train_and_assign(
  model_type = "rf",
  feature_set_name = "Base",
  selected_features = get_training_ready_features(
    "rf", "base", "hospital_expire_flag", "morthosp_feat_sel", feature_selection_results_hospmort,
features_no_sdoh_log_h_mort),
  tuneGrid = tuneGrid_bestrf
)

save_models(mget(ls(pattern = "^rf"), inherits = TRUE), "Morthosp_final", clear = TRUE)

```


```{r feet.sel.final, fig.show='hold'}



top_features_list <- readRDS(here("data", "metadata", "feature_selection_results_30mort.rds"))
# top_features_list <- readRDS(here("data", "metadata", "feature_selection_results_hospmort.rds"))
# top_features_list <- readRDS(here("data", "metadata", "feature_selection_results_readmit.rds"))
# top_features_list <- readRDS(here("data", "metadata", "feature_selection_results_los.rds"))
```

```
top10_features_list <- lapply(top_features_list, function(model_list) {
 if (is.list(model_list) && "selected" %in% names(model_list)) {
  model_list$selected %>%
   slice_head(n = 10)  #
 } else {
  NULL
 }
})

top10_features_list <- Filter(Negate(is.null), top10_features_list)

for (model_name in names(top10_features_list)) {
 df <- top10_features_list[[model_name]]

 # Check if "Overall" exists
 if ("Overall" %in% names(df)) {
  p <- ggplot(df, aes(x = reorder(feature, Overall), y = Overall)) +
   geom_bar(stat = "identity", fill = "steelblue") +
   coord_flip() +
   labs(title = paste("Top 10 Features -", model_name),
        x = "Feature",
        y = "Overall Score") +
   theme_minimal() +
   theme(
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
   )
 } else if ("rank" %in% names(df)) {
  p <- ggplot(df %>% mutate(importance = max(rank) - rank + 1),
              aes(x = reorder(feature, importance), y = importance)) +
   geom_bar(stat = "identity", fill = "steelblue") +
   coord_flip() +
   labs(title = paste("Top 10 Features -", model_name),
        x = "Feature",
        y = "Relative Importance") +
   theme_minimal() +
   theme(
    panel.background = element_rect(fill = "white", color = NA),
    plot.background = element_rect(fill = "white", color = NA)
   )
 } else {
  warning("no score found for: ", model_name)
  next
 }

 ggsave(
  filename = here::here("presentations", "figures", paste0("f_4.7_top10_", model_name, ".png")),
  plot = p,
  width = 6,
  height = 4,
  dpi = 300
 )

 print(p)
```

}

```

#### Model Evaluation

```{r ROC.call}

mort30_final <- load_models_to_list(subfolder = "Mort30_final", strip_train = TRUE) # pattern = "^gbm"
mort30_final <- add_predictions_to_models(mort30_final, target_col = "mortality_30_day")
save_models(mort30_final, subfolder = "Mort30_final")
rm(mort30_final)

morthosp_final <- load_models_to_list(subfolder = "Morthosp_final", strip_train = TRUE) # pattern =
"^gbm"
morthosp_final <- add_predictions_to_models(morthosp_final, target_col = "hospital_expire_flag")
save_models(morthosp_final, subfolder = "Morthosp_final")
rm(morthosp_final)

readmit_final <- load_models_to_list(subfolder = "Readmit_final", strip_train = TRUE, pattern = "^nnet_")
# readmit_final <- add_predictions_to_models(readmit_final, target_col = "readmitted")
save_models(readmit_final, subfolder = "Readmit_final")
rm(readmit_final)

```

```{r 4.8.ROC.call}


mort30_final <- load_models_to_list(subfolder = "Mort30_final", strip_train = TRUE) # pattern = "^gbm"

mort30_roc <- plot_roc_curves(
  models = mort30_final,
  model_names = names(mort30_final),
  target = "mortality_30_day",
  plot_title = "30 Day Mortality Prediction"
)
rm(mort30_final)

morthosp_final <- load_models_to_list(subfolder = "Morthosp_final", strip_train = TRUE) # pattern =
"^gbm"

morthosp_roc <- plot_roc_curves(
  models = morthosp_final,
  model_names = names(morthosp_final),
  target = "hospital_expire_flag",
  plot_title = "Hospital Mortality Prediction"
)
rm(morthosp_final)

# readmit_final <- load_models_to_list(subfolder = "Readmit_final", strip_train = TRUE, pattern =
"^bayesglm") # pattern = "^gbm"
#
# readmit_final <- plot_roc_curves(
#   models = readmit_final,

```
model_names = names(readmit_final),
target = "readmitted",
plot_title = "30 Day Readmission Prediction"
)

```

```{r PR.AUC.call}


mort30_final <- load_models_to_list(subfolder = "Mort30_final", strip_train = TRUE) # pattern = "^gbm"

mort30_pr <- plot_pr_curves(
  models = mort30_final,
  model_names = names(mort30_final),
  target = "mortality_30_day",
  plot_title = "30 Day Mortality Prediction"
)
rm(mort30_final)

# Hospital Mortality Models
morthosp_final <- load_models_to_list(subfolder = "Morthosp_final", strip_train = TRUE) # pattern =
"^gbm"

morthosp_pr <- plot_pr_curves(
  models = morthosp_final,
  model_names = names(morthosp_final),
  target = "hospital_expire_flag",
  plot_title = "Hospital Mortality Prediction"
)
rm(morthosp_final)

# # Readmitted Models
# plot_pr_curves(
#   models = readmit_models,
#   model_names = sapply(readmit_models, function(model) model$method),
#   target = "readmitted",
#   plot_title = "Hospital Readmission Prediction"
# )

```

```{r pred.hist.call}

mort30_final <- load_models_to_list(subfolder = "Mort30_final", strip_train = TRUE)
results_df <- extract_predicted_probs(mort30_final)
plot_prediction_histograms(results_df)

```

```{r 4.10.x.vs.pred.plot}

source(here("scripts", "feature_sets.R"))
source(here("scripts", "utils.R"))
data_hosp <- load_dataset("data_hosp_5.rds")
```

```
source(here("scripts", "figs_funcs.R"))

gbm_final <- load_models_to_list("length_of_stay_final", pattern = "^gbm")
rf_final <- load_models_to_list("length_of_stay_final", pattern = "^rf")
xgbTree_final <- load_models_to_list("length_of_stay_final", pattern = "^xgbTree")

# load_models_as_objects("length_of_stay_final", pattern = "^xgbTree")

model_plot_pairs <- list()
for (model_name in names(gbm_final)) {
  model <- gbm_final[[model_name]]
  plots <- evaluate_model(model, target = "length_of_stay", model_name = model_name)
  model_plot_pairs[[model_name]] <- plots

  save_named_plot(plots$actual_vs_pred, paste0("f_4_10_los_predicted_vs_actual_", model_name))
  save_named_plot(plots$residual_hist, paste0("f_4_10_los_residual_hist_", model_name))
}
rm(list = ls(pattern = "gbm_final"), envir = .GlobalEnv)


model_plot_pairs <- list()
for (model_name in names(rf_final)) {
  model <- rf_final[[model_name]]
  plots <- evaluate_model(model, target = "length_of_stay", model_name = model_name)
  model_plot_pairs[[model_name]] <- plots

  save_named_plot(plots$actual_vs_pred, paste0("f_4_10_los_predicted_vs_actual_", model_name))
  save_named_plot(plots$residual_hist, paste0("f_4_10_los_residual_hist_", model_name))
}

rm(list = ls(pattern = "rf_final"), envir = .GlobalEnv)
rm(list = ls(pattern = "^plots"), envir = .GlobalEnv)

model_plot_pairs <- list()
for (model_name in names(xgbTree_final)) {
  model <- xgbTree_final[[model_name]]
  plots <- evaluate_model(model, target = "length_of_stay", model_name = model_name)
  model_plot_pairs[[model_name]] <- plots

  save_named_plot(plots$actual_vs_pred, paste0("f_4_10_los_predicted_vs_actual_", model_name))
  save_named_plot(plots$residual_hist, paste0("f_4_10_los_residual_hist_", model_name))
}

rm(list = ls(pattern = "xgbTree"), envir = .GlobalEnv)
gc()


```

```{r x.vs.pred.plot.2}

model_files <- list.files(here::here("models", "length_of_stay_final"),
                          pattern = "los_100.*\\.rds$", full.names = TRUE)

fit_data_list <- list()
```

```
for (file in model_files) {
  model <- readRDS(file)

  model_name <- sub("\\.rds$", "", basename(file))
  test_data <- model$test_data
  preds <- predict(model, newdata = test_data)

  fit_data_list[[model_name]] <- data.frame(
    Predicted = preds,
    Actual = test_data$length_of_stay,
    Model = model_name
  )

  # Optional: release model to save RAM
  rm(model)
  gc()
}

combined_df <- do.call(rbind, fit_data_list)


f_4_10_los_all <- ggplot(combined_df, aes(x = Predicted, y = Actual, color = Model)) +
  geom_smooth(method = "lm", se = FALSE, linewidth = 1.2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dotted", color = "grey40", size = 0.7) +
  ggtitle("LOS Actual vs. Predicted Fit Lines") +
  xlab("Predicted LOS") + ylab("Actual LOS") +
  theme_minimal() +
    theme(legend.position = "bottom",
      panel.background = element_rect(fill = "white", color = NA),
      plot.background = element_rect(fill = "white", color = NA)
    )


save_named_plot(f_4_10_los_all, "f_4_10_los_all", width = 9, height = 9, dpi = 400)


```
```
```