Treelet Dimension Reduction of Diagnoses Among Intensive Care Unit Admissions

by

James Dominic DiSanto

BSc Neuroscience, University of Pittsburgh, 2018

Submitted to the Graduate Faculty of

the Department of Biostatistics in the

Graduate School of Public Health in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH

GRADUATE SCHOOL OF PUBLIC HEALTH

This thesis was presented

by

James Dominic DiSanto

It was defended on

December 7, 2020

and approved by

Jeanine M. Buchanich, PhD, Research Associate Professor, Biostatistics
Graduate School of Public Health, University of Pittsburgh

Ada Youk, PhD, Research Associate Professor, Biostatistics
Graduate School of Public Health, University of Pittsburgh

Jenna C. Carlson, PhD, Assistant Professor, Biostatistics
Graduate School of Public Health, University of Pittsburgh

Douglas Landsittel, PhD, Professor, Biomedical Informatics
School of Medicine, University of Pittsburgh

Thesis Advisor: Jeanine M. Buchanich, PhD, Research Associate Professor, Biostatistics
Graduate School of Public Health, University of Pittsburgh

Jeanine M. Buchanich, PhD

Treelet Dimension Reduction of Diagnoses Among Intensive Care Unit Admissions

James Dominic DiSanto, MS

University of Pittsburgh, 2020

Abstract

Background: The objective of this thesis is to apply treelet dimension reduction to ICD-9-CM diagnosis codes and apply the resulting transformation in the prediction of clinical outcomes of in-hospital mortality, unplanned re-admission, and hospital length of stay.

Data: International Classification of Disease, 9th Revision, Clinical Modification (ICD-9-CM) codes and patient demographic data (age, sex, insurance coverage) from the Medical Information Mart for Intensive Care III (MIMIC-III) database prospective cohort study of 38,554 adults admitted to a single intensive care unit from 2001 to 2012.

Methods: We applied treelet dimension reduction to ICD-9-CM diagnosis codes (n=178, $\geq$1% prevalence in the analytic cohort) to identify a transformed feature space of patient diagnoses that we then used, with patient demographic data, to predict in-hospital mortality, unplanned hospital re-admission, and length of hospital stay using logistic and negative binomial regression models.

Results: Treelet dimension reduction for ICD-9-CM diagnosis codes identified reduced feature spaces in prediction of in-hospital mortality, unplanned hospital re-admission, and length of stay. The resulting treelet features for each clinical outcome, in addition to patient age, gender, and payment method, demonstrate improved utility in predicting in-hospital mortality (AUC=0.858) but limited accuracy in prediction hospital re-admission (AUC=0.661). Treelet dimension reduction identifies a sparse number of ICD-9-CM diagnosis codes (107 of 178)

retained in the treelet features included in modeling of length of stay (RMSE=10.29).

Public Health Significance: These analyses leverage a large, public database of critical care admissions, generating predictive models of clinical outcomes using only patient demographic and comorbidity diagnosis information. The presented analysis builds upon previous work by applying the novel treelet dimension reduction model on diagnosis data in a dataset of critical care admissions and demonstrate the utility of diagnosis code data alone in prediction of clinical outcomes.

Keywords: treelet, dimension reduction, diagnosis codes, generalized linear models

Table of Contents

List of Tables

List of Figures

1.0 Introduction

1.1 Clinical Prediction Models

Clinical prediction models present useful, empirical methods to assess patient risk, often modelling outcomes such as mortality, disease-specific remission, hospital resource utilization, etc. Prediction models may inform both patient treatment (e.g. estimating patient recovery prognosis following ischemic stroke, comparing estimated benefit and risk of a specific treatment) and clinical research (e.g. estimating baseline risk of outcome to identify specific risk-groups of patients for study enrollment) (Steyerberg, 2009). Beyond patient-specific prediction, health-policy makers and hospital administrators may use models of outcomes such as in-hospital mortality, hospital length of stay, and unplanned re-admissions as measures of a hospital's case-severity and/or resource utilization (Awad, Bader–El–Den, et al., 2017; Quach et al., 2009).

The prediction of mortality (or assessment of patient mortality risk) is notably used at the hospital- or system-level to account for diversity of illness or injury severity of admissions, allowing for comparison of care quality across health care systems and/or centers (Quach et al., 2009). The performance of existing predictive models of mortality remain limited. Studies have previously explored the predictive utility of comorbidity indices such as the Charlson (Charlson et al., 1987) and Elixhauser (Elixhauser et al., 1998). These measurement systems assess the presence or absence of conditions (19 disease groups in the Charlson index, 31 in the Elixhauser) using a subset of available ICD-9-CM diagnosis codes. Models using these existing indices have estimated concordance values ranging from ~0.71 to ~0.78 (Quach et al., 2009; Snow et al., 2020). The APACHE-II (Knaus et al., 1985) is a disease severity classification system that additionally uses

physiological and temporal measurements (e.g. lab values of hematocrit, creatinine, white blood cell count). Use of APACHE-II scores resulted in improved, but still limited, prediction of in-hospital mortality, with concordance values ranging from ~0.75 to ~0.84 (Awad, Bader-El-Den, et al., 2017; Falcão et al., 2019) (see **Section 2.2.4** for description of concordance and additional model fit metrics).

Looking beyond in-hospital mortality, hospital length of stay is an important metric that inherently captures information related to (and can loosely serve as a proxy measurement of) hospitalization cost and/or hospital resource utilization (Awad, Bader–El–Den, et al., 2017). Similarly, individuals who survive their initial hospital admission remain at risk for adverse post-discharge events and subsequent hospital readmission. In addition to the physical and mental toll of an unplanned hospital readmission and/or a prolonged hospital course, patients experience significant, undue financial burden (Mayr et al., 2017). At the hospital system level, the Center for Medicare and Medicaid Services includes hospital readmission as an assessment of quality of care, including a financial incentive for hospitals to reduce readmission rates (CMS, n.d.). Predictive models of hospital readmission and length of stay remain limited both in their predictive performance and in the availability of the data which the models require (Kansagara et al., 2011). A review of existing prediction models of mortality and length of stay proposes that future models should leverage large, commercially or publicly available critical care databases, such as the Medical Information Mart for Intensive Care III (MIMIC-III) (formerly the Multiparameter Intelligent Monitoring in Intensive Care or MIMIC-II) data used in this work (Awad, Bader–El–Den, et al., 2017).

The growth of healthcare data and statistical learning has further catalyzed interest in statistically-derived prediction models due to the increase in both available sample size and

diversity of available predictors (Steyerberg, 2009). The generalizability of clinical prediction models is limited to the availability of the required data. That is, models built upon esoteric data elements (e.g. hospital-specific variables) or highly granular information (e.g. pre-admission lab values, genetic data), present barriers to effective implementation by requiring collection of the necessary input information, which may be infeasible when applied to a new hospital or clinical setting. As a result, the use of large, healthcare data sources must account for not only the performance of the relevant models but the likely availability of the required data elements.

1.2 Modern Healthcare Data

Due to advances in data collection and management and the digitization of healthcare data into electronic health records (EHR), hospitals and healthcare systems now maintain an enormous amount of patient data, with the opportunity to wield this information to improve patient care (Dash et al., 2019). A single American hospital's EHR captures an estimated 10^7 terabytes of data annually (Pah et al., 2014). With such large data volume, there are significant obstacles to efficiently collecting, managing, and leveraging pertinent information from the variety of data sources that are commonly present in a large hospital.

A large component of a hospital's EHR data comprises patient-level diagnoses of disease, injury, and associated conditions (Pah et al., 2014). The Center for Medicare and Medicaid Services presents a codified system of diagnosing diseases (among other clinical care information such as health services, injury/disease causes, etc.) among patients, referred to as the International Classification of Diseases, Clinical Modification codes (ICD-CM). The 9^{th} version of the system (ICD-9-CM) was adopted in the 1980's and used through 2014, at which point the current 10^{th}

version (ICD-10-CM) was mandated (having been preliminarily adopted in the late 1990's) (Topaz et al., 2013). The ICD-9-CM system included nearly 17,000 unique patient diagnoses, while ICD-10-CM expands this catalog to over 155,000 unique codes (Topaz et al., 2013).

The volume of diagnosis data present in the EHR is a rich resource to support clinical research and improve upon existing predictive clinical models (Kennedy et al., 2013). Groups of ICD-9-CM diagnosis codes in a large dataset may also expectedly represent redundant information. For example, highly correlated or commonly concurrent diagnoses (e.g. hypertension, hyperlipidemia, type-2 diabetes mellitus) could be grouped into a single aggregate input representing this group of diagnoses.

1.3 High-Dimensional Data & Dimension Reduction

High-dimensional data describes data sets with a high number of covariates (or inputs, features, etc.), which may commonly include highly correlated variables. Such data sets present elevated risk of overfitting and, in extreme cases where there are a similar or greater number of predictors than observations, may prevent identification of statistical models using the full feature set (Hastie et al., 2017). Even in data sets with sufficiently high sample sizes to fit models including a large number of predictors, high-dimensional data often contain an unknown but non-negligible amount of noise, correlated pairs of inputs, and/or groups of intercorrelated inputs, and may contain information that is representable by only a subset of the total inputs. Dimension reduction techniques present methods to represent high-dimensional data using only a subset of the input features (such as in feature selection or clustering) or within a new projection of the feature space to a new space of reduced dimensionality (e.g. principal components analysis [PCA]) (Hastie et

al., 2017). In the context of clinical prediction models, dimension reduction may be applied with the specific goal of reducing the number of predictors retained in the final model. The diversity of patient ICD-9-CM diagnosis data within large patient populations may result in highly correlated or possibly redundant diagnoses. As a result, the use of dimension reduction of ICD-9-CM diagnosis codes prior to clinical prediction model fitting may improve model performance and/or identify only a subset of the original diagnoses to include as predictors.

Treelet dimension reduction (also referred to as treelet transformation or simply treelet) is a recent dimension reduction technique proposed by Dr.'s Ann Lee, Boaz Nadler, and Larry Wasserman in their work "*Treelets – An Adaptive Multi-Scale Basis for Sparse Unordered Data*" (A. B. Lee et al., 2008). The authors present treelet as a dimension reduction technique inspired by hierarchical clustering and PCA, that attempts to represent the original input variables in a reduced number of variables *and* identify only a subset of these transformed variables responsible for much of the information present in the original data, ideally both reducing the number of dimension and identifying a sparse space of the original inputs that inform these transformed features. The authors offer example applications of treelet dimension reduction in datasets of cell imaging and DNA microarray data. Beyond these clinical examples, the authors use treelet in a dataset of internet advertisements, transforming data set of 760 original, categorical (binary) predictors which result in improved classification over the original features. This set of binary features parallels the structure of a large data set of diagnosis codes, where binary variables may represent the presence or absence of diagnoses. In fact, treelet has previously been applied in an observational cohort study of traumatic brain injury (Kumar et al., 2018) to identify groups of correlated diseases. However, treelet has not yet been applied to a data set of ICD-9-CM diagnosis codes in the context of critical care admissions or large data sets with a diverse patient population,

which may specifically benefit from the identification of a reduced, sparse feature space of ICD-9-CM diagnosis codes prior to the construction of clinical models.

1.4 Objectives

The objective of this thesis is to transform a large number of ICD-9-CM diagnosis codes into a sparse set of features, using treelet dimension reduction, and apply this new feature space in the prediction of clinical outcomes of in-hospital mortality, unplanned hospital re-admission, and hospital length of stay. The analyses presented in this work leverage a prospective cohort study of intensive care unit (ICU) admissions to identify this new feature space before building and assessing the predictive validity of models built using the treelet-generated features.

Section 2 details the statistical methods used for both dimension reduction (treelet) and regression (logistic and negative binomial) models. **Section 3** presents the results of statistical analyses, including descriptive statistics and the results of treelet dimension reduction and supervised models of this work's three clinical outcomes of interest. **Sections 4** and **5**, respectively, contain a final discussion of these results and their possible implications.

2.0 Methods

2.1 Data

2.1.1 Data Source

Data for these analyses were accessed from the Massachusetts Institute of Technology's MIMIC-III database (Johnson et al., 2016). The MIMIC-III database contains nearly 60,000 admissions to the ICU of the Beth Israel Deaconess Medical Center in Boston, MA between 2001 and 2012. Information contained in the MIMIC-III data is stored in 26 tables containing distinct data elements and related metadata. Patient data and admissions were linked by common patient (*SUBJECT_ID*) and stay/admission (*HADM_ID*) identifiers. Information was pulled from the following MIMIC-III tables:

Table 1: MIMIC-III Data Tables

Table	Data Elements
Diagnoses_ICD	Diagnoses codes for a given patient's hospital stay
D_ICD_Diagnoses	Text descriptions of diagnosis codes
Admissions	Date of admission and discharge for use in isolating first and most recent hospital admissions and length of stay (using admission and discharge times); Insurance/Payment method for a given stay
Patients	Patient-level data including date of birth, sex, and mortality status

A subset of 7,537 patients were admitted multiple times, resulting in inclusions in the MIMIC-III database corresponding to each hospital admission. For these patients, only data from the earliest admission was retained. Data for analysis were restricted to adult patients ($\geq$18 years at date of admission) resulting in 38,554 patients included in the full analytic cohort of mortality and hospital length of stay. In analysis of the hospital re-admission outcome (described further in **Section 2.1.4**), individuals who died within a year of discharge with no hospital re-admission were excluded from analysis (n=9,661), resulting in an analytic subset of 28,893 patients for analysis of unplanned hospital re-admission.

2.1.2 Diagnosis Codes

Each patient admission included one or more clinical diagnoses, designated using ICD-9-CM codes (CMS, 2020, p. 9). To assess diagnosis code validity in the MIMIC-III data, all present codes were confirmed to correspond to valid, ICD-9-CM diagnoses using the *icd* package (Wasey et al., 2020), and sex-specific diagnoses (e.g. codes 600-608 among male patients, 614-630 among female patients) were confirmed to be accurately diagnosed. ICD-9-CM diagnosis codes with a "*V*" or "*E*" prefix (respectively designating health factors/health service interactions and external injury causes) and those with <1% prevalence in the analytic cohort were removed, retaining 178 ICD-9-CM diagnosis codes in the final analytic data set. Patients were then assigned indicator variables corresponding to each diagnosis, with a value of *1* representing presence of a given diagnosis and otherwise a value of *0*.

2.1.3 Covariates

Additional data elements included as covariates in statistical analysis included age, genotypical sex, and primary payment method/insurance coverage. Age values ranged from 18-89+ in the original data. For individuals over 89 years old at time of admission, the MIMIC-III data administrators mask age, such that patient age data was unavailable. As a result, a value of 90 years old (as the minimum possible age for these patients) was assigned to these individuals. Age was assessed continuously, with values ranging (after imputation) from 18 to 90 years old. Primary payment method was categorized in the mutually exclusive categories of "*Medicare*", "*Medicaid*" "*private coverage", "self-pay*", or *"other public assistance*".

2.1.4 Outcomes

Clinical outcomes included in-hospital mortality, hospital re-admission, and general hospital length of stay. Hospital re-admission was identified as a patient having an additional admission within one year of discharge from their earliest admission. Patients who died within a year of their initial discharge with no additional hospital admissions were excluded from re-admission analysis. Lastly hospital length of stay was measured in days of total hospital, from date of admission to discharge (or to date of death for patients who died during their hospital stay).

2.2 Statistical Analysis

Treelet can be heuristically considered a combination of (and was inspired by) the common dimension reductions techniques of PCA, wavelets, and hierarchical clustering (A. B. Lee et al., 2008). In this work, we apply the treelet method to the correlation matrix of ICD-9-CM diagnosis codes to represent these 178 features with reduced dimensionality. The implementation of treelet is discussed in further detail in **Section 2.2.1**. The resulting features are then used in regression modeling of clinical outcomes: in-hospital mortality, hospital length of stay, and hospital re-admission. We use cross-validation (see **Section 2.2.3**) to identify the treelet transformation's basis matrix (or simply the specific transformation of our original input variables) that optimizes the performance of regression and/or classification models (see **Sections 2.2.2, 2.2.4**). This process was repeated for each outcome, such that treelet was fit to the analytic cohort for the respective clinical outcome, cross-validation performed (using the appropriate regression model) within this cohort to identify the final treelet transformation used, and then the performance of the final model assessed in the appropriate, outcome-specific test data set.

2.2.1 Treelet Dimension Reduction

Lee et al. proposed the treelet method as a dimension reduction method to represent the internal or latent structure of noisy, high-dimensional data using a sparse number of features (A. B. Lee et al., 2008). Treelet attempts to identify correlated variables that may be grouped together to serve as these sparse features. The method is proposed to both reflect the underlying structure of the input data (or its similarity matrix) and secondarily to improve regression (or classification) models by using the transformed, sparse feature space.

Let $p = 1,2,\ldots,P$ represent the number of features in and $n = 1,2,\ldots N$ the number of observations for an input data set. Treelet begins with the input of a similarity matrix, which is defined as the 0th level similarity matrix $\mathbf{M_0}$. Commonly (and in this specific analysis) this is the correlation matrix of the input features. Treelet defines a 0th level basis matrix as the identity matrix, such that $\mathbf{B}_0 = \mathbf{I_{P\times P}}$. Using this 0th level matrix, the method repeats the following process for levels of *l=1,2…,p-1*:

In similarity matrix ($\mathbf{M_{l-1}}$), identify the two features of maximum similarity, or:

$$p_i, p_j = argmax_{i,j\in P, i<j}\ (\mathbf{M}_{l-1}) \qquad (1)$$

Then identify a Jacobi rotation matrix for a given level as $\mathbf{J_l}$[1]. Define the angle of rotation $\theta_l = 0.5 \times arctan\left(2\frac{\rho_{(i,j)}}{\rho_{(i,i)}\rho_{(j,j)}}\right)$ of variance for features $p_i, p_j = \rho_{i,i}, \rho_{j,j}$ respectively and similarity of the two features $\rho_{i,j}$. The resulting rotation matrix $\mathbf{J_l}$ is then defined as:

$$\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & \cdots & cos(\theta_l) & \cdots & -sin(\theta_l) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & sin(\theta_l) & \cdots & cos(\theta_l) & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad (2)$$

[1] This rotation is equivalent to performing PCA on features p_i, p_j of our input matrix **X**, where the values $\neq 0, 1$, and the above Jacobi rotation matrix **J** are equal to the transpose of the local PCA's resulting rotation matrix

where $\mathbf{J_{i,i}} = \mathbf{J}_{j,j} = \mathbf{cos}(\theta_l), \mathbf{J}_{i,j} = sin(\theta_l), and\ \mathbf{J}_{j,i} = -sin(\theta_l)$. Using this rotation matrix, both the basis and similarity matrices are updated, where $\mathbf{B_1} = \mathbf{B_{l-1}J_l}$ and $\mathbf{M}_l = \mathbf{J_l^T M_{l-1} J_l}$. Then identify the new features that maximize the updated similarity matrix $\mathbf{M}_l$, and similarly construct the *l*th level's rotation matrix, and repeat this process until *p-1* orthonormal bases of the input matrix have been constructed.

Each rotation can be considered a "grouping" of two features, which may include both an original input variable and/or a previously grouped treelet feature (containing loadings from previously grouped input variables). Each basis matrix can be considered a representation of the cumulative rotations and may be used as the transformation of our original inputs. Relatively small cut-levels (or the basis matrices identified for the number of rotations much smaller than the original *p* inputs) identify transformations with only a small number of rotations, where our basis matric retains much of the original input data with few grouped features (i.e. only slightly reducing or transforming the original data). Large cut-levels (approaching *p-1* transformations) indicate basis matrices containing loadings from a large number of rotations, such that a subset of the columns in these basis matrices likely contain loadings from a large number of the original input data. As a result, even a small number of features (or small *K*) retained from a large basis matrix likely contain loadings from many of the original input variables.

The treelet model identifies *p-1* constructed bases. Of which we must then identify the number of components to retain (or the new dimensionality of the feature space) *K* and the basis matrix whose *K* components are used, $\mathbf{B}_L$, which is analogous to defining a "cut-level" *L* of the tree. For a given number of components retain (or a given *K*), we use Lee et al.'s proposed *normalized energy score* to identify an optimal cut-off (L^*) for the treelet (equivalent to identifying the optimal basis matrix $\mathbf{B}_{L^*}$ from which we extract *K* features). The *l*th basis matrix can be

generally defined by the vectors $\mathbf{B_l} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_P]^{\boldsymbol{T}}$, and the input matrix similarly as $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_P]^{\mathbf{T}}$. The *i*th normalized energy score is then defined as:

$$\varepsilon(\mathbf{w}_i) = \frac{\sum_{n=1}^{N} |\mathbf{w}_i \cdot \mathbf{x}_n|^2}{\sum_{n=1}^{N} ||\mathbf{x}_n||^2}$$

We then arrange the normalized energy scores for each basis matrix $\mathbf{B}_{\boldsymbol{l}}$ in descending order. Then for a given K, we the identify the basis $\mathbf{B}_{L^*} = \boldsymbol{argmax} \sum_{i=1}^{\mathbf{K}} \boldsymbol{\varepsilon}(\mathbf{w}_i)$ (i.e. that which maximizes the summation of the K highest energy scores for a given basis). Thus, for a given K, we can deterministically identify an optimal basis matrix $\mathbf{B}_{L^*}$ as the basis matrix that maximizes the sum of the K highest, normalized energy scores.

The authors propose multiple methods to identify these parameters, dependent upon the goal of the treelet transformation (A. B. Lee et al., 2008). The treelet method itself does not include information from an outcome measure or dependent variable and is constructed using only the structure of the similarity structure (e.g. correlation or covariance matrix) of the input variables. In the absence of an outcome or prediction model of interest, the final treelet transformation may be selected using some *a priori* criteria (e.g. retaining a specific number of treelet features, retaining all treelet features of the maximum cut-level basis matrix, etc.). In the context of regression and/or classification, the authors suggest identifying the treelet transformation parameters that minimize regression or classification error, which we accomplish using cross-validation (described below and in further detail in **Section 2.2.3**). We used the process described above to identify the basis/cut-off (L^* or $L^*|K^*$) for each K parameter that maximizes the normalized energy score. As a result, identifying the final treelet transformation requires simply

identifying the determined pair K and (L^* or $L^*|K^*$) that minimizes our model error (see **Section 2.2.4** for description of measures of model error/fit).

We identify the treelet dimension reduction's optimal value of K (which we refer to as the K^* orthonormal basis) that minimizes cross-validation using over 5-fold cross-validation (described in further detail in **Section 2.2.3**) for the models of our respective clinical outcomes. Thus, while the treelet method itself requires only the input data to identify the *p-1* rotations, we identify the final transformation by observing each transformation's prediction performance for the outcome of interest. As a result, we identify a unique transformation for each of our three respective outcomes (mortality, re-admission, and length of stay).

Once we have identified the optimal K^* dimensions to retain and the resulting cut-off L^*, equivalent to identify the optimal basis matrix $\mathbf{B}_{L^*}$, we restrict inputs to K^* dimensions by retaining only the vectors from the basis matrix with the K^* highest normed energy scores $\varepsilon(\mathbf{w}_i)$. We then project the input matrix to the K^* dimensional space by simply multiplying the original input matrix $\mathbf{X}$ (of $n \times p$ dimensionality) by the newly formed and restricted basis matrix $\mathbf{B}_{L^*}$, resulting in the new ($n \times K$) matrix $\mathbf{X}^*$.

2.2.2 Generalized Linear Modeling

Generalized linear modeling (GLM) is a family of extensions to ordinary least squares linear regression that allows for modeling outcomes variables whose distributions do not follow a standard Gaussian distribution, such as binary outcomes, multinomial outcomes, proportions, counts, etc (Nelder & Wedderburn, 1972). Ordinary least squares regression models assume that the outcomes follow identical, independent standard Gaussian distributions, or:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \epsilon_i \sim \mathbf{N}(\mathbf{0}, \boldsymbol{\sigma}^{\mathbf{2}}) \quad (3)$$

GLM extends this framework while only requiring that the distribution of the outcome, $\mathbf{y}$, follows a distribution of the exponential family. This distribution is also referred to as the *random component* of a GLM. Each specific extension includes a characteristic *link* function that specifies the relationship between the random component and the input data (also referred to as the "systematic component").

Outcomes of in-hospital mortality and hospital re-admission were represented as binary variables, such that the outcomes for these models follow a binomial distribution. As a result, we fit logistic regression models, with a binomially distributed random component and a logit link-function. Logistic regression models, therefore, model the logit or log-odds of the respective outcome's probability (π):

$$logit(\pi_i) = log\left(\frac{\pi_i}{1-\pi_i}\right) = \mathbf{x_i}\boldsymbol{\beta} \quad (4)$$

The hospital length of stay outcome, as the number days between patient admission and discharge, is a count variable, taking only positive values. Count outcomes are commonly modelled using Poisson regression, with a Poisson distributed random component and the log link function. Poisson regression, however, assumes the equality of the mean and variance of the outcome, a characteristic of the Poisson probability distribution. Should this assumption not be met, the outcome variable is described as overdispersed, and the Poisson probability distribution (and Poisson regression) are inappropriate. Overdispersion can be tested by comparing the deviance (defined $\phi = -2ln(L)$, for likelihood L) of a Poisson regression model to its ${\chi_{n-p}}^2$

distribution under the null hypothesis of no overdispersion (or equal mean and variance). Overdispersion can be alternatively assessed by simply comparing the mean and variance of the outcome.

Negative binomial regression is commonly used when the Poisson regression's assumption of equal mean-variance assumption is not met (i.e. the data are overdispersed). The negative binomial's probability mass function may be expressed as:

$$P(y_i) = \frac{\Gamma\left(y_i + \frac{1}{\alpha}\right)}{(y_i!)\Gamma\left(\frac{1}{\alpha}\right)}\left(\frac{1}{1+\alpha\mu_i}\right)^{\frac{1}{\alpha}}\left(\frac{\alpha\mu_i}{1+\alpha\mu_i}\right)^{y_i} \quad (5)$$

$$where\ \mu_i = exp(x_i\beta)$$

The parameters α, β are then derived via maximum likelihood estimation for the resulting likelihood function $\prod_{i=1}^{N} P(y_i)$. The use of the log-link function, $\mu_i = exp(x_i\beta)$, restricts the model's fitted values to be non-negative, matching the characteristic of the count outcome variable.

2.2.3 Cross-Validation

Cross-validation is a useful process of data sampling and splitting to build and assess the predictive ability of statistical models (Harrell, 2001; Shao, 1993). As previously alluded, 5-fold cross-validation was used to identify the optimal value for the parameter K^*. Prior to cross-validation, analytic cohorts were split, such that 20% of each cohort was held-out and remained unused through any cross-validation or model fitting. The remaining 80% of each analytic cohort was then randomly grouped into 5 equal sized subsets as "model-fitting" or "cross-validation" data sets. Because of the additional exclusion criteria for categorization of unplanned hospital

readmission, data splitting was conducted separately for length of stay and mortality analysis (*training n*=30,844; *test n*=7,710) and hospital re-admission (*training n*=23,115; *test n*=5,778).

In one iteration of cross-validation, the first subset was held-out of the cross-validation data, and the treelet model fit on the remaining 4 folds of data. After fitting the treelet model, within this same subset the original ICD-9-CM diagnosis code variables were transformed using the basis matrix $\mathbf{B}_{K,L|K}$ for each pair of parameters $K, L|K$. We then fit the appropriate GLM (using the previously described logistic or negative binomial models as appropriate) for each outcome, using the previously described covariates and transformed input matrix, resulting in 177 (*p-1* for *p*=178) fitted models. Each fitted model was then used to predict the outcome in the first cross-validation fold, which had been held out from this model-fitting step, and the test-error of each model then reported for that fold. This process was then repeated five times, such that each cross-validation fold was used as the cross-validation test data exactly once. We then identified the values of the K parameter (K^*), and the corresponding basis or cut-off level $\mathrm{L}* |K*$ for the final treelet transformation in assessing the average model fit across the five test folds (see **Section 2.2.4** for the performance metrics of model fit/test-error and description of parameter selection).

For identification of the K parameter, cross-validation can identify both the value that maximizes model performance (based on the below described performance measures) and the smallest value of K that produces a performance measure within one standard deviation of the best performing model. This "one standard deviation rule" allows for the identification of a parameter value that produces a further reduced model (by reducing the number of retained features, K) at a tolerable cost to model performance (Hastie et al., 2015, 2017).

2.2.4 Model Fit

The fit of regression models (for each respective outcome and GLM method) was assessed during the cross-validation process and in the final hold-out data using similar metrics. In the logistic regression models of in-hospital mortality and hospital-readmission, the Brier Score measured accuracy of predicted probabilities (Brier, 1950; Rufibach, 2010). For N observations (or $n_i, i = 1,2, \dots, N$) with predicted probabilities of event $\widehat{p_i}$ and observed outcome y_i (where 0 represents "no event" and 1 an observed event), the Brier Score is defined as[2]:

$$\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\widehat{p_i} - y_i)^2 \qquad \textbf{(6)}$$

Smaller Brier scores indicate more accurate prediction (or better prediction model performance for binary classification models). The minimum Brier score of 0 indicates perfect prediction (i.e. predicted probabilities of 0 for all observed non-events and 1 for all observed events).

In negative binomial regression models of hospital length of stay, the mean-squared error (MSE) of predicted values assessed model fit. For an observed length of stay values y_i and corresponding predicted values $\widehat{y_i}$ among N observations, the MSE of a model was calculated as:

$$\sum_{\mathrm{i}=1}^{\mathrm{N}}(\mathrm{y_i} - \widehat{\mathrm{y_i}})^2 \qquad \textbf{(7)}$$

[2] Notice this equation is analogous to the calculation of mean-squared error (MSE) in OLS regression, replacing the predicted outcome $\widehat{y_i}$ in the MSE equation with the predicted probability of $\widehat{p_i}$ calculated from the logit-link function.

Once we have identified the optimal parameters for the treelet models within the 80% cross-validation subset, we fit a GLM using the treelet basis transformation of the input matrix on the full 80% cross-validation subset. The resulting fitted model then predicted the outcome in the hold-out, 20% subset that was not used in the cross-validation process. These test predictions were compared with the observed outcomes in this hold-out set to assess final model fit.

In logistic regression models for binary outcomes of hospital readmission and in-hospital mortality, test-model performance was additionally assessed using the area under receiver operating characteristic (ROC) curve (AUC) values. While the Briers score is used in comparing models internally (i.e. within cross-validation to identify the number of treelet features to retain), AUC values are more commonly presented, allowing for comparison of the presented results to previously reported models. AUC values were attained by the following steps:

1) Identify all possible classification thresholds of predicted probabilities ($\hat{p}$) that result in distinct combinations predictions for n observations
2) For each threshold, calculate the $sensitivity$ $\left(\frac{True\ Positives}{True\ Positives + False\ Negatives}\right)$ and $specificity$ $\left(\frac{True\ Negatives}{True\ Negatives + False\ Positives}\right)$
3) Plot $sensitivity$ against $1 - specificity$ for all thresholds and corresponding values
4) Calculate the area under this curve

This value is similarly a measure of concordance, as the value calculated above is equivalent to the proportion of all pairwise comparisons of individuals between the two observed classes with the predicted probability of an "event" is greater in the individual with an observed

"event". In "ties", or pairs with equivalent predicted probabilities, a value of 0.5 is summed (rather than a 1 or 0 among pairs with non-equal predicted probabilities).

2.2.5 Model Comparisons

To contextualize the results of treelet dimension reduction, the performance of models containing treelet features are compared to a the Charlson and Elixhauser comorbidity indices as well as PCA, a common method of dimension reduction, and lasso regression. The Charlson (Charlson et al., 1987) and Elixhauser (Elixhauser et al., 1998) proposed groups of diagnoses (also indicated by ICD-9-CM diagnosis codes) thought to be predictive of in-hospital and long-term mortality. The Charlson index assesses the presence of 19 groups of conditions, including diagnoses such as history of cerebrovascular infarction, presence of dementia, presence of liver disease, and other chronic conditions. The Elixhauser index groups diagnoses into 31 categories indicating groups of diseases related to chronic diseases such as acquired immunodeficiency syndrome, lymphoma, diabetes, and hypertension (among other conditions). Both indices assess the presence or absence of relevant ICD-9-CM codes for each patient to create categorical variable describing patient membership in each indices' categories. Patients in the presented analyses were assigned 19 and 31 binary variables for the Charlson and Elixhauser indices respectively, describing the presence of absence of ICD-9-CM diagnosis codes for each related disease group.

In contrast to the Charlson and Elixhauser indices, which use subsets of ICD-9-CM diagnosis codes determined independent of the present data, penalized regression aims to identify a subset of predictors based on models fit using each study's analytic cohort. More specifically, lasso regression deliberately bias the β-coefficient estimators through the introduction of a shrinkage penalty, often referenced as λ, in a regression model. While ordinary least squares

regression (i.e. regression with no shrinkage penalty) identifies a set of β-coefficients that minimizes the sum of squared errors as, or $\hat{\beta}_{OLS} = \text{argmin}(\sum_{i=1}^{n}(y_i - \beta x_i)^2)$. Lasso regression introduces the shrinkage penalty, such that the lasso minimizes the sum of squared errors including this small penalty: $\hat{\beta}_{lasso} = \text{argmin}\left(\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda \sum_{j=1}^{p}|\beta_j|\right)$. A range of shrinkage penalty values (or λ values) can be assessed via cross-validation and a value identified that either minimizes test error or following the previously described "one-standard-deviation rule" to identify a further reduced number of predictors to retain. The extension of lasso to logistic and negative binomial regression then simply involves including the shrinkage penalty, λ, in the linear component of each model.

Lastly, PCA is a common dimension reduction technique that projects original input data into a smaller dimension space comprised of orthogonal linear combinations of the original inputs, or principal components. After identifying the resulting principal components, the final transformation of the original input data requires identifying the number of principal components to retain. The number of principal components to retain can be determined by assessing the number of cumulative variance (in the original input data) accounted for by the subsequent number of principal components using some prior determined threshold.

All PCA and lasso analysis used the same data (including the raining and test data-splits of model fitting and assignment of cross-validation folds) as the treelet dimension reduction and resulting model fitting for each respective clinical outcome. In lasso regression, the identified shrinkage penalty was selected that minimized test-error across 5-fold cross-validation for each respective outcome (i.e. a unique shrinkage penalty was identified for each clinical outcome). In the use of PCA prior to model fitting, the minimum number of principal components that accounted for $\geq$60% of the variance among the 178 ICD-9-CM diagnosis codes (for each respective

outcome's analytic cohort) were retained (n=65 for mortality and length of stay, n=66 for hospital re-admission). The results of all fit models are then lastly compared to models including indicator variables of each diagnosis code, with no dimension reduction, transformation, or penalization performed.

2.2.6 Software

All data management, visualization, and analysis were performed in R, version 4.0.0, within RStudio, version 1.3.959. The *treelet* package from Drs. Di Liu and Trent Gaugler was used for treelet dimension reduction (Gaugler, 2015) and the *MASS* package for negative binomial regression modeling (Venables & Ripley, 2002). The *tidyverse* family of packages was used extensively for data wrangling and exploratory data analysis in conjunction with the *here* and *icd* packages (Muller, 2017; Wasey et al., 2020; Wickham et al., 2019). The *MASS* package was used for its implementation of negative binomial regression (Venables & Ripley, 2002). In addition to the tidyverse's *ggplot2,* the *corrplot* and *lares* package were used specifically for exploratory data visualization, and the *gghighlight* extension to *ggplot2* to visualize treelet parameter identification (Lares, 2020; Wei & Simko, 2017; Yutani, 2020). The *glmnet* (Friedman et al., 2010) package within *caret* (Kuhn, 2020) and the *mpath* package (Wang, 2020) were used to extend lasso regression in logistic and negative binomial regression respectively. Lastly, the *pROC* package was used to generate ROC curves and AUC values (Robin et al., 2011).

3.0 Results

Analysis results are presented in two sub-sections, the first (**Section 3.1**) displays a brief characterization of this study cohort and the correlation structure of the 178 retained ICD-9-CM diagnosis codes. **Section 3.2** contains sub-sections corresponding to one of the respective clinical outcomes and containing results related to the cross-validation parameter selection process and the final treelet and regression modeling test fit. Supplemental tables and figures are included in **Appendix A**.

3.1 Descriptive Statistics

3.1.1 Patients

Descriptive outcome and covariate statistics for the analytic cohort of 38,554 patients are included in **Table 2**. This analytic cohort of ICU admissions presents an older sample, with a mean age of nearly 64 years at time of admission, and a majority male but moderately balanced sample of 56.60% patients and 43.40% female patients. Patients had a median hospital length of stay of 7 days, nearly 15% of patients died during their hospital stay.

Table 2: Analytic Patient Cohort Characteristics

	Analytic Cohort (n=38,554)	Hospital Readmission Subset (n=28,894)
Age, Mean (SD)	63.51 (17.55)	60.92 (17.58)
Sex (Male), n (%)	21,820 (56.60%)	16,663 (57.67%)
Hospital Stay (days), Median (IQR)	7 [4-12]	7 [4-11]
Re-Admission, n (%)	*N/A*	2,153 (7.45%)
In-Hospital Mortality, n (%)	5,586 (14.49%)	*N/A*
Number of ICD-9-CM Diagnosis Codes per Patient, Median (IQR)	7 [5-9]	6 [4-9]
Primary Payment Method, n (%)		
Medicare	20,433 (53.00%)	13,633 (47.18%)
Private Insurance	13,243 (34.35%)	11,209 (38.79%)
Self-Pay	546 (1.42%)	440 (1.52%)
Medicaid	3,169 (8.22%)	2,584 (8.94%)
Other Public Assistance	1,163 (3.02%)	1,027 (3.55%)

IQR = Interquartile Range [25th-75th Percentiles]; SD = Standard Deviation

3.1.2 Diagnosis Codes

Of 6,985 unique ICD-9-CM codes, the exclusion of "*E*" and "*V*" codes resulted in 5,992 diagnoses codes remaining. Of these 5,992 diagnosis codes, 178 were retained with $\geq$1% prevalence. **Figure 1A** displays each diagnosis code frequency in descending order, where we see a subset of codes in the left-most portion of the graph with a comparatively higher frequency. **Figure 1B** specifically displays this information for the top 15 of these most frequent diagnosis codes, with proportions ranging from 8.65% for "Pneumonia, NOS" (not otherwise specified [NOS]) to 42.71% for "Hypertension".

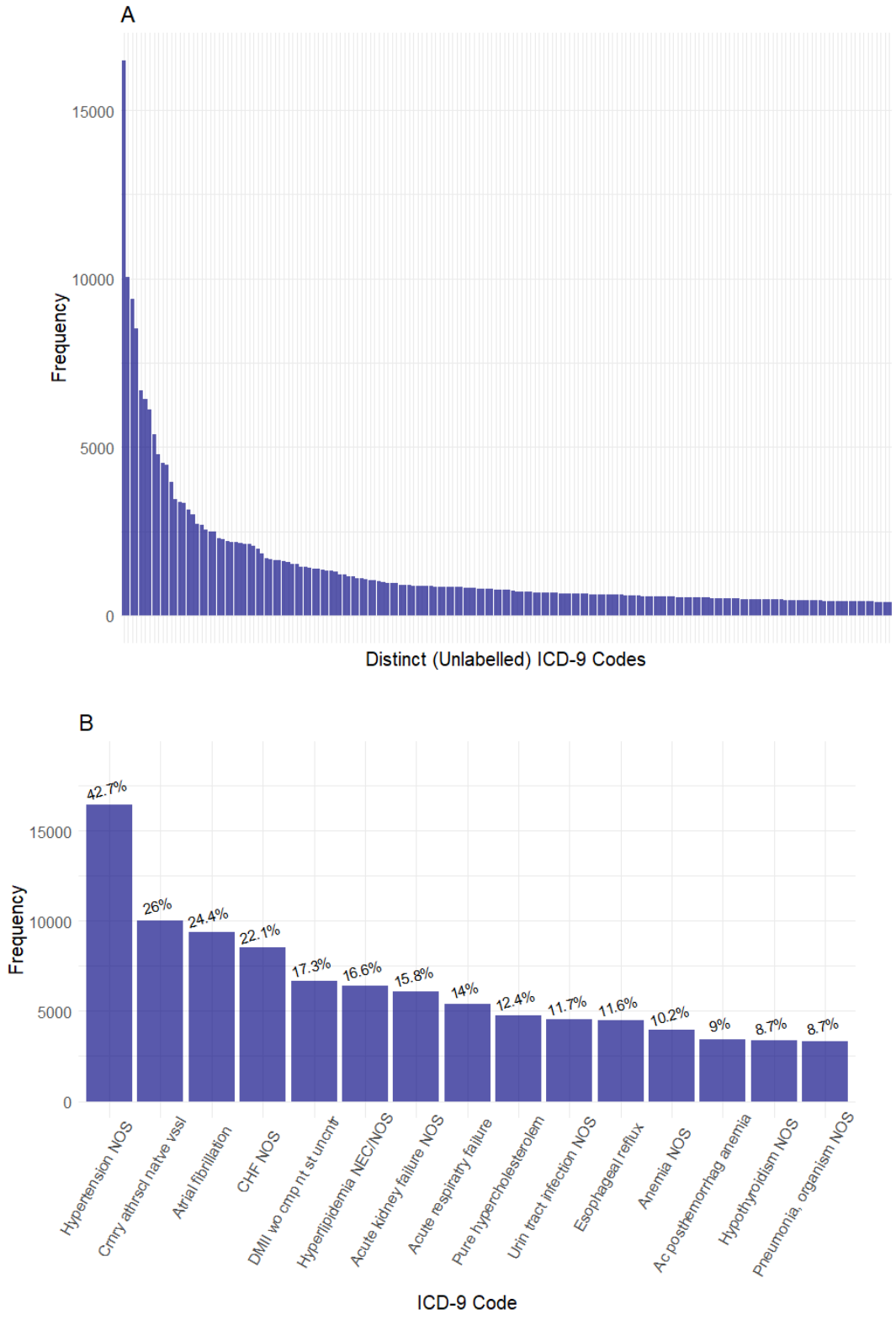


Figure 1: Frequencies of (A) All and (B) 15 Most Common ICD-9-CM Diagnosis Codes

The correlation structure of the diagnosis code data (**Figure 2**) displays pockets of correlated diagnosis codes, most noticeably the groups of dark blue squares near the diagonal. The treelet model uses the correlation structure of the data as the "similarity matrix", with which we will represent the 178 ICD-9-CM diagnosis codes with a comparatively more sparse set of features.

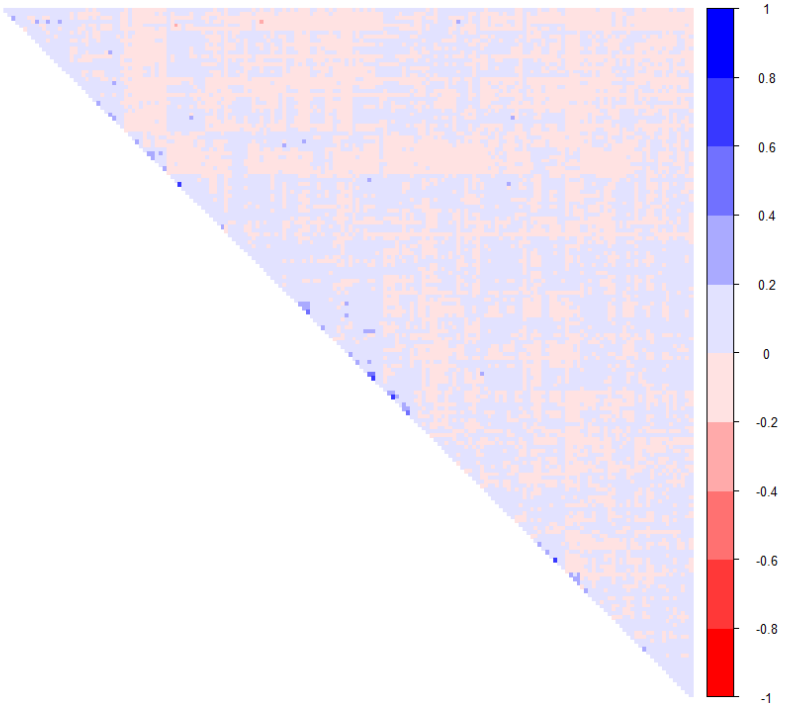
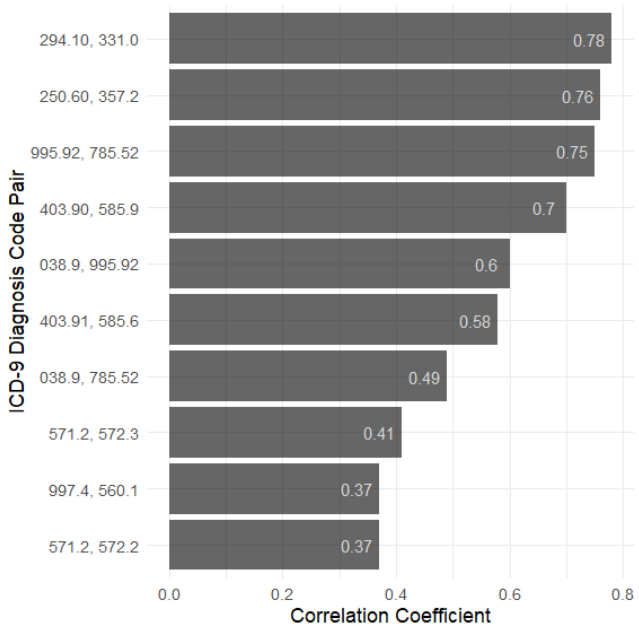


Figure 2: Correlation Matrix of Included ICD-9-CM Diagnosis Codes

The most correlated pairs of diagnosis codes are presented in **Figure 3**. This first pair of the most highly correlated diagnoses are unsurprisingly related diagnoses (294.10, "*Dementia in conditions classified elsewhere*" & 331.0, *"Alzheimer's Disease"*) which will be the first joined pair in the treelet process.



ICD-9-CM Code	Description
038.9	Septicemia NOS
250.60	Diabetes (II) with neurological manifestations
294.10	Dementia without behavioral disturbance
331.0	Alzheimer's disease
357.2	Neuropathy in diabetes
403.90	Hypertensive chronic kidney disease, stage I-IV
403.91	Hypertensive chronic kidney disease, stage V+
560.1	Paralytic ileus
571.2	Alcohol cirrhosis liver
572.2	Hepatic encephalopathy
572.3	Portal hypertension
585.6	End stage renal disease
585.9	Chronic kidney disease NOS
785.52	Septic shock
995.92	Severe sepsis
997.4	Digestive system complications NOS

Figure 3: Ten Most Correlated Pairs of Diagnosis Codes

3.2 Statistical Analysis

3.2.1 In-Hospital Mortality

Figure 4 displays the results of the 5-fold cross-validation of the treelet's K parameter, showing the Brier Score (averaged across the 5 cross-validation folds) for each K (and respective $L|K$) parameter. The red highlighted point indicates the parameters that minimized the Brier Score ($K = 174, L|K = 4$), favoring a model that includes nearly all diagnosis codes. The blue observation indicates the "sparser parameter" ($K = 123\ , L|K = 57$), that is the minimum K value within one standard deviation of the minimized test-error. Using the smaller K parameter allows us to further reduce the feature set with an acceptable "loss" in cross-validation

performance, opting for a sparser model. The final basis matrix for the more sparse (or "one standard deviation" rule) parameters included $K = 123$ dimensions of the cut level (or the Lth basis matrix B_L) of $L|K = 57$. This reduced number of retained features includes loadings from all 178 diagnosis. That is, while we were able to reduce the number of input variables from our 178 original diagnosis codes to 123 treelet features, these retained treelet features do *not* identify a sparse feature space (i.e. still requiring information from all 178 diagnoses in our original input data).

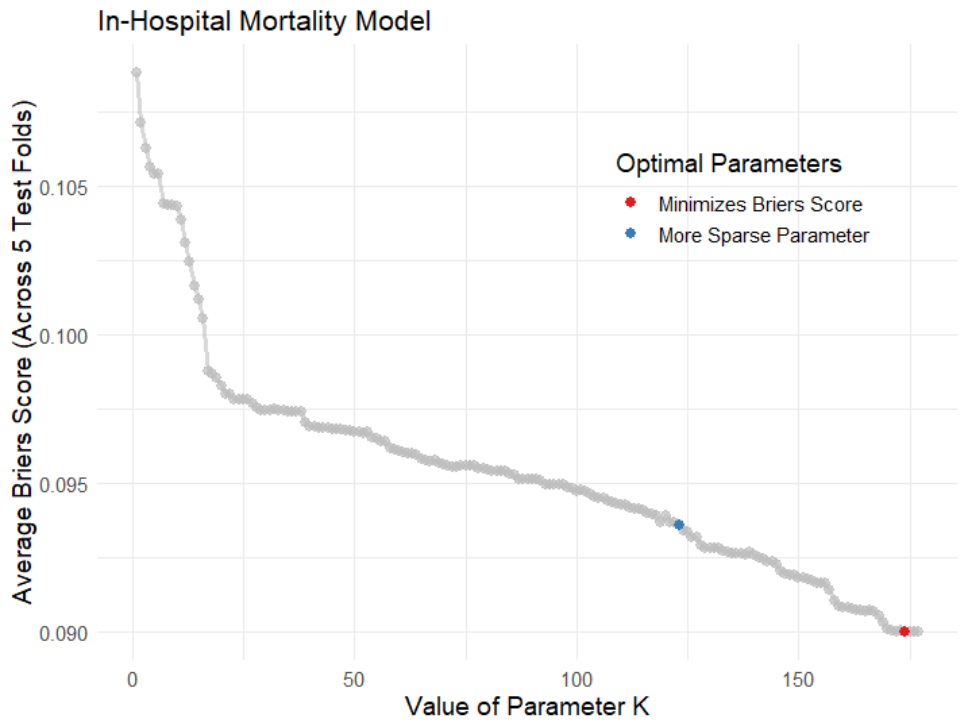


Figure 4: Average Test Briers Score Over 5-Fold Cross-Validation (Mortality Model)

Having identified these parameters, we fit a logistic regression model to the total cross-validation cohort (n=30,844) predicting in-hospital mortality using age, sex, insurance coverage/payment method, and the transformation of the diagnoses codes into the new feature

space using the parameters identified above (patient-level information contained in **Table 3,** full model results including *K=123* treelet features included in **Appendix Table 1**). This fit logistic regression model was then then used to predict probabilities of mortality in the 20% hold-out data set (n=7710), for a final test-performance Brier Score of 0.0916 and AUC of 0.853. **Table 3** additionally contains the results of a model fit including only the patient demographic information, with a final test model Brier Score of 0.1183 and AUC of 0.666. In models of mortality that both include and exclude the treelet features, older age (β=0.031, p<0.001) and a primary payment method of *Self-Pay* (β=1.145 compared to the reference group of "*Other Public Assistance*", p<0.001) demonstrate statistically significant increases in mortality risk. Interestingly, the inclusion of treelet features results in the statistical significance for our covariates of male sex (β=-0.118, p=0.004) and *Medicare* payment method (β=0.328, p=0.032).

Table 3: Logistic Regression Model of Mortality

	Model Excluding Treelet Features			**Model Including Treelet Features***		
Predictor	***β***	**95% CI**	**P-Value**	***β***	**95% CI**	**P-Value**
Intercept	-4.041	[-4.334, -3.747]	<0.001	-5.021	[-5.371, -4.671]	<0.001
Age	0.031	[0.028, 0.034]	<0.001	0.038	[0.035, 0.042]	<0.001
Sex (Male)	-0.050	[-0.115, 0.015]	0.1343	-0.118	[-0.198, -0.037]	0.004
Insurance						
Medicaid	0.374	[0.092, 0.656]	0.0093	0.178	[-0.140, 0.497]	0.273
Medicare	0.252	[-0.014, 0.517]	0.0635	0.328	[0.029, 0.627]	0.032
Private	0.067	[-0.194, 0.328]	0.6142	0.103	[-0.191, 0.397]	0.491
Self-Pay	1.145	[0.788, 1.503]	<0.001	1.174	[0.762, 1.586]	<0.001

Test Model Performance: Brier Score = 0.0917; AUC = 0.858
Test Model Performance (excluding treelet features): Brier Score = 0.1183; AUC = 0.666
Abbreviated model results presented in **Table 3, see **Appendix Table 1** for K=123 included treelet features*

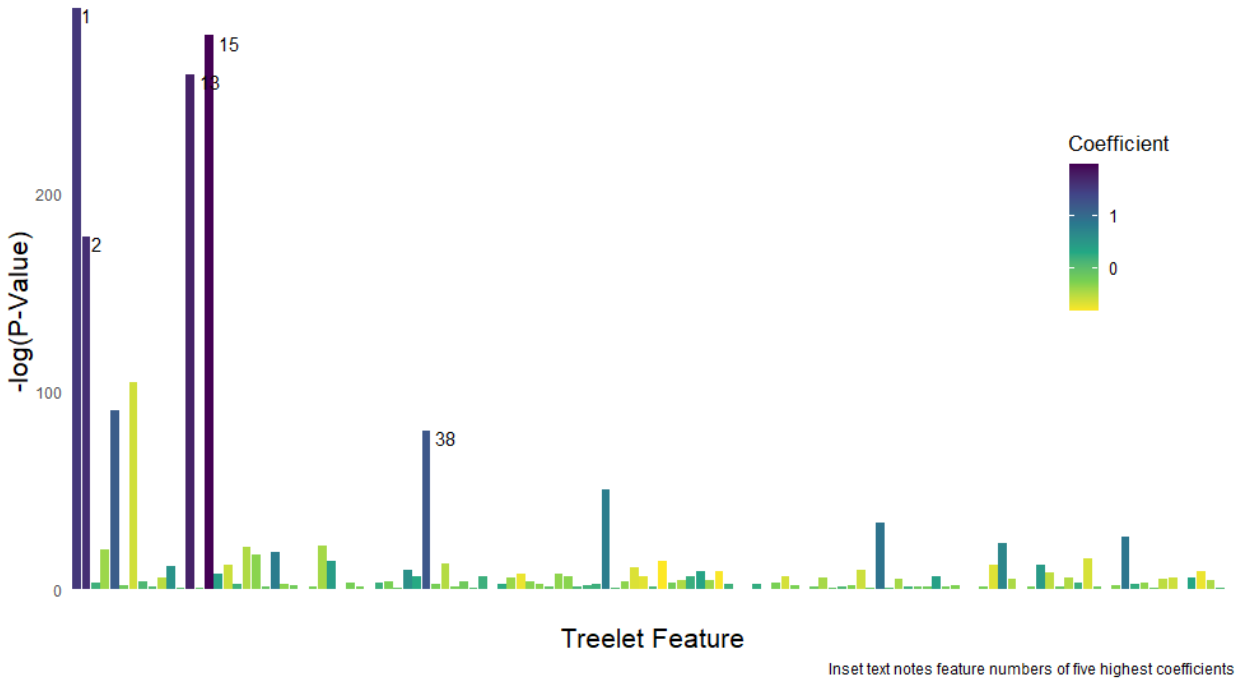


Figure 5: Treelet Feature P-Values & β-Coefficients (Mortality)

For the retained treelet features, the p-value for hypothesis tests of β-coefficients are presented in **Figure 5**, as well as the value of the β-coefficient values, with the five largest coefficients labelled (with coefficients presented in descending order of relative normed energy score). Each bar represents one of K=123 treelet features included in the final model, with the height displaying the those with the highest $-ln(\text{p}-\text{value})$ (equivalent to the *smallest* p-value) and the color displaying the relative value of the point estimate of each feature's β-coefficient.

In the figure we see subset of treelet features with a much lower p-value and comparatively higher β-coefficient than many of the retained treelet features. The five, labelled treelet features (1, 2, 13, 15, and 38) denote features with the highest magnitude beta-coefficient, and the five tallest bars (treelet features 1, 2, 7, 13, and 15) those with the highest $-ln(\text{p}-\text{value})$ (**Appendix Table 4**). Among the treelet features retained in the final model of mortality, feature 1 included

loading from all 178 ICD-9-CM diagnosis codes, where the codes with the highest loading diagnoses corresponding to conditions related to sepsis (codes 995.92 *Sepsis*; 38.9 *Septicemia*) or organ system failure (codes 584.9 *kidney failure;* 518.81 *respiratory failure*) as well as otherwise unspecified pneumonia (code 486.00). Additional features included codes related to cancers/malignancies (feature 2: codes 198.3, *brain/spinal malignancy*; 197.7, *liver* malignancy; 197.0, *lung malignancy*; 198.5, *bone and bone marrow malignant neoplasm*), neurological injury and sequalae (feature 13: 431, *intracranial hemorrhage*; 430, *subarachnoid hemorrhage*; 348.5, *cerebral edema*), and cardiovascular diagnoses (feature 15: *427.5*, *cardiac arrest*; 427.47, *ventricular fibrillation*).

The density curves for the predicted probabilities of the patients in the test (or hold-out) data set (resulting from the final logistic regression model, including all patient demographic covariates and K=123 treelet features) are included in **Figure 6**. The figure presents two density curves, stratified by the patients' observed (or true) mortality status, with the light blue curve representing the distribution of predicted probabilities among patients who survived their hospital stay and the red curve among patients who died. We see that patients observed to have survived their hospital stay have predicted probabilities concentrated below 10% ($0 \leq \hat{p} \leq 0.10$). Patients who died during their hospital stay have more uniformly distributed predicted probabilities, with separation of predicted probabilities between the two groups beginning most notably in the region above 37.5% ($\hat{p} \geq 0.375$).

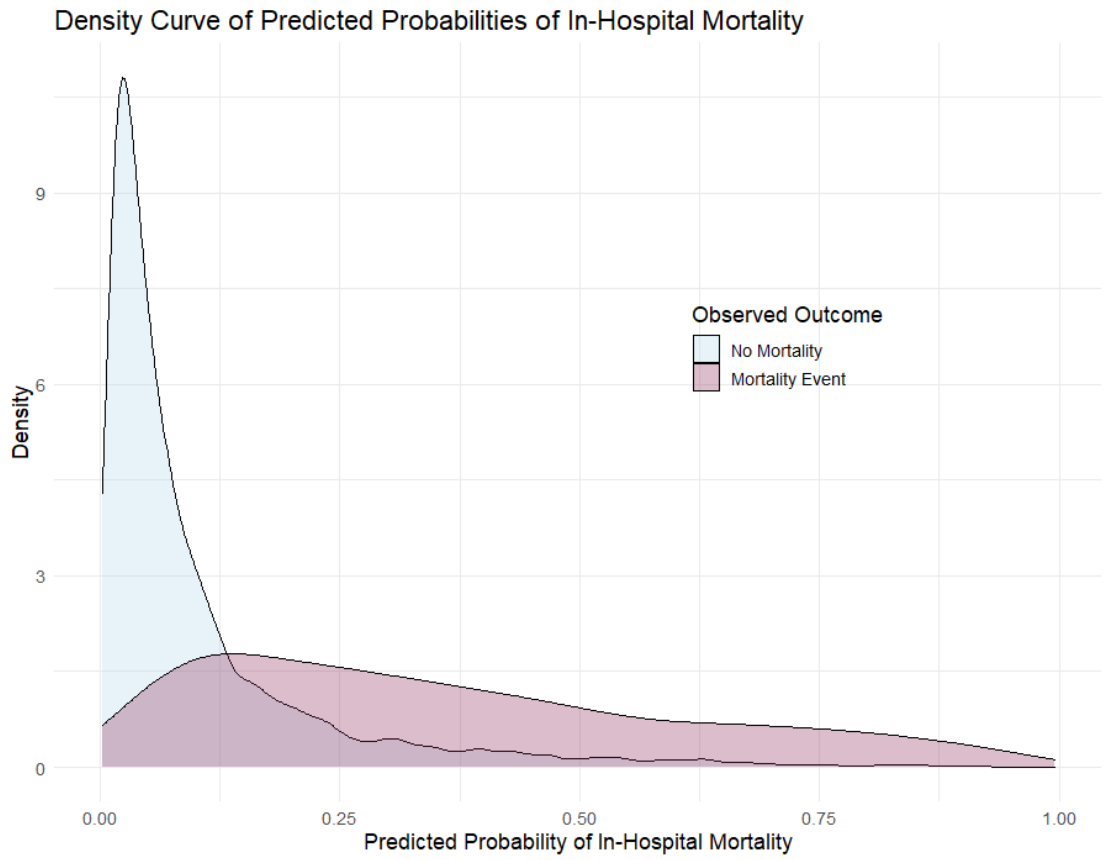


Figure 6: Density Curve of Predicted Probabilities of Mortality

Lastly, **Figure 7** compares the ROC curves and AUC of three logistic regression models of mortality, the first including only the patient demographic variables (represented by the blue curve), the second including patient demographic covariates as well as the five most significant treelet features as outlined above (represented by the orange curve), and the third model including all patient demographic and all $K = 123$ treelet features (represented by the green curve). Inclusion of the treelet features largely improves upon a model built using only the demographic covariates. The predicted probabilities of a model including only demographic covariates also demonstrates poor separation between patients' observed mortality status (**Appendix Figure 1**) compared to the predicted probabilities generated from our model retaining both patient

demographic data and all treelet features (**Figure 6**). Interestingly, in the ROC curve comparison, the model including only the five most significant treelet features appears to be largely responsible for this increase in performance (AUC=0.833 compared to AUC=0.666 for patient demographic covariates alone, AUC=0.858 for demographic covariates and all treelet features).

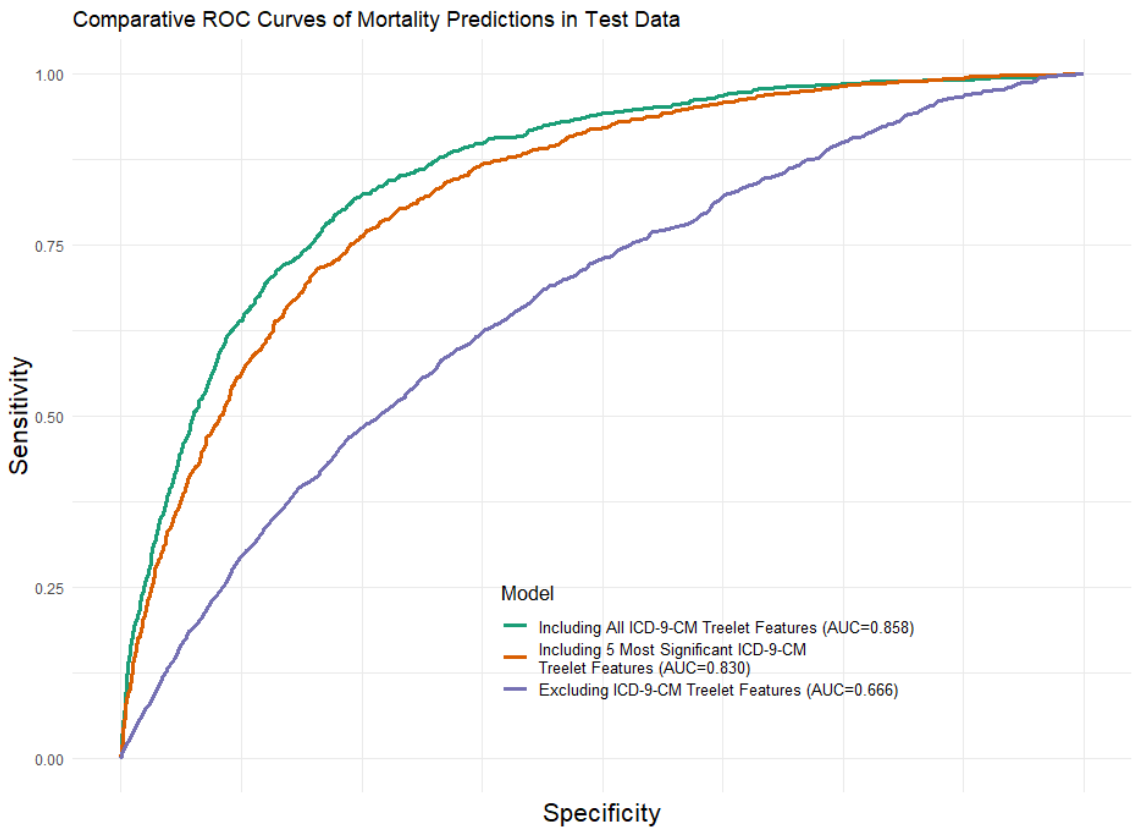


Figure 7: Comparative ROC Curves of Mortality Predictions

3.2.2 Hospital-Readmission

Figure 8 includes the results of 5-fold cross-validation of the treelet's K parameter in prediction of unplanned hospital re-admission, displaying the averaged test performance, measured via Brier Score, for each K (and respective $L|K$) parameter. The plot also highlights

the K and $L|K$ parameter that minimized the Brier Score ($K = 30, L|K = 177$) and the more sparse parameter ($K = 5, L|K = 177$) within one standard deviation of the minimized test-error.

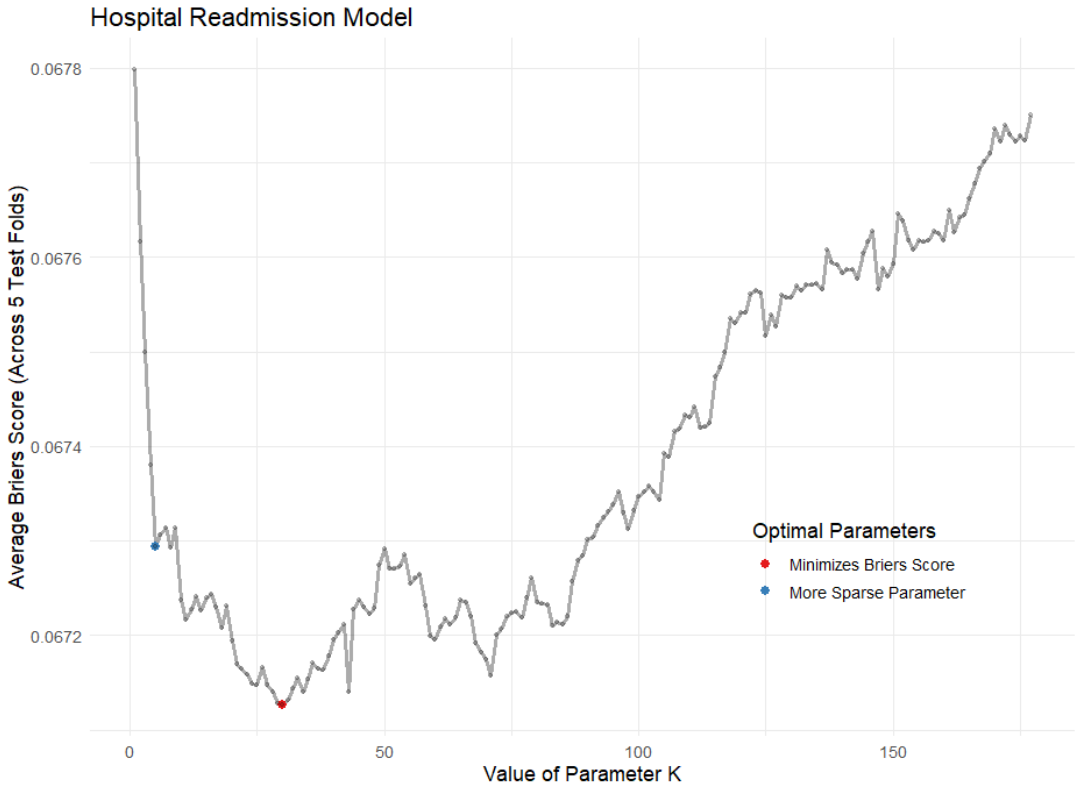


Figure 8: Average Test Briers Score Over 5-Fold Cross-Validation (Readmission Model)

Unlike the results in the mortality treelet, this analysis identifies a reduced feature space, with markedly fewer features than the initial 178 diagnosis codes. However, both the minimizing and more sparse values of K use a large cut-level of $L|K = 177$, or a basis matrix that has undergone all transformations of the treelet model. As a result, although the cross-validation process identifies a much lower number of dimensions to include, the optimal basis matrix (or the identified $L|K$ value) results in basis matrices that similarly include loadings from all 178 diagnosis. As both of the final basis matrices ($K = 30, K|L = 177; K = 5, K|L = 177$) result in

loadings from all diagnosis codes and the K value that minimizes the cross-validation error does reduce the number of input features for diagnosis code data substantially (from 178 to 30), we used the minimizing parameters ($K = 30, K|L = 177$) for the final treelet transformation rather than the more-sparse parameter.

Using the readmission parameters that yielded the lowest Brier Score in cross validation, we fit a logistic regression model to the total training cohort (n=23,115) predicting hospital readmission using age, sex, insurance coverage/payment method, and the transformation of the diagnoses codes into the new feature space (patient covariate results contained in **Table 4**, full model results including *K=30* treelet features presented in **Appendix Table 2**). This logistic regression model was then then used to predict probabilities of unplanned hospital re-admission in the 20% test (or hold-out) data set (n=5,778), for a final test-performance Brier Score of 0.0681 and AUC of 0.661, indicating overall poor predictive performance of this model. We see that only the primary payment methods of *Medicaid*, *Medicare*, and *Self-Pay* were statistically significant predictors of unplanned hospital re-admission, and among these categories only the *Medicaid* category remaining significant with the inclusion of the treelet feature.

Table 4: Logistic Regression Model of Readmission

	Model Excluding Treelet Features			Model Including Treelet Features*		
Predictor	***β***	**95% CI**	**P-Value**	***β***	**95% CI**	**P-Value**
Intercept	-2.608	[-2.942, -2.274]	<0.001	-3.137	[-3.490, 2.783]	<0.001
Age	-0.002	[-0.006, 0.002]	0.3387	0.002	[-0.002, 0.007]	0.455
Sex (Male)	-0.052	[-0.152, 0.048]	0.3078	0.039	[-0.142, 0.064]	0.281
Insurance						
Medicaid	0.619	[0.303, 0.935]	<0.001	0.484	[0.162, 0.806]	0.003
Medicare	0.394	[0.084, 0.705]	0.0129	0.310	[0.005, 0.625]	0.053
Private	-0.086	[-0.384, 0.213]	0.5735	-0.033	[-0.336, 0.271]	0.833
Self-Pay	-0.723	[-1.387, -0.06]	0.0326	-0.608	[-1.278, 0.061]	0.075

Test Model Performance: Brier Score = 0.0681; AUC = 0.661
Test Model Performance (excluding treelet features): Brier Score = 0.0692; AUC = 0.574
Abbreviated model results presented in **Table 4, see **Appendix Table 2** for K=30 included treelet features*

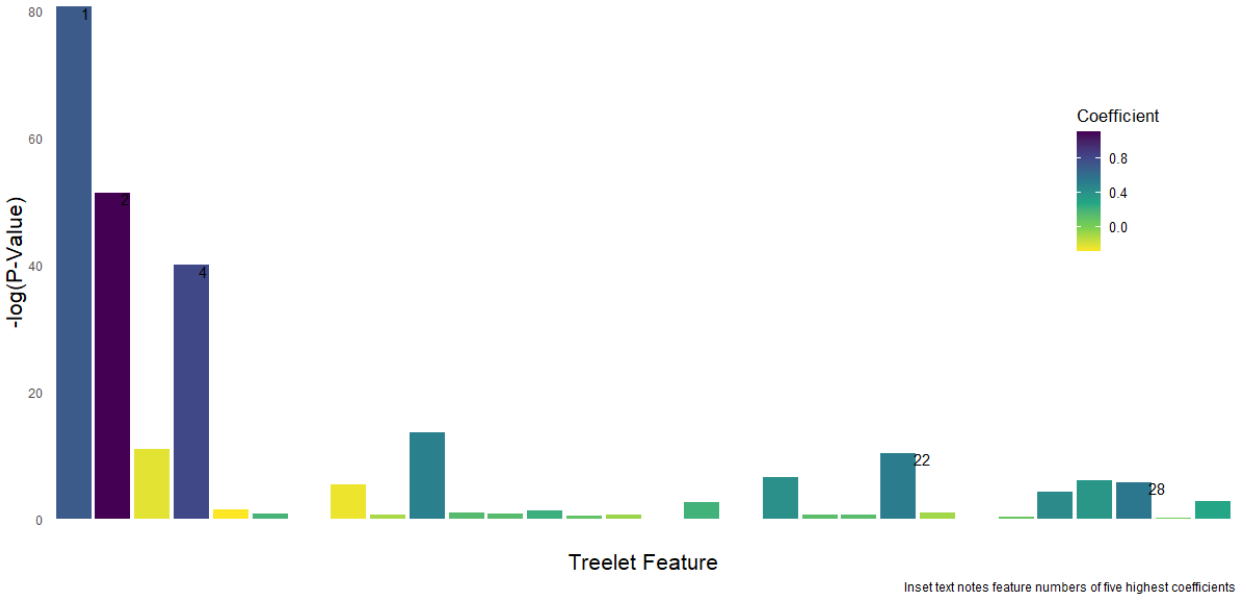


Figure 9: Treelet Feature P-Values & *β*-Coefficients (Readmission)

Figure 9 displays both the p-values and relative magnitude of β-coefficients for the 30 treelet features included in the final regression model, with the five most significant coefficients again labeled. Each bar represents one of K=30 treelet features included in the final model, with the height displaying and the color displaying the value of the point estimate of each feature's β-coefficient. Among the 30 retained treelet features, the graph prominently displays the importance of features 1, 2, and 4 (**Appendix Table 5**). Feature 1 included diagnoses related to organ failure (codes 584.9, *kidney failure*; 518.81, *respiratory failure*; 574.5, *cirrhosis*) and infection (34, *urinary tract infection*). Feature 2 contained similar diagnoses to feature 1 in the treelet features used in mortality, including diagnoses related to sepsis and organ failure. Lastly, feature 4 included diagnoses of diabetes and related complications (codes 250.60, *diabetes mellitus (type II) with neurological manifestations*; 357.2, *diabetes with neuropathy*).

Figure 10 presents the density curves for the predicted probabilities of the patients in the test (or hold-out) data set, resulting from the final logistic regression model, including all patient demographic covariates and K=30 treelet features. The distributions are stratified by the patients' observed readmission status, with the blue curve representing the predicted probabilities of patients with an observed readmission and red curve for those not readmitted. The density curves corroborate the low AUC of the final model in our hold-out test data set, demonstrating a poor separation of predicted probabilities between the patients with observed readmission and those without.

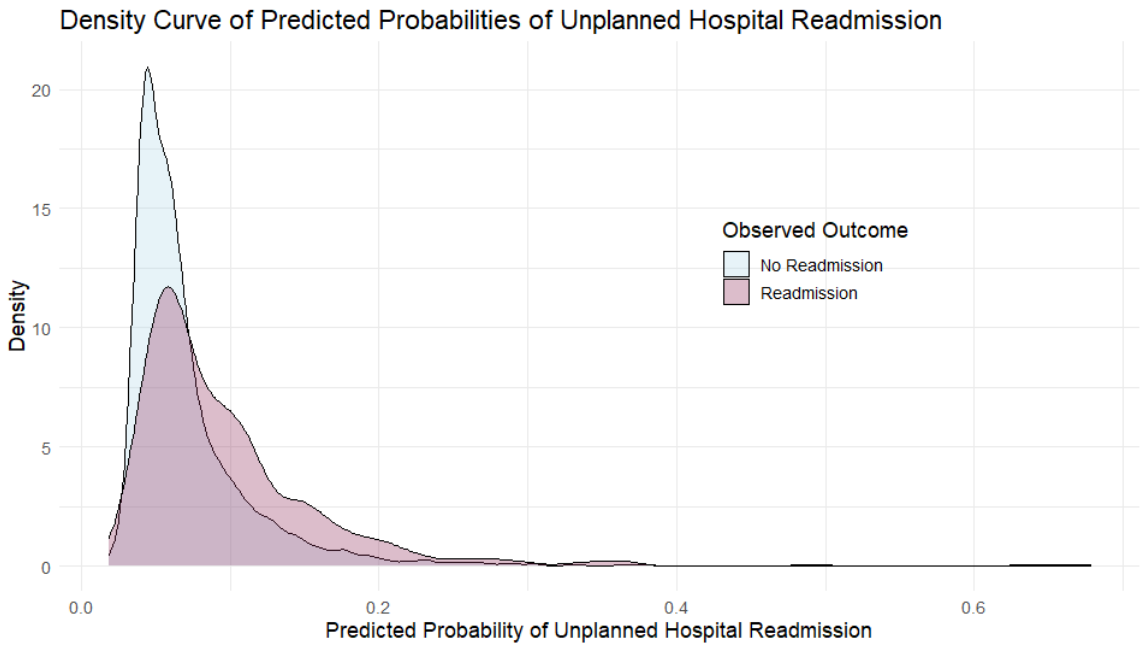


Figure 10: Density Curve of Predicted Probabilities of Readmission

Lastly, **Figure 11** compares the ROC curves and AUC of three models of hospital readmission, the first including only the patient demographic variables (represented by the blue curve), the second including patient demographic covariates as well as the five most-significant treelet features as outlined above (represented by the orange curve), and the third model including all patient demographic and all $K = 30$ treelet features (represented by the green curve). Inclusion of the treelet features slightly improves the upon the model built using only the demographic covariates, indicated by the increased AUC (AUC=0.574 demographic covariates only, AUC=0.661 including demographic covariates and all treelet features). The predicted probabilities of a model including only demographic covariates demonstrate comparatively less separation between patients' observed readmission status (**Appendix Figure 2**) compared to those generated form the model including all treelet features (**Figure 10**). The model including only the five most significant treelet features accounts for nearly all model fit observed by including diagnosis data

(AUC=0.661 including all treelet features, AUC=0.658 including five most significant treelet features).

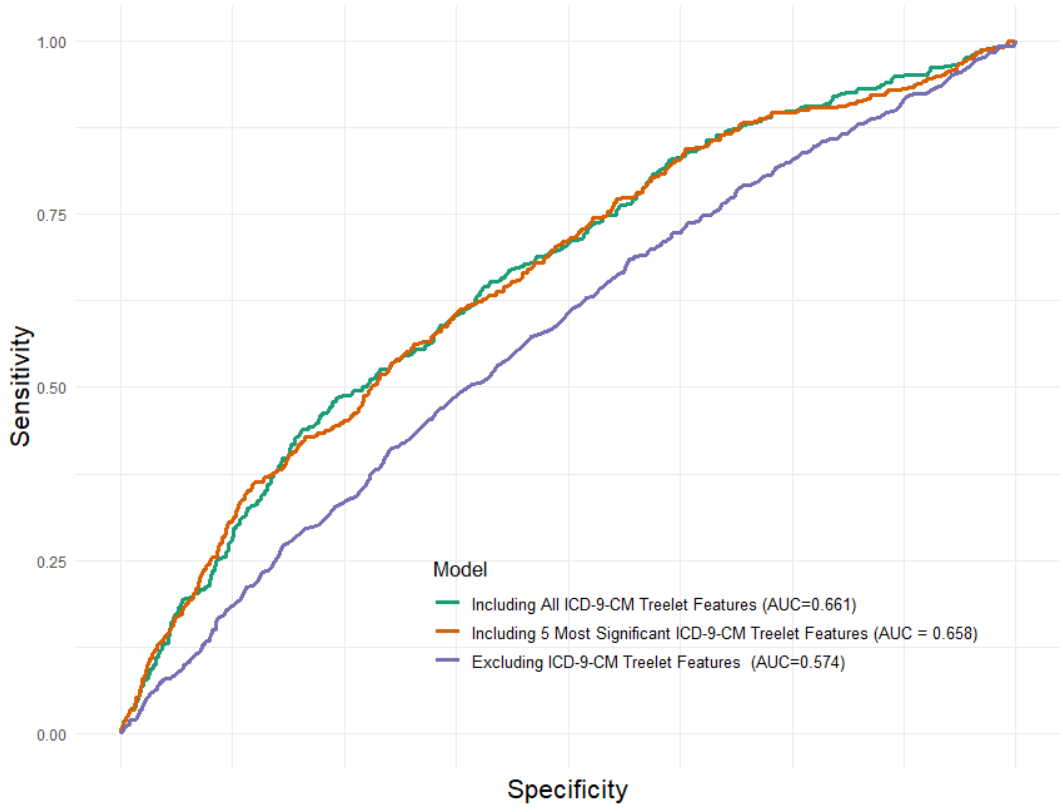


Figure 11: Comparative ROC Curves of Hospital Re-admission Models

3.2.3 Hospital Length of Stay

A negative binomial regression model was fit to predict hospital length of stay, which was considered *overdispersed* with a mean of 9.78 and variance of 112.63 in the full analytic cohort (see **Appendix Figure 3, Appendix B.3** for density curve of length of stay variable). The

Poisson model, fit using the same covariates and observations from our cross-validation data set, also provided evidence of overdispersion ($p<0.001$, data not shown).

Figure 12 includes the results of 5-fold cross-validation for prediction of hospital length of stay, measured via MSE for each K (and respective $L|K$) parameter. The plot includes both the K and $L|K$ parameters that minimized MSE ($K = 115, L|K = 63$) and those that were within one standard deviation of the minimum MSE ($K = 46, L|K = 63$). Contrary to the cross-validation results for predicting in-hospital mortality and hospital readmission, the length-of-stay model identifies a sparse feature space that minimizes MSE. This yielded the parameters identified using one-standard deviation rule to further reduce the feature space rather than merely "correcting" the lack of sparsity. Interestingly, this cross-validation graph also appears to identify values of K that yield overfitting, as MSE *increases* as K increases past values near the minimizing value of 115. The final basis matrix for the more sparse (or "one standard deviation" rule) parameters included $K = 46$ dimensions of the cut level (or the Lth basis matrix $\boldsymbol{B_L}$) of $L|K = 63$ basis matrix, which included loadings from 107 of 178 diagnosis codes.

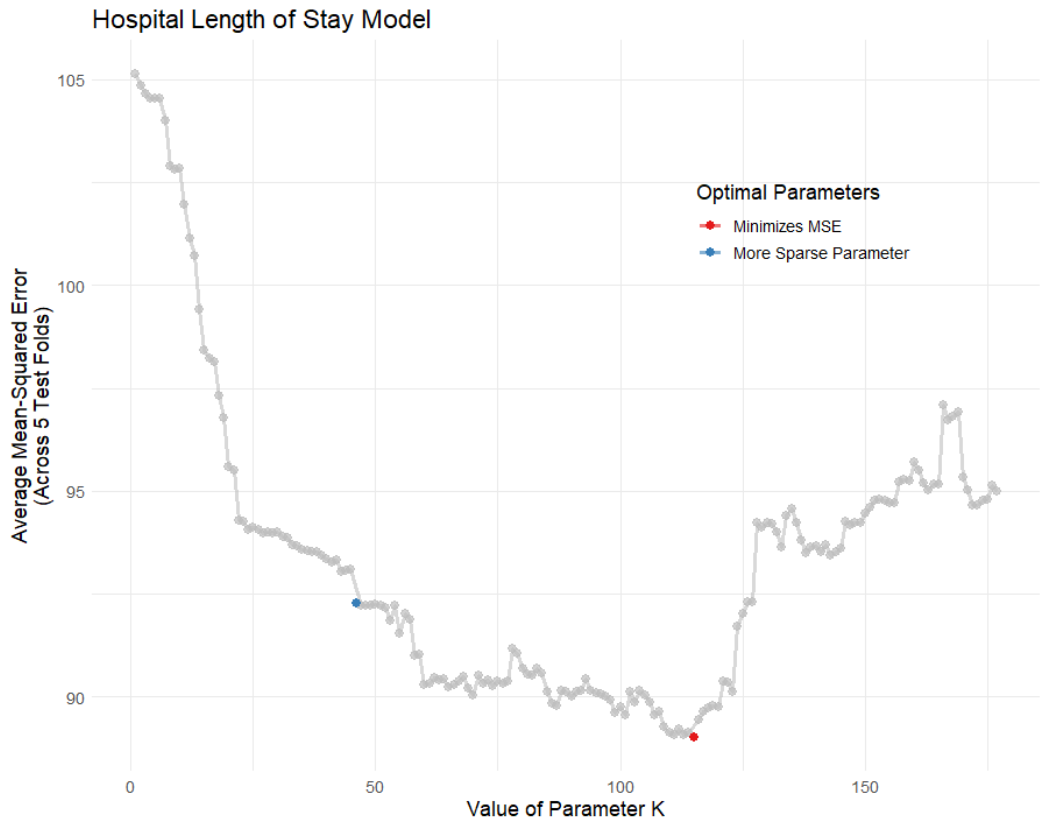


Figure 12: Average Test Briers Score Over 5-Fold Cross-Validation (Length of Stay Model)

We then fit a negative binomial model to the total cross-validation cohort (n=30,884) predicting hospital length of stay using patient-level covariates and diagnoses codes transformed into the new feature space (results for patient demographic covariates information contained in **Table 5**, full model results including $K=46$ treelet features presented in **Appendix Table 3**), which was then then used to predict length of stay values of in the hold-out data set (n=7,710), with an MSE of 105.82.

Table 5: Negative Binomial Model of Length of Stay

	Model Excluding Treelet Features			Model Including Treelet Features*		
Predictor	***β***	**95% CI**	**P-Value**	***β***	**95% CI**	**P-Value**
Intercept	2.310	[2.246, 2.375]	<0.001	2.001	[1.942, 2.061]	<0.001
Age	-0.001	[-0.002, -0.001]	<0.001	-0.002	[-0.003, 0.002]	<0.001
Sex (Male)	0.025	[0.006, 0.044]	0.010	0.053	[0.035, 0.071]	<0.001
Insurance						
Medicaid	0.172	[0.109, 0.235]	<0.001	0.114	[0.058, 0.171]	<0.001
Medicare	0.062	[0.003, 0.122]	0.0388	0.048	[-0.006, 0.101]	0.079
Private	0.008	[-0.049, 0.064]	0.7862	0.039	[-0.12, 0.090]	0.133
Self-Pay	-0.373	[-0.471, -0.274]	<0.001	-0.318	[-0.407, -0.229]	<0.001

Test Model Performance: Root-Mean-Square Error = 10.29
Test Model Performance (excluding treelet features): Root-Mean-Square Error = 11.09
Abbreviated model results presented in **Table 5, see **Appendix Table 3** for K=46 included treelet features*

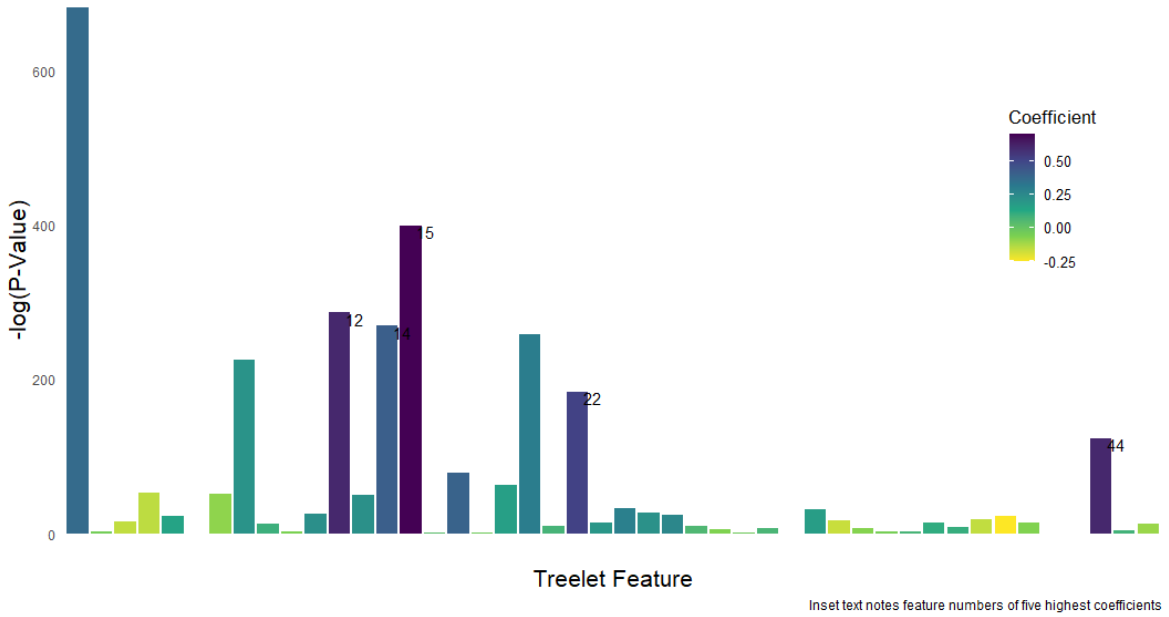


Figure 13: Treelet Feature P-Values & *β*-Coefficients (Length of Stay)

Figure 13 again displays the p-values and relative magnitude of β-coefficients of treelet features included in the final negative binomial model of hospital length of stay. Each bar represents one of *K=46* retained treelet features. Among notable features with high β-coefficients and among the lowest p-values, feature 1 includes diagnoses related to sepsis (995.92, *severe sepsis*; 389 *septicemia*; 785.52, *septic shock*) and organ failure (584.9, *acute kidney failure*; 518.81, *acute respiratory failure*) similar to important features identified in both our models of mortality and readmission. Notably, both feature 12 and feature 14 include only two ICD-9-CM codes, with feature 12 including 997.4 (*digestive complications, not otherwise specified*) and 561.0 (*paralytic ileus*) while feature 14 includes only 518.0 (*pulmonary collapse* and 511.9 (*pleural effusion*).

Figure 14 lastly contains the predicted length of stay values from this negative binomial model against the true, observed length of stay values, with blue dots representing patients with a larger predicted than observed length of stay and red dots patients whose length of stay was underpredicted. The model heavily over-predicted length of stay (with predicted values $\geq$80 days) in a subset of patients with observed length of stays under 50 days while simultaneously underpredicted length of stay (with predictions under 40 days) in a group of patients with observed length of stays over 100 days. However, the bulk of observations are contained in the bottom-left quadrant of the **Figure 14**, with both predicted and observed length of stays concentrated in a range of 0 to 50 days.

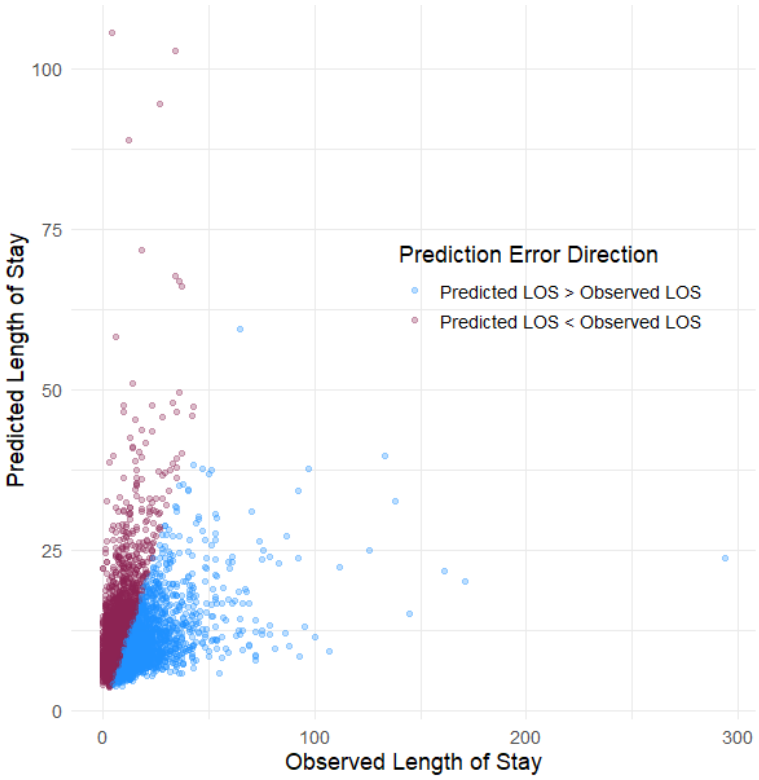


Figure 14: Scatter Plot of Observed and Predicted Length of Stay Values

Figure 15 further demonstrates this concentration of lower length of stay values, separately displaying the density curve of the predicted and observed values of patient length of stay (**Figure 15**). Both the predicted and observed length of stay distributions appear heavily right skewed, with most values contained in under 30 days. The red curve of predicted values tends to underestimate length of stay, evidenced by the higher density of lower length of stay values under 20 days, compared to the blue curve of observed length of stay values which continues with a slightly increased density through 40 days. The underprediction of length of stay values is corroborated by the distribution of the errors of the negative binomial model contained in **Appendix Figure 4**.

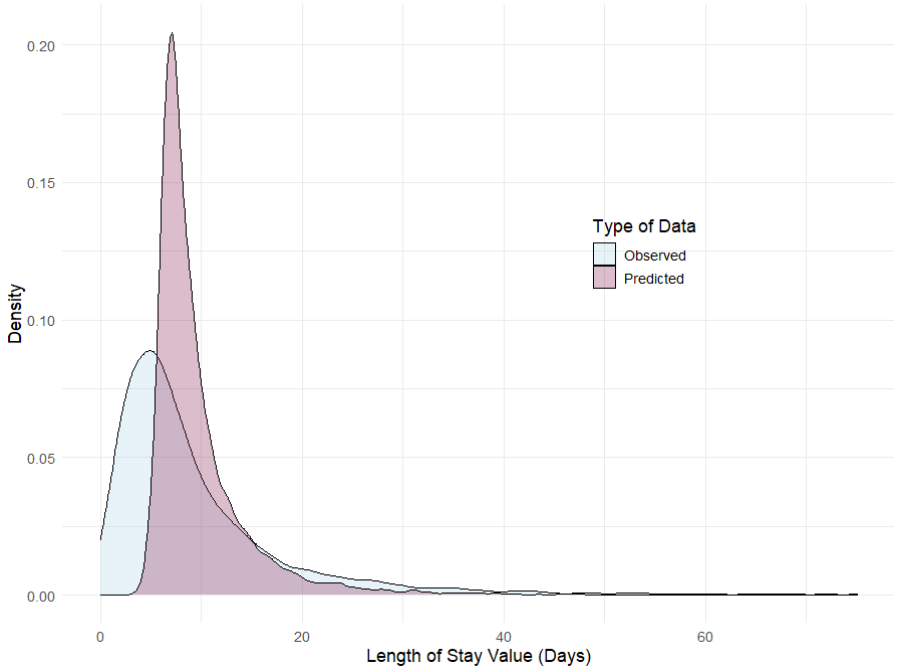


Figure 15: Density Curves of Predicted & Observed Length of Stay Values

Lastly, **Figure 16** displays the root-mean-square error of models including the subsequent addition of the most significant treelet features (from the model fit including all K=46 treelet features and patient demographic covariates). That is, the first point represents a model including patient demographic covariates and treelet feature 1 (the treelet feature with the lowest p-value, seen in **Figure 13**), the second point including the same predictors and adding treelet feature 15 (treelet feature with subsequent lowest p-value, **Figure 13**), and the further points representing the addition of the remaining treelet features, such that the final point in the furthest right portion of the graph represents the final model including patient demographic variables and all K=46 treelet features. The figure displays that the first five treelet features reduces the root-mean-square error from 10.85 to 10.35, while the remaining 41 treelet features only further reduce the root-mean-square error to the final value of 10.29. Thus, similar to the improvement in the AUC in the models of binary outcomes of mortality and unplanned hospital-readmission, the introduction of the five

most significant treelet features appears responsible for the bulk of the model improvement that results from the inclusion if ICD-9-CM diagnosis codes.

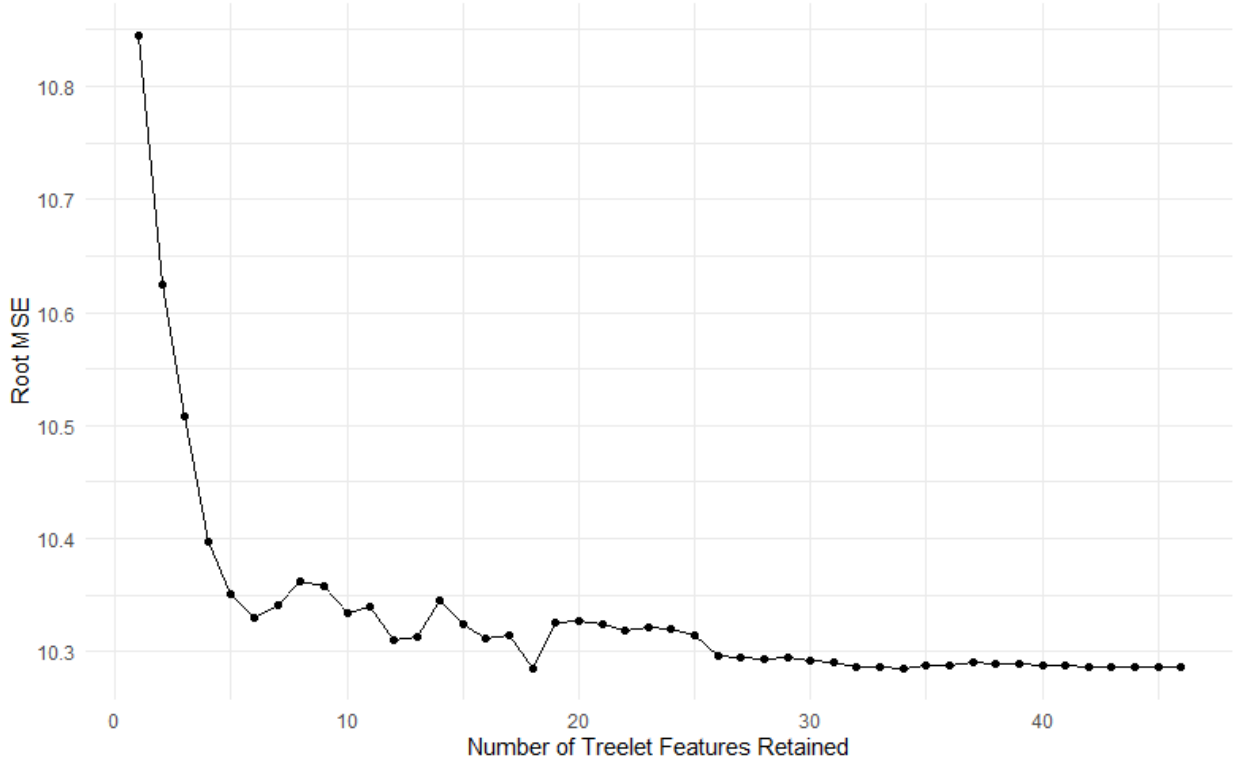


Figure 16: Root-Mean-Square Error by Number of Retained Treelet Features

3.2.4 Comparative Model Fit

In addition to the results of models including the previously described treelet features, **Table 6** contains the test model fit of lasso generalized linear models, models using PCA transformed features, models using the Charlson and Elixhauser comorbidity indices, and models including the original, 178 ICD-9-CM diagnosis codes. The models fit using treelet features are outperformed by the lasso and PCA models across all three clinical outcomes, as well as the models retaining the original 178 diagnosis codes in models of in-hospital mortality and hospital re-

admission. That is, treelet dimension reduction does *not* improve the prediction of our clinical outcomes of in-hospital mortality or hospital re-admission over the original diagnosis data. Similarly, treelet transformed features of ICD-9-CM diagnosis codes do not outperform more common dimension reduction methods of lasso or PCA in all three of our clinical outcomes. Of our dimension reduction models, only the lasso models outperform retaining the original 178 ICD-9-CM diagnosis codes in models of mortality and re-admission. The Charlson and Elixhauser comorbidity indices demonstrate little-to-no classification ability for in-hospital mortality or hospital-readmission and the highest prediction error (compared to the remaining models in **Table 6**) for each clinical outcome.

Table 6: Comparative Results of Model Performance

	Treelet (All Features)	**Treelet (Top 5 Features)**	**Lasso**	**PCA**	**Charlson**	**Elixhauser**	**All ICD Codes**
Mortality*	0.858	0.830	0.868	0.860	0.632	0.615	0.867
Readmission*	0.661	0.658	0.669	0.667	0.502	0.513	0.667
Length of Stay**	10.29	10.35	9.61	10.24	13.48	13.49	11.75

**AUC values reported; **Root-mean-square error presented*

Table 7: Summary of Retained Features and ICD-9-CM Diagnosis Codes

	Treelet (Optimal)	**Treelet (Top 5 Features)**	**Lasso**	**PCA**
In-Hospital Mortality*	123 (178)	5 (38)	170 (170)	66 (178)
Hospital Re-admission**	30 (178)	5 (178)	48 (48)	65 (178)
Hospital Length of Stay*	46 (107)	5 (29)	178 (178)	66 (178)

Number of features retained (Number of ICD-9-CM codes loading onto retained features)
**Models using more-sparse parameters; **Model using test error minimizing parameters*

Table 7 reports the number of retained features and the number of ICD-9-CM diagnosis codes that inform the features for the treelet model, including the optimal *K* features and the models including only the five most significant features (labelled "*Top 5 features*"), as well as the lasso and PCA models. Each PCA model includes loadings from all 178 ICD-9-CM diagnosis codes (as PCA retains information from all of the original input variables), accounting for 60% of the variance in the original ICD-9-CM diagnosis codes in 66 principal components in the analytic cohort of in-hospital mortality and hospital length of stay and in 65 components in the analytic cohort of hospital re-admission. The treelet model identifies the smallest number of features retained in the final model of all three clinical outcomes. The optimal treelet parameter for hospital length of stay identifies the smallest number of required ICD-9-CM diagnosis codes loading onto the final, included features. The lasso model of hospital re-admission identifies a much smaller number of required ICD-9-CM diagnosis codes compared to both the optimal treelet model and the five most significant treelet features.

4.0 Discussion

This work applied treelet dimension reduction to ICD-9-CM diagnosis codes and used the resulting, transformed features to fit models of in-hospital mortality, unplanned hospital readmission, and hospital length of stay in a cohort of critical care admissions. The resulting predictive models require only ICD-9-CM diagnosis code, patient age, sex, and insurance coverage/payment method. Treelet dimension reduction represents the original set of ICD-9-CM code covariates with a smaller number of features and ideally using only a subset of the original input covariates. This analysis built upon previous work through use of treelet dimension reduction and through use of the large, publicly available MIMIC-III database, a publicly available data source of single-center, critical care admissions.

In the analyses of mortality and hospital length of stay, the analytic cohort included 38,554 adult patients (18+ at time of admission). In this cohort, hospital mortality occurred in 14.19% of patients, and the median length of stay was 7 days, while length of stay values ranged from less than a day to 294 days (**Table 2**). Analysis of unplanned hospital re-admission included 28,894 patients, as a subset of patients died within a year of their earliest discharge with no hospital re-admission.

Among the 178 retained ICD-9-CM diagnosis codes, the most prevalent codes included expected conditions such as hypertension (42.7%), atrial fibrillation (24.4%), and congestive heart failure (22.1%) as well as diagnosis of acute kidney (15.8%) and respiratory (14.0%) failure, anemia (10.2%), and pneumonia (8.7%) (**Figure 1**). The high prevalence of the acute organ failure and anemia diagnoses may represent complications among severe trauma admissions (Alder & Tambe, 2020). The prevalence of pneumonia in this cohort align with previous estimates of

nosocomial pneumonia prevalence in American hospital admissions (Shebl & Gulick, 2020). The exploration of the correlation structure of the diagnosis codes, and specifically the examination of the most correlated pairs in **Figure 3**, showed expectedly related pairs of correlated conditions, such as *dementia without behavioral disturbance* and *Alzheimer's disease* as the most correlated pair, and subsequent pairs of correlated codes including *severe sepsis* and *septic shock* as well as *diabetes with neurological manifestations* and *neuropathy in diabetes*. These and other, similar pairs of ICD-9-CM diagnosis codes may be data elements that are seemingly redundant, which may be best represented by a single combined covariate as they are joined through the treelet model.

In the fitting of logistic regression models to predict mortality, the cross-validation of the treelet model identified only minor dimension reduction, as we identified a K parameter (which identified the number of dimensions to retain in the $L|K$ basis) of 123. The 123 features in the transformed treelet basis additionally represented information from all 178 diagnosis codes, such that the treelet model, while moderately reducing the number of covariates in our final model, did not yield a sparse feature space. Among the covariate included from the treelet transformation, feature 1 included loading from all 178 ICD-9-CM diagnosis codes, where the codes with the highest loading diagnosis commonly corresponding to diagnosis related to sepsis, organ failure, and pneumonia. Additional features included codes related to cancers/malignancies, neurological injury, and cardiovascular disease. Severe diagnoses such as malignant neoplasms (Nasir et al., 2017) and traumatic brain injuries (McCredie et al., 2018) are unsurprising risk factors of in-hospital mortality, as well as severe complications such as sepsis or organ failure (Paoli et al., 2018; Rubenfeld et al., 2005).

The resulting model demonstrated good discrimination of in-hospital mortality, evidenced both by the AUC value of 0.858 (**Table 3**) and the separation of predicted probabilities of in-hospital mortality between patients' true, observed in-hospital mortality status (**Figure 6**). The presented prediction model demonstrates improved performance over existing models including those that use similar ICD-9-CM diagnosis code data but models that include additional laboratory and physiological data elements (Awad, Bader-El-Den, et al., 2017; Falcão et al., 2019).

In logistic regression modeling of unplanned hospital re-admission, treelet dimension reduction identified a reduced dimension space, with a selected value of 30 for the K parameter (describing the number of retained dimensions). The included basis (or $L|K$ parameter) was the 177^{th} basis matrix, or the final basis matrix, which included loadings from all 178 diagnosis codes. Thus, although the selected parameters of our treelet model yielded a largely reduced number of included covariates (contrary to the selected K value of 123 in our model of mortality), the identified basis matrix similarly failed to yield a sparse feature space of our 178 ICD-9-CM diagnosis codes. The most significant treelet features observed in **Figure 9** notably included feature 1 (involving diagnoses related to organ failure infection), feature 2 (including diagnoses of sepsis and organ failure similar to those included in feature 1 in the model of mortality), and feature 4 (including diagnoses of diabetes and related complications).

In addition to elevating risk of mortality, the diagnoses related to sepsis and organ systems failure in feature 2 are also associated with highly increased risk of hospital re-admission (Goodwin & Ford, 2018). Feature 1 includes diagnoses commonly observed as risk factors for hospital re-admission in previous research, most notably renal failure and related cirrhosis (Tapper et al., 2016) and diabetes mellitus (and related conditions) (Ostling et al., 2017). Feature 1 interestingly also includes a diagnosis of urinary tract infection, a risk factor of all-cause hospital

re-admission (MacVane et al., 2015) and specifically re-admission following admission for brain and/or spinal cord injuries, which often also include respiratory and/or renal failure complications (Brito et al., 2019; K. Lee & Rincon, 2012; Middleton et al., 2004).

The resulting features did not yield a high-performing prediction model of unplanned hospital re-admission, with a low AUC value (AUC=0.661, **Table 4**) and a poor separation of predicted probabilities (**Figure 10**). The poor performance of our model corroborates previous research, which has identified that comorbidity diagnoses and ICD-9-CM diagnosis codes remain limited in their ability to predict hospital re-admission while demonstrating high performance in prediction of mortality (Awad, Bader–El–Den, et al., 2017). This shortcoming identifies the need in prediction of hospital re-admission to not only use information beyond acute care diagnoses but likely the need to use additional information related to a patient's environment post-discharge. These data elements may include information related to discharge location, social determinants of health (such as social support, nutrition, access to transportation, etc.), assessments of function at time of discharge, or levels of independence (such as ability to complete activities of daily living) (Depalma et al., 2013; Greysen et al., 2015).

Treelet dimensions for negative binomial regression modeling of the last clinical outcome, hospital length of stay, identified a reduced number of covariates to include (K=46) and included loadings from only 107 of the 178 total ICD-9-CM diagnosis codes. Thus, the treelet dimension reduction identified a reduced number of covariates within a sparse feature space, requiring only a subset of the originally included diagnosis codes. Notable treelet features include a group of diagnoses related to sepsis and renal or respiratory failure. This first (and most significant) treelet predictor contains similar diagnoses as important features identified in both the mortality and hospital re-admission models. Sepsis and systemic organ failure are unsurprisingly related to

prolonged hospital stays (Paoli et al., 2018). Interestingly, two treelet features included only two ICD-9-CM codes, with the first including only 997.4 (*digestive complications, not otherwise specified*) and 561.0 (*paralytic ileus*) and the second only 518.0 (*pulmonary collapse* and 511.9 (*pleural effusion*). Bowel obstructions and paralytic ileus are common post-surgery complications that result in prolonged length of hospital stay (Luckey et al., 2003). While few models exist to compare the performance of the model presented in this work, the prediction of hospital length of stay appears to demonstrate only limited utility, with a large root-mean-square error of over 10 days (**Table 5**). The visualization of predicted and observed length of stay durations in **Figure 14** demonstrate that our model is affected by outliers of both large over- and under-prediction of length of stay. Future models assessing patient length of stay may expand upon existing regression modeling by assessing patient length of stay and discharge as dynamic processes (Awad, Bader–El–Den, et al., 2017). While dynamic modeling would require sequentially updated information from a patient's acute stay, models including this additional information may improve prediction by utilizing information related to adverse events and/or complications during acute hospitalization course that causally affect length of stay duration.

The final section of this work compares the results of models fit using ICD-9-CM diagnosis code data transformed using treelet dimension reduction, PCA, lasso, and the use of ICD-9-CM diagnosis codes in the Charlson and Elixhauser indices and simply using indicator variables for the retained 178 diagnosis codes. Interestingly, the Charlson and Elixhauser indices result in the worst model performance among the presented models. These results highlight the improved performance of transformed diagnosis code data in the prediction of clinical outcomes over *a priori* indices such as the Charlson and Elixhauser.

Interestingly, neither treelet transformation nor PCA dimension reduction of the original, 178 ICD-9-CM diagnosis codes improved classification performance of hospital mortality or hospital re-admission compared to models including all 178 diagnosis code variables. Lasso regression models improved only modestly upon the results of models of mortality or re-admission including the original ICD-9-CM diagnosis code variables. In modelling hospital length of stay, models including treelet features, PCA components, and lasso regression models all demonstrate improved prediction over models including the original ICD-9-CM code data, with only the model including treelet features identifying a sparse number of ICD-9-CM diagnosis codes in both the full model (46 treelet features including 107 ICD-9-CM codes) and using only the subset of treelet features (5 treelet features including 29 ICD-9-CM codes). Thus, while outperformed by the lasso negative binomial model and modestly by the model including PCA transformed features, the treelet dimension reduction identified a much smaller number of retained features and required ICD-9-CM diagnosis codes with only a modest reduction in model fit.

Future studies may also explore the comparative performance of indices that use acute physiological or lab measurements, such as the APACHE-II model, to compare performance of transformed physiological data over the original data elements and to compare the added predictive utility of these data elements over models including only diagnosis codes. The models of mortality in this work out-perform previous models presented using physiological data, but we may expect the use of these more granular data elements in the MIMIC-III data set (or similar large EHR databases) to further improve the performance of prediction models over those including only demographic and diagnosis data.

4.1.1 Limitations

Date or time of diagnosis is unavailable in the MIMIC-III data. As a result, such that patients may receive their diagnoses at time of admission or at any point during their acute stay. Thus, we cannot determine whether the presented models rely solely upon *baseline* ICD-9-CM diagnosis codes (i.e. diagnoses present at time of admission). The use of baseline diagnosis code data may improve the generalizability and ease-of-implementation of diagnosis code prediction models at the possible cost of prediction performance. The presented model is generated using an adult population for an all-cause admission ICU. Specialized or sub-population units (e.g. pediatric ICU, neuro ICU, cardiac ICU, etc.) likely require their own predictive models. The inclusion of a diverse patient population may lead to reduced prediction performance in the presented analysis, that may be improved by examining specific sub-populations and relevant data elements or diagnoses. Lastly, although the cross-validation method sought to combat overfitting of the presented models, these results have not been assessed for external predictive performance among new patient populations or data from separate hospitals or healthcare systems.

5.0 Conclusion

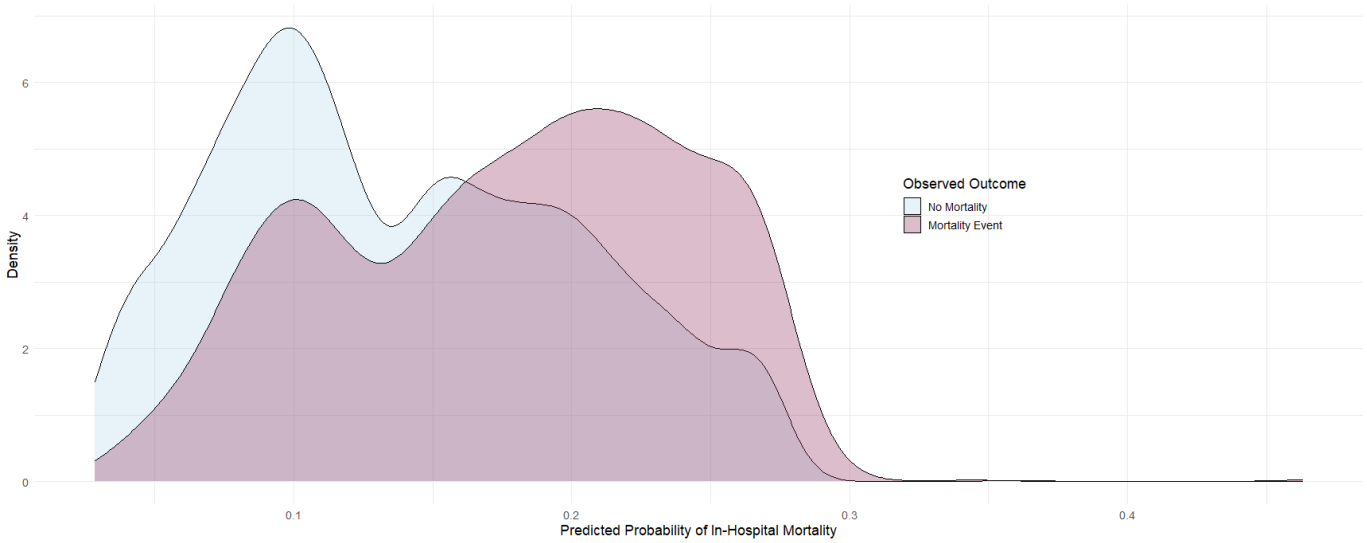
The presented work applies treelet, a novel dimension-reduction model, to ICD-9-CM diagnosis codes. The resulting transformation of diagnosis code with patient demographic variables were used to fit logistic regression models of in-hospital mortality, unplanned hospital re-admission, and hospital length of stay. The proposed objectives aimed to build prediction models requiring minimal information (patient demographic and diagnosis information) and to identify a reduced dimensionality and possibly a sparse set of ICD-9-CM diagnosis codes to consider in predicting patient outcomes. The presented work used data from the Medical Information Mart for Intensive Care (MIMIC-III), a publicly available database of critical care admissions which has been previously outlined as an important yet underutilized critical care admissions data source.

While treelet dimension reduction did not identify a sparse number of codes for in-hospital mortality prediction, the model demonstrated improved model fit performance when compared to previous models using similar data elements (i.e. patient demographic information and ICD-9-CM diagnosis codes) as well as improvement over models including patient physiological and lab measurements over acute hospital stays. Treelet dimension reduction failed to yield a sparse set of ICD-9-CM codes to consider in prediction of hospital re-admission, where logistic regression models failed to adequately predict patients' readmission statuses, aligning with previous research identifying the limitations of diagnosis code prediction of hospital re-admission. Lastly, treelet dimension reduction identified a sparse number of ICD-9-CM diagnosis codes, retaining only 102 of 178 included codes, using a reduced number of covariates in negative binomial regression modeling of hospital length of stay. Evaluation of the negative binomial model of hospital length

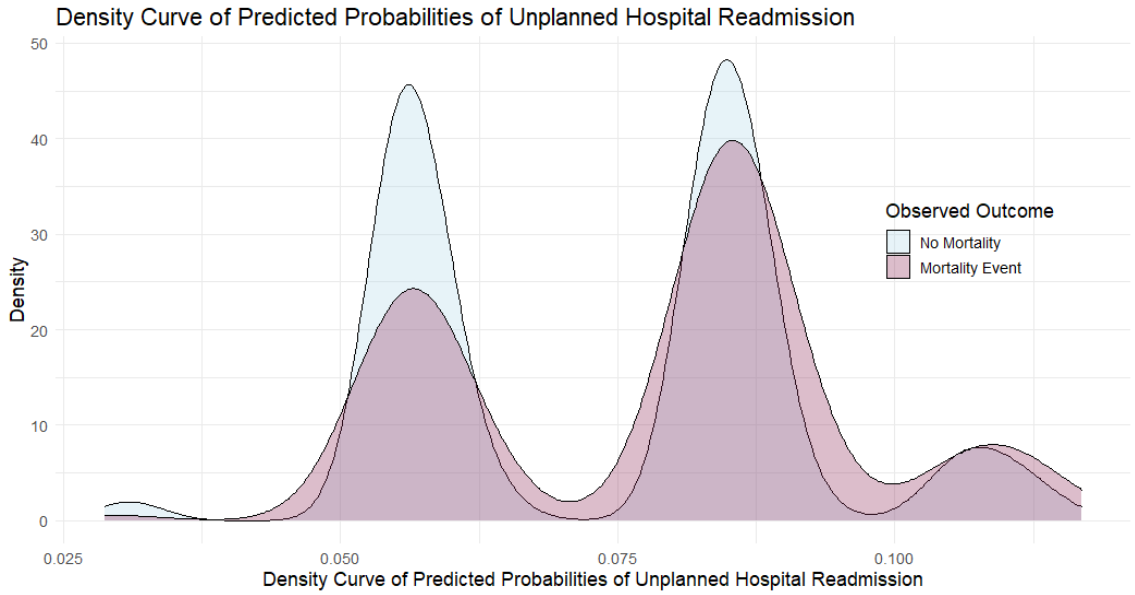
of stay in a final, test data set again demonstrated only limited prediction utility. The retained treelet features in the three included regression models align with previously identified risk factors of mortality, re-admission, and hospital length of stay respectively.

The results of these analyses demonstrate the useful but limited performance of ICD-9-CM diagnosis codes as the primary data element considered in prediction of clinical outcomes. While patient demographic data and diagnosis codes may result in accurate prediction of mortality, additional information is likely required for improved modeling of hospital length of stay and unplanned re-admission. Hospital length of stay modeling may benefit from the use of patient acute care information as well as disease-specific modeling within subset of patients. Hospital re-admission may benefit from using not only acute care but post-discharge data, including elements such as those related to patients' discharge environment and functionality at discharge.

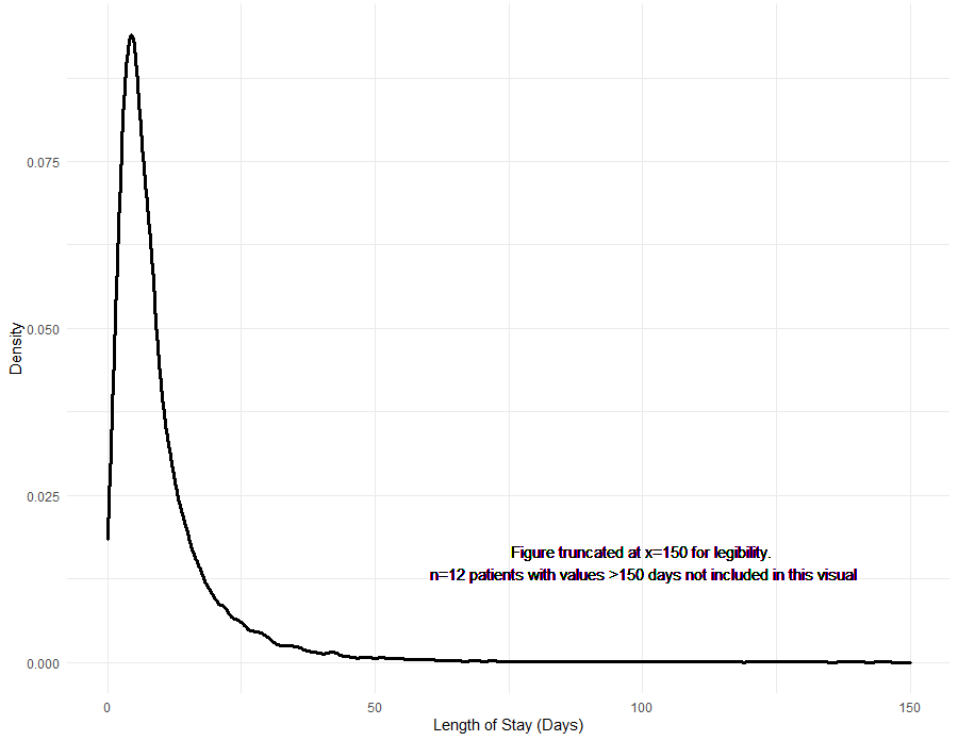
Appendix A Supplemental Tables & Figures



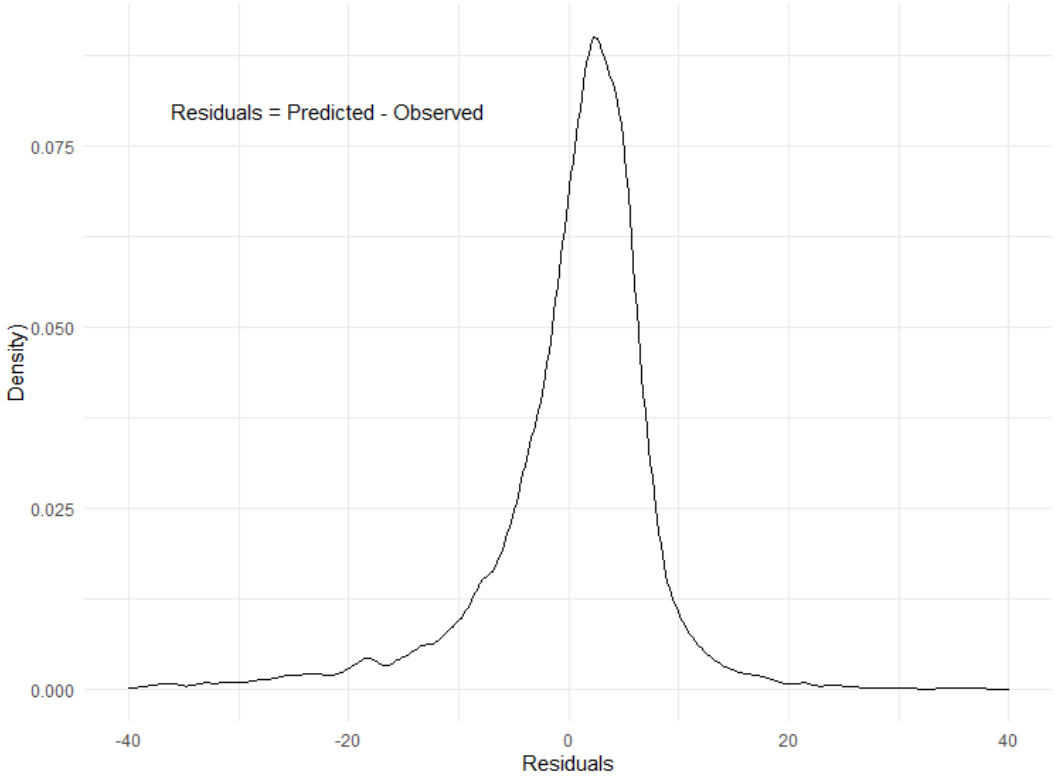
Appendix Figure 1: Density Curve of Mortality Model Predicted Probabilities (Treelet Features Omitted)



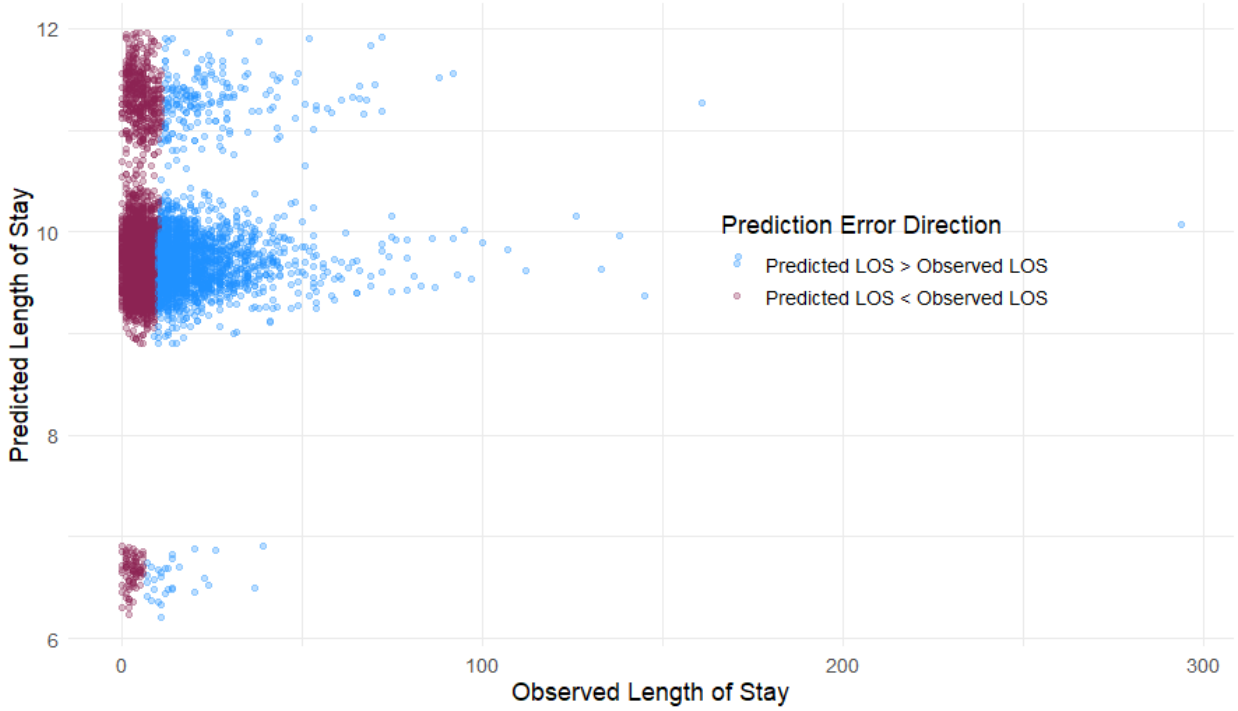
Appendix Figure 2: Density Curve of Readmission Model Predicted Probabilities (Treelet Features Omitted)



Appendix Figure 3: Density Curve of Hospital Length of Stay Predictions



Appendix Figure 4: Density Curve of Residuals in Prediction of Length of Stay



Appendix Figure 5: Scatter Plot of Length of Stay Model Predicted Values (Treelet Features Omitted)

Appendix Table 1: Full Regression Estimates (Mortality)

	β	95% Conf. Interval	P-Value
Intercept	-5.02	[-5.37, -4.67]	<0.001
Sex (Male)	-0.12	[-0.20, -0.04]	0.0043
Age	0.04	[0.03, 0.04]	<0.001
Insurance			
Medicaid	0.18	[-0.14, 0.50]	0.2728
Medicare	0.33	[0.03, 0.63]	0.0317
Private	0.10	[-0.19, 0.40]	0.4911
Self-Pay	1.17	[0.76, 1.59]	<0.001
Treelet Cluster			
1	1.56	[1.49, 1.64]	<0.001
2	1.64	[1.47, 1.81]	<0.001
3	0.19	[0.00, 0.39]	0.0518
4	-0.38	[-0.50, -0.26]	<0.001
5	1.16	[0.99, 1.33]	<0.001
6	-0.15	[-0.36, 0.06]	0.1697
7	-0.60	[-0.68, -0.52]	<0.001
8	0.10	[0.02, 0.19]	0.0191
9	0.09	[-0.07, 0.24]	0.2884
10	-0.50	[-0.84, -0.17]	0.0031
11	0.55	[0.31, 0.79]	<0.001
12	0.05	[-0.18, 0.28]	0.67
13	1.76	[1.61, 1.91]	<0.001
14	-0.03	[-0.19, 0.12]	0.6873
15	1.99	[1.82, 2.15]	<0.001
16	0.41	[0.17, 0.64]	<0.001
17	-0.57	[-0.82, -0.33]	<0.001
18	0.19	[-0.03, 0.41]	0.0936
19	-0.46	[-0.61, -0.32]	<0.001
20	-0.31	[-0.42, -0.20]	<0.001
21	-0.09	[-0.30, 0.12]	0.3833
22	0.83	[0.55, 1.10]	<0.001
23	-0.31	[-0.64, 0.01]	0.0608
24	-0.17	[-0.4, 0.06]	0.146
25	0.02	[-0.22, 0.27]	0.847
26	-0.21	[-0.56, 0.14]	0.2465
27	-0.42	[-0.55, -0.29]	<0.001
28	0.43	[0.26, 0.60]	<0.001
29	0.03	[-0.18, 0.23]	0.8038
30	-0.21	[-0.43, 0.01]	0.0561

31	-0.20	[-0.66, 0.26]	0.3858
32	0.00	[-0.19, 0.18]	0.96
33	0.22	[0.02, 0.42]	0.0353
34	-0.23	[-0.43, -0.03]	0.0231
35	-0.06	[-0.26, 0.14]	0.568
36	0.58	[0.29, 0.87]	<0.001
37	0.30	[0.11, 0.48]	0.0017
38	1.22	[1.03, 1.42]	<0.001
39	-0.21	[-0.46, 0.03]	0.0878
40	-0.48	[-0.68, -0.28]	<0.001
41	-0.22	[-0.58, 0.15]	0.2457
42	-0.15	[-0.28, -0.02]	0.0255
43	0.05	[-0.28, 0.39]	0.7559
44	0.21	[0.08, 0.33]	0.0017
45	-0.02	[-0.34, 0.30]	0.8925
46	0.28	[-0.03, 0.58]	0.0752
47	-0.37	[-0.61, -0.13]	0.0027
48	-0.72	[-1.13, -0.31]	<0.001
49	-0.29	[-0.56, -0.03]	0.0264
50	-0.31	[-0.63, 0.02]	0.0682
51	-0.06	[-0.19, 0.07]	0.4001
52	-0.33	[-0.51, -0.14]	<0.001
53	-0.31	[-0.50, -0.12]	0.0012
54	0.13	[-0.10, 0.35]	0.2803
55	0.18	[-0.08, 0.43]	0.1808
56	0.20	[-0.03, 0.44]	0.083
57	0.81	[0.65, 0.97]	<0.001
58	0.08	[-0.20, 0.36]	0.5807
59	-0.32	[-0.58, -0.05]	0.0189
60	-0.65	[-0.95, -0.35]	<0.001
61	-0.68	[-1.1, -0.27]	0.0013
62	0.11	[-0.14, 0.36]	0.3946
63	-0.81	[-1.13, -0.49]	<0.001
64	-0.19	[-0.36, -0.01]	0.0352
65	-0.45	[-0.82, -0.09]	0.0154
66	0.22	[0.08, 0.37]	0.0022
67	0.36	[0.18, 0.55]	<0.001
68	-0.36	[-0.64, -0.09]	0.0088
69	-0.79	[-1.19, -0.39]	<0.001
70	0.19	[-0.04, 0.42]	0.1056
71	0.04	[-0.28, 0.35]	0.8149
72	0.03	[-0.28, 0.34]	0.8442
73	0.25	[-0.05, 0.55]	0.0967

74	-0.04	[-0.36, 0.29]	0.8319
75	-0.23	[-0.48, 0.01]	0.0574
76	-0.62	[-1.02, -0.22]	0.0022
77	-0.25	[-0.59, 0.10]	0.1611
78	0.03	[-0.43, 0.48]	0.9141
79	-0.11	[-0.36, 0.14]	0.3836
80	-0.42	[-0.70, -0.14]	0.0029
81	-0.06	[-0.30, 0.17]	0.6022
82	0.10	[-0.13, 0.34]	0.4007
83	-0.26	[-0.59, 0.07]	0.1269
84	-0.58	[-0.87, -0.29]	<0.001
85	-0.13	[-0.45, 0.19]	0.4162
86	0.92	[0.69, 1.14]	<0.001
87	0.08	[-0.26, 0.42]	0.6456
88	-0.41	[-0.7, -0.12]	0.0055
89	0.19	[-0.12, 0.49]	0.2321
90	-0.13	[-0.43, 0.17]	0.391
91	-0.17	[-0.47, 0.13]	0.2569
92	0.40	[0.16, 0.65]	0.0014
93	-0.18	[-0.52, 0.16]	0.3024
94	-0.26	[-0.58, 0.06]	0.1117
95	0.02	[-0.25, 0.30]	0.8578
96	0.04	[-0.35, 0.43]	0.832
97	-0.21	[-0.56, 0.14]	0.2325
98	-0.66	[-0.95, -0.38]	<0.001
99	0.70	[0.49, 0.91]	<0.001
100	-0.44	[-0.77, -0.12]	0.0077
101	0.01	[-0.41, 0.42]	0.9725
102	-0.21	[-0.56, 0.14]	0.2424
103	0.47	[0.27, 0.67]	<0.001
104	-0.52	[-0.80, -0.25]	<0.001
105	-0.11	[-0.35, 0.13]	0.3828
106	-0.46	[-0.77, -0.16]	0.0031
107	0.22	[0.02, 0.42]	0.0325
108	-0.63	[-0.87, -0.39]	<0.001
109	-0.14	[-0.41, 0.12]	0.2841
110	0.03	[-0.35, 0.41]	0.8727
111	-0.28	[-0.64, 0.08]	0.1245
112	0.89	[0.64, 1.15]	<0.001
113	0.25	[-0.04, 0.54]	0.0916
114	-0.33	[-0.67, 0.01]	0.0567
115	-0.09	[-0.37, 0.19]	0.5218
116	-0.52	[-0.88, -0.15]	0.0055

117	-0.59	[-0.99, -0.19]	0.0042
118	-0.02	[-0.34, 0.30]	0.9107
119	0.34	[0.12, 0.56]	0.0027
120	-0.72	[-1.10, -0.35]	<0.001
121	-0.45	[-0.80, -0.10]	0.0111
122	0.09	[-0.20, 0.38]	0.545
123	0.00	[-0.30, 0.29]	0.9763

Appendix Table 2: Full Regression Estimates (Readmission)

	β	95% Conf. Interval	P-Value
Intercept	-3.14	[-3.49, -2.78]	<0.001
Sex (Male)	-0.04	[-0.14, 0.06]	0.4548
Age	0.00	[0.00, 0.01]	0.2812
Insurance			
Medicaid	0.48	[0.16, 0.81]	0.0032
Medicare	0.31	[0.00, 0.63]	0.0535
Private	-0.03	[-0.34, 0.27]	0.8331
Self-Pay	-0.61	[-1.28, 0.06]	0.0749
Treelet Feature			
1	0.71	[0.60, 0.82]	<0.001
2	1.10	[0.88, 1.32]	<0.001
3	-0.23	[-0.33, -0.12]	<0.001
4	0.80	[0.62, 0.98]	<0.001
5	-0.29	[-0.74, 0.17]	0.2207
6	0.19	[-0.33, 0.71]	0.4757
7	0.01	[-0.16, 0.17]	0.9227
8	-0.25	[-0.42, -0.08]	0.0048
9	-0.10	[-0.40, 0.20]	0.522
10	0.50	[0.30, 0.70]	<0.001
11	0.15	[-0.16, 0.46]	0.3559
12	0.13	[-0.23, 0.49]	0.4765
13	0.22	[-0.18, 0.62]	0.28
14	0.09	[-0.22, 0.41]	0.5659
15	-0.07	[-0.29, 0.15]	0.5273
16	-0.02	[-0.42, 0.39]	0.9268
17	0.20	[-0.02, 0.43]	0.0733
18	0.01	[-0.25, 0.27]	0.9585
19	0.41	[0.16, 0.66]	0.0015
20	0.12	[-0.28, 0.53]	0.5503
21	0.13	[-0.28, 0.53]	0.5381
22	0.52	[0.27, 0.76]	<0.001
23	-0.08	[-0.25, 0.09]	0.3584
24	0.00	[-0.19, 0.19]	0.9763
25	0.07	[-0.26, 0.39]	0.6824
26	0.43	[0.09, 0.78]	0.0132
27	0.37	[0.13, 0.61]	0.0025
28	0.55	[0.19, 0.92]	0.0032
29	0.02	[-0.21, 0.26]	0.846
30	0.28	[-0.01, 0.58]	0.0594

Appendix Table 3: Full Regression Estimates (Length of Stay)

	β	95% Conf. Interval	P-Value
Intercept	2.00	[1.94, 2.06]	<0.001
Sex (Male)	0.05	[0.04, 0.07]	<0.001
Age	0.00	[0.00, 0.00]	<0.001
Insurance			
Medicaid	0.11	[0.06, 0.17]	<0.001
Medicare	0.05	[-0.01, 0.1]	0.079
Private	0.04	[-0.01, 0.09]	0.134
Self	-0.32	[-0.41, -0.23]	<0.001
Treelet Feature			
1	0.37	[0.35, 0.39]	<0.001
3	-0.16	[-0.22, -0.10]	<0.001
4	-0.15	[-0.18, -0.12]	<0.001
5	0.15	[0.10, 0.19]	<0.001
7	-0.09	[-0.10, -0.07]	<0.001
8	0.21	[0.19, 0.23]	<0.001
9	0.09	[0.05, 0.13]	<0.001
11	0.23	[0.16, 0.29]	<0.001
12	0.61	[0.56, 0.66]	<0.001
13	0.23	[0.18, 0.27]	<0.001
14	0.42	[0.38, 0.45]	<0.001
15	0.7	[0.65, 0.75]	<0.001
17	0.4	[0.34, 0.46]	<0.001
19	0.17	[0.14, 0.20]	<0.001
20	0.3	[0.28, 0.33]	<0.001
22	0.52	[0.47, 0.57]	<0.001
23	0.2	[0.12, 0.28]	<0.001
24	0.29	[0.22, 0.36]	<0.001
25	0.22	[0.16, 0.28]	<0.001
26	0.26	[0.18, 0.34]	<0.001
32	0.17	[0.13, 0.22]	<0.001
33	-0.17	[-0.23, -0.11]	<0.001
37	0.11	[0.07, 0.16]	<0.001
39	-0.16	[-0.21, -0.10]	<0.001
40	-0.25	[-0.33, -0.18]	<0.001
41	-0.07	[-0.10, -0.04]	<0.001
44	0.61	[0.53, 0.68]	<0.001
46	-0.1	[-0.14, -0.06]	<0.001

Appendix Table 4: Abbreviated Treelet Features (Mortality)

Treelet Feature	ICD-9-CM Code	Loading	Code Description
Feature 1*	584.9	0.555	Acute kidney failure NOS
	518.81	0.507	Acute respiratory failure
	995.92	0.347	Severe sepsis
	389	0.309	Septicemia NOS
	785.52	0.272	Septic shock
Feature 2	198.3	0.418	Secondary malignant neoplasm (brain/spine)
	197.7	0.539	Secondary malignant neoplasm (liver)
	197.0	0.475	Secondary malignant neoplasm (lung)
	198.5	0.556	Secondary malignant neoplasm (bone)
Feature 7*	401.9	0.736	Hypertension NOS
	414.01	0.524	Coronary atherosclerosis of native coronary artery
	250.00	0.255	DMII without complications
	272.4	0.238	Hyperlipidemia NEC/NOS
	272.0	0.187	Pure hypercholesterolemia
Feature 13	431	0.815	Intracerebral hemorrhage
	348.5	0.462	Cerebral edema
	331.4	0.144	Obstructive hydrocephalus
	430	0.213	Subarachnoid hemorrhage
	348.4	0.236	Compression of brain
Feature 15	427.5	0.946	Cardiac arrest
	427.41	0.324	Ventricular fibrillation

**Features 1 and 7 contain loadings from 19 and 10 ICD-9-CM codes respectively, only diagnoses with 5 highest loadings presented*

Appendix Table 5: Abbreviated Treelet Features (Readmission)

Treelet Feature	ICD-9-CM Code	Loading	Code Description
Feature 1	584.9	0.604	Acute kidney failure NOS
	518.81	0.418	Acute respiratry failure
	599.0	0.252	Urin tract infection NOS
	403.90	0.216	Hy kid NOS w cr kid I-IV
	585.9	0.215	Chronic kidney dis NOS
	285.9	0.204	Anemia NOS
	995.92	0.197	Severe sepsis
	389.	0.170	Septicemia NOS
Feature 2	571.5	0.405	Cirrhosis of liver NOS
	705.4	0.491	Chrnc hpt C wo hpat coma
	571.2	0.484	Alcohol cirrhosis liver
	572.3	0.413	Portal hypertension
	789.59	0.251	Ascites NEC
Feature 4	357.2	0.573	Neuropathy in diabetes
	403.91	0.494	Hyp kid NOS w cr kid V
	250.60	0.469	DMII neuro nt st uncntrl
	585.6	0.315	End stage renal disease
	362.01	0.218	Diabetic retinopathy NOS
Feature 22	427.1	0.889	Parox ventric tachycard
	425.4	0.406	Prim cardiomyopathy NEC
	410.11	0.116	AMI anterior wall, init
	427.5	0.115	Cardiac arrest
	785.51	0.097	Cardiogenic shock

**All features contain loadings from additional ICD-9-CM codes, only diagnoses with 5 highest loadings presented*

Appendix Table 6: Abbreviated Treelet Features (Length of Stay)

Treelet Feature	ICD-9-CM Code	Loading	Code Description
Feature 1*	584.9	0.555023	Acute kidney failure NOS
	518.81	0.506634	Acute respiratory failure
	995.92	0.346761	Severe sepsis
	389	0.309371	Septicemia NOS
	785.52	0.271532	Septic shock
Feature 2	198.3	0.417658	Sec mal neo brain/spine
	197.7	0.539451	Second malignant neoplasm (liver)
	197	0.47509	Second malignant neoplasm (lung)
	198.5	0.555737	Second malignant neoplasm (bone)
Feature 12	997.4	0.546458	Digestive complications NOS
	560.1	0.837486	Paralytic ileus
Feature 14	518	0.507345	Pulmonary collapse
	511.9	0.861743	Pleural effusion NOS

**Features 1 contains loadings from 19 ICD-9-CM codes respectively, only diagnoses with 5 highest loadings presented*

Appendix B Analytic Code

Raw data publicly accessible (by request) at https://mimic.physionet.org/ (*N.B. Data are collected and stewarded by the Massachusetts Institute of Technology Lab for Computational Physiology, not the author or advisors of this document or any group at the University of Pittsburgh)*

Analytic code is included below as raw, RMarkdown code. Downloadable RMarkdown files (in addition to test data predictions, CSV files cross-validation performance, and figures) additionally available at https://github.com/domdisanto/ICD_Diagnoses_Treelet

Appendix B.1 R Code to Perform Data Cleaning and Exploratory Data Analysis (incl. Descriptive Statistics)

````
---
title: "Treelet Transform: Identifing clusters of ICD-9 Diagnoses in a Boston
Trauma Center"
subtitle: "Data Cleaning, Exploratory Data Analysis & Visualization"
author: "Dominic DiSanto\n Master's Thesis"
date: "Updated 9/20/2020"
output:
  html_document:
    keep_md: true
    toc: true
    toc_depth: '3'
    code_folding: show
---

## Preparation


## Libraries
```{r, message=F, warning=FALSE}
````
````

```
library(magrittr) # Ceci n'est pas une %>%, loaded via dplyr also but liked to include for transparency
library(dplyr) # General data management, cleaning (admittedly I switch between Base R and tidyverse as I code, somewhat stream-of-consciousness ly)
library(ggplot2) # Visualization
library(tidyr) # pivot functions for transposing data to/from long and wide
library(icd) # used in validity check of diagnoses codes
library(lubridate) # used in evaluating dates, most notably in date of death
library(lares) # corr_cross function used to identify the top correlations within a data frame/design matrix
library(corrplot) # used for visualizing correlation matrices
library(here) # Used for data-calls/ease of file path storage usage
```

File Path

This is my first attempt at using the `here` package for improved functionality of this program. I believe to use the `here` package as written in my program, your data simply need to be contained in a sub-folder called **Data** from where you've saved this file. For transparency, I'll describe my general (and I think simplistic) file structure for this analysis: Within a general project folder (say `Treelet`), this script and it's output are contained in an ***"Analysis"*** subfolder and the data within a ***Data*** subfolder of the same project folder. For the raw input data from MIMIC, I included a **Raw** sub-folder within the **Data** folder (to isolate raw MIMIC data from any exported data files or cleaned data).

Because I contain my analysis in a sub-folder of my main project file, I had to therefore manually set my `.here` file one level above my analytic file. If you happen to mirror my file structure, you must simply use the command `set_here("../")`, which will create a `.here` file in your root folder, a level above the analytic subfolder.

```{r}
here()
```

Data Cleaning

I will broadly classify the data cleaning in two areas: **Patient Data** and **Diagnoses Data**. **Patient data cleaning** will include wrangling of patient-level demographic and admissions data, identifying patients with multiple admissions and specifying which admission of interest to use in analysis, and other individual/person cleaning. **Diagnosis data cleaning** will involve identifying and cleaning the ICD-9 diagnoses code data to be included in the treelet transform dimension reduction technique.

These steps are not entirely separate, as the included diagnoses codes will only involve patients in our analytic cohort from the **patient data cleaning**, but this separation is useful and somewhat natural due to the varied input data and steps required in each process.

Cleaning Patient Data

Before meaningfully working with the any data or performing analyses, we must identify our patient cohort to be used in analysis. The first step will be identifying an analytic patient cohort. This will include:
- Identify the admission of interest among patients with multiple stays
    - This will be the earliest admission, which we will synonymously reference as earliest admission or first encounter
- Removing pediatric patients (those under 18 at time of admission)
- Examining and cleaning variables/covariates to an "analytic format", the exact definition which will be data element dependent but will prepare elements for proper analysis, exploration, and visualization

Cohort Identification

As mentioned above, we must identify our analytic cohort by:
- Identify the admission of interest among patients with multiple stays
    - This will be the earliest admission, which we will synonymously reference as earliest admission or first encounter
- Removing pediatric patients (those under 18 at time of admission)

To accomplish both of our goals above, we must first identify the admissions of interest for each patient. As mentioned preivously, we will use the first patient encounter in our data set to identify diagnoses to include in our dimension reduction and information/data to include in our analyses.

```{r}
admit <- read.csv(here("Data", "Raw", "ADMISSIONS.csv"))

# # Number of patients with multiple visits
cat("There are", admit %>% group_by(SUBJECT_ID) %>% count() %>% filter(n>1) %>% nrow(), "patients in our data set with multiple admissions")
cat("These individuals with multiple admissions account for", admit %>% group_by(SUBJECT_ID) %>% count() %>% filter(n>1) %>% ungroup() %>%  select(n) %>% sum(), "visits, including their index dates/first admissions.")
  # # Of the 58,976 visits, 19,993 are duplicate visits (including first encounter) among 7,537 patients
  # # therefore of the 58,976 visits, 12,456 are removed resulting in 46,520 unique patient first-encounters

admit_unq <- admit %>%
  group_by(SUBJECT_ID) %>%
  filter(ADMITTIME==min(ADMITTIME)) %>%
  ungroup() %>%
  select(SUBJECT_ID, HADM_ID, ADMITTIME, DISCHTIME, ADMISSION_TYPE, INSURANCE)

if(admit_unq %>% nrow() !=  admit_unq %>% distinct(SUBJECT_ID) %>% nrow()) {
  print("Problem with admit data, the number of rows and patients in this data frame should be equal but are not")
  break
}
```

We can now merge in our patient data to each admission of interest to calculate age and limit our population

```{r}
pts_raw <- read.csv(here("Data", "Raw", "PATIENTS.csv"))

pts_red <- pts_raw %>% select(SUBJECT_ID, DOB, DOD, GENDER)

admit_pts <- merge(pts_red, admit_unq, by="SUBJECT_ID", all=T) %>%
  mutate(Age=
           (difftime(ADMITTIME, DOB, unit="weeks") %>%
              as.integer()/52) %>%
           floor()) %>%
  select(SUBJECT_ID, Age, everything(), -DOB) %>% filter(Age>=18)


admit_pts %>% distinct(ADMISSION_TYPE)
  # Confirming there are no "NEWBORN" admission types

admit_pts <- admit_pts %>% select(-ADMISSION_TYPE)
```

We have now identified our cohort of interest of adults (patients 18 or older at first admission) and have identified our first-encounters/earliest visits of interest. There is however some additional cleaning necessary for our variables of interest to include in EDA and analysis later.

Covariate Cleaning

Cleaning of patient-level characteristics are carried out and described below. This section will not include analysis or visualization, which are saved for the EDA section of this program.

Age

In examining the data and the MIMIC-III metadata/documentation, I noticed that `Age` values occur of 301, where patients who were older tha 89 at time of admission have their (randomized) date-of-birth's set to 300+ years prior to their hospital admittance. As dates are randomly jittered I am unable to impute these values using admit or discharge times. As a result, I will set these values to simply 1 year higher than the maximum age (that is less than 300).

```{r}
admit_pts %>% count(Age) %>% arrange(desc(Age)) %>% head()

admit_pts <- admit_pts %>%
  mutate(Age =
    case_when(Age>100 ~ 90,
              TRUE ~ Age)
  )

admit_pts %>% count(Age) %>% arrange(desc(Age)) %>% head()
```

Mortality

The MIMIC-III data offers two sources for mortality status (and related date of death):
  `DOD_HOSP` - In-hospital mortality collected and stored in the hospital's local database
  `DOD_SSN` - Date of death as obtained from the social security death index (SSDI), which includes deaths up to 4-years post-discharge

Both of these variables are aggregated into a generic `DOD` variable of date of death, which prioritizes `DOD_HOSP` if both sources have a recorded date of death. In presenting this data to the Capstone committee, we had decided to use "in-hospital mortality", and I planned to simply use the `DOD_HOSP` variable. However I noticed that among patients with multiple visits, `DOD_HOSP` would capture in-hospital mortality at a later visit (and not the visit of interest which we've discussed and isolated). Therefore I will use the generic `DOD` variable, and identify in-hospital mortality as present for any patient with a DOD date equal to or less than their discharge date. Otherwise, in-hospital mortality will be set as surviving the patient's stay.

One detail I will include is that I will *not* consider time differences when assessing this difference. I will simply see if the date of death `DOD` and time of discharge `DISCHTIME` are the same year-month-date or if `DOD` is less than `DISCHTIME`. Lastly, there are patients whose `DOD` is immediately greater than their `DISCHTIME`. As a buffer, I will consider in-patient mortality as present or patients as expiring during their stay if `DOD` is within 24 hours of `DISCHTIME`.

```{r}
admit_pts %>%
  filter(DOD!="" & as.Date(ymd_hms(DOD))!=as.Date(ymd_hms(DISCHTIME))) %>%
  select(SUBJECT_ID, Age, DOD, DISCHTIME) %>% sample_n(5)
  # Examinign random patients with disparate `DOD` and `DISCHTIME` values


admit_pts %>%
  filter(DOD!="" & as.Date(ymd_hms(DOD))>as.Date(ymd_hms(DISCHTIME)))  %>%
  mutate(dodlag = as.integer(difftime(DOD, DISCHTIME, unit="hours"))) %>%
  arrange(dodlag) %>% head()
  # Examining some of the differences in time that are small between DOD and DISCHTIME


admit_pts <- admit_pts %>%
  mutate(InHospMortality =
           case_when(
             DOD!="" & as.Date(ymd_hms(DOD)) <= as.Date(ymd_hms(DISCHTIME)) ~ 1,
             DOD!="" & as.integer(as.Date(ymd_hms(DOD)) - as.Date(ymd_hms(DISCHTIME)))<=24 ~ 1,
             TRUE ~ 0
           ))

```

Payment/Insurance

```{r}
admit_pts %>% count(INSURANCE)
```

The `Self Pay` category is (comparatively) somewhat small, but as of now I don't think there is any need to collapse these groups considerign even this small proportion is nearly 550 observations.

General Hospital Length of Stay

```{r}
admit_pts <- admit_pts %>%
  mutate(HospitalLOS =
           floor(as.numeric(difftime(DISCHTIME, ADMITTIME, unit="days"))))

```

Hospital Re-admission

For re-admission, I will explore whether to use 30-day or 90-day readmission. I found literature using both as "short-term" and "early-" hospital readmission. I will also specifically look at emergency/urgent readmission (not elective).

```{r}
readmit_dts <- admit %>% filter(ADMISSION_TYPE %in% c("EMERGENCY", "URGENT")
& # filtering out elective admissions
                    admit$SUBJECT_ID %in% c(admit_pts %>%
select(SUBJECT_ID))[[1]]) %>%  # identifying only patients in our analytic
cohort
  group_by(SUBJECT_ID) %>% filter(ADMITTIME!=min(ADMITTIME)) %>% # removing
our index visits
  filter(ADMITTIME==min(ADMITTIME)) %>% ungroup() %>%  # now saying give me
the admittime closest to your index admittance
  select(SUBJECT_ID, ReadmitDate=ADMITTIME) # finally, simply selecting the
SUBJECT_ID and readmitdate

admit_pts <- merge(admit_pts,readmit_dts, by="SUBJECT_ID", all.x=T)  %>%
  mutate(TimeToReadmit =
           case_when(
             !is.na(ReadmitDate) ~ as_date(ReadmitDate) - as_date(ADMITTIME),
             TRUE ~ NA_real_
             )
         )
```

I have identified our time to hospital readmission, but have not examined or limited the data. Let's first visualize the distribution:

```{r}
admit_pts %>% filter(!is.na(TimeToReadmit)) %>%
  ggplot(aes(x=TimeToReadmit)) +
  geom_density() + theme_minimal() +
```

```
  xlab("Days to Hospital Readmission") + ylab("Density") +
  ggtitle("Density Curve of Days to Unplanned/Emergent Readmission") +
  geom_vline(aes(xintercept=30, color="30 Days"), alpha=0.4,  lwd=1.2, lty=2)
+
  geom_vline(aes(xintercept=90, color="90 Days"), lwd=1.2, lty=2, alpha=0.4,)
+
  geom_vline(aes(xintercept=365, color="365 Days"), lwd=1.4, lty=2) +
  scale_color_manual(name="Days to Readmission",
                     values=c(`30 Days`="red", `90 Days`="blue", `365
Days`="lightblue")) +
  theme(legend.position=c(0.72, 0.5), legend.box.margin = margin(6, 6, 6, 6))


```


This distribution looks very skewed. I added lines to the 30 and 90 days marks, as I was interested in these benchmarks, but I can't fully assess possible sample size of this group while excluding no readmission (from the above `filter` statement) and from a density curve. Below is a frequency table:

```{r}
admit_pts %>% mutate(ReadmitCats =
                       case_when(
                  is.na(TimeToReadmit)  ~ "No readmit",
                  TimeToReadmit >= 350 ~ "Greater than 1 year",
                  TimeToReadmit >= 90 ~ "From 90 to 365 days",
                  TimeToReadmit >= 30 ~ "30-90 days",
                  TRUE ~ "0-30 days"
                )) %>% count(ReadmitCats)

```

From the above table, with fairly low frequencies for the 0-30 and 30-90 days ranges alone, I will use readmission with the following calendar year (i.e. next 365 days). Within this chunk, we will also limit the analytic cohort specific to readmission variable.

I will ensure that the readmit variable `Yr1Readmit` is only calculated for patients who survived out to one year (i.e. `DOD`-`DISCHTIME`$\leq$ 365 days). When examining readmission, we should only include those patients who 1) survived out to one year (regardless of readmission status) and 2) among patients who died, patients who were readmitted within one year prior to their date of death:

```{r}
admit_pts <- admit_pts %>% mutate(TimeToMort =
                                    case_when(
                                      DOD!="" ~ as.Date(ymd_hms(DOD)) -
as.Date(ymd_hms(DISCHTIME)),
                                      TRUE  ~ 9999
                                      ),
                                  Yr1Readmit =
                                    case_when(
                                      TimeToMort>365 & TimeToReadmit<=365 ~
1,
```

```
                                    TimeToMort>365 ~ 0,
                                    TRUE ~ NA_real_
                                    ),
                                TimeToReadmit_Recalc = # if we do a time-
to-event analysis, including this recalculated variable
                                  case_when(
                                    TimeToReadmit<=365 ~ TimeToReadmit,
                                    TRUE ~ 366
                                    )
                                ) #%>% select(-ReadmitDate, -TimeToReadmit)
```

Diagnoses Codes

Now that we have cleaned our patient-level data elements, we can begin working with the diagnosis code data. This will include:
    1. Removal of V and E diagnoses codes related to health factors and causes of admission respectively outside of morbidity diagnosis
    2. Ensuring the validity of our ICD-9 codes through a definition check and a quick spot-check of gender-specific codes
    3. Limiting our diagnoses codes to only those that met our prevalence threshold of 1%

Imports

```{r}
icd_raw <- read.csv(here("Data", "Raw", "DIAGNOSES_ICD.csv"),
stringsAsFactors = F) %>% select(-ROW_ID)

cat("There are", icd_raw %>% nrow(), "rows in our raw, ICD-9 diagnosis code
data.\n")
cat("There are", icd_raw %>% distinct(SUBJECT_ID) %>% nrow(), "unique
`SUBJECT_ID` values (representign patients) in this data.\n")
cat("Lastly, there are", icd_raw %>% distinct(ICD9_CODE) %>% nrow(),
"distinct ICD-9 diagnosis codes in this data set.")
```

Removing V & E Codes

I will first remove any duplicated diagnoses codes within a patient *and* visit. I will also remove the V and E codes which correspond to Health Services/Factors and Causes of Injury/Illness respectively, separate from diagnoses.

```{r}
icd_precln <- icd_raw %>% distinct(SUBJECT_ID, HADM_ID, ICD9_CODE, .keep_all
= T) %>%
  filter(substring(ICD9_CODE, 1, 1)!="E" & substring(ICD9_CODE, 1, 1)!="V")
#removing V and E codes

icd_precln %>% sample_n(5)
```

```

##### Checking Code Definitions

Using the `icd` package's built-in `is_defined` function, which tests whether a given input value 1) follows valid formatting for an ICD-9 code (5 or less characters, numeric or alphanumeric for V, E codes (which we've excluded)) and 2) is defined using a call to CMS, which keeps a list of what the package refers to as "canonical" ICD-9 codes:

```{r}
icd_precln %>% mutate(valid = is_defined(ICD9_CODE)) %>% filter(valid==F & ICD9_CODE!="")
```

Gender Code Spot-Check

Although we will note exhaustively examine ICD-9 codes that may be mutually exclusive or gender-specific, we can spot check some large ranges of these codes to re-assure ourselves of the data's validity.

ICD-9 codes ranging from 600 to 608 are specific to males, so we can spot-check to see if any female patients were erroneously diagnosed with these codes:

```{r}
icd_precln %>% filter(ICD9_CODE>="6000" & ICD9_CODE<"6090") %>%
  merge(pts_raw, by="SUBJECT_ID") %>% count(GENDER)
```

And we can perform a similar check using female-specific codes ranging from 614 to 629:

```{r}
icd_precln %>% filter(ICD9_CODE>="6140" & ICD9_CODE<"6300") %>%
  merge(pts_raw, by="SUBJECT_ID") %>% count(GENDER)
```

Thankfully our spot checks appear to corroborate the ICD-9 data validity!

Limiting to Our Cohort

We must first limit our ICD data to only those patients/visits of interest for our analysis, which we have thankfully already cleaned and can simply use as a merging "limiter":

```{r}
icd_cohort <- admit_pts %>% select(SUBJECT_ID, HADM_ID) %>%
  merge(icd_precln, by=c("SUBJECT_ID", "HADM_ID"), all.x=T) %>% select(-SEQ_NUM)
```
```

Subsetting by Prevalence

Now we will finalize our diagnosis by subsetting our diagnoses codes to those with a minimum of 1% event rate in our cohort. The relatively large number of unique diagnoses codes contain a number of rare diseases, with extremely low variance. As a result, we will truncate to codes with a sufficiently high proportion or event rate:

```{r}
icd_1pct <- icd_cohort %>% count(ICD9_CODE) %>%
filter(n>(0.01*nrow(admit_pts))) %>% pull(ICD9_CODE)
length(icd_1pct)
icd_cohort <- icd_cohort %>% mutate(ICD9_CODE =
                            case_when(
                              ICD9_CODE %in% icd_1pct ~ ICD9_CODE,
                              TRUE ~ NA_character_)
                            ) %>% distinct(SUBJECT_ID, HADM_ID, ICD9_CODE)
```

Data Cleaning Concluding Notes

The above data wrangling corresponds to the inclusion, validity-checking, and coding of data for categories to be considered analysis. This data cleaning code does *not* perfectly prepare data for analysis. Treelet dimension reduction will require the calculation and input of a correlation and/or variance-covariance matrix, while modelling or descriptive analysis may require coercion of data elements to/from factors and integers or other coding changes. These small changes, which will change the structure of the data but not the content or information contained therein, are left as *ad hoc* programming done within each relevant analytic section.

EDA

Diagnosis Code Data

Diagnosis Frequency

We can look broadly at the frequency of all of our diagnoses codes, with the below plot simply arranged in descending order:

```{r}
icd_cohort %>% count(ICD9_CODE)  %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  ggplot(aes(x=reorder(ICD9_CODE, -n), y=n)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("Frequency Plot of All Diagnoses Codes",
          subtitle = paste("(Including only codes with 1% prevalence or
greater)\nn=", nrow(icd_cohort %>% count(ICD9_CODE)),
                           " unique diagnoses, among",
                           nrow(admit_pts), "patients")) +
    ylab("Frequency") + xlab("Distinct (Unlabelled) ICD-9 Codes") +
theme_minimal() +
    theme(axis.text.x=element_blank(), text=element_text(size=13.5))
```

```

As it is impossible to elucidate much useful information from this visual, due to the volume of data, we can examine simply the most common diagnoses codes, arbitrarily picking the top 15 for legibility of plots:

```{r}
icd_descr <- read.csv(here("Data", "Raw", "D_ICD_DIAGNOSES.csv"))

# Percentages
icd_cohort %>% count(ICD9_CODE) %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  merge(icd_descr, by="ICD9_CODE", all.x=T) %>% arrange(desc(n)) %>%
ungroup() %>%  filter(row_number()<=15) %>%
  mutate(Prop=n/nrow(icd_cohort %>% distinct(SUBJECT_ID))) %>%
select(ICD9_CODE, n, Prop)


icd_cohort %>% count(ICD9_CODE) %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  merge(icd_descr, by="ICD9_CODE", all.x=T) %>% arrange(desc(n)) %>%
ungroup() %>%  filter(row_number()<=15) %>%
  ggplot(aes(x=reorder(SHORT_TITLE, -n), y=n)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("Frequency of the 15 Most Common Diagnoses") +
    ylab("Frequency") + xlab("ICD-9 Code") + theme_minimal() +
    theme(axis.text.x=element_text(angle=60, vjust=0.9, hjust=0.8))
    # this is a really unfortunately x-axis, couldn't find a better angle or
adjustment for the x-axis unfortunately

```

And re-plotting with Thesis-friendly captions:


```{r, freq_plots}
icd_cohort %>% count(ICD9_CODE)  %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  ggplot(aes(x=reorder(ICD9_CODE, -n), y=n)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("A", #"Frequency Plot of All Diagnoses Codes",
          # subtitle = paste("(Including only codes with 1% prevalence or
greater)\nn=", nrow(icd_cohort %>% count(ICD9_CODE)),
          #                  " unique diagnoses, among",
          #                  nrow(admit_pts), "patients")
          ) +
    ylab("Frequency") + xlab("Distinct (Unlabelled) ICD-9 Codes") +
theme_minimal() +
    theme(axis.text.x=element_blank(), text=element_text(size=15))

icd_descr <- read.csv(here("Data", "Raw", "D_ICD_DIAGNOSES.csv"))

icd_cohort %>% count(ICD9_CODE) %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
```
```

```
  merge(icd_descr, by="ICD9_CODE", all.x=T) %>% arrange(desc(n)) %>%
ungroup() %>%  filter(row_number()<=15) %>%
  mutate(Prop = paste0(round(100*n / nrow(icd_cohort %>%
distinct(SUBJECT_ID)), 1), '%')) %>%
  ggplot(aes(x=reorder(SHORT_TITLE, -n), y=n, label=Prop)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("B", #"Frequency of the 15 Most Common Diagnoses"
            ) +
    ylab("Frequency") + xlab("ICD-9 Code") + theme_minimal() +
    theme(axis.text.x=element_text(angle=60, vjust=0.9, hjust=0.8),
text=element_text(size=15)) +
    # geom_text(vjust=1.2, color="white", size=3.31, hjust=0.45)
    geom_text(vjust=-0.9, angle=15) + ylim(c(0, 19000))
    # this is a really unfortunate x-axis, couldn't find a better angle or
adjustment for the x-axis

```


Correlation Matrix Among Top Diagnoses

Looking at the correlation matrix of these most common diagnoses codes (as the correlation of diagnoses is what will determine the hierarchy of clustering in the treelet method):

```{r}
icd_cohort %>% filter(!is.na(ICD9_CODE) & ICD9_CODE %in% (
        icd_cohort %>% count(ICD9_CODE) %>%
        arrange(desc(n)) %>%
        merge(icd_descr, by="ICD9_CODE", all.x=T) %>%
        arrange(desc(n)) %>%
        ungroup() %>%
        filter(row_number()<=15) %>% pull(ICD9_CODE)
        )) %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, x))  %>%
  select(-SUBJECT_ID) %>%
  cor() %>%
  corrplot::corrplot(type="upper", diag=F, order="hclust", method = "shade")
```


Correlation Matrix Among All Included ($1% \geq$ Prevalence) Codes

```{r}
x <- (icd_cohort %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, x))  %>%
  select(-SUBJECT_ID) %>%
  cor())

x[x>1] <- 1
```

```
x %>% corrplot::corrplot(type="upper", diag=F, order = "hclust", method =
"color", tl.pos = "n")
```

Most Correlated Diagnoses

In addition to the `corrplot` package's visualiation of an input correlation matrix, we can use the `corr_cross` package to examine the upper limit of our diagnoses code's correlations. In the plot below, "Correlation %" simply refers to the scaled correlation coefficient (e.g. a "Correlation %" of 89.45% corresponds to a correlation coefficient $\rho=0.8945$):

```{r, warning=F}
top10_corr_plot <- icd_cohort %>%
  filter(!is.na(ICD9_CODE)) %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, 1)) %>%
  select(-SUBJECT_ID) %>%
  corr_cross(top=10, plot=F)
```

```
top10_corr_plot %>% mutate(Pair = paste0(group1, ", ", group2)) %>%
  ggplot(aes(x=reorder(Pair, corr), y=round(corr, 2))) +
  geom_bar(stat='identity', fill="black", alpha=0.6) +
  geom_text(aes(y =round(corr, 2)-0.04, label=round(corr, 2)),
            color="white", alpha=0.75) +
  coord_flip() + theme_minimal() +
  theme(text=element_text(size=13.5)) +
  ylab("Correlation Coefficient") + xlab("ICD-9 Diagnosis Code Pair")
```

And then examine a matrix-plot of these diagnoses as well:

```{r}
top_corr_vars <- c(top10_corr_plot %>% mutate(vars=substr(key, 2,
nchar(key))) %>% pull(vars),
                   top10_corr_plot %>% mutate(vars=substr(mix, 2,
nchar(mix))) %>% pull(vars)) %>% unique()
```

```
icd_cohort %>%
  filter(!is.na(ICD9_CODE)) %>%
    mutate(values=1) %>%
    pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
    mutate_all(function(x) ifelse(is.na(x), 0, 1)) %>%
    select(!!!top_corr_vars) %>% cor() %>%
      corrplot::corrplot(type="upper", diag=F, order="hclust", method =
"shade")
```

```

#### Patient Level Data

We can briefly/descriptively examine some of our patient level data, observing frequencies or distributions of our covariates and examining possible relationships of our patient characteristics to mortality, readmission, and hospital length of stay where appropriate. Much of these visualizations were purely exploratory in nature. In instances where data were changed/re-categorized or otherwise altered based on the visualization, I have included comments/annotations. Otherwise, these figures are presented without commentary.

##### Number of Diagnoses

```{r}
icd_cohort %>% count(SUBJECT_ID) %>% summarise(Mean=quantile(n)[3],
                                               P25=quantile(n)[2],
                                               P75=quantile(n)[4])
```

Mortality

```{r}
admit_pts %>% mutate(MortalityType=
                       factor(case_when(
                         InHospMortality==1 ~ "In-Hospital Mortality",
                         TRUE ~ "Survived to Discharge")
                     )) %>% count(MortalityType) %>%
mutate(prop=paste0(round(n/nrow(admit_pts), 4)*100, '%')) %>%
  ggplot(aes(x=reorder(MortalityType, -n), y=n, fill=MortalityType,
label=prop)) +
    geom_text(position = position_dodge(.9),
              vjust = -0.2,
              size = 4) +
  geom_col() + ylab("Frequency") + xlab("Mortality Status") +
scale_fill_brewer(palette=2, type = "qual") +
  ggtitle("Frequency of In-Hospital Mortality Status") + theme_minimal() +
theme(legend.position = "none")
```

Payment/Insurance

```{r}
admit_pts %>% count(INSURANCE) %>%
mutate(prop=paste0(round(n/nrow(admit_pts), 4)*100, '%')) %>%
  ggplot(aes(x=reorder(INSURANCE, -n), y=n, fill=INSURANCE, label=prop)) +
    geom_text(position = position_dodge(.9),
              vjust = -0.2,
              size = 4) +
```
```

```
  geom_col() + ylab("Frequency") + xlab("Payment Method") +
scale_fill_brewer(palette=2, type = "qual") +
  ggtitle("Frequency of Insurance Status/Payment Method") + theme_minimal() +
theme(legend.position = "none")

```

With the two small groups of `Self Pay` and `Government`, I will contradict what I wrote earlier and collapse these categories. `Self Pay` will be collapsed into the `Private` category, and `Government` will be combined with `Medicaid` as `Medicaid/Non-Medicare Public Assistance`:

```{r}
admit_pts %>%
  count(INSURANCE) %>% mutate(prop=paste0(round(n/nrow(admit_pts), 4)*100,
'%'),
                                  InsuranceBin =
                                    case_when(
                                      INSURANCE == "Self Pay" | INSURANCE ==
"Private" ~ "Private/Self-Pay",
                                      INSURANCE=='Medicaid' | INSURANCE ==
'Government' ~ 'Medicaid/Public Assistance',
                                      TRUE ~ INSURANCE)) %>%
  ggplot(aes(x=reorder(InsuranceBin, -n), y=n, fill=reorder(INSURANCE, n),
label=prop)) +
  geom_col() + ylab("Frequency") + xlab("Payment Method") +
scale_fill_brewer(palette=2, type = "qual") +
  ggtitle("Frequency of Insurance Status/Payment Method") + theme_minimal() +
theme(legend.position = "none")

```


General Hospital Length of Stay

```{r}
los_graph <- admit_pts %>%
  mutate(GenLOS=ceiling(difftime(DISCHTIME, ADMITTIME, units = "days") %>%
as.numeric())) %>% select(SUBJECT_ID, GenLOS, DISCHTIME, ADMITTIME) # %>%

# quantile(los_graph$GenLOS)

los_graph %>%   ggplot(aes(x=GenLOS)) + geom_density(fill="lightblue",
alpha=0.4) + theme_minimal() +
  xlab("General Hospital Length of Stay") + ylab("Density") +
  ggtitle("Distribution of General Hospital Length of Stay (Days)") +
  annotate(geom="text", x=150, y=0.05, label=paste0("Length of stay values
ranged from 1 to ", max(los_graph$GenLOS), " days.")) +
  annotate(geom="text", x=150, y=0.04, label=paste0("Our length of stay
values have a mean of ", round(mean(los_graph$GenLOS), 2), " and variance \n
of ",

round(var(los_graph$GenLOS),2), ", suggesting overdispersion of this
variable."))


```

We unsurprisingly see a heavy skew in our length of stay data which is highly overdispersed (variance of 113 is more than ~13x greater than our mean of under 10 days).

Hospital Re-admission

```{r}
admit_pts %>% filter(!is.na(Yr1Readmit)) %>% count(Yr1Readmit) %>%
mutate(prop=paste0(100*round(n/nrow(admit_pts[!is.na(admit_pts$Yr1Readmit),])
, 4), "%")) %>%
  ggplot(aes(x=reorder(Yr1Readmit, -n), y=n, fill=as.factor(Yr1Readmit),
label=prop)) +
  geom_text(position = position_dodge(.9),
              vjust = -0.32,
              size = 4) +
  geom_bar(stat="identity") + ylab("Frequency") + xlab("Unplanned Readmission
Within One-Year of Discharge") +
  ggtitle("Frequency of Unplanned Readmission in MIMIC Data") +
  scale_fill_brewer(palette=2, type = "qual") + theme_minimal() +
  scale_x_discrete(label= c("No Readmission", "Readmitted")) +
theme(legend.position = "none")

```

Not only can we look at the simple binary readmission status, we also have time to re-admission, which we previously visualized among all patients but can look at simply within our subset of patients who were readmitted within our single, calendar year of interest:

```{r}
admit_pts %>% filter(Yr1Readmit==1) %>%
  ggplot(aes(x=TimeToReadmit_Recalc)) +
  geom_density(fill="white") + theme_minimal() +
  xlab("Days to Hospital Readmission") + ylab("Density") +
  ggtitle("Density Curve of Days to Unplanned/Emergent Readmission") +
  geom_vline(aes(xintercept=30, color="30 Days"), alpha=0.4,  lwd=1.2, lty=2)
+
  geom_vline(aes(xintercept=90, color="90 Days"), lwd=1.2, lty=2, alpha=0.4,)
+
  geom_vline(aes(xintercept=365, color="365 Days"), lwd=1.4, lty=2) +
  scale_color_manual(name="Days to Readmission",
                     values=c(`30 Days`="red", `90 Days`="blue", `365
Days`="lightblue")) +
  theme(legend.position=c(0.72, 0.5), legend.box.margin = margin(6, 6, 6, 6))

```

Age

```{r}
# mean(admit_pts$Age)
# sd(admit_pts$Age)
quantile(admit_pts$Age)
```

```
admit_pts %>%
  ggplot(aes(x=Age)) +
  geom_density(fill="white") + theme_minimal() +
  xlab("Age") + ylab("Density") +
  ggtitle("Density Curve of Age")
```

Gender

```{r}
admit_pts %>% count(GENDER) %>%
mutate(prop=paste0(100*round(n/nrow(admit_pts), 4), "%")) %>%
  ggplot(aes(x=reorder(GENDER, -n), y=n, fill=GENDER, label=prop)) +
  geom_text(position = position_dodge(.9),
            vjust = -0.32,
            size = 4) +
  geom_bar(stat="identity") + ylab("Frequency") + xlab("Gender") +
  ggtitle("Frequency of Gender in MIMIC Data") +
  scale_fill_brewer(palette=2, type = "qual") + theme_minimal()
```

We have a fairly balanced data set with respect to gender, with men outnumbering women (which we would expect in a data set of critical care admissions).

Final Data Export

For the exploratory analysis above, the diagnosis data and patient-characteristic data have been contained in separate data frames. Below I pivot the diagnosis data (as previously done when determining the correlation matrix of our diagnosis data) from `icd_cohort` into `icd_wide` and merge the resulting pivoted data frame with the patient characteristics data contained in `admit_pts`. The final dataframe is then titled `cohort_full`. This data frame is used in this file, but I also export it as the standalone cohort and if in the future I would prefer to separate the data cleaning and EDA from the dimension reduction and regression modelling results of my thesis.

```{r}
icd_wide <- icd_cohort %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, x)) %>% select(-`NA`)

cohort_full <- merge(icd_wide, admit_pts, by="SUBJECT_ID")
```

```
colnames(cohort_full)[c(grep("[0-9]$", colnames(cohort_full)))] <-
paste0("X", colnames(cohort_full)[c(grep("[0-9]$", colnames(cohort_full)))])

write.csv(cohort_full,
          here("Data", "cohort_full.csv"),
          row.names = F)
```


Appendixes


```{r, warning=F, message=F}
require(magrittr) # Ceci n'est pas une %>%
require(dplyr) # General data management, cleaning (admittedly I switch
between Base R and tidyverse as I code, somewhat stream-of-consciousness ly)
require(ggplot2) # Visualization
require(comorbidity) # Used to easily generate Elixhauser comorbdity
grouping/categorization [8/23/2020 Note: may be excluded if Elixhauser or
Charlson not used]
require(tidyr) # pivot functions for transposing data to/from long and wide
require(icd) # used in validity check of diagnoses codes
require(lubridate) # used in evaluating dates, most notably in date of death
require(lares) # corr_cross function used to identify the top correlations
within a data frame/design matrix
require(corrplot) # used for visualizing correlation matrices
require(here) # Used for data-calls/ease of file path storage usage
require(treelet) # for treelet modelling
require(ggdendro) # trying ggplot's dnedrogram extension

if(!("cohort_full" %in% ls())) {
  cohort_full <-  read.csv(here("Data", "cohort_full.csv"))
}

```


Appendix A: Thesis Table & Figure Generation

Redundant code from the main body of cleaning and EDA code, but I wanted to
consolidate relevant table & figure genereation code. This is not an
exhaustive list of tables & figures, including only those captured in the
descriptive analyses (Results 3.1).

Table 2A & 2B: Cohort Descriptives

```{r}
stat_sum <- function(data, var, stat, category=NULL) {
  quovar <- deparse(substitute(var))
  if(stat=="mean") output_txt <- paste0(mean(data[,quovar]) %>% round(2), "
(", sd(data[,quovar]) %>% round(2), ")")
  if(stat=="median") output_txt <- paste0(median(data[,quovar]) %>% round(2),
" [", quantile(data[,quovar])[2] %>% round(2), "-",
quantile(data[,quovar])[4] %>% round(2), "]")
  if(stat=="proportion" | stat=="prop") {
```

```
    if(is.null(category)) stop("When requesting proportion for categorical
vaiablevariable, please specify ")
    freq <- data[data[,quovar]==category & !is.na(data[,quovar]),] %>% nrow()
    prop <- (100*(data[data[,quovar]==category & !is.na(data[,quovar]),] %>%
nrow()) / nrow(data[!is.na(data[,quovar]),])) %>% round(2)
    output_txt <- paste0(freq, " (", prop, "%)")
  }
  return(output_txt)
}

## 1A: Mortality & LOS Cohort (n=38,554)
  # Calculating number of ICD-9-CM Diagnosis Codes per patient (Median [IQR])
before table
    icd_quantiles <- icd_cohort %>% count(SUBJECT_ID) %>% pull(n) %>%
quantile()

  # Generating the table in an easy copy/paste format
    (sumtbl <- cohort_full %>% summarize(
      # `Age, Mean (SD)` = paste0(round(mean(Age), 2), " (",
round(sd(Age),2), ")"),
      # `Sex (Male), n (%)` = paste0(sum(cohort_full$GENDER=="M"), " (",
round(100*sum(cohort_full$GENDER=="M")/nrow(cohort_full), 2), "%)"),
      # `Hospital Stay (days), Median (IQR)` = paste0(median(HospitalLOS), "
[", quantile(HospitalLOS)[2], "-", quantile(HospitalLOS)[4], "]"),
      `Age, Mean (SD)` = stat_sum(data=., var=Age, stat="mean"),
      `Sex (Male), n (%)` = stat_sum(data=., var=GENDER, stat="prop",
category = "M"),
      `Hospital Stay (days), Median (IQR)` = stat_sum(., var=HospitalLOS,
stat="median"),
      `Re-Admission*, n (%)`= "",
      `In-Hospital Mortality, n (%)` = stat_sum(., InHospMortality, "prop",
1),
      `Number of ICD-9-CM Diagnosis Codes per Patient, Median (IQR)` =
paste0(icd_quantiles[3], " [", icd_quantiles[2], "-", icd_quantiles[4], "]"),
      `Primary Payment Method, n (%)` = "",
      `Medicare` = stat_sum(., INSURANCE, "prop", "Medicare"),
      `Private Insurance` = stat_sum(., INSURANCE, "prop", "Private"),
      `Self-Pay` = stat_sum(., INSURANCE, "prop", "Self Pay"),
      `Medicaid` = stat_sum(., INSURANCE, "prop", "Medicaid"),
      `Other Public Assistance` = stat_sum(., INSURANCE, "prop",
"Government")
    ))

    data.frame(colnames(sumtbl),
               t(sumtbl[1,]),
               row.names = NULL) %>% write.table("clipboard")



## 1B: Reamdission Cohort (n=28,893)
    readmit_cohort <- cohort_full %>% filter(!is.na(Yr1Readmit))

    readmit_icds <- icd_cohort %>% filter(SUBJECT_ID %in% (readmit_cohort %>%
pull(SUBJECT_ID)))
    readmit_quantiles <- readmit_icds %>% count(SUBJECT_ID) %>% pull(n) %>%
quantile()
```

```
    (sumtbl_readmit <- readmit_cohort %>% summarize(
      # `Age, Mean (SD)` = paste0(round(mean(Age), 2), " (",
round(sd(Age),2), ")"),
      # `Sex (Male), n (%)` = paste0(sum(cohort_full$GENDER=="M"), " (",
round(100*sum(cohort_full$GENDER=="M")/nrow(cohort_full), 2), "%)"),
      # `Hospital Stay (days), Median (IQR)` = paste0(median(HospitalLOS), "
[", quantile(HospitalLOS)[2], "-", quantile(HospitalLOS)[4], "]"),
      `Age, Mean (SD)` = stat_sum(data=., var=Age, stat="mean"),
      `Sex (Male), n (%)` = stat_sum(data=., var=GENDER, stat="prop",
category = "M"),
      `Hospital Stay (days), Median (IQR)` = stat_sum(., var=HospitalLOS,
stat="median"),
      `Re-Admission*, n (%)`= stat_sum(., Yr1Readmit, "prop", 1),
      `In-Hospital Mortality, n (%)` = stat_sum(., InHospMortality, "prop",
1),
      `Number of ICD-9-CM Diagnosis Codes per Patient, Median (IQR)` =
paste0(readmit_quantiles[3], " [", readmit_quantiles[2], "-",
readmit_quantiles[4], "]"),
      `Primary Payment Method, n (%)` = "",
      `Medicare` = stat_sum(., INSURANCE, "prop", "Medicare"),
      `Private Insurance` = stat_sum(., INSURANCE, "prop", "Private"),
      `Self-Pay` = stat_sum(., INSURANCE, "prop", "Self Pay"),
      `Medicaid` = stat_sum(., INSURANCE, "prop", "Medicaid"),
      `Other Public Assistance` = stat_sum(., INSURANCE, "prop",
"Government")
    ))


# All results combined
data.frame(colnames(sumtbl),
       t(sumtbl[1,]),
       t(sumtbl_readmit[1,]),
       row.names = NULL) %>% write.table("clipboard")


```


Figure 1A & 1B: Diagnosis Code Frequency
```{r}
icd_cohort %>% count(ICD9_CODE)  %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  ggplot(aes(x=reorder(ICD9_CODE, -n), y=n)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("A", #"Frequency Plot of All Diagnoses Codes",
          # subtitle = paste("(Including only codes with 1% prevalence or
greater)\nn=", nrow(icd_cohort %>% count(ICD9_CODE)),
          #                  " unique diagnoses, among",
          #                  nrow(admit_pts), "patients")
          ) +
    ylab("Frequency") + xlab("Distinct (Unlabelled) ICD-9 Codes") +
theme_minimal() +
    theme(axis.text.x=element_blank(), text=element_text(size=15))

icd_descr <- read.csv(here("Data", "Raw", "D_ICD_DIAGNOSES.csv"))
```

```
icd_cohort %>% count(ICD9_CODE) %>% arrange(desc(n)) %>%
filter(!is.na(ICD9_CODE)) %>%
  merge(icd_descr, by="ICD9_CODE", all.x=T) %>% arrange(desc(n)) %>%
ungroup() %>%  filter(row_number()<=15) %>%
  mutate(Prop = paste0(round(100*n / nrow(icd_cohort %>%
distinct(SUBJECT_ID)), 1), '%')) %>%
  ggplot(aes(x=reorder(SHORT_TITLE, -n), y=n, label=Prop)) +
    geom_bar(stat="identity", fill="navyblue", alpha=0.65) +
    ggtitle("B", #"Frequency of the 15 Most Common Diagnoses"
            ) +
    ylab("Frequency") + xlab("ICD-9 Code") + theme_minimal() +
    theme(axis.text.x=element_text(angle=60, vjust=0.9, hjust=0.8),
text=element_text(size=15)) +
    # geom_text(vjust=1.2, color="white", size=3.31, hjust=0.45)
    geom_text(vjust=-0.9, angle=15) + ylim(c(0, 19000))
    # this is a really unfortunate x-axis, couldn't find a better angle or
adjustment for the x-axis
```

Figure 3: Correlation Matrix
```{r}
cormat <- (icd_cohort %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, x))  %>%
  select(-SUBJECT_ID) %>%
  cor())

cormat %>% corrplot::corrplot(type="upper", diag=F, order = "hclust", col =
colorRampPalette(c("red","white", "blue"))(10), method = "color", tl.pos =
"n")
```


Figure 4: Highest Correlation Bar Graph
```{r}
top10_corr_plot <- icd_cohort %>%
  filter(!is.na(ICD9_CODE)) %>%
  mutate(values=1) %>%
  pivot_wider(id_cols="SUBJECT_ID", names_from="ICD9_CODE",
values_from="values") %>%
  mutate_all(function(x) ifelse(is.na(x), 0, 1)) %>%
  select(-SUBJECT_ID) %>%
  corr_cross(top=10, plot=F)

top10_corr_plot <- top10_corr_plot %>%  mutate(group1 =
                              paste0(substr(group1, 0, 3), ".", substr(group1,
4, nchar(group1))),
                            group2 =
                              paste0(substr(group2, 0, 3), ".", substr(group2,
4, nchar(group2))))

barplot <- top10_corr_plot %>% mutate(Pair = paste0(group1, ", ", group2))
%>%
  ggplot(aes(x=reorder(Pair, corr), y=round(corr, 2))) +
```

```
  geom_bar(stat='identity', fill="black", alpha=0.6) +
  geom_text(aes(y =round(corr, 2)-0.04, label=round(corr, 2)),
            color="white", alpha=0.75) +
  coord_flip() + theme_minimal() +
  theme(text=element_text(size=13.5)) +
  ylab("Correlation Coefficient") + xlab("ICD-9 Diagnosis Code Pair")

require(gridExtra)
require(grid)

corrtbl <- c(top10_corr_plot %>% pull(group1) %>% unique(),
             t(t(top10_corr_plot %>% pull(group2) %>% unique()))) %>%
unique() %>% data.frame(ICD9_CODE=.) %>%
  merge(icd_descr %>% select(ICD9_CODE, SHORT_TITLE) %>%
          mutate(ICD9_CODE = paste0(substr(ICD9_CODE, 0, 3), ".",
substr(ICD9_CODE, 4, nchar(ICD9_CODE)))), by="ICD9_CODE", all.x=T) %>%
arrange(ICD9_CODE) %>%
  mutate(SHORT_TITLE = case_when(is.na(SHORT_TITLE) ~ "Digestive system
complications NOS",
                                 ICD9_CODE=="250.60" ~  "Diabetes (II) with
neurological manifestations",
                                 ICD9_CODE=="294.10" ~  "Dementia without
behavioral disturbance",
                                 ICD9_CODE=="403.90" ~  "Hypertensive chronic
kidney disease, stage I-IV",
                                 ICD9_CODE=="403.91" ~  "Hypertensive chronic
kidney disease, stage V+",
                                 ICD9_CODE=="585.9" ~  "Chronic kidney
disease NOS",
                                 TRUE ~ SHORT_TITLE)) %>% select(`ICD-9-CM
Code`=ICD9_CODE, `Description`=SHORT_TITLE)  %>% tableGrob(rows = NULL)


grid.arrange(barplot,
             corrtbl,
             nrow=1,
             as.table=T)
```

Appendix B: Unused Exporatory Analyses

My lazy calling of packages and data, so that analysis does not require running all cleaning and EDA code above:

Precursor Dimension Reduction

Prior to the treelet cross-validation process, Dr. Carlson suggested fitting PCA to evaluate a possible range of values for the K number of clusters parameter to fit in the treelet cross-validation process. I thought it may be interesting to similarly do some (similarly preliminary) agglomerative hierarchical clustering to the data.

PCA Precursor

```{r}
```

```
icd_pca <- cohort_full %>% select(starts_with("X")) %>% prcomp(center=T,
scale=T)

icd_pca_df <- data.frame(PC = 1:178,
                         Var = icd_pca$sdev^2) %>%
              mutate(PropVar = Var / nrow(.),
                     CmltvPropVar = cumsum(PropVar))

icd_pca_df %>% ggplot(aes(x=PC, y=PropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Proportion of Variance Explained") + xlab("Principal Component") +
  ggtitle("Proportion of Variance Explained by Individual Principal
Component")

icd_pca_df %>% ggplot(aes(x=PC, y=CmltvPropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Cumulative Proportion of Variance Explained") + xlab("Principal
Component") +
  ggtitle("Cumulative Proportion of Variance Explained by Principal
Component")
```

Preliminary Treelet

Full dendrogram of our treelet, not particularly useful/insightful but thanfully it is simple and quick to fit our treelet, retaining results for all levels

```{r, warning=F, message=F}
# compute correlation matrix
icd_cor <- cohort_full %>% select(starts_with("X")) %>% cor()

# run treelet
tt_results <- treelet::Run_JTree(icd_cor, nrow(icd_cor)-1, 1:nrow(icd_cor)-1)
```

Treelet Identification

```{r}
tt_results %>% str()
```

Dendrogram Vizualization

```{r}
# Converting the covariance matrix --> correlation matrix --> distance matrix
  # currently simply for the highest level of the covariance matrix
dist_mat <- as.dist(
  1-cov2cor(tt_results$TreeCovs[[nrow(icd_cor)-1]])
  )
```

```
# Making the result easily plotted in a dendrogram
  dendr <- dendro_data(hclust(dist_mat), type="rectangle")

# Modifying the axis position of the labels slightly to reduce length of the
final visual
  dendr$segments[segment(dendr)$yend==0, "yend"] <-
min(segment(dendr)[segment(dendr)$yend>0, "yend"])*0.95
  dendr$labels$y <- min(segment(dendr)[segment(dendr)$yend>0, "yend"])
  dendr$labels$label <- stringr::str_replace(dendr$labels$label, "X", "")

# Plot
  ggplot() +
    geom_segment(data=segment(dendr), aes(x=x, y=y, xend=xend, yend=yend)) +
    geom_text(data=label(dendr), aes(x=x, y=y, label=label, hjust=0), size=3)
+
    coord_flip() + scale_y_reverse(expand=c(0.2, 0)) +
    theme(axis.line.y=element_blank(),
          axis.ticks.y=element_blank(),
          axis.text.y=element_blank(),
          axis.title.y=element_blank(),
          panel.background=element_rect(fill="white")) +
    ggtitle("Example Dendrogram of All Data", subtitle = "Maximum Cut-Off
Chosen Arbitrarily\nVisual and results incomplete, only included
demonstratively")


```
```

The above visualization is impossible to decipher, but (again solely for current presentation and familiaring myself with the treelet function's output structure), we can visualize the treelet for only the first 20 conjoinings/clusterings:

```{r, warning=F, message=F}
# pick zposition of interest (i.e. cut-level) and take the covariance matrix
from that level
  # tt_results$Zpos[1:20,]

# need to extract the numeric label to the actual diagnosis code
  labels_df <- cov2cor(tt_results$TreeCovs[[ncol(icd_cor)-1]]) %>% colnames()
%>% data.frame(code = ., label=1:178)

  codes_mat <- tt_results$Zpos[1:20,] %>% as.data.frame() %>%
    merge(labels_df, by.x="V1", by.y="label", all.x=T)  %>%
    merge(labels_df, by.x="V2", by.y="label", all.x=T) %>%
    select(CodeLab1=code.x, CodeLab2=code.y) %>% as.matrix()

  "X99592" %in% codes_mat



  dist_mat <- as.dist(
    1 - cov2cor(tt_results$TreeCovs[[ncol(icd_cor)-1]]) %>% .[colnames(.)
%in% codes_mat,colnames(.) %in% codes_mat]
```
```

```
  )

dendr <- dendro_data(hclust(dist_mat), type="rectangle")

dendr$segments[segment(dendr)$yend==0, "yend"] <-
min(segment(dendr)[segment(dendr)$yend>0, "yend"])*0.95
dendr$labels$y <- min(segment(dendr)[segment(dendr)$yend>0, "yend"])
dendr$labels$label <- stringr::str_replace(dendr$labels$label, "X", "")

ggplot() +
  geom_segment(data=segment(dendr), aes(x=x, y=y, xend=xend, yend=yend)) +
  geom_text(data=label(dendr), aes(x=x, y=y, label=label, hjust=0), size=3) +
  coord_flip() + scale_y_reverse(expand=c(0.2, 0)) +
  theme(axis.line.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.text.y=element_blank(),
        axis.title.y=element_blank(),
        panel.background=element_rect(fill="white"))


```


Trying to subset labels in the full dendrogram


```{r, warning=F, message=F}
# compute correlation matrix
icd_cor <- cohort_full %>% select(starts_with("X")) %>% cor()

# run treelet
tt_results <- treelet::Run_JTree(icd_cor, nrow(icd_cor)-1, 1:nrow(icd_cor)-1)

# Converting the covariance matrix --> correlation matrix --> distance matrix
  # currently simply for the highest level of the covariance matrix
dist_mat <- as.dist(
  1-cov2cor(tt_results$TreeCovs[[nrow(icd_cor)-1]])
  )

# Making the result easily plotted in a dendrogram
  dendr <- dendro_data(hclust(dist_mat), type="rectangle")

# Modifying the axis position of the labels slightly to reduce length of the
final visual
  dendr$segments[segment(dendr)$yend==0, "yend"] <-
min(segment(dendr)[segment(dendr)$yend>0, "yend"])*0.95
  dendr$labels$y <- min(segment(dendr)[segment(dendr)$yend>0, "yend"])
  dendr$labels[!(dendr$labels$label %in% codes_mat), "label"] <- ""
  dendr$labels$label <- stringr::str_replace(dendr$labels$label, "X", "")

# Plot
  ggplot() +
    geom_segment(data=segment(dendr), aes(x=x, y=y, xend=xend, yend=yend)) +
    geom_text(data=label(dendr), aes(x=x, y=y, label=label, hjust=0), size=3)
+
    coord_flip() + scale_y_reverse(expand=c(0.2, 0)) +
```

```
    theme(axis.line.y=element_blank(),
          axis.ticks.y=element_blank(),
          axis.text.y=element_blank(),
          axis.title.y=element_blank(),
          panel.background=element_rect(fill="white")) +
    ggtitle("Example Dendrogram of All Data", subtitle = "Maximum Cut-Off
Chosen Arbitrarily\nVisual and results incomplete, only included
demonstratively")


```
```

Appendix B.2 R Code to Perform Treelet and GLM Fitting (incl. Cross-Validation)

```
---
title: "Treelet Transform: Identifing clusters of ICD-9 Diagnoses in a Boston
Trauma Center"
subtitle: "Data Analysis: Treelet & GLM Fitting"
author: "Dominic DiSanto\n Master's Thesis"
date: "Updated 9/20/2020"
output:
  html_document:
    keep_md: true
    toc: true
    toc_depth: '3'
    code_folding: show
---

## Preparation


## Libraries
```{r, message=F, warning=FALSE}
library(magrittr) # Ceci n'est pas une %>%, loaded via dplyr also but liked
to include for transparency
library(dplyr) # General data management, cleaning (admittedly I switch
between Base R and tidyverse as I code, somewhat stream-of-consciousness ly)
library(ggplot2) # Visualization
library(tidyr) # pivot functions for transposing data to/from long and wide
library(icd) # used in validity check of diagnoses codes
library(lubridate) # used in evaluating dates, most notably in date of death
library(lares) # corr_cross function used to identify the top correlations
within a data frame/design matrix
```

```
library(corrplot) # used for visualizing correlation matrices
library(here) # Used for data-calls/ease of file path storage usage
library(treelet) # Used for treelet analysis
library(ggdendro) # Used for dendrogram visualization of Treelet analysis
library(gghighlight) # Used in cross-validation visualizations
library(MASS) # Used for glm.nb negative binomial regression function
require(stringr) # Some regex matching for filtering in the visualiation of
p-values & coefficients from GLM's
require(pROC)

select <- dplyr::select # Masking the MASS select function, somethign to do
with ridge regression I think, in favor of dplyr's `select()` function for
wrangling

`%nin%` <- Negate(`%in%`) # Creating the inverse function of %in%, simpler
than working with the !(...) negating logic syntax and saves me the extra
parenthetical blocks
```


File Path & Import

Loading data via `here` package

```{r}
here()

cohort_full <-  read.csv(here("Data", "cohort_full.csv"))
colnames(cohort_full) <- cohort_full %>% colnames() %>% gsub(pattern = "X",
"", x = .)
# cohort_full %>% head()

diagnosis_labs <- read.csv(here("Data", "Raw", "D_ICD_DIAGNOSES.csv"))
```


Treelet Cross-Validation Function

Defining the function that fits the treelet, and retains the characteristics
of:
- The "best K-basis" or the optimal L|K parameter for each K
- The retained K features for each given K
- All p-1 basis matrices from the fit treelet


```{r}
treelet_process <- function(x_mat, cov_mat){

  tt_results <- tt_results <- treelet::Run_JTree(cov_mat, nrow(cov_mat)-1,
1:nrow(cov_mat)-1) # Running the `treelet` package's implementation and
retaining all (1) to (p-1) results
  energy <- list() # empty list to store energy scores

      for(L in 1:length(tt_results$basis)) { # repeating this for all basis
matrices identified in the treelet above
```

```
        basisk <- tt_results$basis[[L]] # storing the specific basis
        w_x <- t(basisk) %*% t(x_mat) # applying the basis matrix to the
original input matri of diagnosis codes

          num_vec <- rowSums(abs(w_x)^2) # numerator vector -> calculation of
the p-1 values for the numerator of the energy score calculation
          den_vec <- x_mat^2 %>% colSums() # similar to the above line but
the denominator calculation, column summed over all n observations
              names(num_vec) <- NULL # removing dimension names o fmatrix
              names(den_vec) <- NULL

        energy[[L]] <- matrix(c(1:ncol(x_mat), num_vec / den_vec), ncol=2,
dimnames = list(NULL, c("W_i", "Energy"))) # generating energy scores
    }

  # Creating blank objects
    optimal_L <- matrix(c(1:length(energy), rep(NA, length(energy))),
nrow=length(energy), dimnames = list(NULL, c("K", "Optimal L"))) # empty list
set
    retained_fts <- rep(list(rep(list(rep(NA, length(energy))),
length(energy))), length(energy)) # empty list set

  # Reordering the energy matrices in descending order of normed energy score
    energy_ordered <- lapply(1:length(energy), function(L)
energy[[L]][energy[[L]][,2] %>% order(decreasing = T),]) # sorting all p-1
energy vectors in descending order

  # Identifying optimal L
    optimal_L <- matrix(c(1:length(energy_ordered), # identifying the basis
matrix with the highest energy summation for every given K
                        sapply(1:length(energy_ordered),
                               function(K) which.max(sapply(1:length(energy),
                                                            function(x)
sum(energy_ordered[[x]][1:K,2])
                                                            )
                                                     ))),
                        ncol=2, dimnames=list(NULL, c("GivenK",
"OptimalBasis_L")))

  # And retained fts
    retained_fts <- lapply(1:length(energy_ordered),
                           function(x)
energy_ordered[[optimal_L[x,2]]][optimal_L[1:x,1], 1]) # then the retained
features of the basis that represent the K highest energy score columns

  return(list(basis_mats=tt_results$basis,
              optimal_params=optimal_L,
              retained_fts=retained_fts))
}
```

Cross-Validation Data Split

Splitting the data into a cross-validation set (80%) and hold-out test set (20%). Within the 80% cross-validation set, I then create a new variable identifying the five folds to be used in the cross-validation process.

The length of stay and mortality models use the same cohort (the same inclusion criteria), so the same data splits are used for both of these analyses Our readmission cohort is limited only to patients who were readmitted or who survived out to one year withour readmission, so the data split is conduced separately for this subset of patients

Mortality & Length of Stay

```{r}
set.seed(2824)

hold_out_pts <- sample(1:nrow(cohort_full), size=nrow(cohort_full)/5, replace
= F)

holdout_test <- cohort_full[hold_out_pts,]
# nrow(holdout_test)

cv_data <- cohort_full[setdiff(1:nrow(cohort_full), hold_out_pts),]
# nrow(cv_data)

(nrow(holdout_test) + nrow(cv_data)) == nrow(cohort_full)

cv_data$fold <- sample(c(rep(1, ceiling(nrow(cv_data)/5)),
                         rep(2, ceiling(nrow(cv_data)/5)),
                         rep(3, ceiling(nrow(cv_data)/5)),
                         rep(4, ceiling(nrow(cv_data)/5)),
                         rep(5, ceiling(nrow(cv_data)/5))
                         ),
                       size=nrow(cv_data), replace=F
                       )

table(cv_data$fold)
cat("\n")
cat("Printing frequency of \"Self-Pay\" insurane category across CV
folds...\n")
count <- cv_data %>% filter(INSURANCE=="Self Pay") %>% count()
cat(paste0("Full analytic data (n=", nrow(cv_data), "): " ,  count, " (",
round(100*count/nrow(cv_data), 2) ,"%)\n"))

for(i in 1:max(cv_data$fold)){
  count <- cv_data %>% filter(fold==i) %>% filter(INSURANCE=="Self Pay") %>%
count()
  cat(paste0("Fold ", i, ": ", count, " (", round(100*count/nrow(cv_data %>%
filter(fold==i)), 2) ,"%)\n"))
}
```

Hospital Readmission

```{r}
set.seed(70221)
```

```
cohort_readmit <- cohort_full %>% filter(!is.na(Yr1Readmit))

hold_out_readmit <- sample(1:nrow(cohort_readmit),
size=nrow(cohort_readmit)/5, replace = F)

holdout_test_readmit <- cohort_readmit[hold_out_readmit,]
# nrow(holdout_test)

cv_data_readmit <- cohort_readmit[setdiff(1:nrow(cohort_readmit),
hold_out_readmit),]
# nrow(cv_data)

(nrow(holdout_test_readmit) + nrow(cv_data_readmit)) == nrow(cohort_readmit)

cv_data_readmit$fold <- sample(c(rep(1, ceiling(nrow(cv_data_readmit)/5)),
                          rep(2, ceiling(nrow(cv_data_readmit)/5)),
                          rep(3, ceiling(nrow(cv_data_readmit)/5)),
                          rep(4, ceiling(nrow(cv_data_readmit)/5)),
                          rep(5, ceiling(nrow(cv_data_readmit)/5))
                          ),
                        size=nrow(cv_data_readmit), replace=F
                        )

table(cv_data_readmit$fold)

cat("\n")
cat("Printing frequency of \"Self-Pay\" insurane category across CV
folds...\n")
count <- cv_data_readmit %>% filter(INSURANCE=="Self Pay") %>% count()
cat(paste0("Full analytic data (n=", nrow(cv_data_readmit), "): " ,  count, "
(", round(100*count/nrow(cv_data_readmit), 2) ,"%)\n"))

for(i in 1:max(cv_data_readmit$fold)){
  count <- cv_data_readmit %>% filter(fold==i) %>% filter(INSURANCE=="Self
Pay") %>% count()
  cat(paste0("Fold ", i, ": ", count, " (",
round(100*count/nrow(cv_data_readmit %>% filter(fold==i)), 2) ,"%)\n"))
}
```

Treelet Cross-Valdiation and Results Export

For each of our outcomes, similar processes are followed, which include:

1) Fitting the treelet model, using the previously defined `treelet_process` function, for each training fold
2) Then for K=1,2,...p-1 in this identified treelet:
    a) Fitting the appropriate regression model in the training data using the K dimensions and corresponding optimal Lth basis
    b) Using this fit model to predict probability of outcome (or outcome, in the length of stay negative binomial model)
    c) Assess test fit (Briers & AUC for mortality, readmission; MSE for length of stay)
    d) Export a data frame that contains test error for each K parameter and fold, and the average test Brier Score and AUC (so 177 observations (for 1 to

p-1 values of K), and 13 columns, a K value, test-fold specific Brier Score and AUC values, and the two averages)

A procedural note, the three outcome specific processes below include the same `brier`, `auc`, and `performance_df` named objects, so that each chunk writes over the results of the previous (not saving the resulting R objects for each section, which are large and tend to obstruct my R session), but each chunk exports the results in its final step.

In-Hospital Mortality

```{r}
brier <- list(c(), c(), c(), c(), c())
auc <- list(c(), c(), c(), c(), c())

for (fold_no in 1:max(cv_data$fold)) {

  train_cv <- cv_data[cv_data$fold!=fold_no, ]
  test_cv <- cv_data[cv_data$fold==fold_no, ]

  train_xmat <- train_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-Yr1Readmit) %>% as.matrix()
  train_cov <- cov(train_xmat)

  test_xmat <- test_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-Yr1Readmit) %>% as.matrix()

  tt_fold <- treelet_process(x_mat = train_xmat, cov_mat = train_cov)

  for(K in 1:length(tt_fold$basis_mats)){
        # basis_l <- tt_fold$basis_mats[[K]][,tt_fold$retained_fts[[K]]]
        basis_no <- tt_fold$optimal_params[i,2]
        basis_l <- tt_fold$basis_mats[[basis_no]][,tt_fold$retained_fts[[K]]]

        k_mat <- train_xmat %*%  basis_l

        train_glm_df <- train_cv %>% select(InHospMortality, GENDER, Age, INSURANCE) %>% cbind(., k_mat)
        if(K==1) colnames(train_glm_df)[5] <- "`1`"     # Account for weirdness in column naming from cbind() when test_kmat has one column
        # train_glm <- glm(train_cv$InHospMortality ~ train_cv$GENDER + train_cv$Age + as.factor(train_cv$INSURANCE) + train_cv$HospitalLOS + k_mat ,
        #                  family = "binomial")

        train_glm <- glm(InHospMortality ~ . , data=train_glm_df,
                         family = "binomial")

        test_kmat <- test_xmat %*% basis_l
        test_glm_df <- test_cv %>% select(InHospMortality, GENDER, Age, INSURANCE) %>% cbind(., test_kmat)
        if(K==1) colnames(test_glm_df)[5] <- "`1`"     # Account for weirdness in column naming from cbind() when test_kmat has one column
```

```
        phat <- predict(object = train_glm, newdata = test_glm_df, 
type="response")

        brier[[fold_no]][K] <- sum((phat - test_cv$InHospMortality)^2) / 
nrow(test_cv)
        auc[[fold_no]][K] <- pROC::roc(test_cv$InHospMortality, phat)$auc

    }
}


performance_df <- data.frame(K=c(1:length(brier[[1]])),
                             BS_F1 = brier[[1]],
                             BS_F2 = brier[[2]],
                             BS_F3 = brier[[3]],
                             BS_F4 = brier[[4]],
                             BS_F5 = brier[[5]],
                             AUC_F1 = auc[[1]],
                             AUC_F2 = auc[[2]],
                             AUC_F3 = auc[[3]],
                             AUC_F4 = auc[[4]],
                             AUC_F5 = auc[[5]])

performance_df2 <- performance_df %>% mutate(BS_TestAvg = 
                            rowMeans(select(performance_df, 
starts_with("BS"))),
                          AUC_TestAvg = 
                            rowMeans(select(performance_df, 
starts_with("AUC"))))

write.csv(performance_df2, 
here("Results/MortalityModel_CVPerformance_NoLOS_NewKLCode.csv"), row.names = 
F)

```


Hospital Readmission

```{r}
brier <- list(c(), c(), c(), c(), c())
auc <- list(c(), c(), c(), c(), c())


for (fold_no in 1:max(cv_data_readmit$fold)) {

  train_cv <- cv_data_readmit[cv_data_readmit$fold!=fold_no, ]
  test_cv <- cv_data_readmit[cv_data_readmit$fold==fold_no, ]

  train_xmat <- train_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
  train_cov <- cov(train_xmat)
```

```
  test_xmat <- test_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()

  tt_fold <- treelet_process(x_mat = train_xmat, cov_mat = train_cov)

  for(K in 1:length(tt_fold$basis_mats)){
        # basis_l <- tt_fold$basis_mats[[K]][,tt_fold$retained_fts[[K]]]
        basis_no <- tt_fold$optimal_params[K,2]
        basis_l <- tt_fold$basis_mats[[basis_no]][,tt_fold$retained_fts[[K]]]

        k_mat <- train_xmat %*%  basis_l

        train_glm_df <- train_cv %>% select(Yr1Readmit, GENDER, Age,
INSURANCE, HospitalLOS) %>% cbind(., k_mat)
        if(K==1) colnames(train_glm_df)[6] <- "`1`"     # Account for
weirdness in column naming from cbind() when test_kmat has one column
        # train_glm <- glm(train_cv$InHospMortality ~ train_cv$GENDER +
train_cv$Age + as.factor(train_cv$INSURANCE) + train_cv$HospitalLOS + k_mat ,
        #                  family = "binomial")

        train_glm <- glm(Yr1Readmit ~ . , data=train_glm_df,
                         family = "binomial")

        test_kmat <- test_xmat %*% basis_l
        test_glm_df <- test_cv %>% select(Yr1Readmit, GENDER, Age, INSURANCE,
HospitalLOS) %>% cbind(., test_kmat)
        if(K==1) colnames(test_glm_df)[6] <- "`1`"     # Account for
weirdness in column naming from cbind() when test_kmat has one column

        phat <- predict(object = train_glm, newdata = test_glm_df,
type="response")

        brier[[fold_no]][K] <- sum((phat - test_cv$Yr1Readmit)^2) /
nrow(test_cv)
        auc[[fold_no]][K] <- pROC::roc(test_cv$Yr1Readmit, phat)$auc

    }
}

performance_df_readmit <- data.frame(K=c(1:length(brier[[1]])),
                             BS_F1 = brier[[1]],
                             BS_F2 = brier[[2]],
                             BS_F3 = brier[[3]],
                             BS_F4 = brier[[4]],
                             BS_F5 = brier[[5]],
                             AUC_F1 = auc[[1]],
                             AUC_F2 = auc[[2]],
                             AUC_F3 = auc[[3]],
                             AUC_F4 = auc[[4]],
                             AUC_F5 = auc[[5]])

performance_df2_readmit <- performance_df_readmit %>% mutate(BS_TestAvg =
                             rowMeans(select(performance_df_readmit,
starts_with("BS"))),
                           AUC_TestAvg =
                             rowMeans(select(performance_df_readmit,
starts_with("AUC"))))
```

```
write.csv(performance_df2_readmit,
here("Results/ReadmissionModel_CVPerformance_NewKLCode.csv"), row.names = F)


```


Hospital Length of Stay

Cross-validation data splits are the same as the mortality data

```{r}

MSE <- list(c(), c(), c(), c(), c())

# fold_no = 5

for (fold_no in 1:max(cv_data$fold)) {

  train_cv <- cv_data[cv_data$fold!=fold_no, ]
  test_cv <- cv_data[cv_data$fold==fold_no, ]

  train_xmat <- train_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
  train_cov <- cov(train_xmat)

  test_xmat <- test_cv %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()

  tt_fold <- treelet_process(x_mat = train_xmat, cov_mat = train_cov)

  for(K in 1:length(tt_fold$basis_mats)){
        # basis_l <- tt_fold$basis_mats[[K]][,tt_fold$retained_fts[[K]]]
        basis_no <- tt_fold$optimal_params[i,2]
        basis_l <- tt_fold$basis_mats[[basis_no]][,tt_fold$retained_fts[[K]]]

        k_mat <- train_xmat %*%  basis_l

        train_glm_df <- train_cv %>% select(HospitalLOS, GENDER, Age,
INSURANCE) %>% cbind(., k_mat)
        if(K==1) colnames(train_glm_df)[5] <- "`1`"     # Account for
weirdness in column naming from cbind() when test_kmat has one column
        # train_glm <- glm(train_cv$InHospMortality ~ train_cv$GENDER +
train_cv$Age + as.factor(train_cv$INSURANCE) + train_cv$HospitalLOS + k_mat ,
        #                  family = "binomial")

        train_glm <- glm.nb(HospitalLOS ~ . , data=train_glm_df)

        test_kmat <- test_xmat %*% basis_l
        test_glm_df <- test_cv %>% select(HospitalLOS, GENDER, Age,
INSURANCE) %>% cbind(., test_kmat)
        if(K==1) colnames(test_glm_df)[5] <- "`1`"     # Account for
weirdness in column naming from cbind() when test_kmat has one column
```

```
        yhat <- predict(object = train_glm, newdata = test_glm_df,
type="response")

        MSE[[fold_no]][K] <- sum((yhat - test_cv$HospitalLOS)^2) /
nrow(test_cv)


    }
}


nb_MSE_df <- data.frame(K=c(1:length(MSE[[1]])),
                            MSE_F1 = MSE[[1]],
                            MSE_F2 = MSE[[2]],
                            MSE_F3 = MSE[[3]],
                            MSE_F4 = MSE[[4]],
                            MSE_F5 = MSE[[5]])

nb_MSE_df2 <- nb_MSE_df %>% mutate(MSE_TestAvg =
                           rowMeans(select(nb_MSE_df, starts_with("MSE"))))

write.csv(nb_MSE_df2, here("Results/LOSModel_MSE_DF_newKLCode.csv"),
row.names = F)
```

Tables & Figures

Having fit the treelet models above, we can now explore the results of the cross-validation processes. Each outcome section below includes chunks that:

1) Plot the cross-validation error for each K parameter (`Cross-Validation Figures`) and identify the K and L|K parameter
2) Export the cluster membership and loadings
3) Build the final model on the full cross-validation set and assess test fit, including object-specific figures to be included in thesis manuscript

Mortality

```{r}
mortality_performance <-
read.csv(here("Results/Treelet_KLOpt_WithinCVLoop/MortalityModel_CVPerformanc
e_NoLOS_NewKLCode.csv"))

k_1sd <-
mortality_performance[mortality_performance$BS_TestAvg<=(min(mortality_perfor
mance$BS_TestAvg) + sd(mortality_performance$BS_TestAvg)), ] %>% .[1,1]

mortality_performance <- mortality_performance %>% mutate(ParamFlag =
                                  case_when(
                                    BS_TestAvg==min(BS_TestAvg) ~ "Minimizes
Briers Score",
                                    K==k_1sd ~ "More Sparse Parameter",
                                    TRUE ~ NA_character_
                                  )) %>% ungroup()
```

```

#### Cross-Validation Figure

```{r, warning=F, message=F}
ggplot(mortality_performance, aes(x=K, y=BS_TestAvg, color =
as.factor(ParamFlag))) +
  geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
  theme_minimal() + ggtitle("In-Hospital Mortality Model") +
  xlab("Value of Parameter K") + ylab("Average Briers Score (Across 5 Test
Folds)") +
  gghighlight(ParamFlag!=0) + labs(color="Optimal Parameters") +
  scale_color_brewer(type = "qual", palette = 6) +
  theme(legend.position=c(0.75, 0.75), text = element_text(size=13.5))


# which.min(mortality_performance$BS_TestAvg)
#
# # AUC Graph if of any interest
# ggplot(mortality_performance, aes(x=K, y=AUC_TestAvg)) +
#   geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
#   theme_minimal() + ggtitle("In-Hospital Mortality Model") +
#   xlab("Value of Parameter K") + ylab("Average AUC (Across 5 Test Folds)")

```

Cluster Membership/Loading Export

```{r}
# Subsetting the highlighted K parameters above
  mortality_performance[!is.na(mortality_performance$ParamFlag),]
  # opt_Ks_mortality <-

# Refitting the treelet process in our training data to pull optimal L's for
our highlighted K's
  cv_xmat <- cv_data  %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
  cv_cov <- cov(cv_xmat)

  tt_fnc_mortality <- treelet_process(cv_xmat, cv_cov)

  tt_fnc_mortality$optimal_params[c(123, 174),]
  tt_fnc_mortality$retained_fts[[123]]

  # For our 1-standard deviation parameter, pulling K-features from the Lth
basis matrix
  final_basis_mortality <-
tt_fnc_mortality$basis_mats[[57]][,tt_fnc_mortality$retained_fts[[123]]] %>%
    as.data.frame() %>%
    mutate(LabelIndex = row_number(),
           RowMissCount = rowSums(.==0)) %>%
    filter(RowMissCount<123)

  labels_df <- cv_cov %>% colnames() %>%
    data.frame(code = ., label=1:ncol(cv_cov))

  loading_mat_mortality <- merge(final_basis_mortality, labels_df,
```
```

```
                                all.x=T, by.y="label", by.x="LabelIndex")


  holder <- sapply(2:(ncol(loading_mat_mortality)-2), function(x)
matrix(c(loading_mat_mortality[loading_mat_mortality[,x]!=0, "code"],

loading_mat_mortality[loading_mat_mortality[,x]!=0, x]),

ncol=2))

  # Lazily using a for loop to transform to an exportable csv
  i = 1
  reformat_loadingmat <- as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2)

  for (i in 2:length(holder)){
  reformat_loadingmat <- rbind(reformat_loadingmat,
                               as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2))
  }


  write.csv(loading_mat_mortality,
            here("Results/LoadingMatrix_Mortality.csv"))

  # Pulling in labels for the full matrix
  reformat_loadingmat_labs <- reformat_loadingmat %>%
mutate(Order=row_number()) %>%
    merge(diagnosis_labs %>% select(ICD9_CODE, SHORT_TITLE), by.x="Code",
by.y="ICD9_CODE", all.x=T) %>% arrange(Order) %>% select(-Order)

  write.csv(reformat_loadingmat_labs,
            here("Results/LoadingMatrix_Mortality_Redux.csv"), na = "")



```


Building Final Model and Assessting Test Fit

Using 1-Standard Deviation Parameter, building the logistic regression model
on our the 80% cross-validation subset

```{r}
final_basis <-
tt_fnc_mortality$basis_mats[[57]][,tt_fnc_mortality$retained_fts[[123]]]

cv_xmat_transform <- cv_xmat %*% final_basis

cv_predictors <- cv_data %>% select(GENDER, Age, INSURANCE, InHospMortality)
%>% cbind(., cv_xmat_transform) %>% as.data.frame()
```

```
dim(cv_predictors)

train_glm <- glm(InHospMortality ~ . , data=cv_predictors, family =
"binomial")
# train_glm %>% summary()
# confint(train_glm, parm = 1:7)

test_xmat <- holdout_test  %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
test_xmat_transform <- test_xmat %*% final_basis
test_predictors <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality) %>% cbind(., test_xmat_transform) %>% as.data.frame()

phat <- predict(object = train_glm, newdata = test_predictors,
type="response")

brier_test <- sum((phat - test_predictors$InHospMortality)^2) /
nrow(holdout_test)
auc_test <- pROC::roc(test_predictors$InHospMortality, phat)$auc

# Exporting full model estimates
ci95s <- confint.default(train_glm) # Using Wald approximation for confidence
intervals, profile likelihood using confint() from MASS takes minutes to run
(when it isn't crashing my R session)

cbind(train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
select(Estimate) %>% round(2),
      CI=paste0("[", (ci95s %>% round(2))[,1], ", ", (ci95s %>%
round(2))[,2], "]"),
      train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
select(`Pr(>|z|)`) %>% round(4)) %>% write.table("clipboard")
```

Graphing P-values/Coefficients

```{r}
# Height = -log(PValue); Color=Coefficient
train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
mutate(Covariate=rownames(.), Order=row_number()) %>%
select(PValue=`Pr(>|z|)`, everything()) %>%
  filter(stringr::str_detect(Covariate, "[0-9]")) %>% arrange(desc(Estimate))
%>% mutate(Label = case_when(row_number()<=5 ~ str_replace_all(Covariate,
"`", ""),

TRUE ~ "")) %>% arrange(Order) %>%
  ggplot(aes(x=reorder(Covariate, Order), y=-log(PValue), fill=Estimate)) +
    geom_bar(stat="identity") + theme_minimal() +
  geom_text(aes(label=Label, group=Label),
            hjust=-0.45, vjust=0.95) +
  theme(axis.text.x = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16),
```

```
        legend.title = element_text(size=12.4),
        legend.text =  element_text(size=10),
        legend.position = c(0.9, 0.65)) +
  labs(caption="Inset text notes feature numbers of five highest
coefficients") +
  xlab ("Treelet Feature") + ylab("-log(P-Value)") +
  scale_fill_continuous(type="viridis", name="Coefficient", direction=-1)

```


Graphing Phat Distributions

```{r}
phat_df <- as.data.frame(cbind("EventProb" = phat,
"ObsOut"=test_predictors$InHospMortality))

phat_df %>% ggplot(aes(x=EventProb, fill=as.factor(ObsOut))) +
  geom_density(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Density") + xlab("Predicted Probability of In-Hospital Mortality")  +
  scale_fill_manual(name="Observed Outcome",
                    labels=c("No Mortality", "Mortality Event"),
                    values=c("lightblue", "violetred4")) +
  ggtitle("Density Curve of Predicted Probabilities of In-Hospital
Mortality")


```


Fitting Model Without ICD Codes

```{r}
cv_predictors_noicd <- cv_predictors %>% select(-matches("[0-9]"))

train_glm_noicd <- glm(InHospMortality ~ . , data=cv_predictors_noicd, family
= "binomial")
#train_glm_noicd %>% summary()
# confint(train_glm, parm = 1:7)

test_predictors_noicd <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality)

phat_noicd <- predict(object = train_glm_noicd, newdata =
test_predictors_noicd, type="response")

brier_test_noicd <- sum((phat_noicd -
test_predictors_noicd$InHospMortality)^2) / nrow(test_predictors_noicd)
auc_test_noicd <- pROC::roc(test_predictors_noicd$InHospMortality,
phat_noicd)$auc

# Exporting full model estimates
ci95s_noicd <- confint.default(train_glm_noicd) # Using Wald approximation
for confidence intervals, profile likelihood using confint() from MASS takes
minutes to run (when it isn't crashing my R session)
```

```
cbind(train_glm_noicd %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(Estimate) %>% round(3),
      CI=paste0("[", (ci95s_noicd %>% round(3))[,1], ", ", (ci95s_noicd %>%
round(3))[,2], "]"),
      train_glm_noicd %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(`Pr(>|z|)`) %>% round(4)) #%>% write.table("clipboard")

```

Fitting with Most Significant Features
```{r}
# Pulling the five most significant
# train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
arrange(`Pr(>|z|)`) %>% tibble::rownames_to_column() %>%
filter(str_detect(rowname,"[0-9]"))

top5_tt_ftrs <- train_glm %>% summary() %>% .$coefficients %>%
as.data.frame() %>% arrange(`Pr(>|z|)`) %>%
  tibble::rownames_to_column() %>% filter(str_detect(rowname, "[0-9]")) %>%
filter(row_number()<=5) %>% pull(rowname) %>%
  str_replace_all("`", "")

top5_train_df <- cv_predictors %>% dplyr::select(GENDER, Age, INSURANCE,
InHospMortality, `1`, `15`, `13`, `2`, `7`)

top5_glm <- glm(InHospMortality ~ ., data=top5_train_df, family="binomial")

test_predictors_top5 <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality) %>% cbind(., test_xmat_transform) %>% as.data.frame() %>%
  select(GENDER, Age, INSURANCE, InHospMortality, all_of(top5_tt_ftrs))

phat_top5 <- predict(object = top5_glm, newdata = test_predictors_top5,
type="response")

brier_test_noicd <- sum((phat_top5 - test_predictors_top5$InHospMortality)^2)
/ nrow(test_predictors_top5)
auc_test_noicd <- pROC::roc(test_predictors_top5$InHospMortality,
phat_top5)$auc


```


Retained ICD-9-CM Does in 5 Features
```{r}
top5_tt_ftrs_cols <- sapply(top5_tt_ftrs, function(x) paste0("V", x))
names(top5_tt_ftrs_cols) <- NULL

loading_mat_mortality %>% select(!!top5_tt_ftrs_cols, code) %>%
  filter(V1!=0 | V15!=0 | V13!=0 | V2!=0 | V7!=0) %>% pull(code) %>% unique()
%>% length()
```


Model of ICD Codes (No treelet features)

```{r}
retained_codes <- loading_mat_mortality$code %>% unique()
length(retained_codes)

retain_traindf <- cv_data %>% select(GENDER, Age, INSURANCE, InHospMortality,
!!retained_codes)

retain_train_glm <- glm(InHospMortality ~ . , data=retain_traindf, family =
"binomial")

retain_test_df <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality, !!retained_codes)
retain_phat <- predict(object = retain_train_glm, newdata = retain_test_df,
type="response")

sum((retain_phat - retain_test_df$InHospMortality)^2) / nrow(retain_test_df)
  pROC::roc(retain_test_df$InHospMortality, retain_phat)$auc


# Logistic Regression of All Codes
all_traindf <- cv_data %>% select(GENDER, Age, INSURANCE, InHospMortality,
matches("[0-9]$"))
all_train_glm <- glm(InHospMortality ~ . , data=all_traindf, family =
"binomial")
all_test_df <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality, matches("[0-9]$"))
all_phat <- predict(object = all_train_glm, newdata = all_test_df,
type="response")

sum((all_phat - all_test_df$InHospMortality)^2) / nrow(retain_test_df)
pROC::roc(all_test_df$InHospMortality, all_phat)$auc

```


Comparative Treelet ROC Curves

```{r, warning=F, message=F}

roc_obj_noicd <- pROC::roc(test_predictors_noicd$InHospMortality, phat_noicd)
roc_obj <- pROC::roc(test_predictors$InHospMortality, phat)
roc_top5 <- pROC::roc(test_predictors_top5$InHospMortality, phat_top5)

pROC::ggroc(list(roc_obj, roc_top5, roc_obj_noicd), lwd=1.4) +
  theme_minimal() +
  xlab("Specificity") + ylab("Sensitivity") +
  theme(axis.text.x = element_blank(),
        # panel.grid.major = element_blank(),
        # panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16),
        legend.title = element_text(size=12.4),
        legend.text =  element_text(size=10),
        legend.position = c(0.6, 0.2)) +
  scale_color_brewer(type="qual", palette=2,
```

```
                     name="Model",
                     labels = c("Including All ICD-9-CM Treelet Features
(AUC=0.858)",
                                "Including 5 Most Significant ICD-9-
CM\nTreelet Features (AUC=0.830)",
                                "Excluding ICD-9-CM Treelet Features
(AUC=0.666)")) +
  ggtitle("Comparative ROC Curves of Mortality Predictions in Test Data")


```````


Probability Distributions of No ICD Model

```{r}
phat_df_noicd_readmit <- as.data.frame(cbind("EventProb" =
phat_readmit_noicd, "ObsOut"=test_predictors_readmit_noicd$Yr1Readmit))

phat_df_noicd_readmit %>% ggplot(aes(x=EventProb, fill=as.factor(ObsOut))) +
  geom_density(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.14, 0.6), text=element_text(size=13.5)) +
  ylab("Density") + xlab("Predicted Probability of In-Hospital Mortality")  +
  scale_fill_manual(name="Observed Outcome",
                     labels=c("No Mortality", "Mortality Event"),
                     values=c("lightblue", "violetred4")) +
  ggtitle("Density Curve of Predicted Probabilities of Hospital Re-
admission")
```

Readmission

Figures of Model Validation


```{r}
readmit_performance <-
read.csv(here("Results/Treelet_KLOpt_WithinCVLoop/ReadmissionModel_CVPerforma
nce_NewKLCode.csv"))
# readmit_performance <- performance_df2_readmit
readmit_performance <- readmit_performance %>% mutate(BS_TestAvg =
                            rowMeans(select(readmit_performance,
starts_with("BS_F"))),
                          AUC_TestAvg =
                            rowMeans(select(readmit_performance,
starts_with("AUC_F"))))

k_1sd_readmit <-
readmit_performance[readmit_performance$BS_TestAvg<=(min(readmit_performance$
BS_TestAvg) + sd(readmit_performance$BS_TestAvg)), ] %>% .[1,1]

readmit_performance <- readmit_performance %>% mutate(ParamFlag =
                                    case_when(
                                      BS_TestAvg==min(BS_TestAvg) ~ "Minimizes
Briers Score",
```

```
                                    K==k_1sd_readmit ~ "More Sparse
Parameter",
                                    TRUE ~ NA_character_
                                  )) %>% ungroup()

ggplot(readmit_performance, aes(x=K, y=BS_TestAvg,
color=as.factor(ParamFlag))) +
  geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
  theme_minimal() + ggtitle("Hospital Readmission Model") +
  xlab("Value of Parameter K") + ylab("Average Briers Score (Across 5 Test
Folds)") +
  gghighlight(ParamFlag!=0) + labs(color="Optimal Parameters") +
  scale_color_brewer(type = "qual", palette = 6) +
  theme(legend.position=c(0.65, 0.75), text = element_text(size=13.5))

readmit_performance %>% mutate(ParamFlag =
                                 case_when(
                                   is.na(ParamFlag) ~ "Unspecified",
                                   TRUE ~ ParamFlag
                                 )) %>%
  ggplot(aes(x=K, y=BS_TestAvg)) +
  geom_line(lwd=1.1, alpha=0.33) + geom_point(aes(x=K, y=BS_TestAvg,
color=as.factor(ParamFlag)), size=2.5, inherit.aes = F) +
  geom_point(size=1, alpha=0.2) +
  theme_minimal() + ggtitle("Hospital Readmission Model") +
  xlab("Value of Parameter K") + ylab("Average Briers Score (Across 5 Test
Folds)") +
  labs(color="Optimal Parameters") +
  scale_color_brewer(type = "qual", palette = 6, limits = c("Minimizes Briers
Score", "More Sparse Parameter")) +
  theme(legend.position=c(0.78, 0.25), text = element_text(size=13.5))

# AUC CV Plot if of any interest later
# ggplot(readmit_performance, aes(x=K, y=AUC_TestAvg)) +
#   geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
#   theme_minimal() + ggtitle("Readmission Mortality Model") +
#   xlab("Value of Parameter K") + ylab("Average AUC (Across 5 Test Folds)")
```

Cluster Membership/Loading Export

```{r}
readmit_performance [!is.na(readmit_performance $ParamFlag),]

cv_readmit_xmat <- cv_data_readmit  %>% select(matches("0|1|2|4|5|6|9")) %>%
select(-Yr1Readmit) %>% as.matrix()
cv_readmit_cov <- cov(cv_readmit_xmat)

tt_fnc_readmit <- treelet_process(cv_readmit_xmat, cv_readmit_cov)

tt_fnc_readmit$optimal_params[c(readmit_performance
[!is.na(readmit_performance $ParamFlag),] %>% pull(K)),]

# Matrix of loadings
```

```
 # For our 1-standard deviation parameter, pulling K-features from the Lth
basis matrix
  final_basis_readmit <-
tt_fnc_readmit$basis_mats[[177]][,tt_fnc_readmit$retained_fts[[30]]] %>%
    as.data.frame() %>%
    mutate(LabelIndex = row_number(),
           RowMissCount = rowSums(.==0)) %>%
    filter(RowMissCount<30)


  labels_df <- cv_readmit_cov %>% colnames() %>%
    data.frame(code = ., label=1:ncol(cv_readmit_cov))

  loading_mat_readmit <- merge(final_basis_readmit, labels_df,
                               all.x=T, by.y="label", by.x="LabelIndex")


  holder <- sapply(2:(ncol(loading_mat_readmit)-2), function(x)
matrix(c(loading_mat_readmit[loading_mat_readmit[,x]!=0, "code"],

loading_mat_readmit[loading_mat_readmit[,x]!=0, x]),

ncol=2))

  # Lazily using a for loop to transform to an exportable csv
  i = 1
  reformat_loadingmat_readmit <- as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2)

  for (i in 2:length(holder)){
  reformat_loadingmat_readmit <- rbind(reformat_loadingmat_readmit,
                              as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2))
  }


  reformat_loadingmat_readmit_labs <- reformat_loadingmat_readmit %>%
mutate(Order=row_number()) %>%
    merge(diagnosis_labs %>% select(ICD9_CODE, SHORT_TITLE), by.x="Code",
by.y="ICD9_CODE", all.x=T) %>% arrange(Order) %>% select(-Order)


  write.csv(loading_mat_readmit,
            here("Results/LoadingMatrix_Readmit.csv"))

  write.csv(reformat_loadingmat_readmit_labs,
            here("Results/LoadingMatrix_Readmit_Redux.csv"), na = "")


```

```
#### Building Final Model & Assessing Test Fit
Using 1-Standard Deviation Parameter, building the logistic regression model
on our the 80% cross-validation subset

```{r}
final_basis_readmit <-
tt_fnc_readmit$basis_mats[[177]][,tt_fnc_readmit$retained_fts[[30]]]

cv_xmat_transform_readmit <- cv_readmit_xmat %*% final_basis_readmit

cv_predictors_readmit <- cv_data_readmit %>% select(GENDER, Age, INSURANCE,
Yr1Readmit) %>% cbind(., cv_xmat_transform_readmit) %>% as.data.frame()

train_glm_readmit <- glm(Yr1Readmit ~ . , data=cv_predictors_readmit, family
= "binomial")
# train_glm_readmit %>% summary()
# confint(train_glm, parm = 1:7)


test_xmat_readmit <- holdout_test_readmit  %>%
select(matches("0|1|2|4|5|6|9")) %>% select(-Yr1Readmit) %>% as.matrix()
test_xmat_transform <- test_xmat_readmit %*% final_basis_readmit

test_predictors_readmit <- holdout_test_readmit %>% select(GENDER, Age,
INSURANCE, Yr1Readmit) %>% cbind(., test_xmat_transform) %>% as.data.frame()

phat_readmit <- predict(object = train_glm_readmit, newdata =
test_predictors_readmit, type="response")

brier_test <- sum((phat_readmit - test_predictors_readmit$Yr1Readmit)^2) /
nrow(test_predictors_readmit)
auc_test <- pROC::roc(test_predictors_readmit$Yr1Readmit, phat_readmit)$auc
brier_test
auc_test

# Exporting full model estimates
ci95s_readmit <- confint.default(train_glm_readmit) # Using Wald
approximation for confidence intervals, profile likelihood using confint()
from MASS takes minutes to run (when it isn't crashing my R session)

cbind(train_glm_readmit %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(Estimate) %>% round(2),
      CI=paste0("[", (ci95s_readmit %>% round(2))[,1], ", ", (ci95s_readmit
%>% round(2))[,2], "]"),
      train_glm_readmit %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(`Pr(>|z|)`) %>% round(4)) %>% write.table("clipboard")
```

Graphing P-values/Coefficients
```{r}
train_glm_readmit %>% summary() %>% .$coefficients %>% as.data.frame() %>%
mutate(Covariate=rownames(.), Order=row_number()) %>%
select(PValue=`Pr(>|z|)`, everything()) %>%
  filter(stringr::str_detect(Covariate, "[0-9]")) %>% arrange(desc(Estimate))
%>% mutate(Label = case_when(row_number()<=5 ~ str_replace_all(Covariate,
"`", ""),
```
```

```
TRUE ~ "")) %>% arrange(Order) %>%
  ggplot(aes(x=reorder(Covariate, Order), y=-log(PValue), fill=Estimate)) +
    geom_bar(stat="identity") + theme_minimal() +
  geom_text(aes(label=Label, group=Label),
            hjust=-0.9, vjust=0.95) +
  theme(axis.text.x = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16),
        legend.title = element_text(size=12.4),
        legend.text =  element_text(size=10),
        legend.position = c(0.9, 0.65)) +
  labs(caption="Inset text notes feature numbers of five highest
coefficients") +
  xlab ("Treelet Feature") + ylab("-log(P-Value)") +
  scale_fill_continuous(type="viridis", name="Coefficient", direction=-1)
```

```
##### Graphing Phat Distributions

```{r}
phat_readmit_df <- as.data.frame(cbind("EventProb" = phat_readmit,
"ObsOut"=test_predictors_readmit$Yr1Readmit))

phat_readmit_df %>% ggplot(aes(x=EventProb, fill=as.factor(ObsOut))) +
  geom_density(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Density") + xlab("Predicted Probability of Unplanned Hospital
Readmission")  +
  scale_fill_manual(name="Observed Outcome",
                    labels=c("No Readmission", "Readmission"),
                    values=c("lightblue", "violetred4")) +
  ggtitle("Density Curve of Predicted Probabilities of Unplanned Hospital
Readmission")


```

Fitting Model Without ICD Codes

```{r}
cv_predictors_readmit_noicd <- cv_predictors_readmit %>% select(-
matches("^[0-9]"))

train_glm_readmit_noicd <- glm(Yr1Readmit ~ . ,
data=cv_predictors_readmit_noicd, family = "binomial")

test_predictors_readmit_noicd <- holdout_test_readmit %>% select(GENDER, Age,
INSURANCE, Yr1Readmit)
```
```

```
phat_readmit_noicd <- predict(object = train_glm_readmit_noicd, newdata =
test_predictors_readmit_noicd, type="response")

brier_test_noicd <- sum((phat_readmit_noicd -
test_predictors_readmit_noicd$Yr1Readmit)^2) /
nrow(test_predictors_readmit_noicd)
auc_test_noicd <- pROC::roc(test_predictors_readmit_noicd$Yr1Readmit,
phat_readmit_noicd)$auc

# Exporting full model estimates
ci95s_readmit_noicd <- confint.default(train_glm_readmit_noicd) # Using Wald
approximation for confidence intervals, profile likelihood using confint()
from MASS takes minutes to run (when it isn't crashing my R session)

cbind(train_glm_readmit_noicd %>% summary() %>% .$coefficients %>%
as.data.frame() %>% select(Estimate) %>% round(3),
      CI=paste0("[", (ci95s_readmit_noicd %>% round(3))[,1], ", ",
(ci95s_readmit_noicd %>% round(3))[,2], "]"),
      train_glm_readmit_noicd %>% summary() %>% .$coefficients %>%
as.data.frame() %>% select(`Pr(>|z|)`) %>% round(4)) #%>%
write.table("clipboard")

```

```
##### Fitting with Most Significant Features
```{r}
# Pulling the five most significant
# train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
arrange(`Pr(>|z|)`) %>% tibble::rownames_to_column() %>%
filter(str_detect(rowname,"[0-9]"))

top5_tt_ftrs_readmit <- train_glm_readmit %>% summary() %>% .$coefficients
%>% as.data.frame() %>% arrange(`Pr(>|z|)`) %>%
  tibble::rownames_to_column() %>% filter(str_detect(rowname, "[0-9]")) %>%
filter(row_number()<=5) %>% pull(rowname) %>%
  str_replace_all("`", "")

top5_train_df_readmit <- cv_predictors_readmit %>% dplyr::select(GENDER, Age,
INSURANCE, Yr1Readmit, all_of(top5_tt_ftrs_readmit))

top5_glm_readmit <- glm(Yr1Readmit ~ ., data=top5_train_df_readmit,
family="binomial")

test_predictors_top5_readmit <- holdout_test_readmit %>% select(GENDER, Age,
INSURANCE, Yr1Readmit) %>% cbind(., test_xmat_transform) %>% as.data.frame()
%>%
  select(GENDER, Age, INSURANCE, Yr1Readmit, all_of(top5_tt_ftrs_readmit))

phat_top5_readmit <- predict(object = top5_glm_readmit, newdata =
test_predictors_top5_readmit, type="response")

brier_test_noicd <- sum((phat_top5_readmit -
test_predictors_top5_readmit$Yr1Readmit)^2) /
nrow(test_predictors_top5_readmit)
auc_test_noicd <- pROC::roc(test_predictors_top5_readmit$Yr1Readmit,
phat_top5_readmit)$auc
```
```

```

###### Retained ICD-9-CM Does in 5 Features
```{r}
top5_tt_ftrs_readmit_cols <- sapply(top5_tt_ftrs_readmit, function(x)
paste0("V", x))
names(top5_tt_ftrs_readmit_cols) <- NULL

loading_mat_readmit %>% select(!!top5_tt_ftrs_readmit_cols, code) %>%
  filter(V1!=0 | V2!=0 | V4!=0 | V10!=0 | V3!=0) %>% pull(code) %>% unique()
%>% length()
```


Model of ICD Codes (No treelet features)

```{r}
retained_codes <- loading_mat_readmit$code %>% unique()
length(retained_codes)

readmit_retain_traindf <- cv_data_readmit %>% select(GENDER, Age, INSURANCE,
Yr1Readmit, !!retained_codes)

readmit_retain_train_glm <- glm(Yr1Readmit ~ . , data=readmit_retain_traindf,
family = "binomial")

readmit_retain_test_df <- holdout_test_readmit %>% select(GENDER, Age,
INSURANCE, Yr1Readmit, !!retained_codes)
readmit_retain_phat <- predict(object = readmit_retain_train_glm, newdata =
readmit_retain_test_df, type="response")

sum((readmit_retain_phat - readmit_retain_test_df$Yr1Readmit)^2) /
nrow(readmit_retain_test_df)
  pROC::roc(readmit_retain_test_df$Yr1Readmit, readmit_retain_phat)$auc


# Logistic Regression of All Codes
all_traindf <- cv_data %>% select(GENDER, Age, INSURANCE, InHospMortality,
matches("[0-9]$"))
all_train_glm <- glm(InHospMortality ~ . , data=all_traindf, family =
"binomial")
all_test_df <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality, matches("[0-9]$"))
all_phat <- predict(object = all_train_glm, newdata = all_test_df,
type="response")

sum((all_phat - all_test_df$InHospMortality)^2) / nrow(retain_test_df)
pROC::roc(all_test_df$InHospMortality, all_phat)$auc

```

Comparative ROC

```{r, warning=F, message=F}

```
roc_obj_noicd_readmit <- pROC::roc(test_predictors_top5_readmit$Yr1Readmit,
phat_readmit_noicd)
roc_obj_readmit <- pROC::roc(test_predictors_top5_readmit$Yr1Readmit,
phat_readmit)
roc_top5_readmit <- pROC::roc(test_predictors_top5_readmit$Yr1Readmit,
phat_top5_readmit)

pROC::ggroc(list(roc_obj_readmit, roc_top5_readmit, roc_obj_noicd_readmit),
lwd=1.4) +
  theme_minimal() +
  xlab("Specificity") + ylab("Sensitivity") +
  theme(axis.text.x = element_blank(),
        # panel.grid.major = element_blank(),
        # panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16),
        legend.title = element_text(size=12.4),
        legend.text = element_text(size=10),
        legend.position = c(0.64, 0.2)) +
  scale_color_brewer(type="qual", palette=2,
                     name="Model",
                     labels = c("Including All ICD-9-CM Treelet Features
(AUC=0.661)",
                                "Including 5 Most Significant ICD-9-CM
Treelet Features (AUC = 0.658)",
                                "Excluding ICD-9-CM Treelet Features
(AUC=0.574)")) +
  ggtitle("Comparative ROC Curves of Hospital Re-admission Predictions in
Test Data")


```



##### Probability Distributions of No ICD Model

```{r}
phat_df_noicd <- as.data.frame(cbind("EventProb" = phat_noicd,
"ObsOut"=test_predictors_noicd$InHospMortality))

phat_df_noicd %>% ggplot(aes(x=EventProb, fill=as.factor(ObsOut))) +
  geom_density(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Density") + xlab("Predicted Probability of In-Hospital Mortality")  +
  scale_fill_manual(name="Observed Outcome",
                     labels=c("No Mortality", "Mortality Event"),
                     values=c("lightblue", "violetred4")) +
  ggtitle("Density Curve of Predicted Probabilities of In-Hospital
Mortality")
```

Length of Stay

Figures of Model Validation

```{r}
los_performance <-
read.csv(here("Results/Treelet_KLOpt_WithinCVLoop/LOSModel_MSE_DF_NewKLCode.csv"))
# los_performance <-
read.csv(here("Results/Treelet_KKOpt_WithinCVLoop/LOSModel_MSE_DF.csv"))

k_1sd_los <-
los_performance[los_performance$MSE_TestAvg<=(min(los_performance$MSE_TestAvg) + sd(los_performance$MSE_TestAvg)), ] %>% .[1,1]

los_performance <- los_performance %>% mutate(ParamFlag =
                                    case_when(
                                      MSE_TestAvg==min(MSE_TestAvg) ~
"Minimizes MSE",
                                      K==k_1sd_los ~ "More Sparse Parameter",
                                      TRUE ~ NA_character_
                                    )) %>% ungroup()


ggplot(los_performance, aes(x=K, y=MSE_TestAvg, color =
as.factor(ParamFlag))) +
  geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
  theme_minimal() + ggtitle("Hospital Length of Stay Model") +
  xlab("Value of Parameter K") + ylab("Average Mean-Squared Error\n(Across 5
Test Folds)") +
  gghighlight(ParamFlag!=0) + labs(color="Optimal Parameters") +
  scale_color_brewer(type = "qual", palette = 6) +
  theme(legend.position=c(0.75, 0.75), text = element_text(size=13.5))

```

Cluster Membership/Loading Export

```{r}
los_performance [!is.na(los_performance $ParamFlag),]

cv_los_xmat <- cv_data  %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
cv_los_cov <- cov(cv_los_xmat)

tt_fnc_los <- treelet_process(cv_los_xmat, cv_los_cov)

tt_fnc_los$optimal_params[c(46, 115),]
tt_fnc_los$retained_fts[[46]]

# Matrix of loadings
 # For our 1-standard deviation parameter, pulling K-features from the Lth
basis matrix
```

```
  final_basis_los <-
tt_fnc_los$basis_mats[[63]][,tt_fnc_los$retained_fts[[46]]] %>%
    as.data.frame() %>%
    mutate(LabelIndex = row_number(),
           RowMissCount = rowSums(.==0)) %>%
    filter(RowMissCount<46)

  labels_df <- cv_los_cov %>% colnames() %>%
    data.frame(code = ., label=1:ncol(cv_los_cov))

  loading_mat_los <- merge(final_basis_los, labels_df,
                                all.x=T, by.y="label", by.x="LabelIndex")


  holder <- sapply(2:(ncol(loading_mat_los)-2), function(x)
matrix(c(loading_mat_los[loading_mat_los[,x]!=0, "code"],

loading_mat_los[loading_mat_los[,x]!=0, x]),
                                                                    ncol=2))

  # Lazily using a for loop to transform to an exportable csv
  i = 1
  reformat_loadingmat_los <- as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2)

  for (i in 2:length(holder)){
  reformat_loadingmat_los <- rbind(reformat_loadingmat_los,
                                   as.data.frame(holder[[i]]) %>%
mutate(Feature=case_when(row_number()==1 ~ paste("Cluster", i),

TRUE ~ NA_character_)) %>% select(Feature, Code=V1, Loading=V2))
  }

  reformat_loadingmat_los_labs <- reformat_loadingmat_los %>%
mutate(Order=row_number()) %>%
    merge(diagnosis_labs %>% select(ICD9_CODE, SHORT_TITLE), by.x="Code",
by.y="ICD9_CODE", all.x=T) %>% arrange(Order) %>% select(-Order)

  length(unique(reformat_loadingmat_los$Code))

  write.csv(loading_mat_los,
            here("Results/LoadingMatrix_LOS.csv"))

  write.csv(reformat_loadingmat_los_labs,
            here("Results/LoadingMatrix_LOS_Redux.csv"), na = "")

# Number of non-zero features
  # More Sparse
    final_basis_los <-
tt_fnc_los$basis_mats[[63]][,tt_fnc_los$retained_fts[[46]]] %>%
    as.data.frame() %>%
    mutate(LabelIndex = row_number(),
           RowMissCount = rowSums(.==0)) %>%
    filter(RowMissCount<46)
  # "Optimal"
```

```
    opt_basis_los <-
tt_fnc_los$basis_mats[[63]][,tt_fnc_los$retained_fts[[115]]] %>%
    as.data.frame() %>%
    mutate(LabelIndex = row_number(),
           RowMissCount = rowSums(.==0)) %>%
    filter(RowMissCount<116)
```

```
#### Building Final Model and Assessting Test Fit

First fitting the Poisson model to assess overdispersion using the resulting
deviance and $\chi^2$ distribution.

```{r}
final_basis <- tt_fnc_los$basis_mats[[63]][,tt_fnc_los$retained_fts[[46]]]

cv_xmat_transform <- cv_los_xmat %*% final_basis


cv_predictors <- cv_data %>% select(GENDER, Age, INSURANCE, HospitalLOS) %>%
cbind(., cv_xmat_transform) %>% as.data.frame()

poisson_los <- glm(HospitalLOS ~ . , data=cv_predictors, family="poisson")
poisson_los$deviance / poisson_los$df.residual # Values near 1 indicate
evenly dispersed data (mean ~= variance), our value of 5.78 indicates
overdispersion (variance ~=5.8*mean)

pchisq(poisson_los$deviance, df=poisson_los$df.residual, lower.tail = F)
  # Unsurprisingly significant

```

Now fitting the negative binomial model, again using the 1-Standard Deviation
Parameter, on our the 80% cross-validation subset

```{r}
final_basis <- tt_fnc_los$basis_mats[[63]][,tt_fnc_los$retained_fts[[46]]]

cv_xmat_transform <- cv_los_xmat %*% final_basis

cv_predictors <- cv_data %>% select(GENDER, Age, INSURANCE, HospitalLOS) %>%
cbind(., cv_xmat_transform) %>% as.data.frame()

dim(cv_predictors)

train_glm_los <- glm.nb(HospitalLOS ~ . , data=cv_predictors)
# train_glm_los %>% summary()
# train_glm_los %>% str()

# confint(train_glm_los, parm = 1:7)
```
```

```
test_xmat <- holdout_test  %>% select(matches("0|1|2|4|5|6|9")) %>% select(-
Yr1Readmit) %>% as.matrix()
test_xmat_transform <- test_xmat %*% final_basis
test_predictors <- holdout_test %>% select(GENDER, Age, INSURANCE,
HospitalLOS) %>% cbind(., test_xmat_transform) %>% as.data.frame()

yhat <- predict(object = train_glm_los, newdata = test_predictors,
type="response")

(MSE <- sum((yhat - test_predictors$HospitalLOS)^2) / nrow(holdout_test)) %>%
sqrt()

# Exporting full model estimates
ci95s_los <- confint.default(train_glm_los) # Using Wald approximation for
confidence intervals, profile likelihood using confint() from MASS takes
minutes to run (when it isn't crashing my R session)

cbind(train_glm_los %>% summary() %>% .$coefficients %>% as.data.frame() %>%
select(Estimate) %>% round(2),
      CI=paste0("[", (ci95s_los %>% round(2))[,1], ", ", (ci95s_los %>%
round(2))[,2], "]"),
      train_glm_los %>% summary() %>% .$coefficients %>% as.data.frame() %>%
select(`Pr(>|z|)`) %>% round(4)) %>% write.table("clipboard")
```

```
##### Plotting residuals
```{r}
residuals <- yhat - test_predictors$HospitalLOS

resid_note <- deparse(bquote("Residuals = "* hat(y) * "-y"))
resid_note_2 <- "Residuals = Predicted - Observed"

data.frame(Residuals=residuals) %>% ggplot(aes(x=Residuals)) + geom_density()
+
  xlim(c(-40, 40)) + theme_minimal() + ylab("Density)") +
  # annotate(geom="text", x=-35, y=0.035, label=resid_note, parse=T) +
  annotate(geom="text", x=-21.8, y=0.08, label=resid_note_2)

qqnorm(residuals)
qqline(residuals)
```


Graphing P-Value/Coefficients
```{r}
train_glm_los %>% summary() %>% .$coefficients %>% as.data.frame() %>%
mutate(Covariate=rownames(.), Order=row_number()) %>%
select(PValue=`Pr(>|z|)`, everything()) %>%
  filter(stringr::str_detect(Covariate, "[0-9]")) %>% arrange(desc(Estimate))
%>% mutate(Label = case_when(row_number()<=5 ~ str_replace_all(Covariate,
"`", ""),

TRUE ~ "")) %>% arrange(Order) %>%
  ggplot(aes(x=reorder(Covariate, Order), y=-log(PValue), fill=Estimate)) +
    geom_bar(stat="identity") + theme_minimal() +
  geom_text(aes(label=Label, group=Label),
            hjust=-0.35, vjust=0.95) +
```
```

```
  theme(axis.text.x = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.background = element_blank(),
        axis.title.x = element_text(size=16),
        axis.title.y = element_text(size=16),
        legend.title = element_text(size=12.4),
        legend.text =  element_text(size=10),
        legend.position = c(0.9, 0.65)) +
  labs(caption="Inset text notes feature numbers of five highest
coefficients") +
  xlab ("Treelet Feature") + ylab("-log(P-Value)") +
  scale_fill_continuous(type="viridis", name="Coefficient", direction=-1)
```

LOS Density Curve
```{r}
# Total Cohort
cohort_full %>% ggplot(aes(x=HospitalLOS)) +
  geom_density(lwd=1.3) + theme_minimal() +
  xlab("Length of Stay (Days)") + ylab("Density") +
  xlim(c(0, 150)) +
  geom_text(label="Figure truncated at x=150 for legibility.\n n=12 patients
with values >150 days not included in this visual", x=100, y=0.015) +
  NULL

(cohort_full %>% arrange(desc(HospitalLOS)) %>% filter(HospitalLOS>=150) %>%
pull(HospitalLOS)) %>% length()

mean(cohort_full$HospitalLOS)
sd(cohort_full$HospitalLOS)

# # CV Training Cohort
# cv_data %>% ggplot(aes(x=HospitalLOS)) +
#   geom_density() + theme_minimal()
#
# mean(cv_data$HospitalLOS)
# sd(cv_data$HospitalLOS)

```


Diagnositics

```{r}
# Poisson
los_poisson <- glm(HospitalLOS ~ ., data=cv_predictors, family="poisson")
los_poisson %>% summary()

# Negative Binomial # using cv_predictors, the resulting `los_nb` object is
the same as the training glm fit earlier, simply renamed not to overwrite
that object
los_nb <- glm.nb(HospitalLOS ~ . , data=cv_predictors)
los_nb %>% summary()
```

```
1/train_glm_los$theta

llik_diff <- -2*(logLik(los_poisson) - logLik(los_nb))

pchisq(llik_diff, df=1, lower.tail = F)

summary(cohort_full$HospitalLOS)

pois_nb_comp <- data.frame(PoissonYhat = predict(object=los_poisson,
newdata=test_predictors),
                           NegBinYhat = predict(object=los_nb,
newdata=test_predictors))

pois_nb_comp %>% ggplot(aes(x=NegBinYhat, y=PoissonYhat)) +
  geom_point() + theme_minimal() + xlab("Negative Binomial Predictions") +
  ylab("Poisson Predictions")

```

Scatterplot of Observed vs Predictive

```{r}
yhat_df <- as.data.frame(cbind("PredictedLOS" = yhat,
"ObservedLOS"=test_predictors$HospitalLOS))


#
# inset <- yhat_df %>% ggplot(aes(x=ObservedLOS, y=PredictedLOS,
color=as.factor(PredictedLOS>ObservedLOS))) +
#   geom_point(alpha=0.3) + theme_minimal() +
#   theme(legend.position="none", text=element_text(size=13.5)) +
scale_color_manual(values=c("lightblue", "violetred4")) +
#   xlim(c(0, 100)) + ylim(c(0, 50)) +
#   # ylab("Predicted Length of Stay") + xlab("Observed Length of Stay")  +
#   # scale_color_manual(name="Prediction Error Direction",
#   #                    labels=c("Predicted LOS > Observed LOS", "Predicted
LOS < Observed LOS"),
#   #                    values=c("lightblue", "violetred4")) +
#   # ggtitle("Scatter Plot of Predicted and Observed Length of Stay Values")
#   NULL
#
#
# inset_tibble <- tibble(y=25, x=200,
#                        plot=list(inset))

yhat_df %>% ggplot(aes(x=ObservedLOS, y=PredictedLOS,
color=as.factor(PredictedLOS>ObservedLOS))) +
  geom_point(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Predicted Length of Stay") + xlab("Observed Length of Stay")  +
  scale_color_manual(name="Prediction Error Direction",
                     labels=c("Observed LOS > Predicted LOS", "Observed LOS
< Predicted LOS"),
                     values=c("dodgerblue", "violetred4")) +
  ggtitle("Scatter Plot of Predicted and Observed Length of Stay Values") +
  # geom_text(x=125, y=30, label="Correlation of Predicted and\nObserved
Length of Stay Values = 0.393") +
```

```
  NULL
```

```
##### Residuals and Number of Diagnoses

```{r}
num_diagnoses_df <- cbind(yhat_df,
                          NumDiagnoses = holdout_test %>%
select(matches("^[0-9]")) %>% rowSums()) %>%
                    mutate(Resid=PredictedLOS-ObservedLOS,
                           absResid = abs(PredictedLOS-ObservedLOS))

num_diagnoses_df %>% ggplot(aes(group=NumDiagnoses, y=Resid)) +
  geom_boxplot() + theme_minimal() + xlab("Number of ICD Diagnoses per
Patient") +
  ylab("Predicted LOS - Observed LOS")

num_diagnoses_df %>% ggplot(aes(x=NumDiagnoses, y=absResid)) +
  geom_point() + theme_minimal() + xlab("Number of ICD Diagnoses per
Patient") +
  ylab("|Predicted LOS - Observed LOS|")

num_diagnoses_df %>% ggplot(aes(x=ObservedLOS, y=Resid)) +
  geom_point() + theme_minimal() + xlab("Observed Length of Stay") +
  ylab("Predicted LOS - Observed LOS")

```
```

```
##### Distribution of Observed and Predicted LOS Values

```{r}
los_dens_df <- rbind(yhat_df %>% select(LOS=PredictedLOS) %>%
mutate(Type="Predicted"),
                     yhat_df %>% select(LOS=ObservedLOS) %>%
mutate(Type="Observed"))

los_dens_df %>% ggplot(aes(x=LOS, fill=as.factor(Type))) +
  geom_density(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Density") + xlab("Length of Stay Value (Days)")  +
  scale_fill_manual(name="Type of Data",
                    labels=c("Observed", "Predicted"),
                    values=c("lightblue", "violetred4")) +
  ggtitle("Density Curve of Predicted & Observed Length of Stay Values") +
  xlim(c(0, 75))


```
```

```
##### Fitting Model Without ICD Codes
```

```{r}
cv_predictors_noicd <- cv_predictors %>% select(-matches("[0-9]"))

train_glm_noicd <- glm.nb(HospitalLOS ~ . , data=cv_predictors_noicd)
#train_glm_noicd %>% summary()
# confint(train_glm, parm = 1:7)

test_predictors_noicd <- holdout_test %>% select(GENDER, Age, INSURANCE,
HospitalLOS)

yhat_noicd <- predict(object = train_glm_noicd, newdata =
test_predictors_noicd, type="response")

(MSE <- sum((yhat_noicd - test_predictors_noicd$HospitalLOS)^2) /
nrow(test_predictors_noicd)) %>% sqrt()

# Exporting full model estimates
ci95s_noicd <- confint.default(train_glm_noicd) # Using Wald approximation
for confidence intervals, profile likelihood using confint() from MASS takes
minutes to run (when it isn't crashing my R session)

cbind(train_glm_noicd %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(Estimate) %>% round(3),
      CI=paste0("[", (ci95s_noicd %>% round(3))[,1], ", ", (ci95s_noicd %>%
round(3))[,2], "]"),
      train_glm_noicd %>% summary() %>% .$coefficients %>% as.data.frame()
%>% select(`Pr(>|z|)`) %>% round(4)) #%>% write.table("clipboard")

```

Fitting with Most Significant Features
```{r}
# Pulling the five most significant
# train_glm %>% summary() %>% .$coefficients %>% as.data.frame() %>%
arrange(`Pr(>|z|)`) %>% tibble::rownames_to_column() %>%
filter(str_detect(rowname,"[0-9]"))

top_tt_ftrs_los <- train_glm_los %>% summary() %>% .$coefficients %>%
as.data.frame() %>% arrange(`Pr(>|z|)`) %>%
  tibble::rownames_to_column() %>% filter(str_detect(rowname, "[0-9]")) %>%
pull(rowname) %>%
  str_replace_all("`", "")

for (i in 1:46){
  los_subdf <- cv_predictors %>% select(GENDER, Age, INSURANCE, HospitalLOS,
all_of(top_tt_ftrs_los[1:i]))
  sub_los_glm <- glm.nb(HospitalLOS ~ ., data=los_subdf)

  los_test_subdf <- holdout_test %>% select(GENDER, Age, INSURANCE,
HospitalLOS) %>% cbind(., test_xmat_transform) %>% as.data.frame() %>%
  select(GENDER, Age, INSURANCE, HospitalLOS, all_of(top_tt_ftrs_los[1:i]))

  yhat_sub <- predict(object = sub_los_glm, newdata = los_test_subdf,
type="response")

  RMSE_sub <- (MSE <- sum((yhat_sub - los_test_subdf$HospitalLOS)^2) /
nrow(los_test_subdf)) %>% sqrt()
```

```
  if(i ==1) RMSE_vec <- RMSE_sub else{
  RMSE_vec <- c(RMSE_vec, RMSE_sub)
  }

}

data.frame(RMSE = RMSE_vec, FeaturesRetained = 1:46)  %>%
  ggplot(aes(x=FeaturesRetained, y=RMSE_vec)) +
  geom_point() + geom_line() +
  theme_minimal() +
  ylab("Root MSE") + xlab("Number of Treelet Features Retained")


```

Retained ICD-9-CM Does in 5 Features
```{r}
top_tt_ftrs_los_cols <- sapply(top_tt_ftrs_los, function(x) paste0("V", x))
%>% .[1:5]
names(top_tt_ftrs_los_cols) <- NULL

loading_mat_los %>% select(!!top_tt_ftrs_los_cols, code) %>%
  filter(V1!=0 | V12!=0 | V14!=0 | V20!=0 | V15!=0) %>% pull(code) %>%
unique() %>% length()
```


Model of ICD Codes (No Treelet Features)

```{r}
retained_codes <- loading_mat_los$code %>% unique()
length(retained_codes)
retain_traindf <- cv_data %>% select(GENDER, Age, INSURANCE, HospitalLOS,
!!retained_codes)
retain_train_glm <- glm.nb(HospitalLOS ~ . , data=retain_traindf)
retain_test_df <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality, !!retained_codes)
retain_yhat <- predict(object = retain_train_glm, newdata = retain_test_df,
type="response")

(sum((retain_yhat - retain_test_df$InHospMortality)^2) /
nrow(retain_test_df)) %>% sqrt()

# And trying all codes
retainall_traindf <- cv_data %>% select(GENDER, Age, INSURANCE, HospitalLOS,
matches("^[0-9]"))
retainall_train_glm <- glm.nb(HospitalLOS ~ . , data=retainall_traindf)
retainall_test_df <- holdout_test %>% select(GENDER, Age, INSURANCE,
InHospMortality, matches("^[0-9]"))
retainall_yhat <- predict(object = retainall_train_glm, newdata =
retainall_test_df, type="response")

(sum((retainall_yhat - retainall_test_df$InHospMortality)^2) /
nrow(retainall_test_df)) %>% sqrt()
```

```

##### Prediction Scatter Plot of No ICD Model

```{r}
yhat_noicd_df <- as.data.frame(cbind("PredictedLOS" = yhat_noicd,
"ObservedLOS"=test_predictors_noicd$HospitalLOS))


yhat_noicd_df %>% ggplot(aes(x=ObservedLOS, y=PredictedLOS,
color=as.factor(PredictedLOS>ObservedLOS))) +
  geom_point(alpha=0.3) + theme_minimal() +
  theme(legend.position=c(0.7, 0.6), text=element_text(size=13.5)) +
  ylab("Predicted Length of Stay") + xlab("Observed Length of Stay")  +
  scale_color_manual(name="Prediction Error Direction",
                     labels=c("Predicted LOS > Observed LOS", "Predicted LOS
< Observed LOS"),
                     values=c("dodgerblue", "violetred4")) +
  ggtitle("Scatter Plot of Predicted and Observed Length of Stay Values") +
  # geom_text(x=125, y=30, label="Correlation of Predicted and\nObserved
Length of Stay Values = 0.393") +
  NULL
```

Appendix Analysis: Comparative Models

Exploratory analysis to see how the results of the treelet modelling above compares with the application of PCA, lasso, and possibly the use of the Charlson and/or Elixhauser comorbidity indexes as a predictor

```{r}
require(caret)

cv5 <- caret::trainControl(method="cv",
                     number=5)

cv_data %>% head()
cv_data_readmit %>% head()
```

Mortality

LASSO

```{r}
lasso_mortality <-  caret::train(as.factor(InHospMortality) ~ .,
                   data = cv_data %>% select(matches("^[0-9]"),
InHospMortality, Age, GENDER, INSURANCE),
                   method="glmnet",
                   metric="AUC",
```
```

```
                  trControl=cv5)

phat_lasso <- predict(object = lasso_mortality, newdata = holdout_test,
type="prob")

((lasso_mortality$finalModel %>%
coef(lasso_mortality$bestTune$lambda))[,1]!=0) %>% sum()
(lasso_mortality$finalModel %>% coef(lasso_mortality$bestTune$lambda))[,1]
%>% .[.==0]
  # Uses 176 of 184 covariates (excludes 250.00, 780.39, 274.9, 714.0, 585.9,
441.2, 491.21, 785.0, and Private Insurance)

(lasso_auc_mortality <- pROC::auc(holdout_test$InHospMortality,
phat_lasso[,1]) %>% round(4))

lasso_mortality$results
```

PCA

```{r}
pca_results <- prcomp(cv_data %>% select(matches("^[0-9]")), center = T,
scale. = T)

(pca_mortality_df <- data.frame(PC = 1:178,
                          Var = pca_results$sdev^2) %>%
              mutate(PropVar = Var / nrow(.),
                     CmltvPropVar = cumsum(PropVar)))


pca_mortality_df %>% ggplot(aes(x=PC, y=PropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Proportion of Variance Explained") + xlab("Principal Component") +
  ggtitle("Proportion of Variance Explained by Individual Principal
Component")


pca_mortality_df %>% ggplot(aes(x=PC, y=CmltvPropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Cumulative Proportion of Variance Explained") + xlab("Principal
Component") +
  ggtitle("Cumulative Proportion of Variance Explained by Principal
Component")



n_retain <- pca_mortality_df %>% filter(CmltvPropVar<=0.5) %>% nrow()

rotate_icd <- (cv_data %>% select(matches("^[0-9]")) %>% as.matrix())  %*%
pca_results$rotation[,1:n_retain]


pca_glm <- glm(InHospMortality ~ . ,
               data = cv_data %>% select(InHospMortality, Age, GENDER,
INSURANCE) %>% cbind(., rotate_icd),
               family="binomial")
```

```
test_rotate <- (holdout_test %>% select(matches("^[0-9]")) %>% as.matrix())
%*%  pca_results$rotation[,1:n_retain]


test_pcadf <- holdout_test %>% select(InHospMortality, Age, GENDER,
INSURANCE) %>% cbind(., test_rotate)

test_pca_phat <- predict(newdata = test_pcadf, object=pca_glm,
type="response")

(pca_auc_mortality <- pROC::auc(predict = test_pca_phat, response =
holdout_test$InHospMortality) %>% round(4))

```


Charlson & Elixhauser

Re-importing ICD data (to include all Charlson & Elixhauser codes)

```{r}
all_diags <- read.csv(here("/Data/Raw/DIAGNOSES_ICD.csv"))

icd_train <- cv_data %>% select(SUBJECT_ID) %>% merge(., all_diags %>%
select(SUBJECT_ID, ICD9_CODE), by="SUBJECT_ID")
icd_test <- holdout_test %>% select(SUBJECT_ID) %>% merge(., all_diags %>%
select(SUBJECT_ID, ICD9_CODE), by="SUBJECT_ID")

# Using Charlson/Elixhauser group membership
  mortality_charlson_train <- icd_train %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., cv_data %>% select(InHospMortality, Age, GENDER,
INSURANCE))
  mortality_elix_train <- icd_train %>% comorbid_elix() %>% as.data.frame()
%>% cbind(., cv_data %>% select(InHospMortality, Age, GENDER, INSURANCE))

  mortality_charlson_test <- icd_test %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., holdout_test %>% select(InHospMortality, Age,
GENDER, INSURANCE))
  mortality_elix_test <- icd_test %>% comorbid_elix() %>% as.data.frame() %>%
cbind(., holdout_test %>% select(InHospMortality, Age, GENDER, INSURANCE))

  charlson_glm <- glm(InHospMortality ~ ., data=mortality_charlson_train,
family = "binomial")
  elix_glm <- glm(InHospMortality ~ ., data=mortality_elix_train, family =
"binomial")

  elix_phat <- predict(object = elix_glm, newdata = mortality_elix_test)
  elix_auc <- pROC::auc(predict = elix_phat,
response=holdout_test$InHospMortality) %>% round(4)

  charlson_phat <- predict(object = charlson_glm, newdata =
mortality_charlson_test)
  charlson_auc <- pROC::auc(predict = charlson_phat,
response=holdout_test$InHospMortality)[1] %>% round(4)
```

```
# Using "score" (sum of group memberships, i.e. number of groups with a
diagnosis)
  charlson_score_train <- icd_train %>% comorbid_charlson() %>%
as.data.frame() %>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., cv_data %>%
select(InHospMortality, Age, GENDER, INSURANCE))
  elix_score_train <- icd_train %>% comorbid_elix() %>% as.data.frame() %>%
mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., cv_data %>%
select(InHospMortality, Age, GENDER, INSURANCE))

  charlson_score_test <- icd_test %>% comorbid_charlson() %>% as.data.frame()
%>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., holdout_test %>%
select(InHospMortality, Age, GENDER, INSURANCE))
  elix_score_test <- icd_test %>% comorbid_elix() %>% as.data.frame() %>%
mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., holdout_test %>%
select(InHospMortality, Age, GENDER, INSURANCE))

  charlson_score_glm <- glm(InHospMortality ~ ., data=charlson_score_train,
family = "binomial")
  elix_score_glm <- glm(InHospMortality ~ ., data=elix_score_train, family =
"binomial")

  elix_score_phat <- predict(object = elix_score_glm, newdata =
elix_score_test)
  elix_score_auc <- pROC::auc(predict = elix_score_phat,
response=holdout_test$InHospMortality) %>% round(4)

  charlson_score_phat <- predict(object = charlson_score_glm, newdata =
charlson_score_test)
  charlson_score_auc <- pROC::auc(predict = charlson_score_phat,
response=holdout_test$InHospMortality)[1] %>% round(4)
```


```{r}
# Printout Results
cat("Elixhauser Categorical AUC: ", elix_auc, "\n")
cat("Charlson Categorical AUC: ", charlson_auc, "\n")
cat("Elixhauser Score AUC: ", elix_score_auc, "\n")
cat("Charlson Score AUC: ", charlson_score_auc, "\n")
cat("PCA AUC (retaining", n_retain, "PC's):", pca_auc_mortality, "\n")
cat("LASSO AUC:", lasso_auc_mortality, "\n")

```


Readmission

LASSO

```{r}
glmnet_readmit <-  train(as.factor(Yr1Readmit) ~ .,
```

```
                  data = cv_data_readmit %>% select(matches("^[0-9]"),
Yr1Readmit, Age, GENDER, INSURANCE),
                  method="glmnet",
                  metric="AUC",
                  trControl=cv5)


phat_readmit <- predict(object = glmnet_readmit, newdata =
holdout_test_readmit, type="prob")

(retained_fts_lasso <- ((glmnet_readmit$finalModel %>%
coef(glmnet_readmit$bestTune$lambda))[,1]!=0) %>% sum())
  # Uses only 48 of 184 covariates (excludes 250.00, 780.39, 274.9, 714.0,
585.9, 441.2, 491.21, 785.0, and Private Insurance)

(lasso_auc_readmit <- pROC::auc(holdout_test_readmit$Yr1Readmit,
phat_readmit[,1]) %>% round(4))
```

PCA

```{r}
pca_readmit <- prcomp(cv_data_readmit %>% select(matches("^[0-9]")), center =
T, scale. = T)

(pca_readmit_df <- data.frame(PC = 1:178,
                         Var = pca_readmit$sdev^2) %>%
              mutate(PropVar = Var / nrow(.),
                     CmltvPropVar = cumsum(PropVar)))


pca_readmit_df %>% ggplot(aes(x=PC, y=PropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Proportion of Variance Explained") + xlab("Principal Component") +
  ggtitle("Proportion of Variance Explained by Individual Principal
Component")


pca_readmit_df %>% ggplot(aes(x=PC, y=CmltvPropVar)) +
  geom_point(size=5, alpha=0.4) + geom_line(lwd=0.75) + theme_minimal() +
  ylab("Cumulative Proportion of Variance Explained") + xlab("Principal
Component") +
  ggtitle("Cumulative Proportion of Variance Explained by Principal
Component")


n_retain <- pca_readmit_df %>% filter(CmltvPropVar<=0.5) %>% nrow()

rotate_readmit <- (cv_data_readmit %>% select(matches("^[0-9]")) %>%
as.matrix())  %*%  pca_readmit$rotation[,1:n_retain]


pca_readmit_glm <- glm(Yr1Readmit ~ . ,
               data = cv_data_readmit %>% select(Yr1Readmit, Age, GENDER,
INSURANCE) %>% cbind(., rotate_readmit),
               family="binomial")
```

```
test_rotate_readmit <- (holdout_test_readmit %>% select(matches("^[0-9]"))
%>% as.matrix())  %*%  pca_readmit$rotation[,1:n_retain]

test_pca_readmitdf <- holdout_test_readmit %>% select(Yr1Readmit, Age,
GENDER, INSURANCE) %>% cbind(., test_rotate_readmit)

test_pca_phat_readmit <- predict(newdata = test_pca_readmitdf,
object=pca_readmit_glm, type="response")

(pca_auc_readmit <- pROC::auc(predict = test_pca_phat_readmit, response =
holdout_test_readmit$Yr1Readmit) %>% round(4))
```

Charlson & Elixhauser

```{r}
all_diags <- read.csv(here("/Data/Raw/DIAGNOSES_ICD.csv"))

icd_train_readmit <- cv_data_readmit %>% select(SUBJECT_ID) %>% merge(.,
all_diags %>% select(SUBJECT_ID, ICD9_CODE), by="SUBJECT_ID")
icd_test_readmit <- holdout_test_readmit %>% select(SUBJECT_ID) %>% merge(.,
all_diags %>% select(SUBJECT_ID, ICD9_CODE), by="SUBJECT_ID")

# Using Charlson/Elixhauser group membership
  readmit_charlson_train <- icd_train_readmit %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., cv_data_readmit %>% select(Yr1Readmit, Age,
GENDER, INSURANCE))
  readmit_elix_train <- icd_train_readmit %>% comorbid_elix() %>%
as.data.frame() %>% cbind(., cv_data_readmit %>% select(Yr1Readmit, Age,
GENDER, INSURANCE))

  readmit_charlson_test <- icd_test_readmit %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., holdout_test_readmit %>% select(Yr1Readmit, Age,
GENDER, INSURANCE))
  readmit_elix_test <- icd_test_readmit %>% comorbid_elix() %>%
as.data.frame() %>% cbind(., holdout_test_readmit %>% select(Yr1Readmit, Age,
GENDER, INSURANCE))

  charlson_glm_readmit <- glm(Yr1Readmit ~ ., data=readmit_charlson_train,
family = "binomial")
  elix_glm_readmit <- glm(Yr1Readmit ~ ., data=readmit_elix_train, family =
"binomial")

  elix_phat_readmit <- predict(object = elix_glm_readmit, newdata =
readmit_elix_test)
  elix_auc_readmit <- pROC::auc(predict = elix_phat_readmit,
response=holdout_test_readmit$Yr1Readmit) %>% round(4)

  charlson_phat_readmit <- predict(object = charlson_glm_readmit, newdata =
readmit_charlson_test)
  charlson_auc_readmit <- pROC::auc(predict = charlson_phat_readmit,
response=holdout_test_readmit$Yr1Readmit)[1] %>% round(4)
```

```
# Using "score" (sum of group memberships, i.e. number of groups with a
diagnosis)
  charlson_score_train_readmit <- icd_train_readmit %>% comorbid_charlson()
%>% as.data.frame() %>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., cv_data_readmit %>%
select(Yr1Readmit, Age, GENDER, INSURANCE))
  elix_score_train_readmit <- icd_train_readmit %>% comorbid_elix() %>%
as.data.frame() %>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., cv_data_readmit %>%
select(Yr1Readmit, Age, GENDER, INSURANCE))

  charlson_score_test_readmit <- icd_test_readmit %>% comorbid_charlson() %>%
as.data.frame() %>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., holdout_test_readmit
%>% select(Yr1Readmit, Age, GENDER, INSURANCE))
  elix_score_test_readmit <- icd_test_readmit %>% comorbid_elix() %>%
as.data.frame() %>% mutate(score = rowSums(.)) %>% select(score) %>%
                                                cbind(., holdout_test_readmit
%>% select(Yr1Readmit, Age, GENDER, INSURANCE))

  charlson_score_glm_readmit <- glm(Yr1Readmit ~ .,
data=charlson_score_train_readmit, family = "binomial")
  elix_score_glm_readmit <- glm(Yr1Readmit ~ .,
data=elix_score_train_readmit, family = "binomial")

  elix_score_phat_readmit <- predict(object = elix_score_glm_readmit, newdata
= elix_score_test_readmit)
  elix_score_auc_readmit <- pROC::auc(predict = elix_score_phat_readmit,
response=holdout_test_readmit$Yr1Readmit) %>% round(4)

  charlson_score_phat_readmit <- predict(object = charlson_score_glm_readmit,
newdata = charlson_score_test_readmit)
  charlson_score_auc_readmit <- pROC::auc(predict =
charlson_score_phat_readmit, response=holdout_test_readmit$Yr1Readmit)[1] %>%
round(4)
```



```{r}

# Printout Results
cat("Elixhauser Categorical AUC: ", elix_auc_readmit, "\n")
cat("Charlson Categorical AUC: ", charlson_auc_readmit, "\n")
cat("Elixhauser Score AUC: ", elix_score_auc_readmit, "\n")
cat("Charlson Score AUC: ", charlson_score_auc_readmit, "\n")
cat("PCA AUC (retaining ", n_retain, " components):", pca_auc_readmit, "\n")
cat("LASSO AUC (retaining", retained_fts_lasso, "features):",
lasso_auc_readmit, "\n")
```


Length of Stay

Due to the length of stay and mortality data sets having the same training data/cross-validation splits, I can simply re-use the PCA and

Charlson/Elixhauser data used in the Mortality section and fit the negative binomial models

LASSO

```{r}
require(mpath)

los_traindf <- cv_data %>% select(matches("^[0-9]"), HospitalLOS, Age,
GENDER, INSURANCE)

lasso_train_results <- glmregNB(formula = HospitalLOS ~ ., data =
los_traindf)

los_train_yhat <- predict(object = lasso_train_results, los_traindf,
type="response")

for(i in 1:ncol(los_train_yhat)) {
  yhat_vec <- los_train_yhat[,i]

  if(i==1) RMSE <- (sum((yhat_vec - los_traindf$HospitalLOS)^2) /
length(yhat_vec)) %>% sqrt() else{
    RMSE <- c(RMSE, (sum((yhat_vec - los_traindf$HospitalLOS)^2) /
length(yhat_vec)) %>% sqrt())
  }
}


lambda_results <- data.frame(lambda=lasso_train_results$lambda,
                             RMSE)

lambda_results <- lambda_results %>% mutate(ParamFlag =
                                  case_when(
                                    RMSE==min(RMSE) ~ "Minimizes RMSE",

lambda==max(lambda_results[lambda_results$RMSE<=(min(lambda_results$RMSE) +
sd(lambda_results$RMSE)), "lambda"]) ~ "More Sparse Parameter",
                                    TRUE ~ NA_character_
                                  )) %>% ungroup()

ggplot(lambda_results, aes(x=lambda, y=RMSE, color=as.factor(ParamFlag))) +
  geom_line(lwd=1.1, alpha=0.6) + geom_point(size=2.5) +
  theme_minimal() + ggtitle("Length of Stay LASSO") +
  xlab("Value of Shrinkage Lambda") + ylab("RMSE (Across 5 Test Folds)") +
  gghighlight(ParamFlag!=0) + labs(color="Optimal Parameters") +
  scale_color_brewer(type = "qual", palette = 6) +
  theme(legend.position=c(0.2, 0.75), text = element_text(size=13.5))


lambda_1sd <- lambda_results %>% filter(ParamFlag == "More Sparse Parameter")
%>% pull(lambda)

los_test_yhat <- predict(object = lasso_train_results, holdout_test,
type="response")

lambda_1sd_test <-
colnames(los_test_yhat)[which.min(abs(colnames(los_test_yhat) %>%
as.numeric() - lambda_1sd))]
```

```
test_rmse <- (sum((los_test_yhat[, lambda_1sd_test] -
holdout_test$HospitalLOS)^2)/nrow(holdout_test)) %>% sqrt()

# And using the optimal lambda
lambda_1sd_opt <- lambda_results %>% filter(ParamFlag == "Minimizes RMSE")
%>% pull(lambda)

lambda_1sd_test_opt <-
colnames(los_test_yhat)[which.min(abs(colnames(los_test_yhat) %>%
as.numeric() - lambda_1sd_opt))]


test_rmse <- (sum((los_test_yhat[, lambda_1sd_test_opt] -
holdout_test$HospitalLOS)^2)/nrow(holdout_test)) %>% sqrt()


# Outputting number of features retained in teh 1-SD lambda
(retained_fts_los_lasso <- lasso_train_results %>% coef(lambda_1sd))

retained_fts_los_lasso %>% nrow()
```

PCA

```{r}
pca_nb <- glm.nb(HospitalLOS ~ . ,
                 data = cv_data %>% select(HospitalLOS, Age, GENDER, INSURANCE)
%>% cbind(., rotate_icd))


n_retain <- pca_readmit_df %>% filter(CmltvPropVar<=0.5) %>% nrow()


test_rotate <- (holdout_test %>% select(matches("^[0-9]")) %>% as.matrix())
%*%  pca_results$rotation[,1:n_retain]

test_los_pcadf <- holdout_test %>% select(HospitalLOS, Age, GENDER,
INSURANCE) %>% cbind(., test_rotate)

test_yhat_pca <- predict(newdata = test_los_pcadf, object=pca_nb,
type="response")

(pca_rmse_los <- (sum((test_yhat_pca -
test_los_pcadf$HospitalLOS)^2)/nrow(test_los_pcadf)) %>% sqrt())

```


Charlson & Elixhauser

```{r}
mortality_charlson_train <- icd_train %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., cv_data %>% select(HospitalLOS, Age, GENDER,
INSURANCE))
```

```
mortality_elix_train <- icd_train %>% comorbid_elix() %>% as.data.frame() %>%
cbind(., cv_data %>% select(HospitalLOS, Age, GENDER, INSURANCE))

mortality_charlson_test <- icd_test %>% comorbid_charlson() %>%
as.data.frame() %>% cbind(., holdout_test %>% select(HospitalLOS, Age,
GENDER, INSURANCE))
mortality_elix_test <- icd_test %>% comorbid_elix() %>% as.data.frame() %>%
cbind(., holdout_test %>% select(HospitalLOS, Age, GENDER, INSURANCE))

charlson_nb <- glm.nb(HospitalLOS ~ ., data=mortality_charlson_train)
elix_nb <- glm.nb(HospitalLOS ~ ., data=mortality_elix_train)

elix_yhat <- predict(object = elix_nb, newdata = mortality_elix_test)
charlson_yhat <- predict(object = charlson_nb, newdata =
mortality_charlson_test)


(charlson_rmse_los <- (sum((charlson_yhat -
mortality_charlson_test$HospitalLOS)^2)/nrow(mortality_charlson_test)) %>%
sqrt() )
(elix_rmse_los <- (sum((elix_yhat -
mortality_elix_test$HospitalLOS)^2)/nrow(mortality_elix_test)) %>% sqrt() )

```

Bibliography

Alder, L., & Tambe, A. (2020). Acute Anemia. In *StatPearls*. StatPearls Publishing. http://www.ncbi.nlm.nih.gov/books/NBK537232/

Awad, A., Bader–El–Den, M., & McNicholas, J. (2017). Patient length of stay and mortality prediction: A survey. *Health Services Management Research*, *30*(2), 105–120. https://doi.org/10.1177/0951484817696212

Awad, A., Bader-El-Den, M., McNicholas, J., & Briggs, J. (2017). Early hospital mortality prediction of intensive care unit patients using an ensemble learning approach. *International Journal of Medical Informatics*, *108*, 185–195. https://doi.org/10.1016/j.ijmedinf.2017.10.002

Brier, G. W. (1950). Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, *78*(1), 1–3.

Brito, A., Costantini, T. W., Berndtson, A. E., Smith, A., Doucet, J. J., & Godat, L. N. (2019). Readmissions After Acute Hospitalization for Traumatic Brain Injury. *The Journal of Surgical Research*, *244*, 332–337. https://doi.org/10.1016/j.jss.2019.06.071

Charlson, M. E., Pompei, P., Ales, K. L., & MacKenzie, C. R. (1987). A new method of classifying prognostic comorbidity in longitudinal studies: Development and validation. *Journal of Chronic Diseases*, *40*(5), 373–383. https://doi.org/10.1016/0021-9681(87)90171-8

CMS. (n.d.). *Hospital Readmissions Reduction Program (HRRP) Overview*. Retrieved October 28, 2020, from https://www.qualitynet.org/inpatient/hrrp

CMS. (2020, April 14). *ICD-9-CM Diagnosis and Procedure Codes: Abbreviated and Full Code Titles*. https://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes

Dash, S., Shakyawar, S. K., Sharma, M., & Kaushik, S. (2019). Big data in healthcare: Management, analysis and future prospects. *Journal of Big Data*, *6*(1), 54. https://doi.org/10.1186/s40537-019-0217-0

Depalma, G., Xu, H., Covinsky, K. E., Craig, B. A., Stallard, E., Thomas, J., & Sands, L. P. (2013). Hospital readmission among older adults who return home with unmet need for ADL disability. *The Gerontologist*, *53*(3), 454–461. https://doi.org/10.1093/geront/gns103

Elixhauser, A., Steiner, C., Harris, D. R., & Coffey, R. M. (1998). Comorbidity measures for use with administrative data. *Medical Care*, *36*(1), 8–27. https://doi.org/10.1097/00005650-199801000-00004

Falcão, A. L. E., Barros, A. G. de A., Bezerra, A. A. M., Ferreira, N. L., Logato, C. M., Silva, F. P., do Monte, A. B. F. O., Tonella, R. M., de Figueiredo, L. C., Moreno, R., Dragosavac, D., & Andreollo, N. A. (2019). The prognostic accuracy evaluation of SAPS 3, SOFA and APACHE II scores for mortality prediction in the surgical ICU: An external validation study and decision-making analysis. *Annals of Intensive Care*, *9*(1), 18. https://doi.org/10.1186/s13613-019-0488-9

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, *33*(1), 1–22.

Gaugler, D. L. and T. (2015). *treelet: An Adaptive Multi-Scale Basis for High-Dimensional, Sparse and Unordered Data* (1.1) [Computer software]. https://CRAN.R-project.org/package=treelet

Goodwin, A. J., & Ford, D. W. (2018). Readmissions Among Sepsis Survivors: Risk Factors and Prevention. *Clinical Pulmonary Medicine*, *25*(3), 79–83. https://doi.org/10.1097/CPM.0000000000000254

Greysen, S. R., Stijacic Cenzer, I., Auerbach, A. D., & Covinsky, K. E. (2015). Functional impairment and hospital readmission in Medicare seniors. *JAMA Internal Medicine*, *175*(4), 559–565. https://doi.org/10.1001/jamainternmed.2014.7756

Harrell, F. E. (2001). *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis* (Updated September 4, 2020). Springer Science & Business Media.

Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edition). Springer.

Hastie, T., Tibshirani, R., & Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC.

Kansagara, D., Englander, H., Salanitro, A., Kagen, D., Theobald, C., Freeman, M., & Kripalani, S. (2011). Risk Prediction Models for Hospital Readmission: A Systematic Review. *JAMA*, *306*(15), 1688. https://doi.org/10.1001/jama.2011.1515

Kennedy, E. H., Wiitala, W. L., Hayward, R. A., & Sussman, J. B. (2013). Improved cardiovascular risk prediction using nonparametric regression and electronic health record data. *Medical Care*, *51*(3), 251–258. https://doi.org/10.1097/MLR.0b013e31827da594

Knaus, W. A., Draper, E. A., Wagner, D. P., & Zimmerman, J. E. (1985). APACHE II: A severity of disease classification system. *Critical Care Medicine*, *13*(10), 818–829.

Kuhn, M. (2020). *caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret

Kumar, R. G., Juengst, S. B., Wang, Z., Dams-O'Connor, K., Dikmen, S. S., O'Neil-Pirozzi, T. M., Dahdah, M. N., Hammond, F. M., Felix, E. R., Arenth, P. M., & Wagner, A. K. (2018). Epidemiology of Comorbid Conditions Among Adults 50 Years and Older With Traumatic Brain Injury. *The Journal of Head Trauma Rehabilitation*, *33*(1), 15–24. https://doi.org/10.1097/HTR.0000000000000273

Lares, B. (2020). *lares: Analytics, Data Mining & Machine Learning Sidekick* (4.9.2) [Computer software]. https://github.com/laresbernardo/lares

Lee, A. B., Nadler, B., & Wasserman, L. (2008). Treelets—An adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, *2*(2), 435–471. https://doi.org/10.1214/07-AOAS137

Lee, K., & Rincon, F. (2012, October 23). *Pulmonary Complications in Patients with Severe Brain Injury* [Review Article]. Critical Care Research and Practice; Hindawi. https://doi.org/10.1155/2012/207247

Luckey, A., Livingston, E., & Taché, Y. (2003). Mechanisms and treatment of postoperative ileus. *Archives of Surgery (Chicago, Ill.: 1960)*, *138*(2), 206–214. https://doi.org/10.1001/archsurg.138.2.206

MacVane, S. H., Tuttle, L. O., & Nicolau, D. P. (2015). Demography and burden of care associated with patients readmitted for urinary tract infection. *Journal of Microbiology, Immunology, and Infection = Wei Mian Yu Gan Ran Za Zhi*, *48*(5), 517–524. https://doi.org/10.1016/j.jmii.2014.04.002

Mayr, F. B., Talisa, V. B., Balakumar, V., Chang, C.-C. H., Fine, M., & Yende, S. (2017). Proportion and Cost of Unplanned 30-Day Readmissions After Sepsis Compared With Other Medical Conditions. *JAMA*, *317*(5), 530. https://doi.org/10.1001/jama.2016.20468

McCredie, V. A., Alali, A. S., Scales, D. C., Rubenfeld, G. D., Cuthbertson, B. H., & Nathens, A. B. (2018). Impact of ICU Structure and Processes of Care on Outcomes After Severe Traumatic Brain Injury: A Multicenter Cohort Study. *Critical Care Medicine*, *46*(7), 1139–1149. https://doi.org/10.1097/CCM.0000000000003149

Middleton, J. W., Lim, K., Taylor, L., Soden, R., & Rutkowski, S. (2004). Patterns of morbidity and rehospitalisation following spinal cord injury. *Spinal Cord*, *42*(6), 359–367. https://doi.org/10.1038/sj.sc.3101601

Muller, K. (2017). *here: A Simpler Way to Find Your Files* (0.1) [Computer software]. https://CRAN.R-project.org/package=here

Nasir, S. S., Muthiah, M., Ryder, K., Clark, K., Niell, H., & Weir, A. (2017). ICU Deaths in Patients With Advanced Cancer. *The American Journal of Hospice & Palliative Care*, *34*(2), 173–179. https://doi.org/10.1177/1049909115625279

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized Linear Models. *Journal of the Royal Statistical Society: Series A (General), 135*(3), 370–384. https://doi.org/10.2307/2344614

Ostling, S., Wyckoff, J., Ciarkowski, S. L., Pai, C.-W., Choe, H. M., Bahl, V., & Gianchandani, R. (2017). The relationship between diabetes mellitus and 30-day readmission rates. *Clinical Diabetes and Endocrinology*, *3*(1), 3. https://doi.org/10.1186/s40842-016-0040-x

Pah, A. R., Rasmussen-Torvik, L. J., Goel, S., Greenland, P., & Kho, A. N. (2014). Big Data: What Is It and What Does It Mean for Cardiovascular Research and Prevention Policy. *Current Cardiovascular Risk Reports*, *9*(1), 424. https://doi.org/10.1007/s12170-014-0424-3

Paoli, C. J., Reynolds, M. A., Sinha, M., Gitlin, M., & Crouser, E. (2018). Epidemiology and Costs of Sepsis in the United States—An Analysis Based on Timing of Diagnosis and Severity Level*. *Critical Care Medicine*, *46*(12), 1889–1897. https://doi.org/10.1097/CCM.0000000000003342

Quach, S., Hennessy, D. A., Faris, P., Fong, A., Quan, H., & Doig, C. (2009). A comparison between the APACHE II and Charlson Index Score for predicting hospital mortality in critically ill patients. *BMC Health Services Research*, *9*(1), 129. https://doi.org/10.1186/1472-6963-9-129

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C., & Müller, M. (2011). PROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics*, *12*, 77.

Rubenfeld, G. D., Caldwell, E., Peabody, E., Weaver, J., Martin, D. P., Neff, M., Stern, E. J., & Hudson, L. D. (2005). Incidence and outcomes of acute lung injury. *The New England Journal of Medicine*, *353*(16), 1685–1693. https://doi.org/10.1056/NEJMoa050333

Rufibach, K. (2010). Use of Brier score to assess binary predictions. *Journal of Clinical Epidemiology*, *63*(8), 938–939. https://doi.org/10.1016/j.jclinepi.2009.11.009

Shao, J. (1993). Linear Model Selection by Cross-validation. *Journal of the American Statistical Association*, *88*(422), 486–494. https://doi.org/10.1080/01621459.1993.10476299

Shebl, E., & Gulick, P. G. (2020). Nosocomial Pneumonia. In *StatPearls*. StatPearls Publishing. http://www.ncbi.nlm.nih.gov/books/NBK535441/

Snow, G. L., Bledsoe, J. R., Butler, A., Wilson, E. L., Rea, S., Majercik, S., Anderson, J. L., & Horne, B. D. (2020). Comparative evaluation of the clinical laboratory-based Intermountain risk score with the Charlson and Elixhauser comorbidity indices for mortality prediction. *PLOS ONE*, *15*(5), e0233495. https://doi.org/10.1371/journal.pone.0233495

Steyerberg, E. (2009). *Clinical Prediction Models: A Practical Approach to Development, Validation, and Updating*. Springer-Verlag. https://doi.org/10.1007/978-0-387-77244-8

Tapper, E. B., Halbert, B., & Mellinger, J. (2016). Rates of and Reasons for Hospital Readmissions in Patients With Cirrhosis: A Multistate Population-based Cohort Study. *Clinical Gastroenterology and Hepatology: The Official Clinical Practice Journal of the American Gastroenterological Association*, *14*(8), 1181-1188.e2. https://doi.org/10.1016/j.cgh.2016.04.009

Topaz, M., Shafran-Topaz, L., & Bowles, K. H. (2013). ICD-9 to ICD-10: Evolution, Revolution, and Current Debates in the United States. *Perspectives in Health Information Management / AHIMA, American Health Information Management Association*, *10*(Spring). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3692324/

Venables, W. N., & Ripley, B. D. (2002). *Modern Applied Statistics with S* (Fourth Edition). Springer. http://www.stats.ox.ac.uk/pub/MASS4

Wang, Z. (2020). *mpath: Regularized Linear Models* (0.4-2.16) [Computer software]. https://CRAN.R-project.org/package=mpath

Wasey, J. O., Murphy, W., Lee, E., Odisho, A., & R Core Team. (2020). *icd: Comorbidity Calculations and Tools for ICD-9 and ICD-10 Codes* (4.0.9) [Computer software]. https://github.com/jackwasey/icd

Wei, T., & Simko, V. (2017). *R package "corrplot": Visualization of a Correlation Matrix* (0.84) [Computer software]. https://github.com/taiyun/corrplot

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., Francois, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Muller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686.

Yutani, H. (2020). *gghighlight: Highlight Lines and Points in "ggplot2"* (0.3.0) [Computer software]. https://CRAN.R-project.org/package=gghighlight