University of Southern Maine
USM Digital Commons

All Theses & Dissertations
Student Scholarship

5-1-2014

Discovering Predictors of Readmission for Acute Myocardial Infarction in a Medicare Population: A Data Mining Approach

Daniel MacDonald Knowles MS
University of Southern Maine

Follow this and additional works at: https://digitalcommons.usm.maine.edu/etd

Part of the Statistics and Probability Commons

Recommended Citation
Knowles, Daniel MacDonald MS, "Discovering Predictors of Readmission for Acute Myocardial Infarction in a Medicare Population: A Data Mining Approach" (2014). *All Theses & Dissertations.* 99.
https://digitalcommons.usm.maine.edu/etd/99

This Open Access Thesis is brought to you for free and open access by the Student Scholarship at USM Digital Commons. It has been accepted for inclusion in All Theses & Dissertations by an authorized administrator of USM Digital Commons. For more information, please contact jessica.c.hovey@maine.edu.

Discovering Predictors of Readmission for Acute Myocardial Infarction in a Medicare Population

- A Data Mining Approach

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Master of Science in Statistics

UNIVERSITY OF SOUTHERN MAINE

Graduate Program in Statistics

BY

Daniel MacDonald Knowles

2014

THE UNIVERSITY OF SOUTHERN MAINE

MASTER OF SCIENCE IN STATISTICS

May 1, 2014

We hereby recommend that the thesis of Daniel MacDonald Knowles entitled

Discovering Predictors of Readmission for Acute Myocardial Infarction in a Medicare Population - A Data Mining Approach

be accepted in partial fulfillment of the requirements for the

Degree of Master of Science in Statistics

Cheng Peng

Cheng Peng

M. Eltaha

Muhammad El-Taha

[illegible]

AbouEl-Makarim Aboueissa

Accepted [illegible]

Dean, College of Sciences, Technology and Health

Acknowledgements

The author wishes to thank the Department of Mathematics and Statistics at the University of Southern Maine as well as Maine Medical Partners for their ongoing support during the development of this work. A special thanks to Dr. Cheng Peng, thesis advisor and Mr. Michael Yunes, MPH. for their support and commitment to see this effort through.

Thanks are extended to the other Committee members Dr. Muhammad El-Taha, graduate advisor and Dr. AbouEl-Makarim Aboueissa as well as Dr. Laurie Woodman, department chair and undergraduate advisor.

There were individuals who acted as mentors and contributed information and/or elements that were very beneficial to the objectives. The list is incomplete but includes; Mr. Michael Yunes, Dr. Tae-Ryong Park, Ms. Joanne Cederna, B.S. and Mr. John DiPalazzo, M.A. Thanks are also extended to reviewers Mr. Richard Shryock and Ms. Christine Berg, B.A., B.S. Without these individuals this work would not have been completed. Thanks Gail for always being there.

In memory of Joanne Cederna

Abstract

Health care costs in the United States have risen at rates far exceeding the cost of living for many years. Previous attempts to control these costs have proven futile. Studies have shown that high per-capita spending in the U.S. does not equate to consistent quality of care or better outcomes. National legislative action in the form of the Affordable Care Act (ACA) has been enacted to attempt yet again to control these spiraling costs. A segment of the ACA instructs the Centers for Medicare and Medicaid Services (CMS) to reduce reimbursements to hospitals experiencing higher than normal 30-day risk adjusted readmission rates for Medicare Beneficiaries initially admitted for incidence of Acute Myocardial Infarction (AMI).

We use this setting as motivation to develop predictive models that may aid our subject hospital in identifying predictors and stratifying an AMI population as to risk of readmission within 30 days of discharge. Since our primary objective is one of classification, we utilize logistic regression and classification decision trees as model methodologies to arrive at a 'best' model to predict readmission.

Our results, though interesting, show fair predictive ability at best. Our chosen logistic regression model has a prediction success rate of 63% while the decision tree had a prediction success rate of 86% (but had very poor predictability for the portion of our cohort experiencing the outcome of interest). It is critical that future work include use of Electronic Medical Records (EMR) data from which we can gain prior hospitalization utilization, problem lists, medication lists, trending lab results, and recent ambulatory activity. The ongoing challenge is selecting data points from an available list that far exceeds in number our study cohort size. Given that

challenge we may investigate more deeply principal component analysis as well as the application of machine learning to our primary objective; prediction of readmission within 30 days.

Table of Contents

Acknowledgements iii

Abstract iv

List of Figures viii

List of Tables viii

Chapter 1 – Research Background and Literature Review 1

1.1 Overview 1
1.2 Research Objectives 3
1.3 Literature Review 4
1.4 Research Methods and Outline 5

Chapter 2 – Study Design and Data Preparation 8

2.1 Study Design and Data Collection 8
2.2 Data Screening and Variable Selection 10
2.2.1 Dimension Reduction 10
2.2.2 Feature Additions 11
2.2.3 Data Profile 12

Chapter 3 – Predictive Models – A Data Mining Perspective 14

3.1 Overview of Classification Models 14
3.1.1 Traditional Probabilistic Classifiers 14
3.1.2 Non-probabilistic Data Mining Classifiers 15
3.1.3 The Classifiers Used in this Research 19
3.2 Logistic Regression Models 19
3.2.1 Model Building 19
3.2.2 Model Selection 23
3.2.3 Modeling Validation and Assessment 24
3.3 Decision Tree Models 25
3.3.1 Model Building Strategy – Growing then Pruning 25
3.3.2 Measure of Performance 31

Chapter 4 – Cross Validation 34

4.1 Motivation of CV in This Study 34
4.2 Common Types of Cross Validations and Implementation 35
4.3 Model Selection with k-fold Cross Validation 36

Chapter 5 – Data Analysis and Results 41

5.1 The Analytic File 41
5.2 Logistic Regression Models 45
5.2.1 Building Models Based on Automatic Variable Selection 45
5.2.2 Scoring using Cross Validation 48
5.2.3 Selecting Final Logistic Model via 5-fold Cross Validation 49
5.3 Decision Tree Model 53
5.4 Interpretation of the Two Models 55

Chapter 6 – Conclusions and Discussion 59

6.1 Conclusions and Comparison of the Final Models 59
6.2 Discussion 64

Appendix A – SAS Code Listings 67

Appendix B – R Code Listings 125

Appendix C – Acronyms used in the Paper 129

Notes 130

References 131

List of Figures

1 Sample decision tree diagram 16

2 Splitting classes linearly with maximum margins separating the classes 18

3 Toy example calculating Gini index based on a binary response. 27

4 Toy example of a fully grown tree. 30

5 Toy example of a pruned tree. 30

6 An example of an ROC Curve for two classes. 33

7 Cross validation holding out first fold. 38

8 Cross validation holding out second fold. 38

9 Cross validation holding out last fold. 39

10 Forward selection successful prediction rates for certain cut points 51

11 Backward selection successful prediction rates for certain cut points 52

12 Stepwise selection successful prediction rates for certain cut points 52

13 Classification Decision Tree for Training Sample.. 54

14 Receiver Operating Characteristic for Final Candidate Models...................... 63

List of Tables

1 Gini index scores for our final training-sample based decision tree 28

2 Sample confusion matrix 31

3 Incidence of medical fact/setting for both outcome cohorts............................ 42

4 Statistics for continuous variables for both outcome cohorts 43

5 Distribution and readmission rate by Gender 43

6 Cohort splits for various samples and their outcome distributions................. 44

7 Full model candidate.. 45

8 Forward selection candidate 46

9 Backward Elimination selection candidate 47

10 Stepwise selection candidate... 47

11 Cross Validation results for Logistic Regression model candidates 49

12 Model fit statistics for all Logistic Regression model candidates 50

13 Odds ratio estimates for chosen Logistic Regression model......................... 56

14 Odds ratios of the predictors from chosen Logistic Regression model.......... 60

15 Variables retained in Logistic variable selection models 61

16 Confusion matrix for the Forward Selection candidate model 62

17 Confusion matrix for the CART candidate model 62

18 Targeted effects for further investigation.. 65

Chapter 1

Research Background and Literature Review

1.1 Overview

In 2010, the U.S. spent $8,246 per person on health care which equivalently was 17.7% of the U.S. economy being devoted to health care. Health care spending is consuming an increasing share of economic activity over time and has exceeded economic growth in every recent decade, though rate of increase in national health spending has declined. The share of economic activity (gross domestic product, or GDP) devoted to health care has increased from 7.2% in 1970 ($255.8 billion) to 17.7% ($2.6 trillion) in 2010 and public expenditure on health was 47.6% of the overall 2010 expenditures (OECD 2013). Projected GDP share is 19.8% ($4.6 trillion) by 2020 (CMS 2009). The World Health Report 2000, Health Systems: Improving Performance, ranked the U.S. health care system 37th in the world (WHO 2010). These continuously rising health care costs are unsustainable (CDC 2011). Addressing this growing cost burden continues to be a major policy priority.

Implementation of the ACA in March 2010 is designed to slow these cost increases in premiums and health claims as well as spread the risk of coverage by extending health insurance coverage to the uninsured and the uninsurable (estimated at ~50 million citizens). The Act also directs CMS to develop programs to encourage health organizations to improve health outcomes and therefore reduce costs of care. Of particular interest to this work is a section of the Act titled the Hospital Readmissions

Reduction program. The Social Security Act establishing the Hospital Readmissions Reduction Program directs CMS to reduce payments to hospitals with excess 30 day readmission rates for specific conditions involving the Medicare Beneficiary population effective for discharges beginning on October 1, 2012 (CMS 2013a).

CMS chose 30 days for their readmission program over longer time periods such as 90 days "because readmissions over longer periods may be impacted by factors outside hospitals' control such as other complicating illnesses, patients' own behavior, or care provided to patients after discharge" (IMA Consulting, 2013).

So we have this landscape in healthcare today. Insurance premiums and the cost of delivering health care continue to climb. At the same time meanwhile, we are the country with the highest per capita spending on health care which does not equate to the highest quality levels of care and outcomes as compared to other industrialized nations (Murray et al. 2010). There is strong evidence that better outcomes lead to reduced costs (CMS 2013b). The national debate on these issues has now developed into rules for reimbursement for better outcomes.

AMI is a high-risk condition and a common cause of admission to hospitals in the United States. The percentage of patients who are readmitted within 30 days of an initial admission for AMI is significant because it is a basis for financial reimbursements and quality ratings by CMS. It is believed that lower readmission rates for an AMI index is an indicator of better health care and patient outcomes. We refer to the initial admission due to AMI as the index case. Note that readmissions can occur at another hospital, therefore note the key role the readmission data from

CMS plays in this work. Additionally, note there are very few exceptions allowed by CMS when excluding an encounter as a readmission. Scheduled (planned) admissions are not considered readmissions by CMS.

1.2 Research Objectives

Our primary objective is to develop one or more predictive models of readmission within 30 days of discharge for an index admission of AMI. The purpose of the predictive models is to identify explanatory variables that may help identify those patients at higher risk of readmission. This would not only aid our index hospital with developing targeted care for these patients to help reduce rates of readmission and improve health outcomes, but also help the index hospital avoid reduced payments for services from CMS for those AMI patients who are fee for service Medicare Beneficiaries. To the best of author's knowledge, there is no study completed to date at the subject hospital to identify predictors of readmission for AMI index cases in a Medicare population.

A secondary objective is to compare predictive model methodologies to see what predictors are discovered as well as compare model performance. Does a statistical model outperform a data mining learning model or vice versa in a classification problem such as ours? Can predictors unique to each model assist our overall objective?

Another objective is to present our results to clinical staff for review and critique. We look to them for clinical interpretation of individual explanatory variables produced by our models. Our hope is this initial work will prompt additional investigation with their guidance.

1.3 Literature Review

Our search and that of others identifies very few published works that identify predictors of readmission in an AMI population.

For some predictive modeling work that has been done in this area see Berkman and Abrams 1986, as well as Maynard, Every and Weaver 1997. We pursue this research objective because "... currently no models exist for measuring readmission rates for hospitals or modeling the risk of readmission for an individual." (Desai et al. 2009). A Poisson regression model (or negative binomial regression model) can be used to model readmission rates. Modeling the risk of readmission is the theme of this research. Desai's work was a systematic review of statistical models and patient predictors of readmission for AMI. Most of the studies (35) examined by Desai's review used medical records (chart reviews) or patient interviews as sources of data. Additionally, these studies were narrowly focused on specific cardiovascular issues within the AMI topic, other comorbidities, or medication modalities. The exploratory study of Berkman and Maynard focused on 30 elderly patients hospitalized with AMI. They identified two psychological factors as significantly related to early readmission: mental status and post discharge stressful events. The work of Maynard, Every and Weaver found predictors of rehospitalization included history of angina pectoris, previous AMI, congestive heart failure, hypertension, or coronary artery surgery. Protective variables included patients receiving thrombolytic therapy, coronary angiography, or coronary artery bypass surgery during the index hospitalization.

1.4 Research Methods and Outline

Our primary objective was to predict a readmission event. Our retrospective study collected electronic admission and readmission data for fee for service Medicare Beneficiary patients ages 65+. The index admission data was collected from a 600+ licensed-bed tertiary care and teaching institution. The readmission data was collected from CMS for both readmitted and non-readmitted patients. We limited the scope of data collection to that which was available at the time of the index admission. Due to the sheer number of variables available as predictors for this cohort we applied dimension reduction techniques to avoid the issue of potential multicollinearity in statistical modeling and the overfitting problem in decision tree modeling and therefore increase the capability of generalization of our candidate model to a new sample. To reduce the available variables we applied non-variant reduction as well as groupers to roll up related variables. We also removed variables that were grossly insignificant in a univariate model setting. We developed new variables derived from available variables to indicate overall disease burden as well as severity of primary diagnosis. The sample was split into three subsamples for the purpose of training, validating, and testing the candidate models. We used a data mining approach to build both logistic regression models and algorithm based decision trees models to the data. Comparison of model candidates was done using non-statistical methods, k-fold cross validation. The predictive power of candidate models was used to assess the performance of the models. The results of the final fitted model presented here were derived by applying candidate models to the testing sample.

Chapter 2 covers study design and data preparation in detail. A review of available variables and their quality is given with an emphasis on the actual data profile. It presents the chosen processes for dimension reduction and grouping as well as deriving new variables to build a more complete patient profile. The chapter finishes with a descriptive list of candidate variables.

Chapter 3 provides an overview of traditional statistical classifiers as well as some of the data mining classifiers. It then provides an in-depth explanation of logistic regression and decision tree models used in our study. The detailed explanations include a mathematical framework, the actual model selection algorithm used, and traditional performance evaluation techniques for both modeling methodologies.

Chapter 4 provides an overview of the concept of cross validation, a non-statistical method to evaluate and compare candidate models. It describes the motivation selecting this method given the special design of this study. This overview covers the key steps of selecting data for the validation step, the role of the hold out sample, and comparing the performance of two or more different types of models through aggregate measures produced by the validation process.

Chapter 5 presents results from critical stages of the model development. These stages include candidate logistic and decision tree models, their variable/feature selection, and their respective performance assessment. A review of model performance and interpretation of the overall predictive power of the final candidates are presented in detail.

Chapter 6 completes the work with conclusions and a discussion. Explanatory variables from the final candidate models are reviewed. Interpretation of the overall results is discussed in detail. Discussion includes issues encountered and shortcomings as well as planned next steps. Suggested future investigation using these results as a baseline is also discussed.

Chapter 2

Study Design and Data Preparation

2.1 Study Design and Data Collection

This is a retrospective study using administrative data of fee-for-service Medicare Beneficiaries age 65+. Target patients are admitted to the hospital for the primary diagnosis of AMI. The AMI admission is referred to as the index admission and the hospital treating the AMI is referred to as the index hospital. The patients must be Medicare beneficiaries for a minimum of 12 months prior to the index admission. Index admissions lasting more than a year, having same day discharges, or actual total charges of $0 are rejected from this work. Further, index admissions where the patient is discharged to another hospital, discharged against medical advice (AMA) or expired are also rejected. Readmissions due to scheduled procedures including Percutaneous Transluminal Coronary Angioplasty (PTCA) and Coronary Artery Bypass Grafting (CABG) are not considered as qualifying unscheduled readmissions and therefore are not used. The above are admission/readmission specifications as defined by CMS. This design uses statistical and data mining modeling techniques to identify key predictors of readmission. It compares model performances using techniques applicable to both families of modeling paradigms.

Our sample, containing 1,049 patients, was treated for AMI during the period July 1, 2009 through June 30, 2012. The index data are administrative data: data compiled for billing purposes. All index admission data are provided by a 600+ licensed-bed tertiary care and teaching institution, the index hospital. Readmission data for these index admissions are supplied by CMS to the index hospital. Patient data from both sources are required to be matched in order for the patient data to be included in this work. All data from both sources is electronic in nature. No manual chart reviews or patient surveys are used.

EMR systems provide a more comprehensive view of a patient's medical history and current medical status. However, EMR data is only available for those patients who are part of the index hospital's practice network. Given that possibility it is quite feasible that some of the patients being admitted to the index hospital may not have any medical history on the index hospital's database. To avoid the possibility of missing data for a portion of our study population we do not utilize EMR-supplied data for this study. We narrow what data would be accepted for this study by only including data collected (historical information, new diagnoses, and procedures performed) during the index admission. The initial pre-processing identifies 1,005 unique diagnosis codes and 155 unique procedure codes for 1,049 patients. The front end data collection process is considered dependable therefore we assume the data is in good condition.

2.2 Data Screening and Variable Selection

2.2.1 Dimension Reduction

Variables are removed from the cohort data that have no variability. For instance, we may have a variable titled admitting subservice that is always set to a single value; it never varies. Since this constant value is not informative we remove admitting subservice from our data. Certain features are manually removed due to lack of information from the feature. An example would be the Medical Record Number. Digital identifying information is not informative. The number of unique secondary diagnoses (N=1,005) and procedures (N=155) for the cohort represents a feature count larger than our study sample. Given the cohort size, dimension reduction is necessary for any possibility of meaningful results from the modeling phase of the work. To that end we investigate available tools to cluster diagnoses and procedures into a manageable number of clinically meaningful categories. Since the index admission diagnoses and procedures are coded using International Classification of Diseases, 9th Revision, Clinical Modification (ICD-9-CM) (NCHS), we choose the Clinical Classification Software (CCS) (HCUP 2009) as our grouping tool. This tool provides a reduced set of clinically meaningful categories of the diagnoses (N=198) and procedures (N=65). The CCS tool also provides yet higher level groupings of these codes for potential further reduction. After investigation it was deemed the higher level groupings were unusable due to the very broad categorization of these groupers. Such broad categorization would hinder any clinically meaningful interpretation of model results.

The Medical Severity Diagnosis Related Grouper (MS-DRG) (Fetter 1976) is included in our data in an attempt to measure the severity level of the patient's condition at index admission. The historical intent of this grouper was for hospital reimbursement. The groupers were originally developed as an expectation of hospital resource use (products provided). We utilize this grouper as a proxy for severity of condition believing that groupers indicating more expensive costs represent higher patient medical severity requiring more expensive procedures at index admission.

Further feature reduction of the MS-DRG and newly defined CCS groupers is performed by eliminating the MS-DRG and CCS groupers underrepresented in the population (< 5%). We chose 5% incidence due to sample size and low percent of patients with the target outcome (~13%).
To this point we have used non-statistical screening methods to reduce features.

2.2.2 Feature Additions

In addition to the variables in the database, we also defined a few more new clinically meaningful variables. These feature additions are: weekday/weekend admit/discharge, daytime/nighttime admit/discharge, discharge disposition (hospice, other care facility, home), and admitting medical subservice. Also missing from our data is a measure of the overall medical condition (total disease burden) of the patient at index admission. We believe that a patient with a chronic overall health condition could have an increased risk of being readmitted within 30 days. Our search for such a measure identifies the Elixhauser comorbidity system (Elixhauser 1998) translated into a single numeric score (van Walraven 2009) that summarizes disease burden. We calculate that score by utilizing the secondary diagnoses of the index admission, diagnoses present on admission, of the study cohort. The result is a single score for the Elixhauser system for each patient (Quan et al. 2005). A higher score represents a higher disease burden. Scores from the algorithm can be negative. All patients have data identified by the Elixhauser system therefore producing a valid score for all; a score of 0 is valid in this case.

The Elixhauser system is intended to predict hospital charges, length of stay, and in-hospital mortality and is developed by identifying comorbidities relevant to hospitalization other than the primary reason for hospitalization and the severity of that condition. As such, the Elixhauser system explicitly excludes important causes of substantial comorbidity; chiefly some of the most common causes of hospitalization and burden of comorbidity in elderly patients, including myocardial infarction and stroke.

Additional work by van Walraven modifies the Elixhauser comorbidity system into a single numeric score that summarizes disease burden. A SAS$^{©}$ program develops the groupers by diagnoses (Turner, Burchill 2006). We leverage that SAS code and integrate the points system by van Walraven associated with each Elixhauser disease group.

2.2.3 Data Profile

We borrow the term data profile here and alter it to mean the major medical features represented by our data after reductions and additions noted above that constitute the patient's medical profile.

After feature additions and dimension reductions the data have key features we want to investigate via predictive modeling. There are patient demographics, index admission features, severity of primary diagnosis, overall disease burden, preexisting CCS diagnosis groupers, and CCS procedure groupers performed during the index admission.

Patient demographics include age, gender and marital status. Index admission features include discharge disposition, admit/discharge day of the week, weekend admit/discharge, day admit/discharge and admitting medical subservice. MS-DRG codes represent the severity of the AMI. Elixhauser total score is used as a proxy for overall disease burden. CCS groupers are used to represent the index procedures and secondary diagnoses, diagnoses known upon admission. An indication of 30 day readmission was supplied by CMS and is our outcome variable. Key elements missing from our profile include prior near term hospital utilization, severity level of other comorbidities, and current medication list.

Chapter 3

Predictive Models – A Data Mining Perspective

3.1 Overview of Classification Models

The outcome variable in this study is categorical in nature with just two values (binary); the patient will be readmitted within 30 days or will not readmit within 30 days. Our primary objective is to develop a predictive model using explanatory feature variables based on statistical and data mining approaches. In a bigger picture, the classification is one of the fundamental problems in pattern recognition. In the machine learning field, classifiers are also called supervised learning algorithms. What follows is an overview of some of the commonly used classifiers in practice.

3.1.1 Traditional Probabilistic Classifiers

Traditional probabilistic classifiers use statistical inference to find the best class for a given observation/instance. There are two ways that can be used to define various classifiers: discriminative and generative. In the discriminative approach, an explicit discriminative function is defined using a regression model in which all effort is placed on defining the overall discriminant function with no consideration for the class-conditional densities which form the discriminant. Most commonly used classifiers in this family are linear and non-linear discriminant function models and logistic discriminant methods. Fisher's two group linear discriminant function may be the earliest statistical classifier while the binary logistic regression may be the most

popular classifier in practice. Multiple-classification can be done by simply extending the binary logistic model with a little more computational effort. In this study, we use binary logistic regression to predict the likelihood of readmission.
In the generative approach, the focus is on estimating the class-condition densities (distributions if the features are discrete) P(x|C=1) and P(x|C=0) and then use these estimates to define posterior probability, hence, the discriminant functions. A simple but popular such classifier is naïve Bayes classifier, One advantage of this type model is that it is simple and intuitive and easy to deal with multi-classification problem.

3.1.2 Non-probabilistic Data Mining Classifiers

In contrast to probabilistic classifiers that may require extensive numerical computation in order to find the explicit discriminant function, non-probabilistic classifiers categorize instances (data points) on the basis of either non-numerical criteria or quantitative measures that can be easily extracted with extensive computations. Non-probabilistic classifiers are also roughly categorized into individual and ensemble algorithms. Ensemble methods use multiple individual classifiers to obtain better predictive performance than could be obtained from any of the individual ones with lower predictive power. Commonly used ensemble classifiers include boosting classifiers (e.g., AdaBoost), bootstrap aggregation classifiers and random forest classifier. Next we provide a brief description of a few individual classifiers.

One of the most popular approaches for predicting and presenting classifiers is Decision Trees. In our specific application we use a decision tree algorithm titled

Classification and Regression Trees (CART) (Breiman et al. 1984). The decision tree methodology is member of the Machine Learning field. A decision tree is a classifier expressed as a recursive partition of the instance space. A decision tree consists of nodes that form a rooted tree, a directed tree with a root node. The root node has no incoming edges.

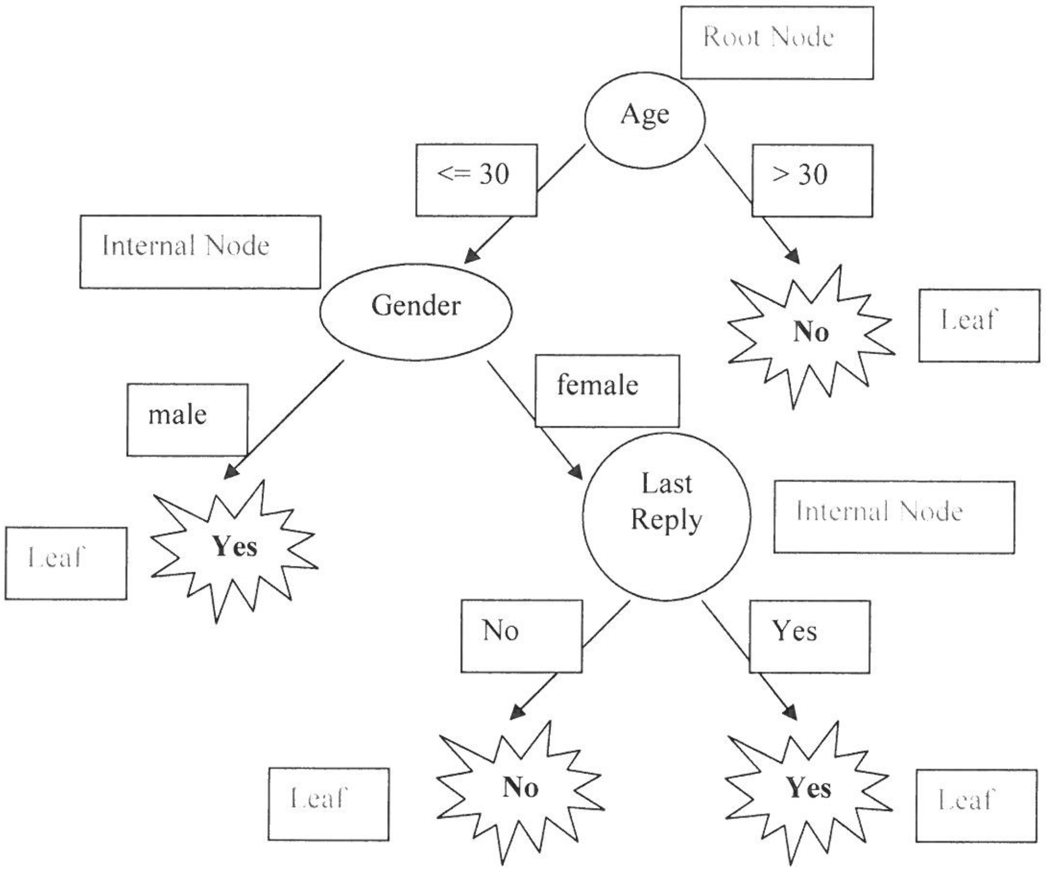


Figure 1: Sample decision tree diagram

The tree consists of nodes, edges and leafs. All nodes other than the root node have one incoming edge. If a node has outgoing edges it is called an internal node otherwise it is a terminal node or leaf. Each leaf is assigned to one class representing the most appropriate target. See Figure 1 for an example.

In our application we direct the observation to a leaf that indicates readmission or not readmission (the class). Observations are classified by navigating them from the root down to a leaf, according to the outcome of the tests along the path. The construction of a tree revolves around: the selection of the splits, when to declare a node terminal or continue splitting, and assigning each terminal node to a class. Trees that become too complex may be thinned via a pruning process. We can convert the final decision tree into a rule induction. In our work we review the final tree and define rules in SAS to predict the classifiers for a given cohort, the test sample.

Next we outline the Support Vector Machines (Cortes and Vapnik 1995) classification algorithm. Support Vector Machines is a classification method that operates on the principle of margin maximization; constructing an optimal separating hyperplane (decision boundary) to correctly classify as much of the population as possible. See Figure 2 (http://en.wikipedia.org/wiki/Support_vector_machine.) which represents a 2 dimensional representation of splitting the 2 groups linearly with maximum margins separating the 2 groups. The maximum margin is sought so that the model of the labeling process generalizes well for future unseen data. The 2 dimensional representations generalize well to high dimensional space.

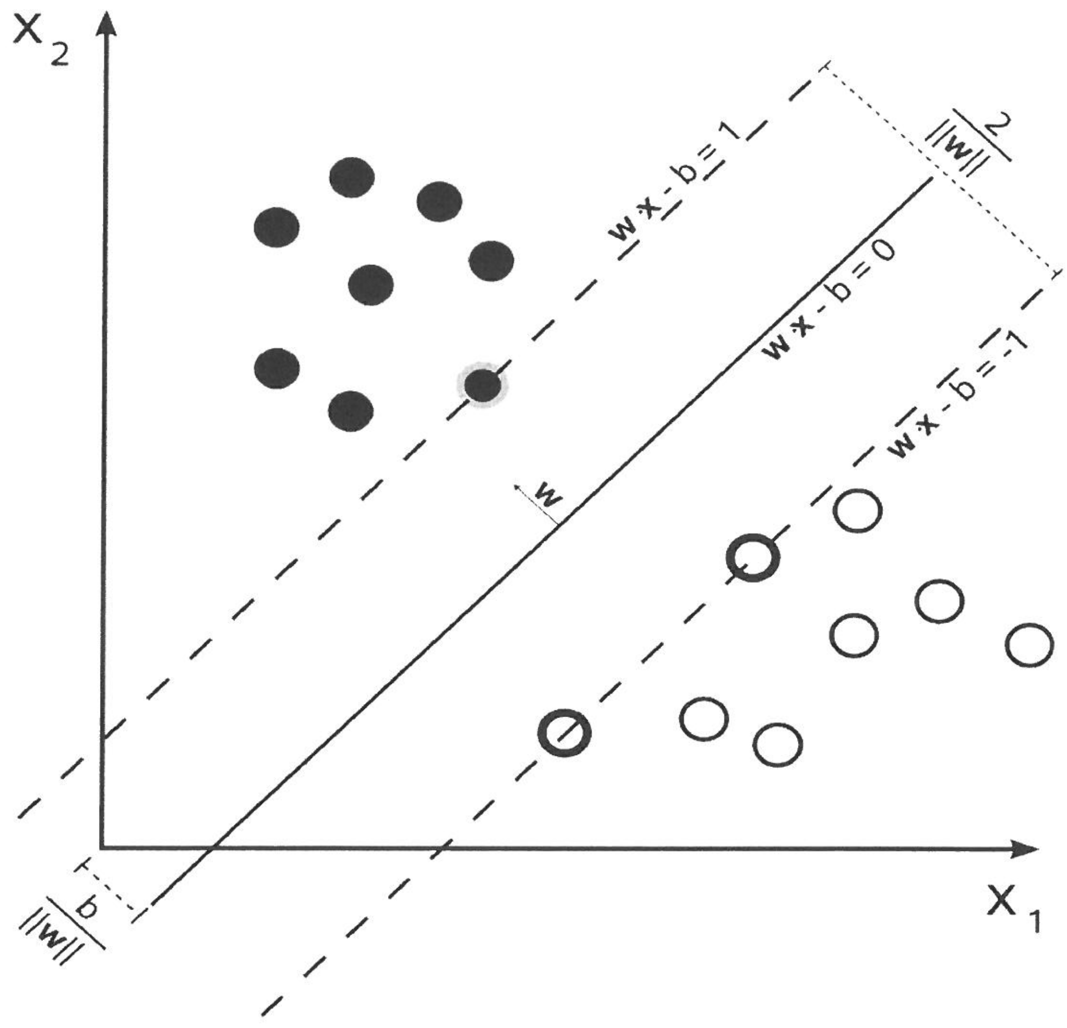


Figure 2. Splitting classes linearly with maximum margins separating the classes

Another data mining classifier is the k-nearest neighbor. The k-nearest neighbor (K-NN) is a non-parametric method for pattern classification and is commonly based on the Euclidean distance between a test sample and the specified training samples. The predicted class of a test sample is set equal to the most frequent true class among k nearest (neighbors) training samples. In other words, the decision is based on the majority vote. K-NN method is a memory-based "lazy" learning algorithm with no explicit training and model formula. The potential problem is that it requires large memory in some practical applications.

3.1.3 The Classifiers Used in this Research

For this work we chose one probabilistic classification model and one non-probabilistic classification model to fit the data. The logistic regression model one of most frequently used models in clinical study and health science. As a parametric statistical model, the logistic model is easy to use and interpret. The non-probabilistic classification model we employ in this study is the decision tree since it is simple, intuitive and understandable. It is also frequently used as a baseline classifier to define more powerful ensemble classification models. One of the practical reasons for choosing these two classifiers is the availability of software for building these models. The logistic regression work was completed using SAS/Stat. The decision tree model was built using R (RPART package).

3.2 Logistic Regression Models

In this section, we briefly describe the process of building the logistic regression model. The detailed technical development of the model development can be found in Kutner et al (2004), Agresti (2007), among others.

3.2.1 Model Building

Our outcome variable is binary (0 = no readmission/1 = readmission), the patient readmitted or did not. Logistic regression is a framework that allows us to discover significant predictors for our outcome event. Since our outcome Y_i is a Bernoulli random variable it has probabilities of

$$P(Y_i = 1) = \pi_i$$

and

$$P(Y_i = 0) = 1 - \pi_i.$$

Then the event Y$_i$ is a Bernoulli random variable with parameter $E(Y_i) = \pi_i$.

The simple logistic regression is defined to be the following generalized linear model (GLM) with form:

$$\text{Log}\frac{E(Y_i)}{1-E(Y_i)} = \beta_0 + \beta_1 X_{1i} + \ldots + \beta_k X_{ki} .,$$

Consequently,

$$E(Y_i) = \pi_i = \frac{\exp(\beta_0 + \beta_1 X_{1i} + \ldots + \beta_k X_{ki})}{1 + \exp(\beta_0 + \beta_1 X_{1i} + \ldots + \beta_k X_{ki})}$$

We use the method of maximum likelihood (MLE) to estimate the model parameters β, a critical step in the model build process. Probability distribution for Y$_i$ is:

$$f_i(Y_i) = \pi_i^{Y_i}(1-\pi_i)^{1-Y_i}. \quad (1)$$

Since the Y$_i$ are independent observations their joint probability function using (1) is:

$$g(Y_1,\ldots,Y_n) = \prod_{i=1}^{n} f_i(Y_i) = \prod_{i=1}^{n} \pi_i^{Y_i}(1-\pi_i)^{1-Y_i}. \quad (2)$$

To make it easier to find the MLE we work with the log of the joint probability:

$$\log_e g(Y_1,\ldots,Y_n) = \log_e \prod_{i=1}^{n} f_i(Y_i) = \log_e \prod_{i=1}^{n} \pi_i^{Y_i}(1-\pi_i)^{1-Y_i} \quad (3)$$

$$= \sum [Y_i \log_e \pi_i + (1-Y_i)\log_e(1-\pi_i)]$$

$$= \sum_{i=1}^{n} \left[Y_i \log_e \left(\frac{\pi_i}{1-\pi_i} \right) \right] + \sum_{i=1}^{n} \log_e (1-\pi_i)$$

Since $E(Y_i) = \pi_i$ for our binary variable we have:

$$1-\pi_i = \left[1+\exp(\beta_0+\beta_1 X_{1i}+\ldots+\beta_k X_{ki}\right]^{-1},$$

To simplify the notation, we use the following vector notation:

$$X_i^{'}\beta = \beta_0 + \beta_1 X_{i,1} + \ldots + \beta_{p-1} X_{i,p-1}.$$

The MLE of the regression coefficients is the solution to maximize the following log-likelihood function:

$$\log_e L(\beta) = \sum_{i=1}^{n} Y_i \left(X_i^{'}\beta\right) - \sum_{i=1}^{n} \log_e \left[1+\exp\left(X_i^{'}\beta\right)\right].$$

The fitted logistic regression model is given by

$$\hat{\pi}_i = \frac{\exp\left(X_i^{'}b\right)}{1+\exp\left(X_i^{'}b\right)} = \left[1+\exp\left(-X_i^{'}b\right)\right]^{-1}.$$

We now present details on how logistic regression helps describe the effects of an explanatory variable on a binary outcome. The logistic regression model has linear form for the logit of the probability:

$$\text{Logit}\left(\pi(\text{x})\right) = \log\left(\pi(\text{x})/1-\pi(\text{x})\right) = \beta_0 + \sum_j \beta_j X_j,$$

where $P(Y=1 \mid X) = \pi(x)$. If we increase x_j by one unit and fix all other explanatory variables and denote

$$\pi_i^j = \frac{\exp(\beta_0+\beta_1 X_{1i}+\ldots+\beta_j X_{ji}+\ldots+\beta_k X_{ki})}{1+\exp(\beta_0+\beta_1 X_{1i}+\ldots+\beta_j X_{ji}+\ldots+\beta_k X_{ki})}$$

and

$$\pi_i^{j+1} = \frac{\exp[\beta_0 + \beta_1 X_{1i} + \ldots + \beta_j (X_{ji} + 1) + \ldots + \beta_k X_{ki}]}{1 + \exp[\beta_0 + \beta_1 X_{1i} + \ldots + \beta_j (X_{ji} + 1) + \ldots + \beta_k X_{ki}]},$$

then

$$\frac{\pi_i^{j+1} / \left(1 - \pi_i^{j+1}\right)}{\pi_i^j / \left(1 - \pi_i^j\right)} = \exp(\beta_j)\ .$$

In other words, β_j is the difference of log odds of being readmitted between the two groups defined by $X_j = x_{ji}$ and $X_j = x_{ji} + 1$ respectively.

In order to predict readmission status, we need to define a cut-off probability, say π_0, so we can compare this cut-off with the predictive probability based on a given new instance (observation) and categorize the subject to either readmission and non-readmission group. This is called the logistic scoring problem. A cross validation procedure to be introduced in next chapter will be used to find the optimal cut-off.

3.2.2 Model Selection

Similar to all other regression models, for a given set of k-explanatory variables, one can fit 2^k different regression models. One has to choose a final model that best fits the data. Since there are different methods and criteria that can be used in searching the final model, different optimal final models can be found in the model selection process.

With logistic regression we can request a full model by not specifying a variable selection method. All predictor variables are held in the model with this method. Otherwise, we focus on three available variable selection methods; forward, backward elimination, and stepwise. All methods manage the available predictor variables differently in the model build process and are available in most statistical software packages.

Forward selection adds variables to the model one at a time. The variables not already in the model are tested for inclusion. The most significant is added if its p-value is below a stated limit. This process continues until no more variables that meet the p-value requirements are identified.

Backward elimination starts with all variables in the model. The least significant variable is removed if its p-value is above a stated limit. The elimination continues until the remaining variables have p-values below a defined limit.

The stepwise method adds variables to the model one at a time. Once a variable has been added, other variables in the model are tested to see if they remained significant. If they are no longer significant they are removed. Variables continue to be added and others in the model checked until there are no others to add that meet the p-value defined limit.

3.2.3 Modeling Validation and Assessment

Once we obtain the fitted logistic response model we examine the appropriateness of the model and then proceed to make inferences about the regression coefficients and their clinical interpretation as well as analyze model performance predicting new observations. Performance evaluation measures and model fit include Akaike Information Criteria (AIC), concordant/discordant pairs, and Hosmer-Lemeshow lack-of-fit test.

The Hosmer-Lemeshow lack-of-fit test groups data with similar fitted values into equivalently populated classes, such as in deciles. If observed and expected subgroup event rates are similar then we conclude that the logistic response function is appropriate. Goodness-of-fit can also be measured by the application of concordant/discordant pairs. A pair is concordant if the observation with the outcome has a higher predicted outcome. A pair is discordant if the observation with the outcome has a lower predicted outcome. The higher the concordance rate means a better fit of the model. AIC determines goodness-of-fit while penalizing for model complexity; number of predictors. Complexity in this case means the number of predictors in our candidate models. AIC penalizes a model having higher complexity. A smaller AIC value implies a better model fit.

In this study, we take a data mining approach. In addition to the aforementioned traditional statistical methods to assess the fit of candidate models, we will also use cross-validation methods to achieve the same goal. By doing this we can provide a consistent comparison between the logistic and decision tree models.

3.3 Decision Tree Models

A decision tree is a tree in which each branch node represents a choice between a number of alternative explanatory variables, and each leaf node represents a classification or decision. For the binary classification tree model, the response variable takes on two values (0 = non-readmission / 1 = readmission). Let $\{X_1, X_2, \ldots, X_p\}$ be a set of p predictor variables. The goal for decision tree model is to predict the value of Y from new X values. There are several decision tree models developed in past three decades. C4.5 of Quinlan (1993) and CART of Breiman et al (1984) are two later classification tree algorithms that are commonly used in practice. We will apply CART to the data in this research.

3.3.1 Model Building Strategy – Growing then Pruning

The construction of the decision tree in C4.5 and CART follows a three step algorithm:

Algorithm *Pseudocode for tree construction by exhaustive search*

```
Step 1: Start at the root node.
Step 2: For each X, find the set S that minimizes the sum of the node impurities in the two child nodes and choose the split $\{X^* S^*\}$ that gives the minimum overall X and S.
Step 3: If a stopping criterion is reached, exit. Otherwise, apply step 2 to each child node in turn.
```

Unlike the C4.5 that uses entropy for its impurity function, CART uses twoing criterion to define its purity function. Note that the twoing rule is identical to the Gini index when the response is binary.

The basic idea of CART tree induction is to let the tree grow all the way out (i.e. until reaching 0% impurity), and then prune back to avoid both overfitting and under-fitting since the predictive power will decrease dramatically if the size of the final tree is too large or too small.

Growing via Splitting

The purpose of splitting rules is to yield low misclassification errors when the tree is growing. One rule uses the Gini index (Shi, 2006.) of diversity as a measure of node impurity. That index is measured as:

$$i(t) = \sum_{i \neq j} p(i \mid t) p(j \mid t)$$

We note in table 1 the Gini index measures for the splits of an example of our decision tree as shown in Figure 3. We further provide in table 1 the balance of Gini index scores for the left-side branch from Node 2 by using the tree shown in Figure 10. Note the large difference in the index scores of node 2 and 3. Gini index and gain provided to readers experienced with interpreting these measures. The Gini for our 2 class problem for a given node t is defined as,

$$Gini = 1 - \sum_i p(i \mid t)^2 .$$

The Gini Gain is defined as,

$$Gini\_Index\_of\_Split = Gini_{p(i|t)} * Gini_{this\_child} + Gini_{p(j|t)} * Gini_{other\_child} .$$

The above is weighted Gini index for both classes. The actual Gini Gain is defined as,

$$Gini\_Gain = Gini_{parent\_node} - Gini\_Index\_of\_Split .$$

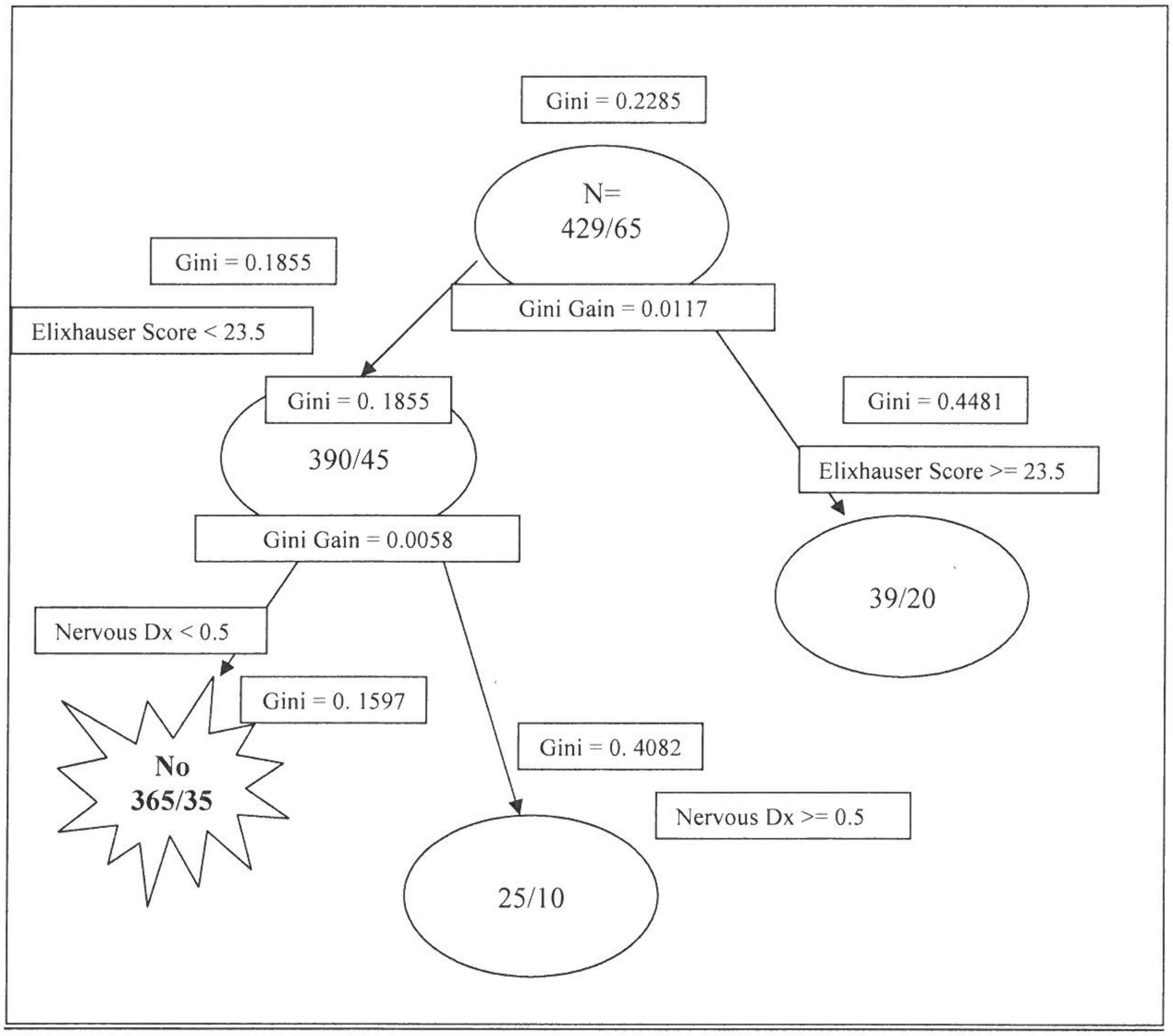


Figure 3. Toy example calculating Gini index based on a binary response.

Table 1. Gini index scores for our final training-sample based decision tree.

Root - Node 1	N	Class 0	Class 1
	494	429	65
Gini([429/494,65/494])			**0.2285**
Node 2	N	Class 0	Class 1
	435	390	45
Gini([390/435,45/435])			**0.1855**
Node 3	N	Class 0	Class 1
	59	39	20
Gini([39/59,20/59])			**0.4481**
Gini Index of the Split - Nodes 2 and 3			**0.2169**
Gini Gain			**0.0117**

Parent - Node 2	N	Class 0	Class 1
	435	390	45
Gini([390/435,45/435])			**0.1855**
Leaf Node 4 - No	N	Class 0	Class 1
	400	365	35
Gini([365/400,35/400])			**0.1597**
Node 5	N	Class 0	Class 1
	35	25	10
Gini([25/35,10/35])			**0.4082**
Gini Index of the Split - Nodes 4 and 5			**0.1797**
Gini Gain			**0.0058**

Parent - Node 5	N	Class 0	Class 1
	35	25	10
Gini([25/35,10/35])			**0.4082**
Leaf Node 8	N	Class 0	Class 1
	12	11	1
Gini([11/12,1/12])			**0.1528**
Node 9	N	Class 0	Class 1
	23	14	9
Gini([14/23,9/23])			**0.4764**
Gini Index of the Split - Nodes 8 and 9			**0.3654**
Gini Gain			**0.0427**

Parent - Node 9	N	Class 0	Class 1
	23	14	9
Gini([14/23,9/23])			**0.4764**
Leaf Node 12 - No	N	Class 0	Class 1
	16	12	4
Gini([12/16,4/16])			**0.3750**
Leaf Node 13 - Yes	N	Class 0	Class 1
	7	2	5
Gini([2/7,5/7])			**0.4082**
Gini Index of the Split - Nodes 12 and 13			**0.3851**
Gini Gain			**0.0913**

Cost-Complexity Pruning

The purpose of pruning is to address trees that are too large or small that may yield a high prediction error. Pruning is used to reduce complexity and improve predictive accuracy by removing potential noise that may be found in terminal nodes.

Cost complexity pruning generates a series of trees $T_o \ldots T_m$ where T_0 is the initial tree and T_m is the root alone. At step *i* the tree is created by removing a subtree from tree *i* - 1 and replacing it with a leaf node with value chosen as in the tree building algorithm. The subtree that is removed is chosen as follows. Define the error rate of tree T over data set S as $err(T,S)$. The subtree that minimizes

$$\frac{err(prune(T,t),S) - err(T,S)}{|leaves(T)| - |leaves(prune(T,t))|},$$

is chosen for removal. The function $prune(T,t)$ defines the tree gotten by pruning the subtrees t from the tree T. The generalization of each pruned tree $T_o \ldots T_m$ is estimated. The best pruned tree is then selected.

Here is an example of pruning a fully grown tree. We have a binary set of classes X and O. We follow down the left side and split t_2 into t_4 and t_5. We further split t_5 into t_6 and t_7 Down the right side we split t_3 into t_8 and t_9. Then we see all terminal nodes are pure, which may not be the case in many trees. We also see that both t_8 and t_9 who have t_3 as their ancestor contain the same class. Making the split of t_3 into t_8 and t_9 was not an improvement. So we will prune t_8 and t_9 back to its ancestor node

t_3. No other child nodes can be recombined without increasing the resubstitution error rate. We have pruned T_{max}. See figures 3 and 4.

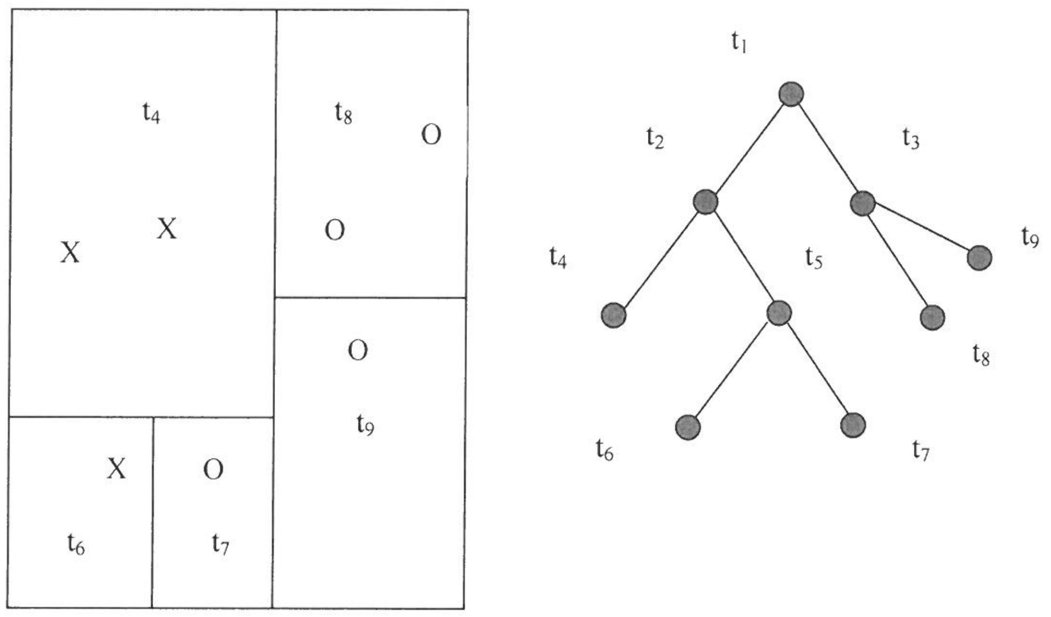


Figure 4. Toy example of a fully grown tree.

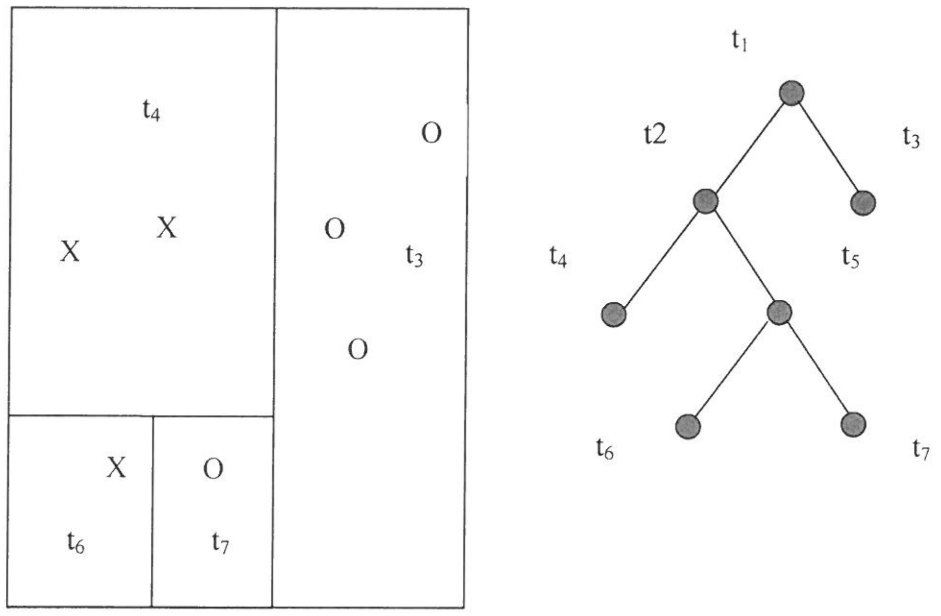


Figure 5. Toy example of a pruned tree.

3.3.2 Measure of Performance

In traditional statistics, a logistic model was built based on the binomial likelihood approach. Some likelihood based measures can be used to assess the model fit and performance. In a decision tree model, likelihood-based measures are not well defined. Therefore we use a confusion matrix to measure prediction performance of the decision tree. The confusion matrix provides a visualization of the performance of a classifier, a decision tree model, by comparing the actual class (row) versus the predicted class (column) for all known classifiers in the data.

Next we will use a 2-by-2 confusion matrix (table 2) to define a few metrics for evaluating the performance of both the logistic regression model and the decision tree model.

Table 2. Sample confusion matrix.

		Predicted class		
		No	**Yes**	**Total**
Actual class				
	No	TN	FP	TN+FP
	Yes	FN	TP	FN+TP
	Total	FN+ TN	TP+FP	

The abbreviations in the above table are defined below.

True Positive (TP) is the proportion of actual positives which are correctly identified as such. In this case the model predicts someone will readmit when they actually readmitted. This measure is also referred to as *Sensitivity*.

False Positive (FP) is the proportion of actual negatives which are incorrectly identified as positives. Model predicts readmission for someone who did not readmit.

True Negative (TN) is the proportion of actual negatives which are correctly identified as such. In this case the model predicts someone will not readmit when they actually did not readmit. This measure is also referred to as *Specificity*.

False Negative (FN) is the proportion of actual positives which are incorrectly identified as negatives. Model predicts non-readmission for someone who did readmit.

The following measures are used to assess the performance of a given classifier or comparing the performance of two more classifiers.

True Positive Rate (sensitivity)	= TP/(TP+FN)
False Positive Rate	= FP/(FP+TN)
False Negative Rate	= FN/(TP+FN)
True Negative Rate (specificity)	= TN/(FP+TN)
Predictive Accuracy rate	= (TP+TN)/(TP+TN+FP+FN)
Predictive Error Rate	= (FP+FN)/ (TP+TN+FP+FN)

The above measures area used to examine the performance of classifiers from different perspectives, see figure 6. The area under the receiver operating characteristic (ROC) curve can be used as a global measure. A ROC curve is defined to be a plot of the True Positive rate versus the False Positive rate.

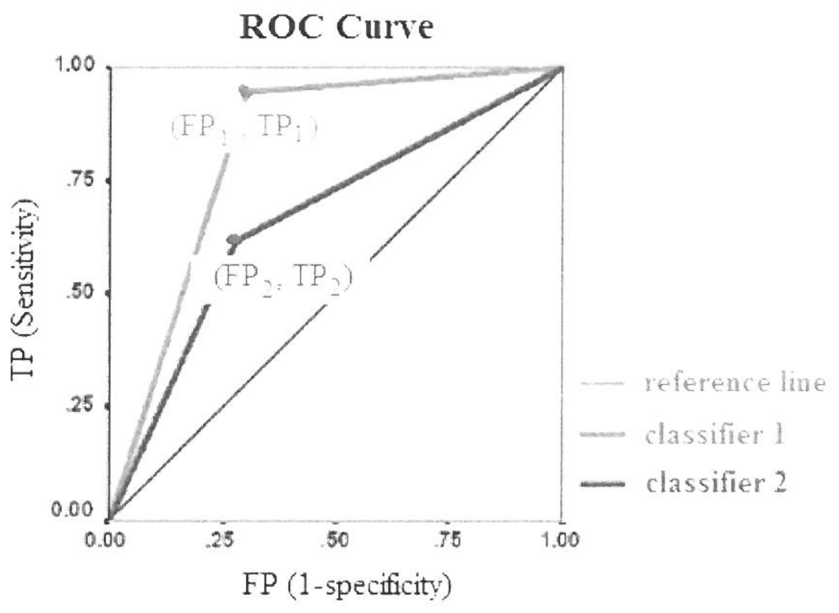


Figure 6. An example of an ROC Curve for two classes.

A larger area under the curve (AUC) indicates better predictive power. We can see from the above ROC curve that the AUC of classifier 1 (in green) is bigger than that of classifier 2 (in red), that is, classifier 1 has higher predictive power than classifier 2.

Chapter 4

Cross Validation

In statistical modeling, various inferential procedures can be developed based on various model assumptions. In a data mining model, it is rare to make assumptions about the population from which the data are collected. It is impossible to define any inferential procedures to validate and assess models. Cross validation as a data driven procedure will be used to perform the same task that inferential procedures do in statistical modeling. Many general descriptions of various cross validations can be found from literature (Refaeilzadeh et al, 2009 as well as Hastie, Tibshirani and Friedman, 2008).

4.1 Motivation of CV in This Study

One of the challenges facing the model selection process is that a candidate model may demonstrate adequate prediction capability for the training set but perform poorly on a future unseen set of data. Cross-validation provides a method for estimating the generalization performance of a candidate model and therefore can be used as a model selection tool. Cross-validation can be used to estimate generalization of a model from available data. It can also be applied to comparing the performance of two or more models. In our work we used cross-validation to compare performance of our candidate models built with logistic regression. Cross validation can also be used to tune the model parameter modeling process. We will use this method to find the optimal cut-off probability to score logistic regression models in order to make prediction.

4.2 Common Types of Cross Validations and Implementation

The two possible goals of any cross-validation method are gauging the generalizability of a model and comparing the performance of two or more models. Some types of cross validation include k-fold cross-validation, repeated k-fold cross-validation, and leave-one-out cross-validation (LOOCV). The most widely used validation procedure and well supported by the literature is 10-fold cross-validation. The strength of k-fold cross-validation is accurate performance estimation. For obtaining potentially more reliable comparisons, the repeated k-fold cross-validation is used. The strength of repeated k-fold cross-validation is it provides a larger number of performance estimates. LOOCV uses all observations except one for training and one observation is held out and used for the validation step. This method is known to have unbiased performance estimation. It is widely used when available data are rare.

Implementation of cross validation is quite straightforward and simple. First we need to choose a metric that measures the performance of the model or the estimated parameters (badness or goodness measure), say prediction error in our case. Cross-validation then directly estimates the generalization (test) error based on the hold-out validating sample and uses this to obtain a prediction error. The steps for performing k-fold cross-validation are outlined in the following algorithm:

Algorithm: K-fold Cross-validation

Step 1: Split the hold-out validation sample into K equal subsets, Z_i, for i = 1,…,K.

Step 2: Do following K times:

1. Estimate the model on subset Z_i
2. Compute the prediction error PE[*i*] of the model based on this subset.

Step 3: compute the average of those K-prediction errors as the overall estimate of the prediction error

$$CV = \frac{1}{K}\sum_{i=1}^{K} PE[i].$$

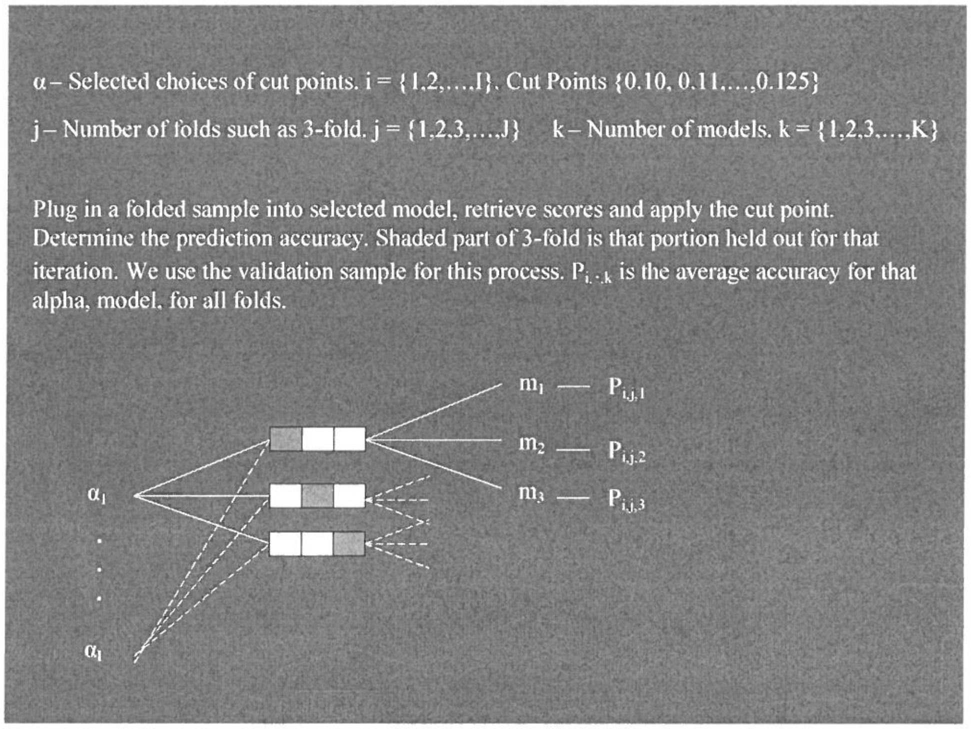
4.3 Model Selection with k-fold Cross Validation

The method of cross-validation starts with splitting the data into training and validation samples. If some data is used for validation in more than one round, the obtained results will be dependent. There must not be any overlap between the data used for learning and the data used for validation. To accommodate that requirement our work isolated the training portion of our sample from the validation portion. We also set aside a testing portion of our sample to report final model performance results in this paper. Our cross-validation effort was limited to using the validation dataset. We attempted to generate as many performance estimates as could be provided by splitting our validation set into k folds. Given the size of our validation sample (N=276) and relatively small population experiencing the outcome (~13%), we selected 5-fold cross-validation.

The validation dataset was then partitioned into 5 folds. In the splitting effort we attempted to define 5 folds that were reasonable copies of the overall sample;

having 13% experiencing the event in each fold. With each iteration a different fold within the 5-fold dataset was held out for validation. Using a candidate model we then ran 4 folds through the model process using the candidate model variables. Then we used the parameters from that training model and measured performance, the success rate of prediction, using the hold out fold. These rates were then averaged for each of the original candidate models. The candidate model with the best average success rate was chosen as our final candidate. The candidate models processed through cross-validation were those built using logistic regression.

We now provide some graphics and text on the following pages to explain the cross validation and cut point selection process. Our goal is 1 model and associated cut point.

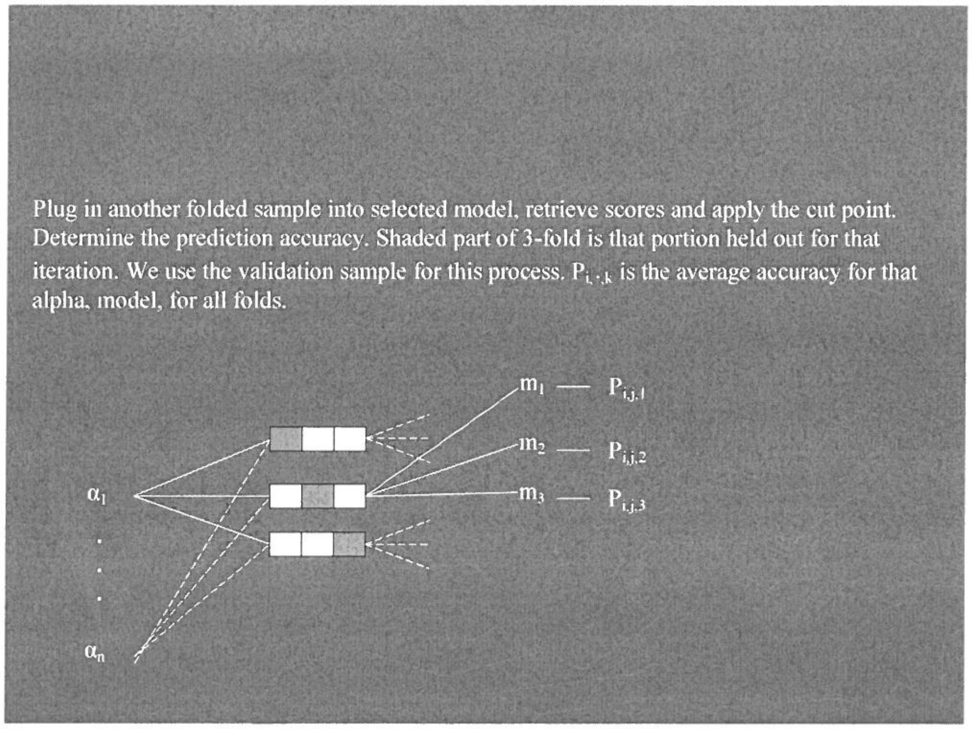


α – Selected choices of cut points. $i = \{1,2,\ldots,I\}$. Cut Points $\{0.10, 0.11,\ldots,0.125\}$

j – Number of folds such as 3-fold. $j = \{1,2,3,\ldots,J\}$ k – Number of models. $k = \{1,2,3,\ldots,K\}$

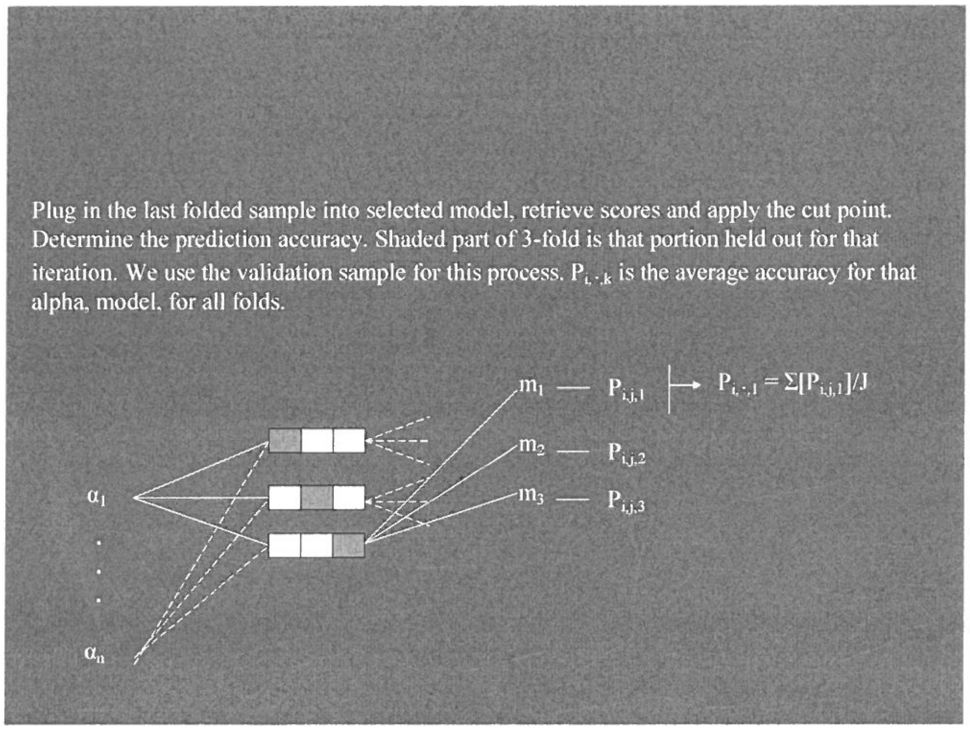
Plug in a folded sample into selected model, retrieve scores and apply the cut point. Determine the prediction accuracy. Shaded part of 3-fold is that portion held out for that iteration. We use the validation sample for this process. $P_{i,\cdot,k}$ is the average accuracy for that alpha, model, for all folds.

Figure 7. Cross validation holding out first fold.



Plug in another folded sample into selected model, retrieve scores and apply the cut point. Determine the prediction accuracy. Shaded part of 3-fold is that portion held out for that iteration. We use the validation sample for this process. $P_{i,\cdot,k}$ is the average accuracy for that alpha, model, for all folds.

Figure 8. Cross validation holding out second fold.



Plug in the last folded sample into selected model, retrieve scores and apply the cut point. Determine the prediction accuracy. Shaded part of 3-fold is that portion held out for that iteration. We use the validation sample for this process. $P_{i,\cdot,k}$ is the average accuracy for that alpha, model, for all folds.

Figure 9. Cross validation holding out last fold.

Each cell in the 3-deminsional matrix (or vector) represents the Prediction Rate for:

cut point i, fold j, model k denoted as:

$P_{i,j,k}$ such that $P_{i,j,k}$ is in [0-1].

For 1 full cross validation (i.e. 10-fold), for 1 alpha, and 1 model our Prediction Rate is the average:

$$P_{i,\cdot,k} = \Sigma[P_{i,j,k}]/J.$$

The maximum Predicted Rate for a given model designates the alpha selected:

$$P_k^{(\alpha)} = \max_i(P_{i,\cdot,k}), \text{ for } i = 1 \text{ to } I.$$

The final candidate designates the alpha and the model as:

$$P^{(\alpha,m)} = \max_k(P_k^{(\alpha)}), \text{ for } k = 1 \text{ to } K.$$

Chapter 5

Data Analysis and Results

In this chapter we will indentify a final logistic regression model from a pool of candidate models obtained from different model building methods as well as a decision tree model using the CART algorithm. The fitted logistic regression, we do both prediction and association analysis while the decision tree model can only be used for prediction. We will utilize AUC to compare the predictive power between the two different predictive models.

5.1 The Analytic File

An analytic file (N=1,049) was created from the univariate step where variables were selected with a liberal p-value <= 0.50. The analytic file from the final dimension reduction and feature additions produced the features shown in tables 3 and 4.

Table 3 covers the dichotomous variables and presents the percent of the two outcome populations who have that medical fact/setting. Rates in bold are significantly different for that outcome group (Chi-Square test p-value < 0.05).

Table 3. Incidence of medical fact/setting for both outcome cohorts

Proportions with Medical Fact	**Readmitted**	
Variable	**Yes**	**No**
Coronary atherosclerosis and other heart disease	94.85	92.22
Pulmonary heart disease	11.03	8.00
Cardiac dysrhythmias	49.26	40.64
Congestive heart failure; nonhypertensive	**52.21**	30.12
Peripheral and visceral atherosclerosis	**22.79**	13.03
Chronic obstructive pulmonary disease and bronchiectas	**33.09**	16.10
Pleurisy; pneumothorax; pulmonary collapse	**12.50**	6.02
Respiratory failure; insufficiency; arrest (adult)	**17.65**	8.43
Esophageal disorders	30.88	26.62
Other gastrointestinal disorders	**16.18**	6.13
Acute and unspecified renal failure	**24.26**	15.44
Chronic kidney disease	**41.91**	25.41
Urinary tract infections	**13.97**	8.00
Osteoarthritis	15.44	10.41
Other connective tissue disease	16.18	11.28
Complications of surgical procedures or medical care	23.53	18.95
E Codes: Adverse effects of medical care	**30.15**	20.26
E Codes: Adverse effects of medical drugs	11.03	7.34
Diabetes mellitus with complications	**10.29**	4.38
Gout and other crystal arthropathies	7.35	5.37
Fluid and electrolyte disorders	**35.29**	17.85
Other nutritional; endocrine; and metabolic disorders	14.71	11.06
Deficiency and other anemia	**30.15**	17.96
Acute posthemorrhagic anemia	7.35	5.81
Coagulation and hemorrhagic disorders	7.35	5.81
Anxiety disorders	8.82	6.35
Delirium, dementia, and amnestic and other cognitive d	12.50	7.89
Mood disorders	13.24	10.95
Screening and history of mental health and substance a	29.41	24.32
Other nervous system disorders	**17.65**	8.00
Heart valve disorders	**27.21**	18.84
Essential hypertension	41.91	**54.98**
Hypertension with complications and secondary hyperten	**34.56**	22.89
Diagnostic ultrasound of heart (echocardiogram)	**18.38**	10.41
Coronary artery bypass graft (CABG)	19.85	14.35
Percutaneous transluminal coronary angioplasty (PTCA)	42.65	**53.45**
Diagnostic cardiac catheterization; coronary arteriography	74.26	78.86
Extracorporeal circulation auxiliary to open heart procedures	19.85	13.91
Other OR procedures on vessels other than head and neck	42.65	**54.11**
Other non-OR therapeutic cardiovascular procedures	41.91	**52.46**
Perc cardiovasc proc w drug-eluting stent w MCC or 4+ vessel	10.29	8.43
Perc cardiovasc proc w drug-eluting stent w/o MCC	11.76	**27.82**
Acute myocardial infarction, discharged alive w MCC	**22.06**	14.79
Acute myocardia infarction, discharged alive w/o CC/MCC	2.94	7.01

DISCH_OTHER_CARE	**37.50**	16.21
DayDischarge	98.53	97.59
MARITAL_STAT	41.18	**53.01**
MEDICAL_SUBSERVICE	34.56	30.12
WeekendAdmit	25.74	32.20
WeekendDischarge	22.79	17.20

From reviewing this table we anticipated for example that congestive heart failure (CHF) could be a strong candidate for prediction. Table 4 covers the continuous variables and presents descriptive statistics for those variables. We made note of the large difference in median values for the Elixhauser Total Score. Table 5 presents statistics for gender and overall readmission rate (~13%).

Table 4. Statistics for continuous variables for both outcome cohorts

Variable	Readmitted	Obs.	Mean	Median	StdDev	Min	Max
Age	Yes	136	78.49	78	7.67	65	95
	No	913	77.41	77	7.82	65	110
Length of Stay	Yes	136	7.88	5	6.93	1	52
	No	913	5.85	4	5.21	1	48
Elixhauser Score	Yes	136	14.46	**14**	11.12	-4	47
	No	913	8.99	5	9.95	-5	51

Table 5. Distribution and readmission rate by Gender

Gender mix and readmission rate			----------**Readmitted**----------		
Gender	**N**	**%**	**Yes**	**No**	**%**
Female	463	44.14	62	401	13.39%
Male	586	55.86	74	512	12.63%
	1,049		136	913	
Overall readmission rate:					12.96%

The variables selected from the univariate step were the source for both the logistic regression work as well as the decision tree efforts. The entire cohort (full sample) was then split into training, validation, and test samples. An attempt was made to have the samples be approximate representatives of the outcome split in the entire cohort (87%, 13%). This was done by creating a routine to test various seeds for the pseudo random number generator used to place a patient in a specific sample. Using the uniform pseudo random number generator in SAS we selected seed 14. Table 6 presents the results of splitting the cohort into 3 samples.

Table 6. Cohort splits for various samples and their outcome distributions.

Training Sample		
Readmitted	**N**	**%**
Yes	65	13.16
No	429	86.84
	494	

Validation Sample		
Readmitted	**N**	**%**
Yes	34	12.32
No	242	87.68
	276	

Test Sample		
Readmitted	**N**	**%**
Yes	37	13.26
No	242	86.74
	279	

5.2 Logistic Regression Models

5.2.1 Building Models Based on Automatic Variable Selection

Model candidates were built using the training sample (N=494). We developed four different logistic regression models. The full model by definition does not specify a variable selection method and was constructed for review purposes only. We specified selection methods: forward selection with a significance level of 0.2 required to allow a variable into the model, backward elimination with a significance level of 0.157 required to allow a variable to stay in the model, and stepwise with a significance level of 0.5 required to allow a variable into the model and with a significance level of 0.157 to allow a variable to stay in the model. See the results of these models in tables 7 through 10.

Table 7. Full model candidate

Analysis of Maximum Likelihood Estimates

Parameter	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	-5.98	2.65	5.09	0.024
Age	0.02	0.02	0.43	0.513
Coronary atherosclerosis and other heart disease	1.38	0.87	2.51	0.113
Pulmonary heart disease	-0.85	0.71	1.45	0.229
Cardiac dysrhythmias	0.11	0.37	0.08	0.773
Congestive heart failure; nonhypertensive	0.95	0.51	3.42	0.065
Peripheral and visceral atherosclerosis	0.25	0.41	0.39	0.534
Chronic obstructive pulmonary disease and bronchiectas	0.78	0.40	3.76	0.053
Pleurisy; pneumothorax; pulmonary collapse	0.91	0.70	1.69	0.193
Respiratory failure; insufficiency; arrest (adult)	0.83	0.52	2.58	0.108
Esophageal disorders	0.08	0.37	0.05	0.828
Other gastrointestinal disorders	0.46	0.57	0.66	0.417
Acute and unspecified renal failure	-0.47	0.53	0.77	0.380
Chronic kidney disease	1.27	0.76	2.80	0.094
Urinary tract infections	1.22	0.51	5.81	0.016
Osteoarthritis	0.16	0.49	0.11	0.741
Other connective tissue disease	0.42	0.46	0.84	0.360
Complications of surgical procedures or medical care	-0.44	0.58	0.56	0.453

E Codes: Adverse effects of medical care	1.17	0.51	5.20	0.023
E Codes: Adverse effects of medical drugs	0.97	0.56	2.96	0.086
Gout and other crystal arthropathies	0.66	0.63	1.09	0.297
Fluid and electrolyte disorders	0.02	0.40	0.00	0.958
Other nutritional; endocrine; and metabolic disorders	0.18	0.53	0.11	0.737
Deficiency and other anemia	0.59	0.38	2.41	0.120
Acute posthemorrhagic anemia	-0.51	0.80	0.41	0.523
Coagulation and hemorrhagic disorders	0.20	0.65	0.10	0.758
Anxiety disorders	-1.12	0.72	2.46	0.117
Delirium, dementia, and amnestic and other cognitive d	0.26	0.61	0.18	0.674
Mood disorders	0.01	0.49	0.00	0.986
Screening and history of mental health and substance a	0.00	0.40	0.00	0.999
Other nervous system disorders	1.22	0.49	6.24	0.013
Heart valve disorders	0.37	0.41	0.82	0.366
Essential hypertension	-0.29	0.43	0.45	0.502
Hypertension with complications and secondary hyperten	-1.59	0.76	4.34	0.037
Diagnostic ultrasound of heart (echocardiogram)	-0.81	0.69	1.40	0.237
Coronary artery bypass graft (CABG)	-1.46	1.80	0.66	0.416
Diagnostic cardiac catheterization; coronary arteriography	0.32	0.45	0.49	0.483
Extracorporeal circulation auxiliary to open heart procedures	1.88	1.71	1.20	0.273
Other OR procedures on vessels other than head and neck	-1.55	1.25	1.53	0.217
Other non-OR therapeutic cardiovascular procedures	1.34	0.92	2.13	0.145
Discharge to Other Care	1.01	0.44	5.21	0.023
Day Discharge	0.85	1.17	0.53	0.467
Length of Stay	-0.09	0.05	3.13	0.077
Marital Status	-0.29	0.34	0.75	0.387
Medical Subservice	-0.10	1.04	0.01	0.921
Perc cardiovasc proc w drug-eluting stent w MCC or 4+ vessel	-0.30	0.65	0.21	0.648
Perc cardiovasc proc w drug-eluting stent w/o MCC	-0.74	0.60	1.54	0.215
Acute myocardial infarction, discharged alive w MCC	-1.02	0.72	1.99	0.159
Acute myocardia infarction, discharged alive w/o CC/MCC	-0.33	0.96	0.12	0.734
Elixhauser Total Score	0.01	0.03	0.06	0.806
Weekend Admit	-0.64	0.38	2.91	0.088
Weekend Discharge	0.23	0.41	0.30	0.583

Table 8. Forward selection candidate

Analysis of Maximum Likelihood Estimates

Parameter	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	-4.17	0.85	23.91	<.0001
Coronary atherosclerosis and other heart disease	1.26	0.80	2.47	0.116
Congestive heart failure; nonhypertensive	1.13	0.36	9.77	0.002
Chronic obstructive pulmonary disease and bronchiectas	0.66	0.34	3.76	0.052
Respiratory failure; insufficiency; arrest (adult)	0.72	0.44	2.71	0.100
Other gastrointestinal disorders	0.68	0.50	1.82	0.178
Urinary tract infections	1.03	0.44	5.43	0.020
E Codes: Adverse effects of medical care	1.07	0.35	9.35	0.002

E Codes: Adverse effects of medical drugs	0.91	0.48	3.59	0.058
Deficiency and other anemia	0.58	0.33	3.08	0.080
Other nervous system disorders	1.09	0.42	6.88	0.009
Discharge to Other Care	1.09	0.37	8.48	0.004
Length of Stay	-0.07	0.03	5.19	0.023
Acute myocardial infarction, discharged alive w MCC	-0.61	0.45	1.83	0.177
Weekend Admit	-0.55	0.33	2.71	0.100

Table 9. Backward Elimination selection candidate

Analysis of Maximum Likelihood Estimates

Parameter	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	-4.17	0.85	23.84	<.0001
Coronary atherosclerosis and other heart disease	1.25	0.80	2.46	0.117
Congestive heart failure; nonhypertensive	1.11	0.37	8.86	0.003
Chronic obstructive pulmonary disease and bronchiectas	0.71	0.34	4.39	0.036
Respiratory failure; insufficiency; arrest (adult)	0.72	0.44	2.67	0.102
Chronic kidney disease	1.27	0.65	3.77	0.052
Urinary tract infections	1.12	0.45	6.31	0.012
E Codes: Adverse effects of medical care	1.00	0.35	8.05	0.005
E Codes: Adverse effects of medical drugs	0.87	0.49	3.16	0.076
Deficiency and other anemia	0.63	0.34	3.49	0.062
Other nervous system disorders	1.09	0.42	6.72	0.010
Hypertension with complications and secondary hyperten	-1.44	0.69	4.41	0.036
Discharge to Other Care	1.01	0.38	7.12	0.008
Length of Stay	-0.06	0.03	3.88	0.049
Acute myocardial infarction, discharged alive w MCC	-0.66	0.46	2.04	0.153
Weekend Admit	-0.57	0.34	2.85	0.091

Table 10. Stepwise selection candidate

Analysis of Maximum Likelihood Estimates

Parameter	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	-4.24	0.84	25.40	<.0001
Coronary atherosclerosis and other heart disease	1.30	0.79	2.72	0.099
Congestive heart failure; nonhypertensive	0.90	0.32	8.02	0.005
Chronic obstructive pulmonary disease and bronchiectas	0.72	0.34	4.58	0.032
Respiratory failure; insufficiency; arrest (adult)	0.63	0.43	2.16	0.142
Urinary tract infections	0.95	0.44	4.70	0.030
E Codes: Adverse effects of medical care	1.08	0.35	9.67	0.002
E Codes: Adverse effects of medical drugs	0.89	0.48	3.49	0.062
Deficiency and other anemia	0.54	0.33	2.76	0.097
Other nervous system disorders	0.96	0.40	5.81	0.016

Discharge to Other Care	0.99	0.36	7.48	0.006
Length of Stay	-0.06	0.03	3.31	0.069
Weekend Admit	-0.55	0.33	2.72	0.099

5.2.2 Scoring using Cross Validation

The next task was to choose a model from the logistic model candidates having the best predictive power. This was completed in two steps. First the candidate models were validated using the validation sample. Then the scores from the validation step were compared and a best candidate selected.

Cross validation involved multiple steps (see Chapter 4). First, we split the validation sample into five folds. Second, using four of the five folds we ran the candidate models without any selection method specified. Third, the fold held out was used to score that fold's population using the parameter estimates provided by step two for the various model candidates. Steps two and three were repeated five times, holding out a different fold each time. Finally, the scores were averaged for each model candidate. The highest average prediction rate was the candidate model selected.

5.2.3 Selecting Final Logistic Model via 5-fold Cross Validation

The model producing the highest average predictive score (in bold) was chosen. See table 11.

Table 11. Cross Validation results for Logistic Regression model candidates.

	Forward Selection Candidate		
Lower 95% CL for Mean	**Upper 95% CL for Mean**	**Mean**	**Std Dev**
0.597	0.720	**0.659**	0.049
	Backward Elimination Candidate		
Lower 95% CL for Mean	**Upper 95% CL for Mean**	**Mean**	**Std Dev**
0.555008	0.656028	0.605518	0.040679
	Stepwise Candidate		
Lower 95% CL for Mean	**Upper 95% CL for Mean**	**Mean**	**Std Dev**
0.577457	0.696979	0.637218	0.048130

The cross validation process was run for the forward, backward elimination, and stepwise selection method candidate models. The three candidates produced mean predictive results of 66%, 61%, and 64% respectively. We compared these results with traditional model fit statistics in table 12. The Akaike information criteria values are too similar to aid a selection decision. Note that the Hosmer-Lemeshow test of goodness-of-fit for all 3 models shows no lack of fit to the data. However, we note that the p-value of the Chi-Square test for the forward selection was the smallest at 0.1792.

Table 12. Model fit statistics for all Logistic Regression model candidates.

Akaike Information Criteria		Hosmer-Lemeshow Lack-of-Fit Test		
Model	AIC	Chi-Square	DF	Pr > ChiSq
Forward	345.592	11.4156	8	0.1792
Backward	345.075	7.9924	8	0.4342
Stepwise	345.294	9.0324	8	0.3396

We chose the forward selection method candidate model and scored this model using the test sample. The correct classification rate for this model with the test sample was 67%. The predictive success for the test population using the final candidate model was fair at best.

We need to note how we set a scoring cut point for the probabilities (scores) returned by the different logistic models. Cut points stratify the population scores into predicted event/non-event cohorts. Understand that a risk score of 0.50 would not work as a cut point that declares scores above 0.50 as prediction of readmission and less than 0.50 as prediction of non-readmission. We make this statement based on the nature of the explanatory variables most of which are binary and the actual event proportion being ~13%. We tested various cut points for each of the three logistic models and selected one for each model type that had the best prediction rate, especially where the readmission prediction was relatively high. We sought a balance where the two cohorts were well predicted given certain cut points. We graphed the prediction rates for both the readmission and non-readmission populations. See figures 7 through 9. After review of the data the final cut points for the candidate logistic models are:

Forward selection method cut point: 0.1075

Backward elimination selection method cut point: 0.11

Stepwise selection method cut point: 0.115.

These cut points were then used in the cross validation and test samples to stratify the two outcome groups when their risk scores were calculated.

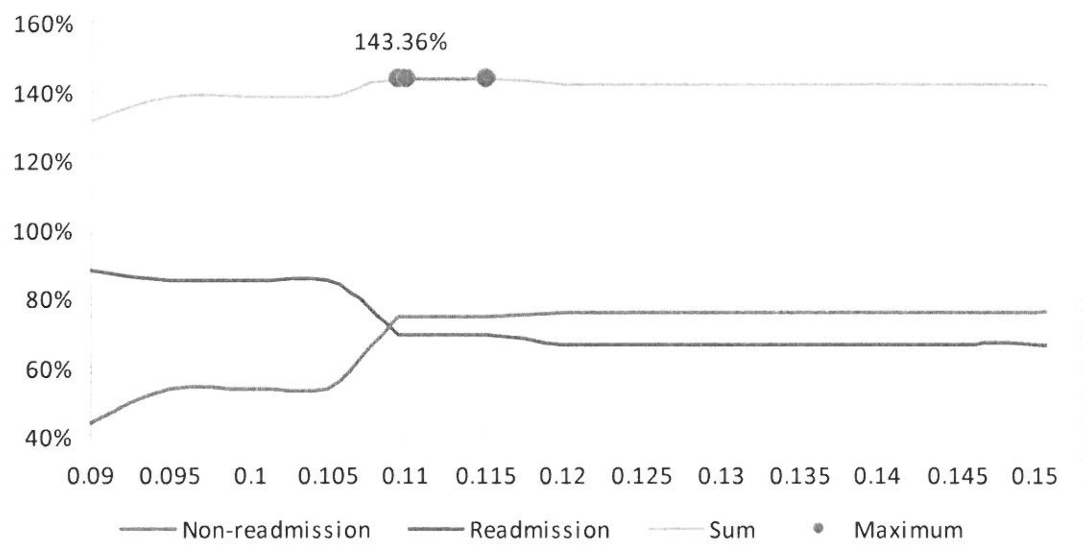


Figure 10. Forward selection successful prediction rates for certain cut points

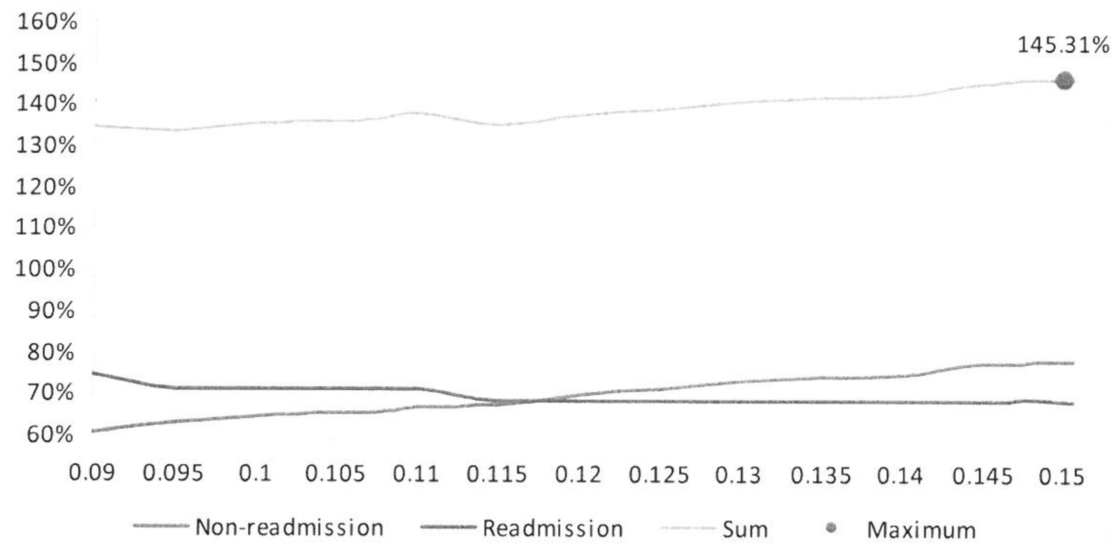


Figure 11. Backward selection successful prediction rates for certain cut points

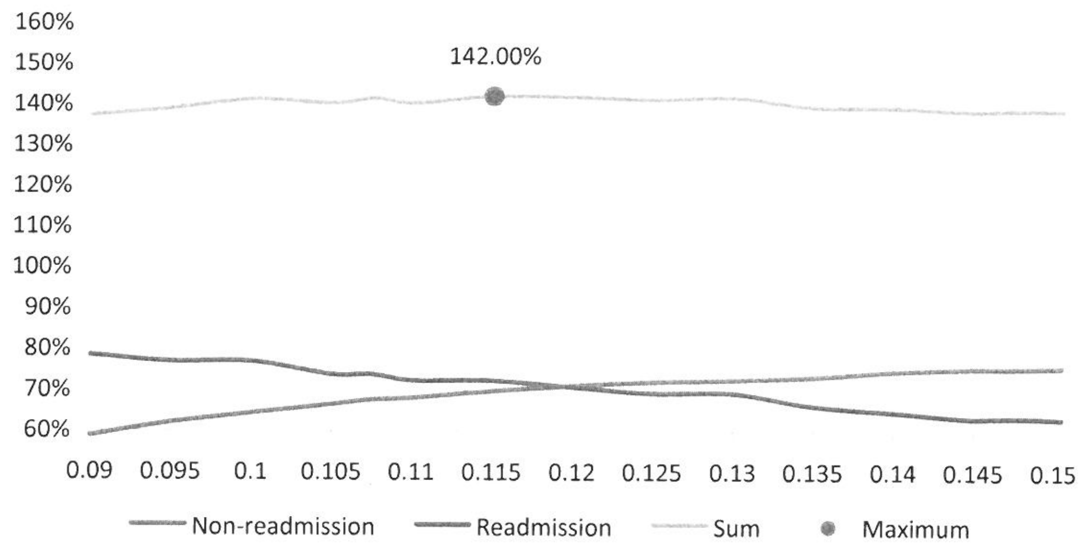


Figure 12. Stepwise selection successful prediction rates for certain cut points

We note that the chosen logistic candidate shared the same explanatory variables and parameter signs with the other 2 candidate models with the exception of other gastrointestinal disorders and AMI discharged alive with medical complications (MCC).

5.3 Decision Tree Model

From the CART modeling methodology we grew a decision tree shown in Figure 10. The predictive success using the training sample for our CART model was 88.5% and the predictive success for the test sample was 86%. Although at first glance this seems to be an excellent predictive scoring ratio it is misleading. The CART model using the test sample correctly predicted readmission for only 11% of the readmission cohort.

What is interesting is that Elixhauser Total Score plays an overwhelming role in the tree but does not appear in any of our logistic regression candidate models. This variable was used for the root splitting decision as well as the splitting decision for node 3. This variable directed 468 of a possible 494 training sample patients available to the decision tree model. An interesting note is Elixhauser scores less than 23.5 and greater than or equal to 27.5 directed high percentages of non-readmits.

Other variables used for splitting were: the derived variable weekend admit meaning the patient was admitted on the weekend and age with a split of 82.5. Additionally, two CCS grouped diagnosis families appeared in the tree: other nervous system disorders and chronic obstructive pulmonary disease. These other variables directed a very small percentage of the cohort.

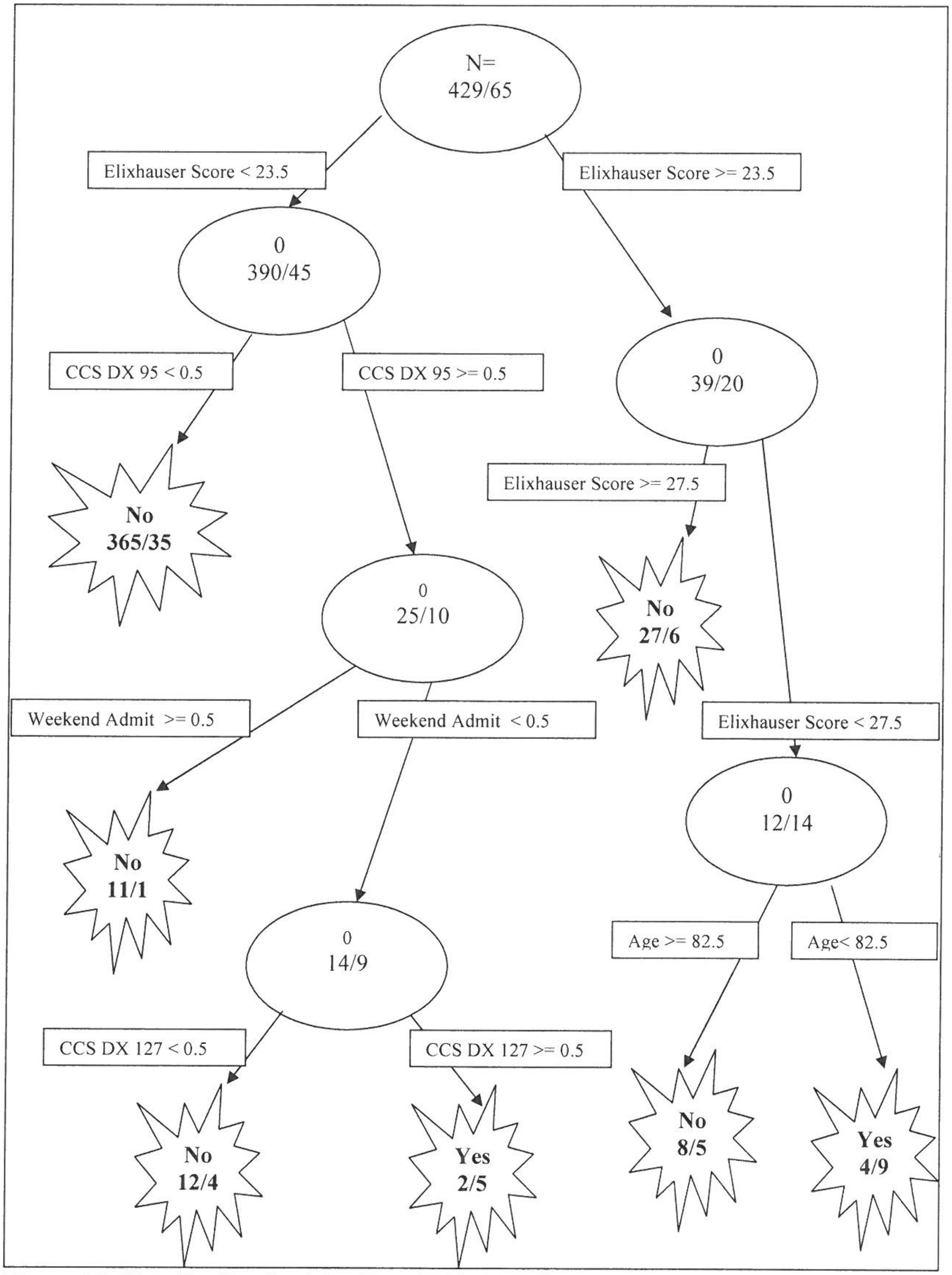


Figure 13. Classification Decision Tree for Training Sample.

5.4 Interpretation of the Two Models

We note that the explanatory variables for the logistic candidate are all predictive with the exception of Length of Stay, Acute Myocardial Infarction discharged alive w MCC, and Weekend Admit which are protective. See table 13. Also noted is a series of the variables having Wald confidence limits that include 1. Those not including 1 are in bold. We include chronic obstructive pulmonary disease since the lower confidence limit is in excess of .99. Those 8 appear to be strong predictors with minimum point estimate of ~2, meaning those patients with these comorbidities/facts have a 2 times or higher chance of readmission within 30 days of discharge of the index admission. For every increase of 1 day in length of stay, the patient has an approximately 7% less chance of readmission. A variable that surprises us is urinary tract infections. This may be an indication of poor general health and/or ability to care for one's self. The variable titled other nervous system disorders (a CCS grouper) needs to be unbundled for the event population and studied in more detail in future work. We also plan to develop a new model in the future with the reduced set of variables where the Wald confidence limits do not include 1. We did not include E Codes: Adverse effects of medical drugs due to a lower confidence limit of 0.97 but this decision may be too strident. This will be discussed in the future.

Table 13. Odds ratio estimates for chosen Logistic Regression model

Odds Ratio Estimates **Effect**	**Point Estimate**	**95% Wald Confidence Limits**	
Coronary atherosclerosis and other heart disease	3.53	0.73	17.05
Congestive heart failure; nonhypertensive	3.10	**1.53**	**6.31**
Chronic obstructive pulmonary disease and bronchiectas	1.93	**0.99**	**3.75**
Respiratory failure; insufficiency; arrest (adult)	2.05	0.87	4.82
Other gastrointestinal disorders	1.97	0.74	5.27
Urinary tract infections	2.80	**1.18**	**6.66**
E Codes: Adverse effects of medical care	2.92	**1.47**	**5.79**
E Codes: Adverse effects of medical drugs	2.48	0.97	6.32
Deficiency and other anemia	1.79	0.93	3.45
Other nervous system disorders	2.97	**1.32**	**6.70**
Discharge to Other Care	2.96	**1.43**	**6.15**
Length of Stay	0.93	**0.87**	**0.99**
Acute myocardial infarction, discharged alive w MCC	0.54	0.22	1.32
Weekend Admit	0.58	0.30	1.11

We use the splitting decisions to interpret the decision tree results. The following logic predicts non-readmission. These are presented based on splitting that produces the best 'purer' dependent nodes.

If the Elixhauser total score is less than 23.5 and the patient does not have other nervous system disorders then the prediction is no readmission.

If the Elixhauser total score is less than 23.5 and the patient does have other nervous system disorders and the patient was admitted on the weekend then the prediction is no readmission.

If the Elixhauser total score is less than 23.5 and the patient does have other nervous system disorders and the patient was not admitted on the weekend and the patient does not have chronic obstructive pulmonary disease then the prediction is no readmission.

If the Elixhauser total score is greater than or equal to 27.5 then the prediction is no readmission.

If the Elixhauser total score is greater than or equal to 23.5 and the Elixhauser total score is less than 27.5 and age is greater than or equal to 82.5 then the prediction is no readmission.

For predictions of readmission we extract the following logic from the decision tree.

If the Elixhauser total score is less than 23.5 and the patient does have other nervous system disorders and the patient was not admitted on the weekend and the patient does have chronic obstructive pulmonary disease then the prediction is readmission.

If the Elixhauser total score is greater than or equal to 23.5 and the Elixhauser total score is less than 27.5 and age is less than 82.5 then the prediction is readmission.

The logistic and decision tree finalists share the explanatory variables chronic obstructive pulmonary disease and other nervous system disorders. In the logistic candidate these variables are strong predictors. However, in the decision tree they appear to play different roles depending on the splitting logic.

The logistic and decision tree finalists share the explanatory variable weekend admission which is protective in the logistic candidate but in the decision tree it appears to play different roles depending on the splitting logic. We note the complete absence of the variable Elixhauser total score from the logistic candidate model. This may make sense since the decision tree candidate does a poor job of correctly predicting the readmission cohort.

Clinical interpretation of the decision tree model may be challenging given the different roles (predict/protect) that some of the variables play.

Chapter 6

Conclusions and Discussion

6.1 Conclusions and Comparison of the Final Models

This study attempted to identify potential predictors of readmission for AMI index cases when applying the methods of logistic regression and decision trees.

Specification of the p-value (significance) of the predictors for the logistic regression work was relaxed and chosen to be p = .157 (which is larger than the default 0.05). Literature was supportive of this action (Steyerberg, et al 2000, Lee and Koval, Shtatland, Cain and Barton). With additional time a specific p-value could be chosen via several trials and analyses.

In spite of the more liberal critical p-value the performance of the logistic models was fair at best. The attempt to compare logistic regression and classification decision trees was hampered by the univariate nature of the key predictor in the decision tree results. The internal nodes of the tree did not provide substantive predictors beyond the initial split at the root node. This was revealed when setting the priors option for the RPART fit to our known readmission rate of 13%.

Subsampling of the full sample led to very small data sets when validating and scoring results with individual folds of the validation sample. Selection of prediction cutoff probabilities was driven by a need to predict the event successfully and yet predict non-event correctly due to the large difference in split of non-outcome/outcome (87%, 13%). If clinical staff decides to alter processes based on this study's findings, it could be costly to predict at a high rate non-outcome patients as potential readmits.

We interpret results first for the chosen logistic regression model (table 14). We note that of the 14 explanatory variables in this model there are 11 that are predictive. The odds of readmission for these 14 variables range from 79% to 253% more likely for those patients who have that medical fact/setting than those who do not. The odds of readmission for the 3 protective variables are 7% (length of stay) to 42% (weekend admit) less likely for those patients who have that medical fact/setting than those who do not. Note that 8 of the 14 explanatory variables have confidence limits that include 1.This signifies that those variables are not significant at the .05 level. Recall that our specified level of significance ranged from 0.157 to 0.20.

Table 14. Odds ratios of the predictors from chosen Logistic Regression model

Odds Ratio Estimates			
Effect	**Point Estimate**	**95% Wald Confidence Limits**	
Coronary atherosclerosis and other heart disease	3.53	**0.73**	**17.05**
Congestive heart failure; nonhypertensive	3.10	1.53	6.31
Chronic obstructive pulmonary disease and bronchiectas	1.93	**0.99**	**3.75**
Respiratory failure; insufficiency; arrest (adult)	2.05	**0.87**	**4.82**
Other gastrointestinal disorders	1.97	**0.74**	**5.27**
Urinary tract infections	2.80	1.18	6.66
E Codes: Adverse effects of medical care	2.92	1.47	5.79
E Codes: Adverse effects of medical drugs	2.48	**0.97**	**6.32**
Deficiency and other anemia	1.79	**0.93**	**3.45**
Other nervous system disorders	2.97	1.32	6.70
Discharge to Other Care	2.96	1.43	6.15
Length of Stay	0.93	0.87	0.99
Acute myocardial infarction, discharged alive w MCC	0.54	**0.22**	**1.32**
Weekend Admit	0.58	**0.30**	**1.11**

We note that the three logistic models using variable selection methods all share a significant common subset of explanatory variables (table 15). Note those variables in bold are protective. The sign for all the parameters match in all three models.

Table 15. Variables retained in Logistic variable selection models.

Variable	Description
CCS_DX_CATEGORY_101	Coronary atherosclerosis and other heart disease
CCS_DX_CATEGORY_108	Congestive heart failure; nonhypertensive
CCS_DX_CATEGORY_127	Chronic obstructive pulmonary disease and bronchiectas
CCS_DX_CATEGORY_131	Respiratory failure; insufficiency; arrest (adult)
CCS_DX_CATEGORY_159	Urinary tract infections
CCS_DX_CATEGORY_2616	E Codes: Adverse effects of medical care
CCS_DX_CATEGORY_2617	E Codes: Adverse effects of medical drugs
CCS_DX_CATEGORY_59	Deficiency and other anemia
CCS_DX_CATEGORY_95	Other nervous system disorders
DISCH_OTHER_CARE	Discharge to Other Care
LENGTH_OF_STAY	**Length of Stay**
WeekendAdmit	**Weekend Admit**

The classification decision tree (Figure 10) identifies the Elixhauser score as the initial splitting choice at the root node. This split directs 88% of all members of the training sample to the left node (#2) driven by a significant non-readmission population count. Additionally, the Elixhauser score actively directs 95% of all the members of the cohort; see nodes 2 and 3. Only 21% of the 65 readmitted members are correctly identified with this tree. The tree correctly classifies over 98% of the non-readmission category. A unique variable in the tree and not the logistic model is age, but it directs a total of 26 members. We see 3 other variables in the tree that we also see in the logistic model: chronic obstructive pulmonary disease and bronchiectas, other nervous system disorders, and weekend admit. Again these variables directed a minor number of members overall in the decision tree.

We finish this comparison with a pair of confusion matrixes, tables 16 and 17, highlighting the performance of the two major candidate models, the forward selection logistic model and the CART model. The logistic overall accuracy was 67%. Note the true positive rate for the Forward Selection model was only 18%. Although the overall accuracy of the CART model was 88%, the true positive rate was a mere 11%.

Table 16. Confusion matrix for the Forward Selection candidate model

	Predicted		
Observed	No	Yes	Total
No	**171**	71	242
Yes	16	**21**	37
Total	187	92	279

Table 17. Confusion matrix for the CART candidate model

	Predicted		
Observed	No	Yes	Total
No	**236**	6	242
Yes	33	**4**	37
Total	269	10	279

We now apply the performance measure introduced in Chapter 3 to our final candidate models. We calculated coordinates based on the confusion matrix for each candidate. Refer to figure 11 and following discussion.

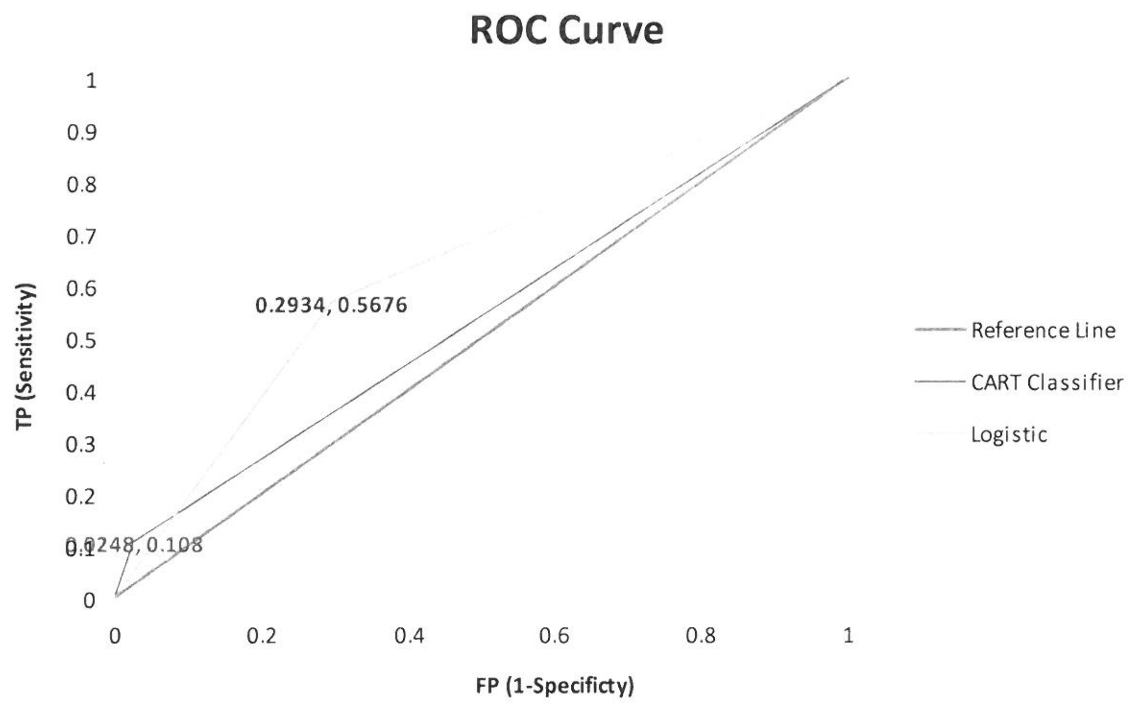


Figure 14. Receiver Operating Characteristic for Final Candidate Models.

The AUC for the Logistic candidate is 0.6371 and for the CART candidate it is 0.5417. Our conclusion about their respective predictive power is poor for the Logistic candidate and failure for the CART candidate. The logistic candidate is poor because of fairly low predictive power. We tag the CART candidate as a failure because its AUC is close to the predictive capability of flipping a coin. The ability of both models to correctly classify patients is disappointing. The CART model has an overall accuracy rate of 0.86, but its accuracy for those actually readmitted is only 16%.

6.2 Discussion

This paper assumed that the index hospital medical protocols for treating age 65+ patients for AMI remained unchanged during the study period. Therefore we assumed data collection and treatment protocols were consistent for all patients throughout the study period.

Though the data collection was limited to information recorded during the time of the index admission, the cohort represented a large number of unique variables. This number actually exceeded the sample population size. With dimension reduction via grouping we lost the granularity of identifying specific predictors as well as discrimination between outcome classes vis-à-vis predictors. Future investigation of the unique variables at a univariate level may help identify potential predictors for both classes.

Limitations applied to our study. Collinearity of the covariates was not investigated directly using such techniques as principal component analysis (PCA). In a future continuation of this work we will investigate multicollinearity using PCA given that we deal with diseases. External knowledge was not incorporated in the modeling process, especially clinical input. The decision tree results essentially were defined by one covariate, the Elixhauser total score. A brief investigation found no internal knowledge or application of the Elixhauser metric.

The set of effects produced by the selected logistic regression model and not including 1 in their Wald Confidence Limits will be investigated further and presented to clinical staff (table 18).

Table 18. Targeted effects for further investigation

Effect	Point Estimate	95% Wald Confidence Limits	
Congestive heart failure; non-hypertensive	3.10	1.53	6.31
Urinary tract infections	2.80	1.18	6.66
E Codes: Adverse effects of medical care	2.92	1.47	5.79
Other nervous system disorders	2.97	1.32	6.70
Discharge to Other Care	2.96	1.43	6.15
Length of Stay	0.93	0.87	0.99

Clinically defined predictors were not available due to time constraints and lack of access to clinicians. Additionally, no predictors were added based on literature review. These are shortcomings in this effort and will be addressed in a future version of this work.

A possible alternative to model evaluation would be to split the sample only between training and testing, dropping the validation sample, and validating with re-sampling techniques such as the Bootstrap (Steyerberg et al. 2001). This would provide us a larger training and test samples from which to train, score, and report. Part of the motivation for this method is the low representation of the outcome population in the validation folds and test sample.

A future version of this work should include medications, lab results and recent biometrics. These information sources can provide a proxy for overall symptom load and AMI severity at a patient specific level. For instance, daily biometric measures of blood pressure could be a predictor of AMI severity and therefore of readmission. Additionally, with implementation of an EMR system at the index hospital, we could acquire a more holistic and dependable view of the patient's overall health history, symptom pathway, as well as prior hospital utilization. The caveat with using an EMR as noted at the start of this paper is the reality of missing data for certain AMI patients not found on the EMR; patients who aren't part of the index hospital's network.

There are serious ongoing challenges selecting from the vast number of variables available to use in predictive modeling, given the relatively small number of readmissions in the targeted cohort. It should be noted that AMI model performance provided to CMS by Yale New Haven Health Services Corporation (YNHHSC) had a mean c-statistic covering 2007-2009 of 0.629 (Bernheim et al. 2011). Given the number of observations (N=559,430) and hospitals (N=4,576) that YNHHSC drew from for their predictive modeling work; their resulting c-statistic is poor.

A model was developed (Krumholz, et al. 2011) to produce hospital-specific risk-standardized estimates of 30-day readmission rates for AMI index cases. This model is being used to publicly report variation in readmission rates among hospitals across the United States. We could apply their model variables to our population but that presents issues. Their model had 31 variables. But this model was developed from a full sample of over 200,000 admissions from 4,171 hospitals. Their model c-statistic was 0.63. It is not surprising that 31 variables would be significant in their model given that sample size. Applying their 31 variables would be inappropriate for a sample size as small as ours and not produce good predictive power.

Appendix A – SAS Code Listings

Run the SAS code modules in the following order:

Macro: getReadmissionFlagsV6 to get a copy of it into memory for later use.

Macro: GetYear_Quarter to get a copy of it into memory for later use.

Macro: Get_AMI_Cases to get a copy of it into memory for later use.

Cohort_Identification

Variable_Additions

Initial_Variable_Reduction

Cohort_w_Facts

Cohort_Facts_Grouped_by_CCS

Elixhauser_Comorbdity

Merge_Cohort_CCS_Elix

Variable_Reduction_By_Groupers

MS_DRG_Table

Univariate_Regression

Split_Cohort

Macro: mdl_calculating_riskscore to get a copy of it into memory for later use.

Build_Model_Candidates

Cross_Validation

Score_Using_Test_Sample

CART

getReadmissionFlagsV6

```
*****************************************************************************************************************
*  Code courtesy of Ms. JoanneCederna
*  Purpose:   To generate a macro that pulls all readmission information from a dataset
*           in compliance with CMS criteria as of most current releases found on October, 2011:
*
*           NO ADMISSION CAN BE CONSIDERED BOTH AN INDEX ADMISSION AND A READMISSION. SO
*             ADDITIONAL ADMISSIONS WITHIN XX DAYS OF DISCHARGE FROM AN INDEX ADMISSION CAN
*             ONLY BE CONSIDERED AS POTENTIAL READMISSIONS
*
*           Update CMS disease definitions. PNEU, CHF, AMI and COPD re-written to reflect current oct/2011 cms definitions.
*               This can be found in a macro called:GET_CMS_DISEASE_POP  THIS would need TO BE RUN FIRST to identify
*               the disease types
*
*           This macro attributes all readmissions back to the month/year of the index stay  Should you choose to look at rates
*              based on provider the last macrovariable attribute_field will attribute the readmission back to the physician
*               identified on the index stay.
*
*           Numerators and Denominators are set in this  macro for calculation of rates.
*
*  Parameters use uppercase please:
*          rLimit: Number of Days to a readmit flag
*          IndexDisease: ALL, COPD, CHF, PN, AMI, OTHER_DISEASE
*          ReadmissionSameDisease: SAME Keep only Readmissions having the same disease as the INDEX.
*          WhichReadmission: FIRST, ALL
*          attribute_field: Your choice of field name
*
*  Exclusions: NONE
*
*  Caution:
*          1)Use the Readmit_flag column for additional work outside of this macro
*          2)If you are reporting out by FY you need to recalulate FY and CANNOT us the field Discharge_FY in SDSM
*
*****************************************************************************************************************;
%macro getReadmissionFlagsV6(rlimit,IndexDisease,ReadmissionSameDisease,WhichReadmission,attribute_field);
options mprint mlogic symbolgen;

  /**************************************************************************************************************
  set Flags on population data
  **************************************************************************************************************/
  Data work.ReadmissionFlags;
   set work.ReadmissionFlags;

   /***Create from_date from dischDate date*/
   format from_date date9 ;
   from_date=(INTNX('month',dischDate,0)) ;


   /* If EPIC data then we need to use Inpatient Admit date for admitdate in v6!!! */
     if source='EPIC' then do;
        admitDate=InpatientAdmitDate;
        admitDate_Readmission=admitDate;
     end;


   /****Identify Age group gte 65 years*****/
   If age >= 65 then Age_65PLUS_FLAG=1;
   else Age_65PLUS_FLAG=0;
  run;

  /*Sort data set for subsequent work*/
  proc sort data=WORK.ReadmissionFlags;
       by Medical_Record_number dischDate;
  run;

  /* The sequentialized table is now ready for readmission analysis
     Used for setting up the Flags and information for each entcounter
     First to do: rLimit parameter: Set up READMIT_FLAG: INITIAL or READMISSION */
  data WORK.ReadmissionFlags;
   set WORK.ReadmissionFlags;
   /* Data set will be processed by Medical Record Number */
     by Medical_Record_number dischDate;
```

```
 /* retain index discharge date for analysis*/
 RETAIN CURRENT_INDEX_DATE;
 format READMIT_FLAG $char12.;
 FORMAT CURRENT_INDEX_DATE DATE9.;
 format days_since_current_INDEX best10.;/* will count the days between all admissions between indexs based on the rlimit*/

 /* Set a field for the Number of Days for Readmission */
 Rlimit=&rlimit;

 /* If first Medical Record Number then INDEX and go to next record */
 if first.Medical_Record_number then do;
  READMIT_FLAG='INDEX';
  CURRENT_INDEX_DATE=dischDate;
  days_since_current_INDEX=0;
 end;  /* Medical_Record_number */

 else do;
  days_since_current_INDEX=intck("day",CURRENT_INDEX_DATE,admitDate); /* calculate days since current index
discharge*/
  if days_since_current_INDEX >&rlimit then do;
   READMIT_FLAG='INDEX';
   CURRENT_INDEX_DATE=dischDate;
   days_since_current_INDEX=0;
  end;
  else do;
   READMIT_FLAG='READMISSION';
  end;
  end;
run;

/* Looking at INDEX records if disease specific requested
  If INDEX then save all their respective READMITS */
 %if %UPCASE(&IndexDisease)=%STR(CHF) OR %UPCASE(&IndexDisease)=%STR(COPD) OR
%UPCASE(&IndexDisease)=%STR(PN) OR
   %UPCASE(&IndexDisease)=%STR(AMI)OR %UPCASE(&IndexDisease)=%STR(OTHER_DISEASE)  %then %do;
 data WORK.ReadmissionFlags;
   set WORK.ReadmissionFlags;
  FORMAT READMITS_KEEP_FLAG $3 ;
  RETAIN READMITS_KEEP_FLAG;
  /* Dealing with an INDEX reocrd */
  if READMIT_FLAG='INDEX' then do;
   /*  Only save disease specific INDEX and the READMITS */
   if &&IndexDisease=1 then do;
    output; /* Output the INDEX record */
    READMITS_KEEP_FLAG='YES';
   end;
   else READMITS_KEEP_FLAG='NO';
  end;
  /* Saving only READMITS that have an INDEX saved. */
  else if READMITS_KEEP_FLAG='YES' then output;
 run;
 %end;

/* Looking at READMIT records.
  If requested to save only READMITS with the same disease as the INDEX. */
 %if %UPCASE(&ReadmissionSameDisease)=%STR(SAME) %then %do;
 data WORK.ReadmissionFlags;
  set WORK.ReadmissionFlags;
   /* Keep only READMITS with the same disease as the INDEX */
   if READMIT_FLAG='READMISSION' then do;
  if &&IndexDisease=1 then output;
 end;
 /* Keep all INDEX records */
 else output;
 run;
 %end;

/* Looking at READMIT records.
  If requested to save only first READMITS only. */
 %if %UPCASE(&WhichReadmission)=%STR(FIRST) %then %do;
 data WORK.ReadmissionFlags (DROP=readmissionNumber);
  set WORK.ReadmissionFlags;
 RETAIN readmissionNumber;
   /* Keep all Index and first readmits */
   if READMIT_FLAG='INDEX' then do;
```

```
      readmissionNumber=0;
      output;
     end;
     else do;
      readmissionNumber=readmissionNumber+1;
      if readmissionNumber=1 then output;
     end;
    run;
    %end;

   /* FIRST INDEX DATE STUFF GOES HERE */
   /***********************************************************************************************************
   Now that we have the data we want. let's get FIRST_INDEX_DATE and put on all records
   ************************************************************************************************************/
   Data work.ReadmissionFlags;
    set work.ReadmissionFlags;
    BY Medical_Record_number;
    RETAIN FIRST_INDEX_DATE CURRENT_INDEX_DATE;
    FORMAT FIRST_INDEX_DATE CURRENT_INDEX_DATE DATE9.;

    if FIRST.Medical_Record_number then do;
     FIRST_INDEX_DATE=dischDate;
     days_since_FIRST_INDEX=0;
    end;  /* Medical_Record_number */
    else do;
     days_since_FIRST_INDEX=intck("day",FIRST_INDEX_DATE,admitDate);/* calculate days since 1st index discharge*/
    end;
    if READMIT_FLAG='INDEX' then do;
     CURRENT_INDEX_DATE=dischDate;
     days_since_CURRENT_INDEX=0;
    end;
   else do;
     days_since_CURRENT_INDEX=intck("day",CURRENT_INDEX_DATE,admitDate);/* calculate days since current index
discharge*/
    end;
  run;

 /*attribution back to index from date
   When reporting out readmission, all readmissions are attributed back to their index case
   When reporting out by month you would use from_date

   !!!CAUTION!!! If you are reporting out by FY you need to recalulate FY and CANNOT us the field Discharge_FY in SDSM!!!
*/
   DATA WORK.ReadmissionFlags;
    SET WORK.ReadmissionFlags;
    from_date=(INTNX('month',current_index_date,0)) ; /*Index from date based on current_index_date. atribution back to index
from date*/
   RUN;

  %if &attribute_field~=%STR() %then %do;
proc sort data=WORK.ReadmissionFlags;
BY MEDICAL_RECORD_NUMBER dischDate &&attribute_field;
run;

/*attributions*/
        data work.ReadmissionFlags;
         set work.ReadmissionFlags;
         FORMAT INDEX_attribute_field $100.;
         BY MEDICAL_RECORD_NUMBER dischDate &&attribute_field;
               FORMAT INDEX_attribute_field $25.;
               format test $50.;
               test=&&attribute_field; /*this field is used for validation*/
         RETAIN
                INDEX_attribute_field;
         if FIRST.MEDICAL_RECORD_NUMBER then do;
          INDEX_attribute_field="";
         end;

         /*  INDEX CASE Process */
         if READMIT_FLAG='INDEX' then do;
           INDEX_attribute_field=&&attribute_field;
           output;
          end;
         /* Dealing with a Readmit */
         /* attribution on the readmit is back to the attributed field */
```

```
            else do;
               &&attribute_field=INDEX_attribute_field;
             output;
             end;
           run;
           %end;

 /* Sort data */
   PROC SORT DATA=work.ReadmissionFlags;
     BY Medical_Record_number dischDate;
   run;

/* Setting the numerator and denominator all cause numerator all readmisions numerator*/
   data work.readmissionflags;
     set work.ReadmissionFlags;
     BY Medical_Record_number dischDate;
   RETAIN READMIT_NUMBER;
   denominator=0;
   numerator=0;
   if READMIT_FLAG='INDEX' then do;
    /* This will be used to number the readmissions after EACH INDEX */
    READMIT_NUMBER=0;
    denominator=1;
    OUTPUT;
   end; /* INDEX */

   else if READMIT_FLAG='READMISSION' then do;
      READMIT_NUMBER=READMIT_NUMBER+1;
      numerator=1;
      OUTPUT;
   end; /* READMISSION */
   run;

%if %UPCASE(&IndexDisease)=%STR(AMI)  %then %do;
/***************************************************************************************************************************
   Part 2 of AMI Cohort

        Begin Page 16 of document:

     3.2.2 Admissions Not Counted As Readmissions
     Some AMI patients have planned readmissions for revascularization procedures .
     for example, to perform percutaneous transluminal coronary angioplasty (PTCA)
     on a second vessel or a second location in the same vessel, or to perform coronary
     artery bypass graft (CABG) surgery after AMI and a period of recovery outside the
     hospital. Because admissions for PTCA or CABG may be staged or scheduled readmissions.
     we do not count as readmissions those admissions after discharge that include PTCA or
     CABG procedures unless the principal discharge diagnosis for the readmission is one of
     the following acute diagnoses, which are not consistent with a scheduled readmission: HF.
     AMI, unstable angina, arrhythmia, and cardiac arrest (i.e., readmissions with these
     diagnoses and a PTCA or CABG procedure are counted as readmissions).

     The ICD ]9 ]CM procedure codes associated with PTCA and CABG revascularization procedures are:
        PTCA: 00.66, 36.06, 36.07
        CABG: 36.10 36.16

     The ICd9 CM diagnosis codes associated with HF, AMI, unstable angina, arrhythmia, and cardiac arrest are:
        HF: 402.01, 402.11, 402.91, 404.01, 404.03, 404.11, 404.13, 404.91, 404.93, 428.xx
        AMI: 410.xx, except 410.x2 (AMI, subsequent episode of care)
        Unstable angina: 411.xx
        Arrhythmia: 427.xx, except 427.5
        Cardiac arrest: 427.5
***************************************************************************************************************************/


/*put this in the readmit macro new for ami*/
DATA work.ReadmissionFlags;
SET work.ReadmissionFlags;
FORMAT RESET $100.;
LENGTH RESET $100.;
/*convert readmits to PLANNED where pci or cabg were scheduled.
   Unscheduled readmission  would also have a diagnosis in the CK_ADMIT_NOT_COUNTED_AS_READ column
*/
IF (UPCASE(READMIT_FLAG)="READMISSION" AND EVAL_AMI_READMIT_CHANGE=1) THEN DO;
```

```
      RESET="planned pci or cabg, Readmission changed to Index case";
      READMIT_FLAG="PLANNED";
      FLAG="PLANNED";
      DENOMINATOR=0;
      NUMERATOR=0;

END;
RUN;
%end;

/*****************************************************************************************************************************
*Reset flag=readmit_flag for use in earlier versions if the macro==> both flag and readmit_flag are in your output.
*****************************************************************************************************************************/
   data work.readmissionflags;
    set work.readmissionflags;
    /* for those that used V3 */
    flag=READMIT_FLAG;
    base_days=days_since_current_INDEX;
     /* If EPIC data then we need to use Inpatient Admit date for admitdate in v6!!! */
     if source='EPIC' then do;
        admitDate=admitDate_Readmission;
     end;
   run;



%if 0 = 0 %then %goto finished;
%finished:
%mend getReadmissionFlagsV6;
```

GetYear_Quarter

```
*******************************************************************
* Thank to Ms. Joanne Cederna                                      *
*   This macro computes the Fiscal and Calendar fields             *
*   Needs an input file with a from_date                           *
*******************************************************************
%macro GetYear_Quarter;
  options mprint mlogic symbolgen;

data WORK.Year_Quarter(drop=MnC DyC YrC);
 set work.Year_Quarter;
 FORMAT QUARTER_DATE QUARTER_START_MONTH QUARTER_END_MONTH DATE9.;
 MONTH=MONTH(from_date);
 YEAR=YEAR(from_date);
    QUARTER_DATE=INTNX('QUARTER',from_date,0,'B');
   QUARTER_START_MONTH=INTNX('QUARTER',from_date,0,'B');
   QUARTER_END_MONTH=INTNX('QUARTER',from_date,0,'E');
QUARTER_END_MONTH=INTNX('MONTH',QUARTER_END_MONTH,0,'B');

  if MONTH=10 or MONTH=11 or MONTH=12 then do;
   FIS_QUARTER=1; FIS_YEAR=YEAR+1;
  end;
  else if MONTH=1 or MONTH=2 or MONTH=3 then do;
   FIS_QUARTER=2; FIS_YEAR=YEAR;
  end;
  else if MONTH=4 or MONTH=5 or MONTH=6 then do;
   FIS_QUARTER=3; FIS_YEAR=YEAR;
  end;
  else if MONTH=7 or MONTH=8 or MONTH=9 then do;
   FIS_QUARTER=4; FIS_YEAR=YEAR;
  end;
  FIS_QUARTER_YEAR=PUT(FIS_YEAR,4.)||'Q'||PUT(FIS_QUARTER,1.);

  if MONTH=1 or MONTH=2 or MONTH=3 then do;
   CAL_QUARTER=1; CAL_YEAR=YEAR;
  end;
  else if MONTH=4 or MONTH=5 or MONTH=6 then do;
   CAL_QUARTER=2; CAL_YEAR=YEAR;
  end;
  else if MONTH=7 or MONTH=8 or MONTH=9 then do;
   CAL_QUARTER=3; CAL_YEAR=YEAR;
  end;
  else if MONTH=10 or MONTH=11 or MONTH=12 then do;
   CAL_QUARTER=4; CAL_YEAR=YEAR;
  end;
  CAL_QUARTER_YEAR=PUT(CAL_YEAR,4.)||'Q'||PUT(CAL_QUARTER,1.);

MnC = month(from_date);
DyC = day(from_date);
YrC = year(from_date);

if Mnc < 10 then do;
   YearMo = strip(YrC) || '-0' || strip(MnC);
end;
else do;
   YearMo = strip(YrC) || '-' || strip(MnC);
end;
run;

/* Let's put in the latest date for each quarter */
PROC SQL;
  CREATE TABLE WORK.Year_Quarter_QUARTER_MONTH AS
  SELECT DISTINCT t1.from_date,
       t1.FIS_QUARTER_YEAR,
       t1.CAL_QUARTER_YEAR
    FROM WORK.Year_Quarter t1
    ORDER BY t1.FIS_QUARTER_YEAR,t1.from_date;
QUIT;

data work.Year_Quarter_QUARTER_MONTH;
 set work.Year_Quarter_QUARTER_MONTH;
 FORMAT LATEST_MONTH DATE9.;
 by FIS_QUARTER_YEAR from_date;
```

```
  if LAST.FIS_QUARTER_YEAR then do;
    LATEST_MONTH=from_date;
    OUTPUT;
  end;
run;

PROC SQL;
  CREATE TABLE WORK.Year_Quarter AS
  SELECT t1.*,
       t2.LATEST_MONTH
    FROM WORK.YEAR_QUARTER t1
         LEFT JOIN WORK.YEAR_QUARTER_QUARTER_MONTH t2 ON (t1.FIS_QUARTER_YEAR = t2.FIS_QUARTER_YEAR)
AND
       (t1.CAL_QUARTER_YEAR = t2.CAL_QUARTER_YEAR);
QUIT;

data work.Year_Quarter;
  set work.Year_Quarter;
  FORMAT LATEST_MONTH_CHAR $20 ;
  if MONTH(LATEST_MONTH)=1 then LATEST_MONTH_CHAR='January';
  else if MONTH(LATEST_MONTH)=2 then LATEST_MONTH_CHAR='February';
  else if MONTH(LATEST_MONTH)=3 then LATEST_MONTH_CHAR='March';
  else if MONTH(LATEST_MONTH)=4 then LATEST_MONTH_CHAR='April';
  else if MONTH(LATEST_MONTH)=5 then LATEST_MONTH_CHAR='May';
  else if MONTH(LATEST_MONTH)=6 then LATEST_MONTH_CHAR='June';
  else if MONTH(LATEST_MONTH)=7 then LATEST_MONTH_CHAR='July';
  else if MONTH(LATEST_MONTH)=8 then LATEST_MONTH_CHAR='August';
  else if MONTH(LATEST_MONTH)=9 then LATEST_MONTH_CHAR='September';
  else if MONTH(LATEST_MONTH)=10 then LATEST_MONTH_CHAR='October';
  else if MONTH(LATEST_MONTH)=11 then LATEST_MONTH_CHAR='November';
  else if MONTH(LATEST_MONTH)=12 then LATEST_MONTH_CHAR='December';
run;

%if 0 = 0 %then %goto finished;
 %finished:
%mend GetYear_Quarter;
```

Get_AMI_Cases

```
/*-----------------------------------------------------------------------------------------------------------------------------
Thanks to Ms. Joanne Cederna

NAME:           Get_AMI_Cases (BEGIN_Date=,END_Date=,CMS_Disease =)

DESCRIPTION:        Pulls CMS Readmission data for Hospital for AMI, CHF, COPD, or PN based upon the CMS definition of
the disease.
            Stationary Start date on data= 01OCT2009
            Data end date is 2 MONTHS BACK (from the run data) at the end of that month
-----------------------------------------------------------------------------------------------------------------------------*/
%macro Get_AMI_Cases (BEGIN_Date=,END_Date=,CMS_Disease =);
options mprint mlogic symbolgen;
%INCLUDE inclcode(IncludeLib_WDATA);
/* Set the end date back two months to the end of the month. This is done for static readmission data.*/
    %let TWO_Months_Ago       = %sysfunc(putn(%sysfunc(intnx(MONTH, '31Aug2012'd, -2,  E)),date9.));

/*Get AMI Readmission DATA Set*/
    %if &CMS_Disease=%STR(AMI) %then %do;
    data WORK.ReadmissionFlags;
    set wData.CMS_DISEASE_POP (where=(from_date >= "&Begin_date."d and from_date <= "&TWO_Months_Ago."d));
    run;
    /*Run thru readmission module*/
    %getReadmissionFlagsV6(30,AMI,,FIRST,);
    %end;

/*Changes were made to the from_date in the readmission macro...Need to re-calculate fiscal year and quarter*/
    Data WORK.Year_Quarter;
    set WORK.ReadmissionFlags;
    run;
    /*Need to re-calculate fiscal year and quarter*/
    %GetYear_Quarter;
    /*rename to Send to wdata2, run thru proc report and drop extra fields*/
    Data MMP_Readmit_&CMS_Disease (drop=     base_days
                            /*flag*/
                            QUARTER_DATE
                            QUARTER_START_MONTH
                            QUARTER_END_MONTH
                            MONTH
                            YEAR
                            FIS_QUARTER
                            FIS_QUARTER_YEAR
                            CAL_QUARTER
                            CAL_YEAR
                            CAL_QUARTER_YEAR
                            YearMo
                            LATEST_MONTH
                            LATEST_MONTH_CHAR);
    set WORK.Year_Quarter;
    run;
    /*Sort output data by E# and Discharge date*/
    Proc sort data=MMP_Readmit_&CMS_Disease.;
      by MEDICAL_RECORD_NUMBER dischDate;
    run;
/*Prepare for proc report*/

            TITLE1 "30 Day All Cause FIRST Readmission";
            TITLE2 "For Patients with &CMS_Disease Index ";
            footnote;
            footnote1 "Center for Performance Improvement";
            FOOTNOTE2 "Source of data:CMS_DISEASE_POP which comes from: EPIC(CPI_EPIC_VISITS_5YRS)/ SCM
(LOBDB5years)";
            footnote3 "Date of Report: &sysdate";
            proc format;
              value blankValue other=' ';
            run;
    *proc report for the monthly sum of numerator, denominator and calculation or rate;
            proc report data=MMP_Readmit_&CMS_Disease
                    out =MMP_Readmit_summary_&CMS_Disease
                    nowd;
              column FIS_YEAR from_date Numerator Denominator Readmission_Rate;

                define from_date        / group "Year-Month" missing;
                format from_date YYMMD7.;
              define FIS_YEAR          / group  "Fiscal Year" order=INTERNAL; /**/
              define Numerator        /  "Readmission" sum;
              define Denominator        /  "Index" sum;

              define Readmission_Rate   / computed format=percent ;
                compute Readmission_Rate;
                  Readmission_Rate = (numerator.sum/denominator.sum);
                endcomp;

            break after FIS_YEAR    / summarize;
          run;

          quit;
%mend Get_AMI_Cases;
```

Cohort_Identification

```
/*---------------------------------------------------------------------------------------------------------
QNet Northeast Health Care Quality - QNET - Person assigned for CMS questions.
https://data.medicare.gov/Hospital-Compare/Hospital-General-Information/v287-28n3

The trace command allows us to see what objects are created by various SAS procedures.
We can then redirect the object outputs to a file for subsequent manipulation.
ods trace on;    ods trace off;

To access DS_PERSON - use MRN mapped to MPI_NUMBER.
---------------------------------------------------------------------------------------------------------*/

LIBNAME Wdata2 BASE "" ;
LIBNAME WdataMMP BASE "" ;
%INCLUDE inclcode(Includelib_wdata)/source2;

options mprint symbolgen VALIDVARNAME=V7;


/*---------------------------------------------------------------------------------------------------------
The following macro graciously provided by Ms. J. Cederna B.S. This macro provides us
all cases of AMI for all ages.
---------------------------------------------------------------------------------------------------------*/
%Get_AMI_Cases (BEGIN_Date=%STR(01Jul2009),END_Date=%STR(31Aug2012),CMS_Disease =AMI);
title; footnote;
proc sql;
  drop table readmissionflags, year_quarter, year_quarter_quarter_month, mmp_readmit_summary_ami;
quit;


/*---------------------------------------------------------------------------------------------------------
We want persons age 65 and older at time of their admission for Heart Attack - Acute Myocardial Infarction (AMI)
This admission is referred to as their Index admission.

What we keep and what we discard:
The data set we use contains pre-EPIC (EMR) data as well as EPIC EMR data. We will not use the EPIC data bacause
we do not have CMS supplied readmission data. We identify EPIC data as that which has values for pat_id.

If a patient has a dicharge disposition of (B - Diacharged to other hospital, G - Against Medical Advice, H - Expired)
for their Index case, we reject these cases as does CMS. We should not have CMS readmission data for this group.

If patient Length of Stay > 365 days we also reject those index cases.

Reject patients who are admitted and discharged the same day for their index case.
---------------------------------------------------------------------------------------------------------*/
%macro Cohort_Identification;

  proc sql;
    create table
      mmp_admission_ami as
    select *
    from
      mmp_readmit_ami
    where
      readmit_flag = 'INDEX'                    /* The AMI Index cases only */
      and age >= 65                             /* Medicare population only */
      and pat_id is null                        /* pre-EPIC data only */
      and DISCHARGE_DISPOSITION not in ('B' 'G' 'H')    /* */
      and LENGTH_OF_STAY <= 365                 /* CMS rejects stays of over 1 year */
      and admitDate ^= dischDate                /* For index cases - Same day discharge rejected */
      and ACTUAL_TOTAL_CHARGE > 0               /* CMS requirment of total charges > 0. */
    order by MEDICAL_RECORD_NUMBER, admitDate;
  quit;


  /*---------------------------------------------------------------------------------------------------------
  Retrieve CMS supplied data that has the AMI and readmission data whether a readmission exists or not.
  ---------------------------------------------------------------------------------------------------------*/
  proc import
    datafile    = '\\sasmeta2\projects\EG Projects\Knowld1\Thesis\200009_I_Readmission_HSR_dmk.XLS'
    out         = cms_data_200009_JULY2012
    dbms        = excel replace;
    getnames    = yes;
    mixed       = yes;
    sheet       = "I.3 30-Day R Discharges";
  run;


  /*-----------------------------------------------------------------------------------------------------------------
  Data cleanup: Build a proper admit date and drop the leading 0 from the CMS Medical Record Number
  -----------------------------------------------------------------------------------------------------------------*/
  data cms_data_200009_JULY2012;
    set cms_data_200009_JULY2012 (where=(Exclusion_Reason = '0' and Measure='AMI' and
upcase(Planned_Readmission____Yes_No_) = 'NO' and length(trim(MEDICAL_RECORD_NUMBER)) > 2));
    length admitDate 8.;
    format admitDate Date9.;

    admitDate =
mdy(input(substr(Admission_Date_of_Index_Stay,1,2),2.0),input(substr(Admission_Date_of_Index_Stay,4,2),2.0),input(substr(Ad
mission_Date_of_Index_Stay,7,4),4.0));
    MEDICAL_RECORD_NUMBER = substr(MEDICAL_RECORD_NUMBER,2,length(MEDICAL_RECORD_NUMBER)-1);
  run;
  proc sort data=cms_data_200009_JULY2012; by MEDICAL_RECORD_NUMBER admitDate; run;


  /*-----------------------------------------------------------------------------------------------------------------
  Merge the AMI Index cases with CMS readmission data by medical record number and admit date of the Index case.
  We place mismatches into their appropriate output file
  -----------------------------------------------------------------------------------------------------------------*/
  data admissions_w_readmissions nomatch_ami nomatch_cms;
    merge    mmp_admission_ami (in = a)
          cms_data_200009_JULY2012 (in = b);
    by MEDICAL_RECORD_NUMBER admitDate;

    if a and b then output admissions_w_readmissions;
    else if a and not b then output nomatch_ami;
    else if not a and b then output nomatch_cms;
  run;

  proc sql noprint;
    drop table
      MATCH_MRN,                        /* created in call Get_AMI_Cases */
      MMP_READMIT_AMI,                    /* created in call Get_AMI_Cases */
      mmp_admission_ami,
      cms_data_200009_JULY2012,
      nomatch_ami,
      nomatch_cms;
  quit;

  proc sql; select * from admissions_w_readmissions where age >= 100; quit;
%mend Cohort_Identification;

%Cohort_Identification;

/*-----------------------------------------------------------------------------------------------------------------
NOTES: Further investigation matching only on MRN produced no matches between hospital and CMS data.


MS-DRGS

1970's Yale University developed a classification system to relate the
resource consumption of an inpatient stay (based on clinical
conditions of the patient). This classification system was referred
to as Diagnosis Related Groups (DRGs).

1980's Medicare adopted a version of the Yale University's DRG system
into a hospital inpatient prospective payment system.
Other payers soon followed Medicare's new payment system.
Some adopted Medicare's DRG system. Other payers, and some
states, adopted different versions of the original DRG system.
Elevated concern of coding guidelines and compliance.

1990's Medicare adopts additional prospective payment systems for other
types of claims (e.g. hospital outpatient (APGs); skilled nursing
(RUGs); inpatient rehabilitation (CMGs), etc.)

2000's March, 2005: Medicare Payment Advisory Commission
(MedPAC) published Report to Congress: Physician-owned
```

Specialty Hospitals which included recommendations to:
- Improve payment accuracy in the hospital inpatient
prospective payment system by:
    Refining the current DRGs to more fully capture
    differences in severity of illness among patients;
    Basing the DRG relative weights on the
    estimated cost of providing care rather than
    charges, and
    Basing the weights on the national average of
    hospitals' relative values in each DRG

July, 2007: RAND Report published: Evaluation of Severity-
Adjusted DRG Systems evaluating 6 DRG system (including the Medicare-severity DRGs proposed by CMS)

August 22, 2007: CMS publishes the Inpatient Prospective Payment System Update for FY 2008 in the Federal Register (Final Rule).

WHAT IS A DIAGNOSIS RELATED GROUP (DRG)?
A grouping of disease and disorders into medically meaningful sets as developed by the Centers for Medicare & Medicaid Services (CMS).
This reimbursement system consists of established payment levels for groupings of patients according to medically meaningful characteristics. There are six major criteria.
which are utilized in assigning a particular admission to a specific DRG. These consist of:
- Patient's principal diagnosis
- Procedures performed on the patient
- Patient's age
- Patient's gender
- Patient's discharge status
- Multiple diagnoses, complications or comorbid conditions.

--*/

Variable_Additions

```
/*------------------------------------------------------------------------------------------------------------
We attach the marital status. Our intent is to separate the statuses so that they might
indicate someone home or not.
------------------------------------------------------------------------------------------------------------*/
proc sql;
   create table
      admissions_w_readmits as
   select
      admissions_w_readmissions.*, DS_ENCOUNTER.MARITAL_STATUS
   from
      admissions_w_readmissions left join Wdata2.DS_ENCOUNTER on
         admissions_w_readmissions.ENCOUNTER_NUMBER = DS_ENCOUNTER.ENCOUNTER_NUMBER;
quit;


/*------------------------------------------------------------------------------------------------------------
Some of the following additions are time of day/week specific variables. We also change
some variables into binary variables. We drop DISCH_HOME as our reference variable.
We set it in code only for documentation purposes.
------------------------------------------------------------------------------------------------------------*/
data admissions_w_readmissions (drop = DISCH_HOME DISCHARGE_DISPOSITION);
   set admissions_w_readmits;

   length AdmitDay DischargeDay 3.;
   format AdmitDay DischargeDay DOWNAME9.;

   AdmitDay          = 0;
   DischargeDay      = 0;
   WeekendAdmit      = 0;
   WeekendDischarge  = 0;
   DayAdmit          = 0;
   DayDischarge      = 0;

   AdmitDay = weekday(admitDate);                    /* Capture Day of Week for Admission. */
   DischargeDay = weekday(dischDate);                 /* Capture Day of Week for Discharge. */

   if AdmitDay in (1, 2) then WeekendAdmit       = 1;         /* Was this a weekend admit? */
   if DischargeDay in (1, 2) then WeekendDischarge = 1;            /* Was this a weekend discharge? */

   if ADMIT_TIME ^= 0 and (70000 <= ADMIT_TIME <= 190000) then DayAdmit = 1;          /* Day admit between 7 and 7. */
   if dischTime ^= '00:00't and ('07:00't <= dischTime <= '19:00't) then DayDischarge = 1;   /* Day discharge between 7 and 7. */

   MALE = (SEX = 'M');

   MARITAL_STAT = 0;                                 /* Split Marital Statuses into with/without someone. */
   if MARITAL_STATUS = ('D') then MARITAL_STAT = 0;
   if MARITAL_STATUS = ('M') then MARITAL_STAT = 1;
   if MARITAL_STATUS = ('P') then MARITAL_STAT = 1;
   if MARITAL_STATUS = ('S') then MARITAL_STAT = 0;
   if MARITAL_STATUS = ('U') then MARITAL_STAT = 0;
   if MARITAL_STATUS = ('W') then MARITAL_STAT = 0;
   if MARITAL_STATUS = ('X') then MARITAL_STAT = 0;

   DISCH_HOSPICE    = 0;                             /* Discharge split into Hospice, Home and Other Medical Care. */
   DISCH_HOME       = 0;
   DISCH_OTHER_CARE = 0;
   if DISCHARGE_DISPOSITION in ('50' '51')       then DISCH_HOSPICE   = 1;
   if DISCHARGE_DISPOSITION in ('A' 'F' 'I')     then DISCH_HOME      = 1;
   if DISCHARGE_DISPOSITION in ('C' 'P' 'S' 'T')  then DISCH_OTHER_CARE = 1;

   MEDICAL_SUBSERVICE    = 0;                        /* Make a binary variable from admitting subservice. */
   MEDICAL_SUBSERVICE = (ADMIT_SUBSERVICE = 'M');
run;

proc freq data=admissions_w_readmissions;
   table AdmitDay;
   table DischargeDay;
   table WeekendAdmit;
   table WeekendDischarge;
   table DayAdmit;
   table DayDischarge;

   table MALE;
   table MARITAL_STAT;
   table DISCH_HOSPICE;
   table DISCH_OTHER_CARE;
   table MEDICAL_SUBSERVICE;
run;


   proc sql noprint;
     drop table
        admissions_w_readmits;
   quit;


/*
Discharge Disposition defined choices:
1  A   Home        Discharge Home
1  B   OTHER HOSP   DISCH OTH HOSP AS IP
1  C   SNF         DISCH SNF CODE 61
1  D   ICF         DISCHARGED ICF
1  E   OTHER HCF    DISCH CANCER/CTR/CHLD
1  F   HHS         DISCHARGED HOME HEALTH
1  G   AMA         DISCHARGED AMA
1  H   EXPIRED     EXPIRED
1  J   STILL PAT    STILL PATIENT
1  M   DISC-OP      DISCHARGED OTHER HOSP OP
1  P   DISC-REHAB   DISCHARGED OTHER REHAB
1  Q   DISC LTC     DISCHARGED LONG TERM CARE
1  R   Disch Fed    Discharged Federal Hospit
1  S   Disch Psyc   Discharged Psych
1  T   Other Hlth   Other Hlth Care
1  50   DC Hospice   Discharged Hospice/Home
1  51   DC HospicF   Discharged Hospice/Fac
1  I  DC CourLaw   Discharged Court/Law Fac
1  ZZ   IP to EPIC  Inhouse converted EPIC


Discharge Disposition represented in our data:
DISCHARGE_DISPOSITION    Frequency   Percent Cumulative Frequency Cumulative Percent
      50              3      0.29          3         0.29
      51              4      0.38          7         0.67
      A             571     54.43        578          55.10
      C             160     15.25        738          70.35
      F             271     25.83       1009          96.19
      I               1     0.10       1010         96.28
      P              36      3.43       1046          99.71
      S               1      0.10       1047          99.81
      T               2      0.19       1049         100.00


Marital Statuses represented in our data:
1  D  DIVORCED        DIVORCED       82
2  M  MARRIED        MARRIED       530
3  P  LIFE PRTNR      LIFE PARTNER    10
4  S  SINGLE        SINGLE       109
5  U  UNKNOWN        UNKNOWN        2
6  W  WIDOWED        WIDOWED       312
7  X  SEPARATED       SEPARATED       4

We separate Marital Status to indicate someone is present at home with patient - married/life partner - all others treated as 'alone'


Race:
99.24% are white patients. We do not include race.
*/
```

Initial_Variable_Reduction

```
%macro initial_variable_reduction;

  /*-----------------------------------------------------------------------------------------------------------------
  Initial Variable Reduction
  There are columns on the source file that are always empty - as well as non-empty
  columns that never vary. The next few steps remove these invariant columns
  Start by capturing a list of the columns from the source file
  -----------------------------------------------------------------------------------------------------------------*/
  proc contents data = admissions_w_readmissions out = admit_w_readmit_cols noprint; run;


  /*-----------------------------------------------------------------------------------------------------------------
  Now count the number of columns found. Place the names of the columns into macro
  variables. Then perform a do loop using the count as your upper end of the loop.
  -----------------------------------------------------------------------------------------------------------------*/
  proc sql noprint;
     select left(put(count(name), 5.))
     into :colcnt
     from  admit_w_readmit_cols;

     select left(trim(name))
     into :col1 - :col&colcnt.
     from  admit_w_readmit_cols;
  quit;


  /*-----------------------------------------------------------------------------------------------------------------
  Loop for all column names - perform a frequency through each column name and if the
  frequency result is 1 row. it means the column values are restricted to 1 value - they are
  invariant. So we reject those. All other column names are concatenated into a string –
  keep_vars - to identify those columns we keep in our data.

  The ODS statement below directs the frequency output to a file we can interrogate.
  -----------------------------------------------------------------------------------------------------------------*/
  %let keep_vars = ;

  %do i = 1 %to &colcnt.;
     ODS OUTPUT OneWayFreqs = _freqs;          /* We will interrogate this frequency file for # of rows. */
        proc freq data = admissions_w_readmissions;
           table &&col&i..;
        run;


        proc sql noprint;
           select left(put(count(*), 5.))
           into :freqcnt
           from _freqs;
        quit;

        %if &freqcnt. > 1 %then %do;          /* This column has varying values to keep this column. */
           %let keep_vars = &keep_vars. &&col&i..;
        %end;
     ODS OUTPUT close;
  %end;

  %put &keep_vars.;                          /* QA step - will show all columns being kept - see the log. */

  /*-----------------------------------------------------------------------------------------------------------------
  The steps above identified those columns having variation. They are specified in a string
   titled keep_vars. The next step then states to keep only those variables. The
   admissions_w_readmissions file has 188 variables.
  -----------------------------------------------------------------------------------------------------------------*/
  data cohort;
     set admissions_w_readmissions (keep = &keep_vars );
  run;
  proc sort data = cohort; by MEDICAL_RECORD_NUMBER AdmitDate; run;


  /*-----------------------------------------------------------------------------------------------------------------
  Upon review additional columns are dropped. Our keep list is shorter so we designate a
  keep statement. This following step reduces the variable count to 32.
  -----------------------------------------------------------------------------------------------------------------*/
  data cohort;
    set cohort (drop =
                   ACTUAL_TOTAL_CHARGE
                   ADMIT_DATE
                   ADMIT_DIAGNOSIS
                   ADMIT_FISCAL_YEAR
                   ADMIT_PHYSICIAN
                   ADMIT_SOURCE
                   ADMIT_TYPE
                   AMI_READMIT_Qualifier
                   ATTENDING_PHYSICIAN
                   Admission_Date_of_Index_Stay
                   Beneficiary_DOB
                   CURRENT_INDEX_DATE
                   days_since_FIRST_INDEX
                   DISCHARGE_DATE
                   DISCHARGE_FISCAL_YEAR
                   Discharge_Date_of_Index_Stay
                   Discharge_Date_of_Readmission
                   EVAL_AMI_READMIT_CHANGE
                   FINANCIAL_CLASS
                   FIRST_INDEX_DATE
                   FIS_YEAR
                   HICNO_SSN
                   ID_Number
                   LAST_UPDATE_DATE
                   MARITAL_STATUS
                   NET_REVENUE__CCM_
                   OTHER_PHYSICIAN_1
                   PAYOR_CODE_1
                   PRODUCT_LINE_CODE
                   Principal_Discharge_Diagnosis_o0
                   Principal_Discharge_Diagnosis_of
                   Provider_ID_of_Readmitting_Hospi
                   RACE
                   READMITS_KEEP_FLAG
                   SECONDARY_DIAGNOSIS
                   SEC_DIAG_USER_DATE
                   TIME_OF_DISCHARGE
                   ZIP_CODE
                   ZIP_CODE_NUMERIC
                   admitMD
                   );

    readmit_30days = 0;
    readmit_30days = (upcase(Unplanned_Readmission_within_30_) = 'YES');
  run;


  /*------------------------------------------------------------------------------------------------
  Drop interim tables.
  ------------------------------------------------------------------------------------------------*/
  proc sql noprint;
    drop table admit_w_readmit_cols;
  quit;

%mend initial_variable_reduction;

%initial_variable_reduction;
```

Cohort_w_Facts

```
/*---------------------------------------------------------------------------------------------------------
Retrieve all procedures for the AMI encounters for our Cohort.
---------------------------------------------------------------------------------------------------------*/
  proc sql;
    create table
      DS_ENCOUNTER_PROCEDURE as
    select
      DS_ENCOUNTER_PROCEDURE.ENCOUNTER_NUMBER,
DS_ENCOUNTER_PROCEDURE.PROCEDURE_CODE__ENCTR_,
      DS_ICD9_PROC_TABLE_VALUE.ICD9_PROC_DESC_50,
DS_ENCOUNTER_PROCEDURE.PRINCIPLE_SECONDARY_PROC
    from
      Wdata2.DS_ENCOUNTER_PROCEDURE inner join Wdata2.DS_ICD9_PROC_TABLE_VALUE on
        DS_ENCOUNTER_PROCEDURE.PROCEDURE_CODE__ENCTR_ = DS_ICD9_PROC_TABLE_VALUE.ICD9_PROC
    where
      DS_ENCOUNTER_PROCEDURE.ENCOUNTER_NUMBER in (select ENCOUNTER_NUMBER from cohort);
  quit;
  proc sort data=DS_ENCOUNTER_PROCEDURE NODUP; by _ALL_; run;


/*---------------------------------------------------------------------------------------------------------
Prepare the procedures in our cohort by removing embedded periods in the ICD 9 procedure codes. These codes will
then match with the CCS groupers.
---------------------------------------------------------------------------------------------------------*/
  data DS_ENCOUNTER_PROCEDURE(drop=period_loc);
    set DS_ENCOUNTER_PROCEDURE;
    length ICD_9_CM_CODE $5 ;

    period_loc = index(PROCEDURE_CODE__ENCTR_,'.');
    if period_loc = 0  then ICD_9_CM_CODE = PROCEDURE_CODE__ENCTR_;
    else ICD_9_CM_CODE = substr(PROCEDURE_CODE__ENCTR_,1,period_loc-1) ||
substr(PROCEDURE_CODE__ENCTR_,period_loc+1,length(PROCEDURE_CODE__ENCTR_)-period_loc);
  run;


/*---------------------------------------------------------------------------------------------------------
Retrieve all non-primary diagnoses for the AMI encounters for our Cohort.
---------------------------------------------------------------------------------------------------------*/
  proc sql;
    create table DS_ENCOUNTER_DIAGNOSIS as
    select DS_ENCOUNTER_SECONDARY_DIAG.ENCOUNTER_NUMBER,
DS_ENCOUNTER_SECONDARY_DIAG.SECONDARY_DIAG_SEQUENCE,
DS_ENCOUNTER_SECONDARY_DIAG.SECONDARY_DIAGNOSIS, DS_ICD9_DIAG_TABLE_VALUE.ICD9_DIAG_DESC_50,
DS_ENCOUNTER_SECONDARY_DIAG.PRESENT_ON_ADMISSION_FLAG
    from Wdata2.DS_ENCOUNTER_SECONDARY_DIAG inner join Wdata2.DS_ICD9_DIAG_TABLE_VALUE
      on DS_ENCOUNTER_SECONDARY_DIAG.SECONDARY_DIAGNOSIS =
DS_ICD9_DIAG_TABLE_VALUE.ICD9_DIAGNOSIS
    where DS_ENCOUNTER_SECONDARY_DIAG.SECONDARY_DIAG_SEQUENCE ^= 1
      and DS_ENCOUNTER_SECONDARY_DIAG.ENCOUNTER_NUMBER in
        (select ENCOUNTER_NUMBER from cohort);
  quit;
  proc sort data=DS_ENCOUNTER_DIAGNOSIS NODUP; by _ALL_; run;


/*---------------------------------------------------------------------------------------------------------
Prepare the diagnoses in our cohort by removing embedded periods in the ICD 9 diagnosis codes. These codes will
then match with the CCS groupers.
---------------------------------------------------------------------------------------------------------*/
  data DS_ENCOUNTER_DIAGNOSIS(drop=period_loc);
    set DS_ENCOUNTER_DIAGNOSIS;
    length ICD_9_CM_CODE $5 ;

    period_loc = index(SECONDARY_DIAGNOSIS,'.');
    if period_loc = 0  then ICD_9_CM_CODE = SECONDARY_DIAGNOSIS;
    else ICD_9_CM_CODE = substr(SECONDARY_DIAGNOSIS,1,period_loc-1) ||
substr(SECONDARY_DIAGNOSIS,period_loc+1,length(SECONDARY_DIAGNOSIS)-period_loc);
  run;
```

Cohort_Facts_Grouped_by_CCS

```
/*-------------------------------------------------------------------------------------------------------------
Variable Reduction.
The attack for variable reduction is:
   Remove columns that were identified by MMP Quality team members.
   Group similar diagnosis and procedure codes using an industry accepted grouper dataset (Recommended by C. Peng)
   Use CCS groupers for both Diagnosis and Procedure codes found in our data.
   Only those groups of diagnosis and procedure codes represented in the cohort at .3% or higher are kept.

   CCS Groupers acquired from Healthcare Cost and Utilization Project (HCUP) - Grouper Data downloaded 10/28/2013
   at site: http://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccsfactsheet.jsp (site last visited - 10/30/2013)
   Note the grouper data has 2 files for diagnosis and 2 files for procedure grouper data. One of the two files has
   the detail data and the other contains the full label of the grouper. These 2 files will be merged.
   We use single level categories for our work with groupers.

Developed at the Agency for Healthcare Research and Quality (AHRQ), the Clinical Classifications Software (CCS) is a
tool for clustering patient diagnoses and procedures into a manageable number of clinically meaningful categories.
CCS collapses diagnosis and procedure codes from the International Classification of Diseases, 9th Revision,
Clinical Modification (IDC-9-CM), which contains more than 14,000 diagnosis codes and 3,900 procedure codes.
Without the CCS tool, the large number of ICD-9-CM codes makes statistical analysis and reporting difficult and
time-consuming.
CCS consists of two related classification systems, single-level and multi-level, which are designed to meet different
needs. Single-level CCS is most useful for ranking of diagnoses and procedures and for direct integration into risk
adjustment and other software. Multi-level CCS is ideal for evaluating larger aggregations of conditions and procedures
or exploring these groupings in greater detail.
-------------------------------------------------------------------------------------------------------------*/


/*-------------------------------------------------------------------------------------------------------------
Import the detailed and summary CCS procedure groupers.
-------------------------------------------------------------------------------------------------------------*/
proc import
   datafile    = '\\sasmeta2\projects\EG Projects\Knowld1\Thesis\CCS\Single_Level_CCS_2014\$prref 2014.csv'
   out         = ccs_proc_groupers
   dbms         = csv replace;
   getnames      = yes;
run;
proc sort data=ccs_proc_groupers; by CCS_CATEGORY; run;

proc import
   datafile    = '\\sasmeta2\projects\EG Projects\Knowld1\Thesis\CCS\Single_Level_CCS_2014\prlabel 2009.csv'
   out         = ccs_proc_labels
   dbms         = csv replace;
   getnames      = yes;
run;
proc sort data=ccs_proc_labels(rename=(CCS_PROCEDURE_CATEGORIES=CCS_CATEGORY)); by CCS_CATEGORY; run;
proc sql noprint; delete from  ccs_proc_labels where CCS_CATEGORY in ('*****','.A','.Z',' '); quit;


data ccs_procedures;
   merge ccs_proc_groupers ccs_proc_labels;
   by CCS_CATEGORY;
run;


/*-------------------------------------------------------------------------------------------------------------
Merge the cohort procedure codes with those in the grouper detail file just imported.
-------------------------------------------------------------------------------------------------------------*/
proc sql;
   create table
      DS_ENCOUNTER_PROCEDURE_GROUPED as
   select
      DS_ENCOUNTER_PROCEDURE.*,
      ccs_procedures.CCS_CATEGORY,
      ccs_procedures.CCS_CATEGORY_DESCRIPTION,
      ccs_procedures.ICD_9_CM_CODE_DESCRIPTION,
      ccs_procedures.CCS_PROCEDURE_CATEGORIES_LABELS
   from
      DS_ENCOUNTER_PROCEDURE left join ccs_procedures on
         DS_ENCOUNTER_PROCEDURE.ICD_9_CM_CODE = ccs_procedures.ICD_9_CM_CODE
   ;
quit;
```

```
/*-----------------------------------------------------------------------------------------------------------
Now review frequencies of grouped procedures for our cohort.
-----------------------------------------------------------------------------------------------------------*/
title 'Procedure CCS Groupers.';
proc freq data = DS_ENCOUNTER_PROCEDURE_GROUPED order = freq;
   table CCS_CATEGORY/missing;
run;



/*-----------------------------------------------------------------------------------------------------------
Import the detailed and summary CCS diagnosis groupers.
-----------------------------------------------------------------------------------------------------------*/
proc import
   datafile    = '\\sasmeta2\projects\EG Projects\Knowld1\Thesis\CCS\Single_Level_CCS_2014\$dxref 2013.csv'
   out         = ccs_dx_groupers
   dbms        = csv replace;
   getnames    = yes;
run;
proc sort data=ccs_dx_groupers; by CCS_CATEGORY; run;

proc import
   datafile    = '\\sasmeta2\projects\EG Projects\Knowld1\Thesis\CCS\Single_Level_CCS_2014\dxlabel 2013.csv'
   out         = ccs_dx_labels
   dbms        = csv replace;
   getnames    = yes;
run;
proc sort data=ccs_dx_labels(rename=(CCS_DIAGNOSIS_CATEGORIES=CCS_CATEGORY)); by CCS_CATEGORY; run;
proc sql noprint; delete from  ccs_dx_labels where CCS_CATEGORY in ('*****' '.A' '.Z' ' '); quit;


data ccs_diagnoses;
   merge ccs_dx_groupers ccs_dx_labels;
   by CCS_CATEGORY;
run;


/*-----------------------------------------------------------------------------------------------------------
Merge the cohort diagnosis codes with those in the grouper detail file just imported.
-----------------------------------------------------------------------------------------------------------*/
proc sql;
   create table
      DS_ENCOUNTER_DIAGNOSIS_GROUPED as
   select
      DS_ENCOUNTER_DIAGNOSIS.*,
      ccs_diagnoses.CCS_CATEGORY,
      ccs_diagnoses.CCS_CATEGORY_DESCRIPTION,
      ccs_diagnoses.ICD_9_CM_CODE_DESCRIPTION,
      ccs_diagnoses.OPTIONAL_CCS_CATEGORY,
      ccs_diagnoses.OPTIONAL_CCS_CATEGORY_DESCRIPTI,
      ccs_diagnoses.CCS_DIAGNOSIS_CATEGORIES_LABELS
   from DS_ENCOUNTER_DIAGNOSIS left join ccs_diagnoses on
      DS_ENCOUNTER_DIAGNOSIS.ICD_9_CM_CODE = ccs_diagnoses.ICD_9_CM_CODE
   ;
quit;


/*-----------------------------------------------------------------------------------------------------------
Now review frequencies of grouped diagnoses for our cohort.
-----------------------------------------------------------------------------------------------------------*/
title 'Diagnosis CCS Groupers.';
proc freq data=DS_ENCOUNTER_DIAGNOSIS_GROUPED order = freq;
   table CCS_CATEGORY/missing;
run;



/*-----------------------------------------------------------------------------------------------------------
How many distinct diagnosis and procedure codes do we have?
-----------------------------------------------------------------------------------------------------------*/
title 'Distinct Diagnosis and Procedure codes.';
proc sql; select count (distinct ICD_9_CM_CODE) from DS_ENCOUNTER_DIAGNOSIS_GROUPED ; quit;
proc sql; select count (distinct ICD_9_CM_CODE) from DS_ENCOUNTER_PROCEDURE_GROUPED ; quit;
```

```
  /*-----------------------------------------------------------------------------------------------------
  Drop interim tables.
  ------------------------------------------------------------------------------------------------------*/
  proc sql noprint;
     drop table
        DS_ENCOUNTER_DIAGNOSIS,
        DS_ENCOUNTER_PROCEDURE,
        ccs_dx_groupers,
        ccs_dx_labels,
/*         ccs_diagnoses.*/
        ccs_proc_groupers,
        ccs_proc_labels
/*         ccs_procedures*/
        ;
  quit;

title;
```

Elixhauser_Comorbdity

```
/*------------------------------------------------------------------------------------------------------------
From the website - http://mchp-appserv cpe umanitoba ca/Upload/SAS/_ElixhauserICD9CM sas txt

Original work - Comorbidity measures for use with administrative data
A Elixhauser. C Steiner. DR Harris. RM Coffey - Medical care. Volume 36. Number 1. pp 8-27


Van Walraven. Carl. Austin. Peter C ; Jennings. Alison. Quan. Hude. Forster. Alan J. (2009)
"A Modification of the Elixhauser Comorbidity Measures into a Point System for Hospital Death Using Administrative Data"
Medical Care 47 (6): 626–33. doi 10.1097/MLR 0b013e31819432e5. PMID 19433995

See: Systematic review of comorbidity indices for administrative data.
Med Care. 2012 Dec;50(12):1109-18. doi: 10.1097/MLR.0b013e31825f64d0.
Sharabiani MT. Aylin P. Bottle A.
------------------------------------------------------------------------------------------------------------*/

data  DS_ENCOUNTER_DIAGNOSIS_ELIX(drop=i);
   set DS_ENCOUNTER_DIAGNOSIS_GROUPED (where=(PRESENT_ON_ADMISSION_FLAG = ('1') or
PRESENT_ON_ADMISSION_FLAG = ('Y')));

  /*----------------------------------------------------------------------------------------------------------
  set up array for individual ELX group counters
  ----------------------------------------------------------------------------------------------------------*/
  array ELX_GRP (31) ELX_GRP_1 - ELX_GRP_31;

  /*----------------------------------------------------------------------------------------------------------
  initialize all ELX group counters to zero
  ----------------------------------------------------------------------------------------------------------*/
  do i = 1 to 31;
   ELX_GRP(i) = 0;
  end;
  TOT_GRP = 0;

  /*----------------------------------------------------------------------------------------------------------
  Congestive Heart Failure
  ----------------------------------------------------------------------------------------------------------*/
  if ICD_9_CM_CODE IN: ('39891','40201','40211','40291','40401','40403','40411','40413','40491',
  '40493','4254','4255','4257','4258','4259','428') then do;
     ELX_GRP_1 = 1;
     TOT_GRP = TOT_GRP + (7);
  end;
  LABEL ELX_GRP_1='Congestive Heart Failure';

  /*----------------------------------------------------------------------------------------------------------
  Cardiac Arrhythmia
  ----------------------------------------------------------------------------------------------------------*/
  if ICD_9_CM_CODE IN: ('4260','42613','4267','4269','42610','42612','4270','4271','4272','4273',
         '4274','4276','4278','4279','7850','99601','99604','V450','V533') then do;
     ELX_GRP_2 = 1;
     TOT_GRP = TOT_GRP + (5);
  end;
     LABEL ELX_GRP_2='Cardiac Arrhythmia';

  /*----------------------------------------------------------------------------------------------------------
  Valvular Disease
  ----------------------------------------------------------------------------------------------------------*/
  if ICD_9_CM_CODE IN: ('0932','394','395','396','397','424','7463','7464','7465','7466','V422','V433')
         then do;
     ELX_GRP_3 = 1;
     TOT_GRP = TOT_GRP + (-1);
  end;
     LABEL ELX_GRP_3='Valvular Disease';

  /*----------------------------------------------------------------------------------------------------------
  Pulmonary Circulation Disorders
  ----------------------------------------------------------------------------------------------------------*/
  if ICD_9_CM_CODE IN: ('4150','4151','416','4170','4178','4179') then do;
     ELX_GRP_4 = 1;
     TOT_GRP = TOT_GRP + (4);
  end;
     LABEL ELX_GRP_4='Pulmonary Circulation Disorders';
```

```
/*---------------------------------------------------------------------------------------------------------------
Peripheral Vascular Disorders
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('0930','4373','440','441','4431','4432','4438','4439','4471','5571','5579','V434')
        then do;
   ELX_GRP_5 = 1;
   TOT_GRP = TOT_GRP + (2);
end;
   LABEL ELX_GRP_5='Peripheral Vascular Disorders';

/*---------------------------------------------------------------------------------------------------------------
Hypertension Uncomplicated
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('401') then do;
   ELX_GRP_6 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
   LABEL ELX_GRP_6='Hypertension Uncomplicated';

/*---------------------------------------------------------------------------------------------------------------
Hypertension Complicated
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('402','403','404','405') then do;
   ELX_GRP_7 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
   LABEL ELX_GRP_7='Hypertension Complicated';

/*---------------------------------------------------------------------------------------------------------------
Paralysis
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('3341','342','343','3440','3441','3442','3443','3444','3445','3446','3449')
        then do;
   ELX_GRP_8 = 1;
   TOT_GRP = TOT_GRP + (7);
end;
  LABEL ELX_GRP_8='Paralysis';

/*---------------------------------------------------------------------------------------------------------------
Other Neurological Disorders
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('3319','3320','3321','3334','3335','33392','334','335','3362','340','341',
        '345','3481','3483','7803','7843') then do;
   ELX_GRP_9 = 1;
   TOT_GRP = TOT_GRP + (6);
end;
  LABEL ELX_GRP_9='Other Neurological Disorders';

/*---------------------------------------------------------------------------------------------------------------
Chronic Pulmonary Disease
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('4168','4169','490','491','492','493','494','495','496','500','501','502',
        '503','504','505','5064','5081','5088') then do;
   ELX_GRP_10 = 1;
   TOT_GRP = TOT_GRP + (3);
end;
  LABEL ELX_GRP_10='Chronic Pulmonary Disease';

/*---------------------------------------------------------------------------------------------------------------
Diabetes Uncomplicated
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2500','2501','2502','2503') then do;
   ELX_GRP_11 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
  LABEL ELX_GRP_11='Diabetes Uncomplicated';

/*---------------------------------------------------------------------------------------------------------------
Diabetes Complicated
---------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2504','2505','2506','2507','2508','2509') then do;
   ELX_GRP_12 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
  LABEL ELX_GRP_12='Diabetes Complicated';
```

```
/*-----------------------------------------------------------------------------------------------------------
Hypothyroidism
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2409','243','244','2461','2468') then do;
   ELX_GRP_13 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
  LABEL ELX_GRP_13='Hypothyroidism';

/*-----------------------------------------------------------------------------------------------------------
Renal Failure
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('40301','40311','40391','40402','40403','40412','40413','40492','40493',
        '585','586','5880','V420','V451','V56') then do;
   ELX_GRP_14 = 1;
   TOT_GRP = TOT_GRP + (5);
end;
  LABEL ELX_GRP_14='Renal Failure';

/*-----------------------------------------------------------------------------------------------------------
Liver Disease
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('07022','07023','07032','07033','07044','07054','0706','0709','4560','4561',
        '4562','570','571','5722','5723','5724','5728','5733','5734','5738','5739','V427')
        then do;
   ELX_GRP_15 = 1;
   TOT_GRP = TOT_GRP + (11);
end;
  LABEL ELX_GRP_15='Liver Disease';

/*-----------------------------------------------------------------------------------------------------------
Peptic Ulcer Disease excluding bleeding
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('5317','5319','5327','5329','5337','5339','5347','5349')
        then do;
   ELX_GRP_16 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
  LABEL ELX_GRP_16='Peptic Ulcer Disease excluding bleeding';

/*-----------------------------------------------------------------------------------------------------------
AIDS/HIV
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('042','043','044')  then do;
   ELX_GRP_17 = 1;
   TOT_GRP = TOT_GRP + (0);
end;
  LABEL ELX_GRP_17='AIDS/HIV';

/*-----------------------------------------------------------------------------------------------------------
Lymphoma
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('200','201','202','2030','2386') then do;
   ELX_GRP_18 = 1;
   TOT_GRP = TOT_GRP + (9);
end;
  LABEL ELX_GRP_18='Lymphoma';

/*-----------------------------------------------------------------------------------------------------------
Metastatic Cancer
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('196','197','198','199') then do;
   ELX_GRP_19 = 1;
   TOT_GRP = TOT_GRP + (12);
end;
  LABEL ELX_GRP_19='Metastatic Cancer';

/*-----------------------------------------------------------------------------------------------------------
Solid Tumor without Metastasis
------------------------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('140','141','142','143','144','145','146','147','148','149','150','151','152',
        '153','154','155','156','157','158','159','160','161','162','163','164','165','166','167',
        '168','169','170','171','172','174','175','176','177','178','179','180','181','182','183',
        '184','185','186','187','188','189','190','191','192','193','194','195')
        then do;
    ELX_GRP_20 = 1;
    TOT_GRP = TOT_GRP + (4);
 end;
   LABEL ELX_GRP_20='Solid Tumor without Metastasis';

/*-----------------------------------------------------------------------------------------------
Rheumatoid Arthritis/collagen
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('446','7010','7100','7101','7102','7103','7104','7108','7109','7112','714',
        '7193','720','725','7285','72889','72930') then do;
    ELX_GRP_21 = 1;
    TOT_GRP = TOT_GRP + (0);
end;
   LABEL ELX_GRP_21='Rheumatoid Arthritis/collagen';

/*-----------------------------------------------------------------------------------------------
Coagulopathy
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('286','2871','2873','2874','2875')  then do;
    ELX_GRP_22 = 1;
    TOT_GRP = TOT_GRP + (3);
end;
   LABEL ELX_GRP_22='Coagulopathy';

/*-----------------------------------------------------------------------------------------------
Obesity
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2780') then do;
    ELX_GRP_23 = 1;
    TOT_GRP = TOT_GRP + (-4);
end;
   LABEL ELX_GRP_23='Obesity';

/*-----------------------------------------------------------------------------------------------
Weight Loss
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('260','261','262','263','7832','7994') then do;
    ELX_GRP_24 = 1;
    TOT_GRP = TOT_GRP + (6);
end;
   LABEL ELX_GRP_24='Weight Loss';

/*-----------------------------------------------------------------------------------------------
Fluid and Electrolyte Disorders
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2536','276') then do;
    ELX_GRP_25 = 1;
    TOT_GRP = TOT_GRP + (5);
end;
   LABEL ELX_GRP_25='Fluid and Electrolyte Disorders';

/*-----------------------------------------------------------------------------------------------
Blood Loss Anemia
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2800') then do;
    ELX_GRP_26 = 1;
    TOT_GRP = TOT_GRP + (-2);
end;
   LABEL ELX_GRP_26='Blood Loss Anemia';

/*-----------------------------------------------------------------------------------------------
Deficiency Anemia
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2801','2808','2809','281') then do;
    ELX_GRP_27 = 1;
    TOT_GRP = TOT_GRP + (-2);
end;
   LABEL ELX_GRP_27='Deficiency Anemia';

/*-----------------------------------------------------------------------------------------------
Alcohol Abuse
-----------------------------------------------------------------------------------------------*/
if ICD_9_CM_CODE IN: ('2652','2911','2912','2913','2915','2918','2919','3030','3039','3050',
        '3575','4255','5353','5710','5711','5712','5713','980','V113') then do;
```

```
     ELX_GRP_28 = 1;
     TOT_GRP = TOT_GRP + (0);
   end;
     LABEL ELX_GRP_28='Alcohol Abuse';

   /*--------------------------------------------------------------------------------------------------------
   Drug Abuse
   ---------------------------------------------------------------------------------------------------------*/
   if ICD_9_CM_CODE IN: ('292','304','3052','3053','3054','3055','3056','3057','3058','3059','V6542')
          then do;
     ELX_GRP_29 = 1;
     TOT_GRP = TOT_GRP + (-7);
   end;
     LABEL ELX_GRP_29='Drug Abuse';

   /*--------------------------------------------------------------------------------------------------------
   Psychoses
   ---------------------------------------------------------------------------------------------------------*/
   if ICD_9_CM_CODE IN: ('2938','295','29604','29614','29644','29654','297','298')
          then do;
     ELX_GRP_30 = 1;
     TOT_GRP = TOT_GRP + (0);
   end;
     LABEL ELX_GRP_30='Psychoses';

   /*--------------------------------------------------------------------------------------------------------
   Depression
   ---------------------------------------------------------------------------------------------------------*/
   if ICD_9_CM_CODE IN: ('2962','2963','2965','3004','309','311') then do;
     ELX_GRP_31 = 1;
     TOT_GRP = TOT_GRP + (-3);
   end;
     LABEL ELX_GRP_31='Depression';
run;


/*--------------------------------------------------------------------------------------------------------
Sum up the Elixhauser individual scores by encounter
---------------------------------------------------------------------------------------------------------*/
proc sql;
   create table
      ELIXHAUSER as
   select
      *, sum(TOT_GRP) as TOTAL_ELIX
   from
      DS_ENCOUNTER_DIAGNOSIS_ELIX
   group by
      encounter_number;
quit;


/*--------------------------------------------------------------------------------------------------------
Keep the last Elixhauser score record by encounter
---------------------------------------------------------------------------------------------------------*/
data ELIXHAUSER;
   set ELIXHAUSER;
   by encounter_number;
   if last.encounter_number then output;
run;


/*--------------------------------------------------------------------------------------------------------
Attach the depemndemt variable to the Elixhauser score record by encounter.
---------------------------------------------------------------------------------------------------------*/
proc sql;
   create table
      ELIXHAUSER_READMIT as
   select
      E.*, A.readmit_30days
   from
      ELIXHAUSER as E inner join Cohort as A on
         E.encounter_number = A.encounter_number;
quit;
```

```
/*---------------------------------------------------------------------------------------------------------------
Drop interim tables.
---------------------------------------------------------------------------------------------------------------*/
proc sql noprint;
   drop table DS_ENCOUNTER_DIAGNOSIS_ELIX, ELIXHAUSER;
quit;
```

Merge_Cohort_CCS_Elix

```
/*---------------------------------------------------------------------------------------------------------------
Transpose the CCS Categories for the Diagnoses. Set up a dummy value/column for the transpose work.
---------------------------------------------------------------------------------------------------------------*/
proc sort data=DS_ENCOUNTER_DIAGNOSIS_GROUPED; by ENCOUNTER_NUMBER CCS_CATEGORY; run;

data DS_ENCOUNTER_DIAGNOSIS_GROUPED;
   set DS_ENCOUNTER_DIAGNOSIS_GROUPED;
   by ENCOUNTER_NUMBER CCS_CATEGORY;
   if last.CCS_CATEGORY then do;
      val = 1;
      output;
   end;
run;


proc transpose data = DS_ENCOUNTER_DIAGNOSIS_GROUPED
   out = ENCOUNTER_DIAGNOSIS_TRANSPOSED prefix = CCS_DX_CATEGORY_;
 by ENCOUNTER_NUMBER;
 id CCS_CATEGORY;
 var val;
run;


data ENCOUNTER_DIAGNOSIS_TRANSPOSED (drop=i);
   set ENCOUNTER_DIAGNOSIS_TRANSPOSED (drop=_NAME_);
   array all(*) _numeric_;

   do i=1 to dim(all);
      if all(i)=. then all(i)=0;
   end;
run;
proc sort data=ENCOUNTER_DIAGNOSIS_TRANSPOSED; by ENCOUNTER_NUMBER; run;


/*---------------------------------------------------------------------------------------------------------------
Transpose the CCS Categories for the Procedures. Set up a dummy value/column for the transpose work.
---------------------------------------------------------------------------------------------------------------*/
proc sort data=DS_ENCOUNTER_PROCEDURE_GROUPED; by ENCOUNTER_NUMBER CCS_CATEGORY; run;

data DS_ENCOUNTER_PROCEDURE_GROUPED;
   set DS_ENCOUNTER_PROCEDURE_GROUPED;
   by ENCOUNTER_NUMBER CCS_CATEGORY;
   if last.CCS_CATEGORY then do;
      val = 1;
      output;
   end;
run;


proc transpose data = DS_ENCOUNTER_PROCEDURE_GROUPED
   out = ENCOUNTER_PROCEDURE_TRANSPOSED prefix = CCS_PROC_CATEGORY_;
 by ENCOUNTER_NUMBER;
 id CCS_CATEGORY;
 var val;
run;


data ENCOUNTER_PROCEDURE_TRANSPOSED (drop=i);
   set ENCOUNTER_PROCEDURE_TRANSPOSED (drop=_NAME_);
   array all(*) _numeric_;

   do i=1 to dim(all);
      if all(i)=. then all(i)=0;
   end;
run;
proc sort data=ENCOUNTER_PROCEDURE_TRANSPOSED; by ENCOUNTER_NUMBER; run;


/*---------------------------------------------------------------------------------------------------------------
For the Elixhauser score we keep the key and the score
---------------------------------------------------------------------------------------------------------------*/
```

```
data ELIXHAUSER_FINAL;
   set ELIXHAUSER_READMIT (keep=ENCOUNTER_NUMBER TOTAL_ELIX);
run;
proc sort data=ELIXHAUSER_FINAL; by ENCOUNTER_NUMBER; run;


/*--------------------------------------------------------------------------------------------------------------
Transpose the Medical Severity Diagnosis Related Groupers (MS_DRG).
--------------------------------------------------------------------------------------------------------------*/
data COHORT;
   set COHORT;
   val = 1;
run;

proc transpose data = COHORT
  out = MS_DRG_TRANSPOSED prefix = MS_DRG_;
  by ENCOUNTER_NUMBER;
  id MS_DRG;
  var val;
run;

data MS_DRG_TRANSPOSED (drop=i);
   length ENCOUNTER_NUMBER $20 ;
   set MS_DRG_TRANSPOSED (drop=_NAME_);
   array all(*) _numeric_;

   do i=1 to dim(all);
      if all(i)=. then all(i)=0;
   end;
run;
proc sort data=MS_DRG_TRANSPOSED; by ENCOUNTER_NUMBER; run;


/*--------------------------------------------------------------------------------------------------------------
Drop variables no longer needed. MARITAL_STATUS RACE DISCHARGE_DISPOSITION
--------------------------------------------------------------------------------------------------------------*/
data COHORT_FINAL;
   length ENCOUNTER_NUMBER $20 ;
   set COHORT (drop=MEDICAL_RECORD_NUMBER SEX  ADMIT_SUBSERVICE MS_DRG PRINCIPAL_DIAGNOSIS
FROM_DATE ADMITDATE DISCHDATE
         ADMIT_TIME PRINCIPAL_PROCEDURE DISCHTIME LASTUPDATE PRESENT_ON_ADMISSION_FLAG DX1_ICD9_5
Unplanned_Readmission_within_30_
         Readmission_Date Readmission_to_Same_Hospital____  ADMITDAY DISCHARGEDAY VAL);
run;
proc sort data=COHORT_FINAL; by ENCOUNTER_NUMBER; run;



/*--------------------------------------------------------------------------------------------------------------
Now merge all the files so that we have 1 row per Encounter. Recall that 1 patient may have more than 1 AMI row.
--------------------------------------------------------------------------------------------------------------*/
data COHORT_DX;
   merge COHORT_FINAL ENCOUNTER_DIAGNOSIS_TRANSPOSED;
   by ENCOUNTER_NUMBER;
run;


data COHORT_PROC;
   merge COHORT_DX ENCOUNTER_PROCEDURE_TRANSPOSED;
   by ENCOUNTER_NUMBER;
run;


data COHORT_ELIX;
   merge COHORT_PROC ELIXHAUSER_FINAL;
   by ENCOUNTER_NUMBER;
run;


data COHORT_MS_DRG;
   merge COHORT_ELIX MS_DRG_TRANSPOSED;
   by ENCOUNTER_NUMBER;
run;
```

```
data ANALYTIC_CANDIDATE;
   set COHORT_MS_DRG;
run;


/* Some may have no procedure codes so we set those flags to 0 from null. */
data ANALYTIC_CANDIDATE(drop=i);
   set ANALYTIC_CANDIDATE;
   array all(*) _numeric_;

   do i=1 to dim(all);
      if all(i)=. then all(i)=0;
   end;
run;
```

Variable_Reduction_By_Groupers

```
%macro Reduce_CCS_Groupers(presence=);

proc contents data = ANALYTIC_CANDIDATE out = ANALYTIC_CANDIDATE_cols noprint; run;

/*-------------------------------------------------------------------------------------------------------
Variable Reduction - CCS Diagnosis Groupers
-------------------------------------------------------------------------------------------------------*/
proc sql noprint;
   select left(put(count(name), 5.))
   into :ccscnt
   from  ANALYTIC_CANDIDATE_cols
   where NAME like 'CCS_DX_CATEGORY_%';

   select left(trim(name))
   into :ccs1 - :ccs&ccscnt.
   from  ANALYTIC_CANDIDATE_cols
   where NAME like 'CCS_DX_CATEGORY_%';
quit;

/*-------------------------------------------------------------------------------------------------------
Loop for all column names - perform a frequency through each column name and if the frequency result is 1 row, it
means the column values are restricted to 1 value - they are invariant. So we reject those.
All other column names are concatenated into a string - keep_vars - to identify those columns we keep in our data

The ODS statement below directs the frequency output to a file we can interrogate
-------------------------------------------------------------------------------------------------------*/
%let keep_vars = ;
%let drop_vars = ;

%do i = 1 %to &ccscnt ;
   ODS OUTPUT OneWayFreqs = _freqs;              /* Capture the frequencies to a file for reduction process. */
      proc freq data = ANALYTIC_CANDIDATE;
         table &&ccs&i..;
      run;
   ODS OUTPUT close;

   proc sql noprint;
      select count(*) into :pctcnt
      from _freqs
      where &&ccs&i.. = 1 and percent >= &presence ;
   quit;

   %if &pctcnt. = 1 %then %do;               /* If the CCS Grouper is present in 1% of the cohort - keep it. */
      %let keep_vars = &keep_vars. &&ccs&i..;
   %end;

   %if &pctcnt. = 0 %then %do;               /* If the CCS Grouper is NOT present in 1% of the cohort - drop it. */
      %let drop_vars = &drop_vars. &&ccs&i..;
   %end;
%end;


%put &keep_vars. = ;
%put &drop_vars. = ;


data ANALYTIC_CANDIDATE;
   set ANALYTIC_CANDIDATE (drop=&drop_vars.);
run;



/*-------------------------------------------------------------------------------------------------------
Variable Reduction - CCS Procedure Groupers
-------------------------------------------------------------------------------------------------------*/
proc sql noprint;
   select left(put(count(name), 5.))
   into :ccscnt
   from  ANALYTIC_CANDIDATE_cols
   where NAME like 'CCS_PROC_CATEGORY_%';
```

```
    select left(trim(name))
    into :ccs1 - :ccs&ccscnt.
    from  ANALYTIC_CANDIDATE_cols
    where NAME like 'CCS_PROC_CATEGORY_%';
 quit;

 /*-------------------------------------------------------------------------------------------------------------
 Loop for all column names - perform a frequency through each column name and if the frequency result is 1 row, it
 means the column values are restricted to 1 value - they are invariant. So we reject those.
 All other column names are concatenated into a string - keep_vars - to identify those columns we keep in our data.

 The ODS statement below directs the frequency output to a file we can interrogate
 -------------------------------------------------------------------------------------------------------------*/
 %let keep_vars = ;
 %let drop_vars = ;

 %do i = 1 %to &ccscnt.;
   ODS OUTPUT OneWayFreqs = _freqs;              /* Capture the frequencies to a file for reduction process */
      proc freq data = ANALYTIC_CANDIDATE;
         table &&ccs&i.;
      run;
   ODS OUTPUT close;

   proc sql noprint;
      select count(*) into :pctcnt
      from _freqs
      where &&ccs&i.. = 1 and percent >= &presence.;
   quit;

   %if &pctcnt. = 1 %then %do;              /* If the CCS Grouper is present in the specified percent of the cohort - keep it. */
      %let keep_vars = &keep_vars. &&ccs&i..;
   %end;

   %if &pctcnt. = 0 %then %do;              /* If the CCS Grouper is NOT present in the specified percent of the cohort - drop it
*/
      %let drop_vars = &drop_vars. &&ccs&i..;
   %end;
%end;


%put &keep_vars. = ;
%put &drop_vars. = ;


data ANALYTIC_CANDIDATE;
   set ANALYTIC_CANDIDATE (drop=&drop_vars.);
run;



%mend Reduce_CCS_Groupers;
%Reduce_CCS_Groupers(presence=5.0);    /* If a CCS Grouper is present in 5% or more of the population, we keep it. */



%macro Reduce_MSDRG_Groupers(presence=);

/*-------------------------------------------------------------------------------------------------------------
Now count the number of columns found. Place the names of the columns into macro variables. Then perform a do loop
using the count as your upper end of the loop.
-------------------------------------------------------------------------------------------------------------*/
proc sql noprint;
   select left(put(count(name), 5.))
   into :mscnt
   from  ANALYTIC_CANDIDATE_cols
   where NAME like 'MS_DRG_%';

   select left(trim(name))
   into :ms1 - :ms&mscnt.
   from  ANALYTIC_CANDIDATE_cols
   where NAME like 'MS_DRG_%';
quit;

/*-------------------------------------------------------------------------------------------------------------
Loop for all column names - perform a frequency through each column name and if the frequency result is 1 row, it
```

```
means the column values are restricted to 1 value - they are invariant. So we reject those.
All other column names are concatenated into a string - keep_vars - to identify those columns we keep in our data.

The ODS statement below directs the frequency output to a file we can interrogate.
-----------------------------------------------------------------------------------------------------------------*/
%let keep_vars = ;
%let drop_vars = ;

%do i = 1 %to &mscnt ;
  ODS OUTPUT OneWayFreqs = _freqs;              /* Capture the frequencies to a file for reduction process. */
    proc freq data = ANALYTIC_CANDIDATE;
      table &&ms&i..;
    run;
  ODS OUTPUT close;

  proc sql noprint;
    select count(*) into :pctcnt
    from _freqs
    where &&ms&i.. = 1 and percent >= &presence ;
  quit;

  %if &pctcnt. = 1 %then %do;             /* If the CCS Grouper is present in the specified percent of the cohort - keep it. */
    %let keep_vars = &keep_vars. &&ms&i..;
  %end;

  %if &pctcnt. = 0 %then %do;             /* If the CCS Grouper is NOT present in the specified percent of the cohort - drop it.
*/
    %let drop_vars = &drop_vars. &&ms&i..;
  %end;
%end;


%put &keep_vars. = ;
%put &drop_vars. = ;


data ANALYTIC_CANDIDATE;
  set ANALYTIC_CANDIDATE (drop=&drop_vars.);
run;


%mend Reduce_MSDRG_Groupers;
%Reduce_MSDRG_Groupers(presence=5.0);     /* If an MS-DRG Grouper is present in 5% or more of the population, we keep it.
*/

proc contents data=ANALYTIC_CANDIDATE varnum; run;
```

MS_DRG_Table

```
proc sort data=WDATA2.DS_MS_DRG_TABLE_VALUE out=DS_MS_DRG_TABLE_VALUE; by MS_DRG; run;
proc sort data=wdata2.DS_MS_DRG_WEIGHT out=DS_MS_DRG_WEIGHT; by MS_DRG; run;

data MS_DRG_Table(rename = (MS_DRG = CATEGORY));
   merge DS_MS_DRG_TABLE_VALUE (keep = MS_DRG MS_DRG_LONG_DESCRIPTION MEDICAL_OR_SURGICAL_FLAG)
DS_MS_DRG_WEIGHT (keep = MS_DRG GEOMETRIC_MEAN_LOS ARITHMETIC_MEAN_LOS);
   by MS_DRG;
run;

proc sort data = MS_DRG_Table nodupkey out = DISTINCT_MS_DRG; by CATEGORY; run;
```

Univariate_Regression

```
/*--------------------------------------------------------------------------------------------------------
Make a copy of the ANALYTIC_CANDIDATE file and drop Encounter Number: not needed for modeling or linking.
--------------------------------------------------------------------------------------------------------*/
data UNIVARIATE;
   set ANALYTIC_CANDIDATE (drop=ENCOUNTER_NUMBER);
run;


/*--------------------------------------------------------------------------------------------------------
Capture a copy of the columns and store into a file.
--------------------------------------------------------------------------------------------------------*/
proc contents data = UNIVARIATE out = UNIVARIATE_cols noprint; run;


/*--------------------------------------------------------------------------------------------------------
Keep all columns except the dependent variable. Change column name to variable.
--------------------------------------------------------------------------------------------------------*/
data UNIVARIATE_cols;
   set UNIVARIATE_cols (where = (variable^='readmit_30days')rename=(name=variable));
   length variable $32 ;
run;


%macro univariate;

/*--------------------------------------------------------------------------------------------------------
If there are any predictors that must be modeled (held) in the univariate process then specify them here in these
3 macro variables.
--------------------------------------------------------------------------------------------------------*/
   %let controller_1 =;
   %let controller_2 =;
   %let controller_3 =;


/*--------------------------------------------------------------------------------------------------------
Make sure the append base file is removed prior to starting the next step. Otherwies you may collect data from
prior instances of your testing.
--------------------------------------------------------------------------------------------------------*/
   proc sql;
      drop table append_logit_stats;
   quit;


/*--------------------------------------------------------------------------------------------------------
Count all the columns and place them into a macro variable array for processing 1 at a time
--------------------------------------------------------------------------------------------------------*/
   proc sql;
      select left(put(count(*), 5.))  into :cnt
      from UNIVARIATE_cols
      where lowcase(variable) not in ("&controller_1.","&controller_2.","&controller_3.");

      select trim(left(variable))  into :var1 - :var&cnt.
      from UNIVARIATE_cols
      where lowcase(variable) not in ("&controller_1.","&controller_2.","&controller_3.");
   quit;


/*--------------------------------------------------------------------------------------------------------
Perform a univariate model process for each column - your predictor columns.
--------------------------------------------------------------------------------------------------------*/
   %do i = 1 %to &cnt.; ***** Start of i Loop ***;

      /*--------------------------------------------------------------------------------------------------------
      Create a copy of the data keeping just those variables needed for the univariate process.
      --------------------------------------------------------------------------------------------------------*/
      data pl_file;
         set UNIVARIATE (keep = readmit_30days &&var&i.. &controller_1. &controller_2. &controller_3. );
      run;


      /*--------------------------------------------------------------------------------------------------------
      Perform a univariate model process for each column. Capture the parameter estimates and the odds ratio information.
      -----------------------------------------------------------------------------------------------------------*/
      proc logistic data= pl_file descending namelen = 32;
         model readmit_30days = &controller_1. &controller_2. &controller_3. &&var&i. ;

         ods output ParameterEstimates = LogitEstimates&i._&i. ;
         ods output OddsRatios = OddsRatios&i._&i. ;
      run;


      /*-----------------------------------------------------------------------------------------------------------
      Force the column titled -variable- to have a consistent length of 32.
      -----------------------------------------------------------------------------------------------------------*/
      data LogitEstimates&i._&i.;
         length variable $32.;
         set LogitEstimates&i._&i. ;
      run;


      /*-----------------------------------------------------------------------------------------------------------
      Append all univariate results - parameter estimates - into 1 file for later significance level review.
      This step will append a different column each time so we need to use the force option. The base file will grow
      width wise with the addition of each new predictor variable.
      -----------------------------------------------------------------------------------------------------------*/
      proc append data = LogitEstimates&i._&i. base = append_logit_stats force;
      run;

   %end; ***** End of i Loop ***;


   /*-----------------------------------------------------------------------------------------------------------
   Make a copy of all the columns and those from the univariate process. Then merge the 2 by the column titled
   -variable-. We keep all the columns found in the original column list: a superste of the columns run through the
   univariate process.
   -----------------------------------------------------------------------------------------------------------*/
   data result_logit_stats;
      set append_logit_stats;
   run;

   proc sort data = UNIVARIATE_cols out = init_iv_list; by variable; run;
   proc sort data = result_logit_stats; by variable; run;


   data ANALYTIC;
      merge init_iv_list  (in=a)
           result_logit_stats ;
      by variable;
      if a ;
   run;

%mend univariate;
%univariate;


/*-----------------------------------------------------------------------------------------------------------
If a column is significant from the above model process, then keep that column.
-----------------------------------------------------------------------------------------------------------*/
data ANALYTIC_COLUMNS;   /* 53 columns*/
  set ANALYTIC (where=(ProbChiSq <= 0.50));   /* 309 columns*/
run;


/*
proc freq data = analytic_candidate;
  table DISCH_OTHER_CARE;
run;
*/

proc means data = analytic_candidate;
  var LENGTH_OF_STAY;
run;

/*
CCS_DX_CATEGORY_108
CCS_DX_CATEGORY_158
```

```
CCS_DX_CATEGORY_2616
CCS_DX_CATEGORY_55
CCS_DX_CATEGORY_98
CCS_DX_CATEGORY_99
CCS_PROC_CATEGORY_45
CCS_PROC_CATEGORY_61
CCS_PROC_CATEGORY_63
DISCH_OTHER_CARE
LENGTH_OF_STAY
MARITAL_STAT
MS_DRG_00247
TOTAL_ELIX

CCS_DX_CATEGORY_108
CCS_DX_CATEGORY_128
CCS_DX_CATEGORY_130
CCS_DX_CATEGORY_131
CCS_DX_CATEGORY_155
CCS_DX_CATEGORY_158
CCS_DX_CATEGORY_159
CCS_DX_CATEGORY_2616
CCS_DX_CATEGORY_3
CCS_DX_CATEGORY_50
CCS_DX_CATEGORY_55
CCS_DX_CATEGORY_87
CCS_DX_CATEGORY_95
CCS_DX_CATEGORY_98
CCS_DX_CATEGORY_99
CCS_PROC_CATEGORY_45
CCS_PROC_CATEGORY_61
CCS_PROC_CATEGORY_63
DISCH_OTHER_CARE
LENGTH_OF_STAY
MARITAL_STAT
MS_DRG_00233
MS_DRG_00247
TOTAL_ELIX
*/
```

Split_Cohort

```
/*-----------------------------------------------------------------------------------------------------------
Split the entire sample (N=1049) into training. validation and test samples.
Training sample (N=494) validation sample (N=276) test sample(N=279).
----------------------------------------------------------------------------------------------------------*/

data training_sample validation_sample test_sample (drop = sample);
   set ANALYTIC_CANDIDATE (drop = ENCOUNTER_NUMBER);

   readmitted = readmit_30days;                /* We do this because the model development overwrites DV readmit_30days.
*/
   sample = uniform(14);                      /* Seed provided acceptable representation for readmission cohort in all 3
samples. */
   if sample <= 0.50 then output training_sample;
   else if sample <= 0.75 then output validation_sample;
   else output test_sample;
run;


title "Frequency of readmits in the Training sample.";
proc freq data = training_sample;
   table readmitted;                     /*13.16*/
run;


title "Frequency of readmits in the Validation sample.";
proc freq data = validation_sample;
   table readmitted;                     /*12.32*/
run;


title "Frequency of readmits in the Test sample.";
proc freq data = test_sample;
   table readmitted;                     /*13.26*/
run;

title ;
```

mdl_calculating_riskscore

```
*=========================================================================================;
*=== File: mdl_calculating_riskscore.sas                                ===;
*=== Author: Many thanks to Tae Park                                    ===;
*=== Date: 5/15/2008                                         ===;
*=== Desc: This macro code is to designed to calculate risk score for both          ===;
*===       the Logistic and Reg Models.                                 ===;
*=========================================================================================;
*=== Fixed Variables:                                          ===;
*===                                                  ===;
*=== VARIABLE   = the field name indicating parameter estimates.                ===;
*=== ESTIAMTE   = the field name indicating the predicted risk score.            ===;
*=========================================================================================;
*=== Macro Reference:                                           ===;
*===                                                  ===;
*=== INPUT_DATA_FILE = the input dataset as an analytic file                  ===;
*=== INPUT_BETA_LIST = the input dataset of variable list.                  ===;
*=== PRED_FIELD_NAME = the field name indicating parameters.                    ===;
*=== DV_DIST        = the field name indicating parameters.               ===;
*=== VAR_FIELD_NAME  = the field name indicating parameters.                   ===;
*=== SCORED_FILE     = the output dataset with scored risk values.               ===;
*=========================================================================================;


%macro mdl_calculating_riskscore(input_data_file =,
                      input_beta_list =,
                      pred_field_name =,
                      dv_dist       =,
                      scored_file    = );

   ***** Pull IVs and lamda of dv ****;
   proc sql noprint;
      select left(put(count(variable),8.)) into :tcnt
      from &input_beta_list.;

      select variable into :iv1 - :iv&tcnt.
      from &input_beta_list.;

      select estimate into :est1 - :est&tcnt.
      from &input_beta_list.;
   quit;


   ***** Calculate Scores ****;
   data &scored_file ;
      set &input_data_file.;

      intercept = 1;

      ******** Weigh betas by predictor ****;
      %do k = 1 %to &tcnt.;
         wt_iv&k. = &&iv&k.. * &&est&k.. ;
      %end;

      ********* Calculate Risk Sccore ****;
      ********* http://luna.cas.usf.edu/~mbrannic/files/regression/Logistic.html ****;
      %if %upcase(&dv_dist.) = B  %then %do;
         log_odds = sum(of %do k = 1 %to &tcnt.; wt_iv&k. %end;);
         &pred_field_name. = (1/(1+(exp(-1*log_odds))));
      %end;

      %if %upcase(&dv_dist.) = C  %then %do;
         &pred_field_name. = sum(of %do k = 1 %to &tcnt.; wt_iv&k. %end;);
      %end;


      drop wt_: log_odds intercept;
   run;


%mend mdl_calculating_riskscore;
```

Build_Model_Candidates

```
%macro runit;


/*------------------------------------------------------------------------------------------------------
From the prior variable reduction step. Univariate_Regression. use the remaining variables for model building.
Results - Forward and stepwise are identical so we will cross validate forward and backward only.
------------------------------------------------------------------------------------------------------*/

/*------------------------------------------------------------------------------------------------------
We use the reduced column set identified at the end of the Univariate_Regression step and found in the file
ANALYTIC_COLUMNS.

Count and store variable list into macro array. We may drop certain variables because of correlation worries
as well as those causing quai-separation in the backward selection methodology - see the line of filtering.
------------------------------------------------------------------------------------------------------*/
proc sql noprint;
   select left(put(count(*), 5.))  into :cnt
   from ANALYTIC_COLUMNS
   where VARIABLE not in
      ('CCS_PROC_CATEGORY_45' 'CCS_DX_CATEGORY_50')
   ;

   select trim(left(VARIABLE))  into :var1 - :var&cnt.
   from ANALYTIC_COLUMNS
   where VARIABLE not in
      ('CCS_PROC_CATEGORY_45' 'CCS_DX_CATEGORY_50')
   ;
quit;


/*------------------------------------------------------------------------------------------------------
                        Step 1
Use the training sample to build candidate logistic models. We designate the variable selection methods
forward, backward and stepwise. Per C. Peng we also build a model with all variables included. In that model
there is no variable selection.
Designate descending in all cases so that the model trains on Readmission = True (readmit_30days = 1)

From SAS website
(http://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_reg_sect030.htm)
visited 12/1/22013.

--Full Model Fitted (NONE)
This method is the default and provides no model selection capability. The complete model specified in the MODEL
statement is used to fit the model. For many regression analyses, this might be the only method you need.

Forward Selection (FORWARD)

The forward-selection technique begins with no variables in the model. For each of the independent variables,
the FORWARD method calculates  statistics that reflect the variable's contribution to the model if it is included.
The -values for these  statistics are compared to the SLENTRY= value that is specified in the MODEL statement
(or to 0.50 if the SLENTRY= option is omitted). If no  statistic has a significance level greater than the
SLENTRY= value, the FORWARD selection stops. Otherwise, the FORWARD method adds the variable that has the largest
statistic to the model. The FORWARD method then calculates  statistics again for the variables still remaining
outside the model, and the evaluation process is repeated. Thus, variables are added one by one to the model until
no remaining variable produces a significant  statistic. Once a variable is in the model, it stays.

Backward Elimination (BACKWARD)

The backward elimination technique begins by calculating  statistics for a model which includes all of the independent
variables. Then the variables are deleted from the model one by one until all the variables remaining in the model produce
statistics significant at the SLSTAY= level specified in the MODEL statement (or at the 0.10 level if the SLSTAY= option is
omitted).
At each step, the variable showing the smallest contribution to the model is deleted.

Stepwise (STEPWISE)

The stepwise method is a modification of the forward-selection technique and differs in that variables
already in the model do not necessarily stay there. As in the forward-selection method, variables are
added one by one to the model, and the  statistic for a variable to be added must be significant at the SLENTRY= level.
After a variable is added, however, the stepwise method looks at all the variables already included in
```

```
the model and deletes any variable that does not produce an  statistic significant at the SLSTAY= level.
Only after this check is made and the necessary deletions are accomplished can another variable be added
to the model. The stepwise process ends when none of the variables outside the model has an  statistic
significant at the SLENTRY= level and every variable in the model is significant at the SLSTAY= level.
or when the variable to be added to the model is the one just deleted from it.

Reviews of model-selection methods by Hocking (1976) and Judge et al. (1980) describe these and other
variable-selection methods.
----------------------------------------------------------------------------------------------------*/
   title "NO selection method - Candidate Build Step.";
   proc logistic data = training_sample descending;                /*N=494*/
      model readmit_30days = %do i = 1 %to &cnt.; &&var&i.. %end;
      /selection = none rsq lackfit;
      ods output ParameterEstimates = LogitEstsFull;
      ods output OddsRatios = OddsRatiosFull;
      ods output Association = AssocFull;
   run;

ods trace on;
   title "Forward variable selection method - Candidate Build Step.";
   proc logistic data = training_sample descending;
      model readmit_30days = %do i = 1 %to &cnt.; &&var&i.. %end;
      /selection = forward slentry = 0.20 rsq lackfit;           /* SAS default for SLENTRY = 0.05 */
      ods output ParameterEstimates = LogitEstsFwd;
      ods output OddsRatios = OddsRatiosFwd;
      ods output Association = AssocFwd;
      ods output ModelBuildingSummary = ModelBuildSum;
   run;
ods trace off;

   title "Backward Elimination variable selection method - Candidate Build Step";
   proc logistic data = training_sample descending;
      model readmit_30days = %do i = 1 %to &cnt.; &&var&i.. %end;
      /selection = backward slstay = 0.157 rsq lackfit;          /* SAS default SLSTAY = 0.05 */
      ods output ParameterEstimates = LogitEstsBkwd;
      ods output OddsRatios = OddsRatiosBkwd;
      ods output Association = AssocBkwd;
   run;

   title "Stepwise variable selection method - Candidate Build Step";
   proc logistic data = training_sample descending;
      model readmit_30days = %do i = 1 %to &cnt.; &&var&i.. %end;
      /selection = stepwise slentry = 0.50 slstay = 0.157 rsq lackfit;
      ods output ParameterEstimates = LogitEstsStep;
      ods output OddsRatios = OddsRatiosStep;
      ods output Association = AssocStep;
   run;

title;

/*--------------------------------------------------------------------------------------------------
The above model results are reviewed manually.
We find in the above 3 methods using selection criteria that the models are different in the predictors
and/or coefficients. We proceed with score cutoff and cross validation processes for each of the 3 models
----------------------------------------------------------------------------------------------------*/


/*--------------------------------------------------------------------------------------------------
                    Step 2
Per C. Peng - Use the full 70% sample to establish score cutoff point for those readmitted/not readmitted.
Use the variable names and coefficients from the above forward selection model. Calculate risk score for
the training_set.
----------------------------------------------------------------------------------------------------*/
%mdl_calculating_riskscore(input_data_file  = training_sample,
                           input_beta_list = LogitEstsFull,
                           pred_field_name = readmit_30days,
                           dv_dist        = B,
                           scored_file    = training_set_scored_full);

%mdl_calculating_riskscore(input_data_file  = training_sample,
                           input_beta_list = LogitEstsFwd,
                           pred_field_name = readmit_30days,
                           dv_dist        = B,
                           scored_file    = training_set_scored_fwd);
```

```
%mdl_calculating_riskscore(input_data_file  = training_sample,
                    input_beta_list = LogitEstsBkwd,
                    pred_field_name = readmit_30days,
                    dv_dist         = B,
                    scored_file     = training_set_scored_bkwd);

%mdl_calculating_riskscore(input_data_file  = training_sample,
                    input_beta_list = LogitEstsStep,
                    pred_field_name = readmit_30days,
                    dv_dist         = B,
                    scored_file     = training_set_scored_step);


/*-----------------------------------------------------------------------------------------------------------
Build a histogram for class members and choose a score cutoff. Check results with SQL statements. This score
cutoff will be set at a point and above which will be considered a readmission and below which will be
considered a non-readmission. We find that 66% of non-readmissions have a score < 0.12. We find that 66% of
readmissions have a score >= 0.12. We decide to use a model score of 0.12 as a cutoff point. We have such a
low cutoff because the maximum model scores for the 2 outcome values (0,1) are 0.71 and 0.73 respectively
-----------------------------------------------------------------------------------------------------------*/

/* The following cutoff set and the percent correct for Yes and No. */
title "Histogram of distribution of ---NO--- Selection scored results for readmit/no readmit.";
proc univariate data = training_set_scored_full noprint;
   class readmitted;
   histogram readmit_30days / normal;
run;

title "---NO--- Selection correctly scored this percent of the readmit population using a probability cutoff of 12.5% or higher.";
proc sql;
   select (count(*)/65) into :Y_1
   from training_set_scored_full
   where readmit_30days >= 0.125        /*Result at this cutoff - 0.707692*/
   and readmitted = 1;
quit;

title "---NO--- Selection correctly scored this percent of the non-readmit population using a probability cutoff less than 12.5%.";
proc sql;
   select (count(*)/429) into :Y_0        /*Result at this cutoff - 0.708625*/
   from training_set_scored_full
   where readmit_30days < 0.125
   and readmitted = 0;
quit;


/* The following cutoff set and the percent correct for Yes and No. */
title "Histogram of distribution of Forward Selection scored results for readmit/no readmit.";
proc univariate data = training_set_scored_fwd noprint;
   class readmitted;
   histogram readmit_30days / normal;
run;

title "Forward Selection correctly scored this percent of the readmit population using a probability cutoff of 10.75% or higher.";
proc sql;
   select count(*)/65 as Y_1        /*Result at this cutoff - 0.723077*/
   from training_set_scored_fwd
   where readmit_30days >= 0.1075
   and readmitted = 1;
quit;

title "Forward Selection correctly scored this percent of the non-readmit population using a probability cutoff less than 10.75%.";
proc sql;
   select count(*)/429 as Y_0        /*Result at this cutoff - 0.641026*/
   from training_set_scored_fwd
   where readmit_30days < 0.1075
   and readmitted = 0;
quit;


/* The following cutoff set and the percent correct for Yes and No. */
title "Histogram of distribution of Backward Selection scored results for readmit/no readmit.";
proc univariate data = training_set_scored_bkwd noprint;
   class readmitted;
   histogram readmit_30days / normal;
run;
```

```
title "Backward Selection correctly scored this percent of the readmit population using a probability cutoff of 11% or higher.";
proc sql;
   select count(*)/65 as Y_1       /*Result at this cutoff - 0.692308*/
   from training_set_scored_bkwd
   where readmit_30days >= 0.11
   and readmitted = 1;
quit;

title "Backward Selection correctly scored this percent of the non-readmit population using a probability cutoff less than 11%.";
proc sql;
   select count(*)/429 as Y_0      /*Result at this cutoff - 0.685315*/
   from training_set_scored_bkwd
   where readmit_30days < 0.11
   and readmitted = 0;
quit;


/* The following cutoff set and the percent correct for Yes and No. */
title "Histogram of distribution of Stepwise Selection scored results for readmit/no readmit.";
proc univariate data = training_set_scored_step noprint;
   class readmitted;
   histogram readmit_30days / normal;
run;

title "Stepwise Selection correctly scored this percent of the readmit population using a probability cutoff of 11.5% or higher.";
proc sql;
   select count(*)/65 as Y_1       /*Result at this cutoff - 0.723077*/
   from training_set_scored_step
   where readmit_30days >= 0.115
   and readmitted = 1;
quit;

title "Stepwise Selection correctly scored this percent of the non-readmit population using a probability cutoff less than 11.5%.";
proc sql;
   select count(*)/429 as Y_0      /*Result at this cutoff - 0.641026*/
   from training_set_scored_step
   where readmit_30days < 0.115
   and readmitted = 0;
quit;



%mend runit;


%runit;


options symbolgen;

%macro record_cutoff(cutoff =, selection = );

    data success_rate_&selection. (keep= readmitted readmit_30days prediction cutoff);
       set training_set_scored_&selection.;
       length cutoff 8.;
       cutoff = &cutoff;
       if readmit_30days < &cutoff then prediction = 0;
       if readmit_30days >= &cutoff then prediction = 1;
    run;

    proc append base = success_rate_&selection._base data = success_rate_&selection.; run;

%mend record_cutoff;

proc sql; drop table success_rate_full_base; quit;
proc sql; drop table success_rate_fwd_base; quit;
proc sql; drop table success_rate_bkwd_base; quit;
proc sql; drop table success_rate_step_base; quit;

%record_cutoff (cutoff = 0.09, selection = full);
%record_cutoff (cutoff = 0.095, selection = full);
%record_cutoff (cutoff = 0.10, selection = full);
%record_cutoff (cutoff = 0.105, selection = full);
%record_cutoff (cutoff = 0.1075, selection = full);
```

```
%record_cutoff (cutoff = 0.1095, selection = full);
%record_cutoff (cutoff = 0.11, selection = full);
%record_cutoff (cutoff = 0.115, selection = full);
%record_cutoff (cutoff = 0.12, selection = full);
%record_cutoff (cutoff = 0.125, selection = full);
%record_cutoff (cutoff = 0.13, selection = full);
%record_cutoff (cutoff = 0.135, selection = full);
%record_cutoff (cutoff = 0.14, selection = full);
%record_cutoff (cutoff = 0.145, selection = full);
%record_cutoff (cutoff = 0.15, selection = full);
%record_cutoff (cutoff = 0.25, selection = full);
%record_cutoff (cutoff = 0.35, selection = full);
%record_cutoff (cutoff = 0.45, selection = full);

%record_cutoff (cutoff = 0.09, selection = fwd);
%record_cutoff (cutoff = 0.095, selection = fwd);
%record_cutoff (cutoff = 0.10, selection = fwd);
%record_cutoff (cutoff = 0.105, selection = fwd);
%record_cutoff (cutoff = 0.1075, selection = fwd);
%record_cutoff (cutoff = 0.1095, selection = fwd);
%record_cutoff (cutoff = 0.11, selection = fwd);
%record_cutoff (cutoff = 0.115, selection = fwd);
%record_cutoff (cutoff = 0.12, selection = fwd);
%record_cutoff (cutoff = 0.125, selection = fwd);
%record_cutoff (cutoff = 0.13, selection = fwd);
%record_cutoff (cutoff = 0.135, selection = fwd);
%record_cutoff (cutoff = 0.14, selection = fwd);
%record_cutoff (cutoff = 0.145, selection = fwd);
%record_cutoff (cutoff = 0.15, selection = fwd);
%record_cutoff (cutoff = 0.25, selection = fwd);
%record_cutoff (cutoff = 0.35, selection = fwd);
%record_cutoff (cutoff = 0.45, selection = fwd);

%record_cutoff (cutoff = 0.09, selection = bkwd);
%record_cutoff (cutoff = 0.095, selection = bkwd);
%record_cutoff (cutoff = 0.10, selection = bkwd);
%record_cutoff (cutoff = 0.105, selection = bkwd);
%record_cutoff (cutoff = 0.1075, selection = bkwd);
%record_cutoff (cutoff = 0.11, selection = bkwd);
%record_cutoff (cutoff = 0.115, selection = bkwd);
%record_cutoff (cutoff = 0.12, selection = bkwd);
%record_cutoff (cutoff = 0.125, selection = bkwd);
%record_cutoff (cutoff = 0.13, selection = bkwd);
%record_cutoff (cutoff = 0.135, selection = bkwd);
%record_cutoff (cutoff = 0.14, selection = bkwd);
%record_cutoff (cutoff = 0.145, selection = bkwd);
%record_cutoff (cutoff = 0.15, selection = bkwd);
%record_cutoff (cutoff = 0.25, selection = bkwd);
%record_cutoff (cutoff = 0.35, selection = bkwd);
%record_cutoff (cutoff = 0.45, selection = bkwd);

%record_cutoff (cutoff = 0.09, selection = step);
%record_cutoff (cutoff = 0.095, selection = step);
%record_cutoff (cutoff = 0.10, selection = step);
%record_cutoff (cutoff = 0.105, selection = step);
%record_cutoff (cutoff = 0.1075, selection = step);
%record_cutoff (cutoff = 0.11, selection = step);
%record_cutoff (cutoff = 0.115, selection = step);
%record_cutoff (cutoff = 0.12, selection = step);
%record_cutoff (cutoff = 0.125, selection = step);
%record_cutoff (cutoff = 0.13, selection = step);
%record_cutoff (cutoff = 0.135, selection = step);
%record_cutoff (cutoff = 0.14, selection = step);
%record_cutoff (cutoff = 0.145, selection = step);
%record_cutoff (cutoff = 0.15, selection = step);
%record_cutoff (cutoff = 0.25, selection = step);
%record_cutoff (cutoff = 0.35, selection = step);
%record_cutoff (cutoff = 0.45, selection = step);


title "Full Selection correctly scored percents of the non-readmit/readmit population cutoffs from 9% to 15% and 25/35/45%.";
  ODS OUTPUT CrossTabFreqs = cross_tabs_full;      /* Capture the frequencies to a file for reduction process */
  proc freq data=success_rate_full_base;
    by cutoff;
```

```
      tables readmitted * prediction;
   run;
   ODS OUTPUT close;

proc sql;
   create table full as
   select cutoff, readmitted, prediction, frequency, rowpercent
   from cross_tabs_full
   where readmitted ^= . and readmitted = prediction;
quit;

proc sort data=full; by cutoff; run;

data full1;
   set full;
   by cutoff;
   length freq_sum 8.; retain freq_sum;
   if first.cutoff then freq_sum = 0;
   freq_sum + frequency;
   if last.cutoff then do;
      success_pred = freq_sum/494;
      output;
   end;
run;

data full2;
   merge full (in=a) full1 (in=b keep=cutoff success_pred);
   by cutoff;
run;


title "Forward Selection correctly scored percents of the non-readmit/readmit population cutoffs from 9% to 15% and 25/35/45%.";
   ODS OUTPUT CrossTabFreqs = cross_tabs_fwd;      /* Capture the frequencies to a file for reduction process */
   proc freq data=success_rate_fwd_base;
      by cutoff;
      tables readmitted * prediction;
   run;
   ODS OUTPUT close;

proc sql;
   create table fwd as
   select cutoff, readmitted, prediction, frequency, rowpercent
   from cross_tabs_fwd
   where readmitted ^= . and readmitted = prediction;
quit;

proc sort data=fwd; by cutoff; run;

data fwd1;
   set fwd;
   by cutoff;
   length freq_sum 8.; retain freq_sum;
   if first.cutoff then freq_sum = 0;
   freq_sum + frequency;
   if last.cutoff then do;
      success_pred = freq_sum/494;
      output;
   end;
run;

data fwd2;
   merge fwd (in=a) fwd1 (in=b keep=cutoff success_pred);
   by cutoff;
run;


title "Backward Selection correctly scored percents of the non-readmit/readmit population cutoffs from 9% to 15% and
25/35/45%.";
   ODS OUTPUT CrossTabFreqs = cross_tabs_bkwd;      /* Capture the frequencies to a file for reduction process */
   proc freq data=success_rate_bkwd_base;
      by cutoff;
      tables readmitted * prediction;
   run;
```

```
    ODS OUTPUT close;

  proc sql;
    create table bkwd as
    select cutoff, readmitted, prediction, frequency, rowpercent
    from cross_tabs_bkwd
    where readmitted ^= . and readmitted = prediction;
  quit;

  proc sort data=bkwd; by cutoff; run;

  data bkwd1;
    set bkwd;
    by cutoff;
    length freq_sum 8.; retain freq_sum;
    if first.cutoff then freq_sum = 0;
    freq_sum + frequency;
    if last.cutoff then do;
       success_pred = freq_sum/494;
       output;
    end;
  run;

  data bkwd2;
    merge bkwd (in=a) bkwd1 (in=b keep=cutoff success_pred);
    by cutoff;
  run;


  title "Stepwise Selection correctly scored percents of the non-readmit/readmit population cutoffs from 9% to 15% and 25/35/45%.";
    ODS OUTPUT CrossTabFreqs = cross_tabs_step;     /* Capture the frequencies to a file for reduction process. */
    proc freq data=success_rate_step_base;
       by cutoff;
       tables readmitted * prediction;
    run;
    ODS OUTPUT close;

  proc sql;
    create table step as
    select cutoff, readmitted, prediction, frequency, rowpercent
    from cross_tabs_step
    where readmitted ^= . and readmitted = prediction;
  quit;

  proc sort data=step; by cutoff; run;

  data step1;
    set step;
    by cutoff;
    length freq_sum 8.; retain freq_sum;
    if first.cutoff then freq_sum = 0;
    freq_sum + frequency;
    if last.cutoff then do;
       success_pred = freq_sum/494;
       output;
    end;
  run;

  data step2;
    merge step (in=a) step1 (in=b keep=cutoff success_pred);
    by cutoff;
  run;
```

Cross_Validation

```
/*-----------------------------------------------------------------------------------------------------------
Generate the 5-fold cross-validation sample into folds 1-5 or 20% of population per fold. We select 5-fold
because we desire a larger validation population that realizes the outcome for scoring than is provided by
traditional 10-fold. Random assignment is done separartely for non-readmits and readmits. Try to get
representative folds.

Macro establish_seed attempts to find the smallest standard deviation among the 5-fold samples within those
who experience the outcome. It also identifies the smallest standard deviation among the entire sample folds.
We determine to use seed = 80 with respective STD values: 0.8366600265 and 4.9699094559.

This is an attempt to keep the Cross Validation results stable.
-----------------------------------------------------------------------------------------------------------*/
  %macro establish_seed;
    proc sql; drop table base_freqs, nonreadmit_base_freqs; quit;

    %do i = 1 %to 80;
    data kfold_validation_sample(drop=sample);
      set validation_sample;

      sample = uniform(&i );
      if sample >= 0.80 then fold = 5;
      else if sample >= 0.60 then fold = 4;
      else if sample >= 0.40 then fold = 3;
      else if sample >= 0.20 then fold = 2;
      else fold = 1;
    run;

    title "Split of readmit/no readmit population using seed: &i..";
    ODS output CrossTabFreqs = _xfreqs;            /* Capture the frequencies to a file for reduction process. */
    proc freq data=kfold_validation_sample;
      tables fold*readmit_30days;
    run;
    ODS output close;                              /* Capture the frequencies to a file for reduction process. */

    data freqs (keep = seed std);
      set _xfreqs (where=(readmit_30days=1 and fold >= 1)) end=eof;
      array freqs{5} freq1 freq2 freq3 freq4 freq5;
      retain freqs;
      seed = &i.;
      row = _N_;
      freqs(row) = frequency;
      if eof then do;
        std = std(of freq1-freq5);
        output;
      end;
    run;

    data nonreadmit_freqs (keep = seed std);
      set _xfreqs (where=(readmit_30days=. and fold >= 1)) end=eof;
      array freqs{5} freq1 freq2 freq3 freq4 freq5;
      retain freqs;
      seed = &i.;
      row = _N_;
      freqs(row) = frequency;
      if eof then do;
        std = std(of freq1-freq5);
        output;
      end;
    run;

    proc append base= base_freqs data=freqs; run;
    proc append base= nonreadmit_base_freqs data=nonreadmit_freqs; run;
    %end;

    title "Seeds that produce the lowest STD for readmit segment.";
    proc sql; select seed from base_freqs where std = select min(std) from base_freqs; quit;
    title "Seeds that produce the lowest STD for overall population segmentation.";
    proc sql; select seed from nonreadmit_base_freqs where std = select min(std) from nonreadmit_base_freqs; quit;

  %mend establish_seed;
/* %establish_seed; */
```

```
/*-----------------------------------------------------------------------------------------------------------
Now repeat the following steps 5 times.
Hold out 1 of the 5 subsamples and run the balance of the samples through the candidate model produced by
the build steps above (Step 1). We expect that the predictor variables in the build candidates will be reduced
during the following step and the associated beta values changed. Capture this model output for scoring.
-----------------------------------------------------------------------------------------------------------*/
%macro CV(model = , cutoff = );

   %do i = 1 %to 5;    ***** 5-fold loop. *****;

      title "Cross Validation of Build Model produced by &model. variable selection. We hold out fold - &i..";
      proc logistic data = kfold_validation_sample (where=(fold ^= &i.)) descending namelen = 32;
         model readmit_30days = %do j = 1 %to &cnt.; &&var&j.. %end;
         ;
         ods output ParameterEstimates    = LogitEstsStep&i.;
         ods output OddsRatios            = OddsRatiosStep&i.;
         ods output Association           = AssocStep&i.;
      run;
      title ;


   /*-----------------------------------------------------------------------------------------------------
   Capture hold out fold into a test sample. Capture the coefficients/variables from the above model results.
   Score the holdout (test) sample using the model results from the previous step.
   -----------------------------------------------------------------------------------------------------*/
      proc sql noprint;
         create table test_sample&i. as
         select *
         from kfold_validation_sample where fold = &i.;

         select count(*) into :holdcnt
         from test_sample&i.;
      quit;

      data beta_list;
         set LogitEstsStep&i. (keep = variable estimate);
      run;


      /*-----------------------------------------------------------------------------------------------------
      Calculate risk score for the fold sample that was held out.
      -----------------------------------------------------------------------------------------------------*/
      %mdl_calculating_riskscore(input_data_file  = test_sample&i.,
                          input_beta_list = beta_list,
                          pred_field_name = readmit_30days,
                          dv_dist         = B,
                          scored_file     = &model._test_sample_scored&i.);

      /*-----------------------------------------------------------------------------------------------------
      The actual decimal score (0-1) from the macro is stored in the variable readmit_30days.
      -----------------------------------------------------------------------------------------------------*/
      data &model._test_sample_scored&i.;
         set &model._test_sample_scored&i.;
         scored_as_readmit = 0;
         if readmit_30days >= &cutoff. then scored_as_readmit = 1;
      run;

      title "Scored Results from Cross Validation of Build Model produced by &model. variable selection. We use fold - &i..";
      proc sql;
         select count(*)/&holdcnt. as Correct_Classification_Rate
         from &model._test_sample_scored&i.
         where scored_as_readmit = readmitted;
      quit;

      proc sql noprint;
         select count(*)/&holdcnt. into :Correct_Classification_Rate&i.
         from &model._test_sample_scored&i.
         where scored_as_readmit = readmitted;
      quit;
   %end;    ***** End of 5-fold loop. *****;


   /*-----------------------------------------------------------------------------------------------------
   Capture the correct scoring percentage for all 5 folds and report Mean/STD/LCL-UCL.
```

```
        ------------------------------------------------------------------------------------------------------- */
        data &model._statistics;
          %do i = 1 %to 5;
            Correctly_Scored = &&Correct_Classification_Rate&i. ;
            output;
          %end;
        run;

        proc means data=&model._statistics fw=8 maxdec=6 alpha=0.05 clm mean std;
          var Correctly_Scored;
          title "Two-sided Confidence Limits for Correctly Scored patients using Model: &model..";
        run;

%mend CV;


/*-----------------------------------------------------------------------------------------------------
Using the seed found above, split the validation population of 275 into 5 folds.
-----------------------------------------------------------------------------------------------------*/
data kfold_validation_sample(drop=sample);
  set validation_sample;

  sample = uniform(80);
  if sample >= 0.80 then fold = 5;
  else if sample >= 0.60 then fold = 4;
  else if sample >= 0.40 then fold = 3;
  else if sample >= 0.20 then fold = 2;
  else fold = 1;
run;


/*-----------------------------------------------------------------------------------------------------
Place Predictor variable count and list into macro array. These are parameters identified in step 1.
-----------------------------------------------------------------------------------------------------*/
proc sql noprint;
  select left(put(count(*), 5.))  into :cnt
  from LogitEstsFwd
  where upcase(variable) ^= 'INTERCEPT';

  select trim(left(variable))  into :var1 - :var&cnt.
  from LogitEstsFwd
  where upcase(variable) ^= 'INTERCEPT';
quit;
%CV(model = FWD, cutoff = 0.1075);   /* The cutoffs were established in: Build_Model_Candidates */


proc sql noprint;
  select left(put(count(*), 5.))  into :cnt
  from LogitEstsBkwd
  where upcase(variable) ^= 'INTERCEPT';

  select trim(left(variable))  into :var1 - :var&cnt.
  from LogitEstsBkwd
  where upcase(variable) ^= 'INTERCEPT';
quit;
%CV(model = BKWD, cutoff = 0.11);   /* The cutoffs were established in: Build_Model_Candidates */


proc sql noprint;
  select left(put(count(*), 5.))  into :cnt
  from LogitEstsBkwd
  where upcase(variable) ^= 'INTERCEPT';

  select trim(left(variable))  into :var1 - :var&cnt.
  from LogitEstsBkwd
  where upcase(variable) ^= 'INTERCEPT';
quit;
%CV(model = STEP, cutoff = 0.115);   /* The cutoffs were established in: Build_Model_Candidates */
```

Score_Using_Test_Sample

```
%macro Score_Test_Sample;
/*---------------------------------------------------------------------------------------------------------
We have selected the candidate model that uses the forward selection process. This model used the training
sample (N=494) as well as the validation sample (N=276).

We now use that candidate model to score the test sample(N=279). The scoring success using this candidate
model and test sample will is reported in the paper.

Capture the coefficients/variables from the forward selection candidate model. Score the test sample and
produce the Correct Classification Rate.

The model parameters, coefficients and classification rate are reported in the paper.
---------------------------------------------------------------------------------------------------------*/
  proc sql noprint;
    select count(*) into :holdcnt
    from test_sample;
  quit;

  data beta_list;
    set LogitEstsFwd (keep = variable estimate);
  run;


  /*---------------------------------------------------------------------------------------------------------
  Calculate risk score for the test sample after choosing best model from step 2 - cross validation.
  This was determined by the model that identified the highest percentage of readmits while balancing
  the best percentage of non-readmits.
  ---------------------------------------------------------------------------------------------------------*/
  %mdl_calculating_riskscore(input_data_file   = test_sample,
                             input_beta_list = beta_list,
                             pred_field_name = readmit_30days,
                             dv_dist        = B,
                             scored_file     = test_sample_scored);

  data test_sample_scored;
    set test_sample_scored;
    scored_as_readmit = 0;
    if readmit_30days >= 0.13 then scored_as_readmit = 1;   /* 1075 Cutoff taken from module Build_Model_Candidates */
  run;

  title "Correct Classification Rate for Original Test Sample using Forward Selection candidate model.";
  proc sql;
    select count(*)/&holdcnt. as Correct_Classification_Rate
    from test_sample_scored
    where scored_as_readmit = readmitted;
  quit;

  proc freq data = test_sample_scored;
    tables scored_as_readmit * readmitted;
  run;


/*
  proc sql noprint;
    select left(put(count(*), 5.)) into :rcnt
    from test_sample_scored
    where readmitted = 1;
  quit;

  title "Percent scored correctly for Original Holdout Sample for Readmits.";
  proc sql;
    select count(*)/&rcnt. as Pct_Scored_Properly
    from test_sample_scored
    where scored_as_readmit = 1 and readmitted = 1;
  quit;


  proc sql noprint;
    select left(put(count(*), 5.)) into :rcnt
    from test_sample_scored
    where readmitted = 0;
  quit;
```

```
   title "Percent scored correctly for Original Holdout Sample for NON-Readmits.";
   proc sql;
      select count(*)/&rcnt. as Pct_Scored_Properly
      from test_sample_scored
      where scored_as_readmit = 0 and readmitted = 0;
   quit;
*/
%mend Score_Test_Sample;

%Score_Test_Sample;
```

CART

```
/*
The recursive structure of CART models is ideal for uncovering complex dependencies among predictor variables.

Recursive Partitioning : Tree-structured models for regression, classification and survival analysis,
following the ideas in the CART book, are implemented in rpart (shipped with base R) and tree
Package rpart is recommended for computing CART-like trees.

From: http://cran.r-project.org/web/views/MachineLearning.html visited 11/20/2013

We'll start with 0.70 sample to train the decision tree. Then use 0.30 sample to test performance.

In information theory, entropy is a measure of the uncertainty in a random variable.[1]
In this context, the term usually refers to the Shannon entropy, which quantifies the expected value of the
information contained in a message.[2]

Entropy is typically measured in bits, nats, or bans.[3] Shannon entropy is the average unpredictability in a
random variable, which is equivalent to its information content. Shannon entropy provides an absolute limit
on the best possible lossless encoding or compression of any communication, assuming that[4] the communication
may be represented as a sequence of independent and identically distributed random variables.

A single toss of a fair coin has an entropy of one bit. A series of two fair coin tosses has an entropy of two bits.
The number of fair coin tosses is its entropy in bits. This random selection between two outcomes in a sequence
over time, whether the outcomes are equally probable or not, is often referred to as a Bernoulli process.
The entropy of such a process is given by the binary entropy function. The entropy rate for a fair coin toss is
one bit per toss. However, if the coin is not fair, then the uncertainty, and hence the entropy rate, is lower.
This is because, if asked to predict the next outcome, we could choose the most frequent result and be
right more often than wrong. The difference between what we know, or predict, and the information that the unfair
coin toss reveals to us is less than one heads-or-tails "message", or bit, per toss.[5]


*/

%macro runit;

/*-----------------------------------------------------------------------------------------------------
Place Predictor variable count and list into macro array. These are parameters identified in step 1.
'TOTAL_ELIX'  'CCS_DX_CATEGORY_106' 'CCS_PROC_CATEGORY_61' 'MS_DRG_00247' 'CCS_DX_CATEGORY_206'
'CCS_DX_CATEGORY_104'

proc contents data = ANALYTIC_CANDIDATE out = ANALYTIC_CANDIDATE_cols noprint; run;
-----------------------------------------------------------------------------------------------------*/
%let keep_vars = ;

proc sql noprint;
   select left(put(count(*), 5.))  into :cnt
   from ANALYTIC_COLUMNS
   ;

   select trim(left(VARIABLE))  into :var1 - :var&cnt.
   from ANALYTIC_COLUMNS
    ;
quit;

%do i = 1 %to &cnt.;
   %let keep_vars = &keep_vars. &&var&i..;
%end;

data training_set (keep=&keep_vars. readmitted);
   set training_sample;
run;

data test_set (keep=&keep_vars. readmitted);
   set test_sample;
run;


proc export data=training_set
  outfile='\\sasmeta2\projects\EG Projects\Knowld1\Thesis\R\training_set.csv'
  dbms=csv replace;
run;

proc export data=test_set
```

```
   outfile='\\sasmeta2\projects\EG Projects\Knowld1\Thesis\R\test_set.csv'
   dbms=csv replace;
 run;

 %mend runit;
 %runit;

 /*
 From: http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Classification/Decision_Trees
 Visited: 11/20/2013

 training = read.table (file='T:/EG Projects/Knowld1/Thesis/R/training_set.csv',header=TRUE)
 fit <- rpart(readmitted ~ AGE + LENGTH_OF_STAY + WeekendAdmit + WeekendDischarge + DayDischarge + DayAdmit +
 MEDICAL_SUBSERVICE + MARITAL_STAT + MALE + CCS_DX_CATEGORY_108 + CCS_DX_CATEGORY_131 +
 CCS_DX_CATEGORY_138 + CCS_DX_CATEGORY_253 + CCS_DX_CATEGORY_53 + CCS_DX_CATEGORY_55 +
 CCS_DX_CATEGORY_651 + CCS_DX_CATEGORY_95 + CCS_DX_CATEGORY_98 + CCS_DX_CATEGORY_101 +
 CCS_DX_CATEGORY_155 + CCS_DX_CATEGORY_158 + CCS_DX_CATEGORY_50 + CCS_DX_CATEGORY_87 +
 CCS_DX_CATEGORY_99 + CCS_DX_CATEGORY_146 + CCS_DX_CATEGORY_259 + CCS_DX_CATEGORY_3 +
 CCS_DX_CATEGORY_103 + CCS_DX_CATEGORY_163 + CCS_DX_CATEGORY_257 + CCS_DX_CATEGORY_60 +
 CCS_DX_CATEGORY_62 + CCS_DX_CATEGORY_653 + CCS_DX_CATEGORY_255 + CCS_DX_CATEGORY_2617 +
 CCS_DX_CATEGORY_663 + CCS_DX_CATEGORY_133 + CCS_DX_CATEGORY_118 + CCS_DX_CATEGORY_159 +
 CCS_DX_CATEGORY_49 + CCS_DX_CATEGORY_110 + CCS_DX_CATEGORY_164 + CCS_DX_CATEGORY_205 +
 CCS_DX_CATEGORY_122 + CCS_DX_CATEGORY_128 + CCS_DX_CATEGORY_130 + CCS_DX_CATEGORY_2616 +
 CCS_DX_CATEGORY_54 + CCS_DX_CATEGORY_88 + CCS_DX_CATEGORY_97 + CCS_DX_CATEGORY_107 +
 CCS_DX_CATEGORY_29 + CCS_DX_CATEGORY_2621 + CCS_DX_CATEGORY_249 + CCS_PROC_CATEGORY_45 +
 CCS_PROC_CATEGORY_47 + CCS_PROC_CATEGORY_63 + CCS_PROC_CATEGORY_49 + CCS_PROC_CATEGORY_48
 + CCS_PROC_CATEGORY_231 + CCS_PROC_CATEGORY_54 + CCS_PROC_CATEGORY_216 +
 CCS_PROC_CATEGORY_39 + MS_DRG_00233 + MS_DRG_00249 + MS_DRG_00282 + MS_DRG_00248 + MS_DRG_00281
 + MS_DRG_00246 + MS_DRG_00234,data=training,method="class")
 summary(residuals(fit))
 plot(predict(fit),residuals(fit))
 printcp(fit)
 plotcp(fit)
 rsq.rpart(fit)
 print(fit)

 108   Congestive heart failure; nonhypertensive
 55   Fluid and electrolyte disorders
 87   Retinal detachments; defects; vascular occlusion; and retinopathy
 */


/* For the following scoring refer to the contents of the fit model from R results. */
data training_sample_tree_scored(keep=readmitted tree_score);
   set training_sample;

   /* Left most edge from root node where the starting training population is split (No readmit/readmit). */

   /* Left most leaf from root node Not Readmitted. */
   if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95< 0.5 then tree_score = 0;
   else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit>=0.5 then tree_score = 0;
   else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127<
0.5 then tree_score = 0;
   else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127>=
0.5 then tree_score = 1;

   else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX>= 27.5 then tree_score = 0;
   else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age>=82.5 then tree_score = 0;
   else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age<82.5 then tree_score = 1;
run;


title "Cross Tab for CART scores for the training sample.";
proc freq data=training_sample_tree_scored;
   tables readmitted*tree_score;
run;


data test_sample_tree_scored(keep=readmitted tree_score);
   set test_sample;

   /* Left most edge from root node where the starting training population is split (No readmit/readmit). */

   /* Left most leaf from root node Not Readmitted. */
```

```
    if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95< 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit>=0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127<
 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127>=
 0.5 then tree_score = 1;

    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX>= 27.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age>=82.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age<82.5 then tree_score = 1;
 run;


 title "Cross Tab for CART scores for the test sample.";
 proc freq data=test_sample_tree_scored;
    tables readmitted*tree_score;
 run;


 data full_sample;
    set training_sample validation_sample test_sample;
 run;

 data full_sample_tree_scored(keep=readmitted tree_score);
    set full_sample;

    /* Left most edge from root node where the starting training population is splt (No readmit/readmit. */

    /* Left most leaf from root node Not Readmitted. */
    if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95< 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit>=0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127<
 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127>=
 0.5 then tree_score = 1;

    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX>= 27.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age>=82.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age<82.5 then tree_score = 1;
 run;


 title "Cross Tab for CART scores for the full sample.";
 proc freq data=full_sample_tree_scored;
    tables readmitted*tree_score;
 run;



 data validation_sample;
    set validation_sample;
 run;

 data validation_sample_tree_scored(keep=readmitted tree_score);
    set validation_sample;

    /* Left most edge from root node where the starting training population is splt (No readmit/readmit. */

    /* Left most leaf from root node Not Readmitted. */
    if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95< 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit>=0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127<
 0.5 then tree_score = 0;
    else if TOTAL_ELIX< 23.5 and CCS_DX_CATEGORY_95>= 0.5 and WeekendAdmit<0.5 and CCS_DX_CATEGORY_127>=
 0.5 then tree_score = 1;

    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX>= 27.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age>=82.5 then tree_score = 0;
    else if TOTAL_ELIX>= 23.5 and TOTAL_ELIX<27.5 and Age<82.5 then tree_score = 1;
 run;


 title "Cross Tab for CART scores for the validation sample.";
 proc freq data=validation_sample_tree_scored;
    tables readmitted*tree_score;
```

```
run;

title;
```

ROC_Scores

```
/* Foward Selection scroing for ROC Curve. */

proc sort data = Test_Sample_Scored; by readmit_30days; run;

data ROC_Points_Logistic (drop = CutPoint1 CutPoint2 CutPoint3 CutPoint4 CutPoint5 CutPoint6 CutPoint7 CutPoint8 CutPoint9
CutPoint10 );
   set TEST_SAMPLE_SCORED (keep = readmit_30days Readmitted scored_as_readmit);

   array CutPoint{10};
   retain CutPoint1 CutPoint2 CutPoint3 CutPoint4 CutPoint5 CutPoint6 CutPoint7 CutPoint8 CutPoint9 CutPoint10;
   length TP TN FP FN 8.;

   if _N_ = 1 then do;
      CutPoint1 = 0.026329063;
      CutPoint2 = 0.041698392;
      CutPoint3 = 0.044787703;
      CutPoint4 = 0.063577925;
      CutPoint5 = 0.080705485;
      CutPoint6 = 0.108640845;
      CutPoint7 = 0.151819648;
      CutPoint8 = 0.237723702;
      CutPoint9 = 0.369887376;
      CutPoint10 = 0.8733821822;
   end;


   if readmit_30days <= CutPoint1 then do;
      Point = 1;
      if Readmitted = scored_as_readmit then do;
         if Readmitted = 1 then TP = 1;
         else TN = 1;
      end;
      if Readmitted ^= scored_as_readmit then do;
         if Readmitted = 1 then FN = 1;
         else FP = 1;
      end;
   end;


   else if readmit_30days <= CutPoint2 then do;
      Point = 2;
      if Readmitted = scored_as_readmit then do;
         if Readmitted = 1 then TP = 1;
         else TN = 1;
      end;
      if Readmitted ^= scored_as_readmit then do;
         if Readmitted = 1 then FN = 1;
         else FP = 1;
      end;
   end;


   else if readmit_30days <= CutPoint3 then do;
      Point = 3;
      if Readmitted = scored_as_readmit then do;
         if Readmitted = 1 then TP = 1;
         else TN = 1;
      end;
      if Readmitted ^= scored_as_readmit then do;
         if Readmitted = 1 then FN = 1;
         else FP = 1;
      end;
   end;


   else if readmit_30days <= CutPoint4 then do;
      Point = 4;
      if Readmitted = scored_as_readmit then do;
         if Readmitted = 1 then TP = 1;
         else TN = 1;
      end;
      if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else if readmit_30days <= CutPoint5 then do;
   Point = 5;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
      else TN = 1;
   end;
   if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else if readmit_30days <= CutPoint6 then do;
   Point = 6;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
      else TN = 1;
   end;
   if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else if readmit_30days <= CutPoint7 then do;
   Point = 7;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
      else TN = 1;
   end;
   if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else if readmit_30days <= CutPoint8 then do;
   Point = 8;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
      else TN = 1;
   end;
   if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else if readmit_30days <= CutPoint9 then do;
   Point = 9;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
      else TN = 1;
   end;
   if Readmitted ^= scored_as_readmit then do;
      if Readmitted = 1 then FN = 1;
      else FP = 1;
   end;
end;


else do;
   Point = 10;
   if Readmitted = scored_as_readmit then do;
      if Readmitted = 1 then TP = 1;
```

```
         else TN = 1;
      end;
      if Readmitted ^= scored_as_readmit then do;
         if Readmitted = 1 then FN = 1;
         else FP = 1;
      end;
   end;

run;


proc sort data = ROC_Points_Logistic; by Point; run;


data ROC_Points_Logistic_Final(drop = TP FP TN FN readmitted scored_as_readmit);
   set ROC_Points_Logistic;
   by Point;

   length TP_Sum FP_Sum TN_Sum FN_Sum 8.;
   retain TP_Sum FP_Sum TN_Sum FN_Sum;

   if first.Point then do;
      TP_Sum = 0;
      FP_Sum = 0;
      TN_Sum = 0;
      FN_Sum = 0;
   end;

   if TP ^= . then TP_Sum = TP_Sum + TP;
   if FP ^= . then FP_Sum = FP_Sum + FP;
   if TN ^= . then TN_Sum = TN_Sum + TN;
   if FN ^= . then FN_Sum = FN_Sum + FN;

   if last.Point then do;               /* Points are designated as FP,TP. */
      FP_Rate = FP_Sum/(FP_Sum + TN_Sum);
      TP_Rate = TP_Sum/(TP_Sum + FN_Sum);
      output;
   end;
run;
```

Appendix B – R Code Listings

```
# Classification Tree with RPART

library(RPART)



training = read.csv (file='T:/EG
Projects/Knowld1/Thesis/R/training_set.csv',header=TRUE)

readmit <- factor(training$readmitted, levels=0:1,
labels=c("Not_Readmit","Readmitted"))




# grow tree

fit <- rpart(readmitted ~       AGE+

CCS_DX_CATEGORY_101+

CCS_DX_CATEGORY_103+

CCS_DX_CATEGORY_106+

CCS_DX_CATEGORY_108+

CCS_DX_CATEGORY_114+

CCS_DX_CATEGORY_127+

CCS_DX_CATEGORY_130+

CCS_DX_CATEGORY_131+

CCS_DX_CATEGORY_138+

CCS_DX_CATEGORY_155+

CCS_DX_CATEGORY_157+

CCS_DX_CATEGORY_158+
```

CCS_DX_CATEGORY_159+

CCS_DX_CATEGORY_203+

CCS_DX_CATEGORY_211+

CCS_DX_CATEGORY_238+

CCS_DX_CATEGORY_2616+

CCS_DX_CATEGORY_2617+

CCS_DX_CATEGORY_50+

CCS_DX_CATEGORY_54+

CCS_DX_CATEGORY_55+

CCS_DX_CATEGORY_58+

CCS_DX_CATEGORY_59+

CCS_DX_CATEGORY_60+

CCS_DX_CATEGORY_62+

CCS_DX_CATEGORY_651+

CCS_DX_CATEGORY_653+

CCS_DX_CATEGORY_657+

CCS_DX_CATEGORY_663+

CCS_DX_CATEGORY_95+

CCS_DX_CATEGORY_96+

CCS_DX_CATEGORY_98+

CCS_DX_CATEGORY_99+

CCS_PROC_CATEGORY_193+

CCS_PROC_CATEGORY_44+

CCS_PROC_CATEGORY_45+

```
CCS_PROC_CATEGORY_47+

CCS_PROC_CATEGORY_50+

CCS_PROC_CATEGORY_61+

CCS_PROC_CATEGORY_63+

DISCH_OTHER_CARE+

DayDischarge+

LENGTH_OF_STAY+

MARITAL_STAT+

MEDICAL_SUBSERVICE+

MS_DRG_00246+

MS_DRG_00247+

MS_DRG_00280+

MS_DRG_00282+

TOTAL_ELIX+

WeekendAdmit+

WeekendDischarge

,

                method="class", data=training,

                parms = list(prior = c(.87,.13), split = "information"))


printcp(fit) # display the results

plotcp(fit)  # visualize cross-validation results

summary(fit) # detailed summary of splits
```

```
fit

readmitted


# plot tree

plot(fit, uniform=TRUE,

  main="Classification Tree for Readmission")

text(fit, use.n=TRUE, all=TRUE, cex=.8,fancy=T)


# create attractive postscript plot of tree

post(fit, file = "c:/tree.ps",

  title = "Classification Tree for Readmission")




# prune the tree

pfit<- prune(fit, cp=   fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])


# plot the pruned tree

plot(pfit, uniform=TRUE,

  main="Pruned Classification Tree for Readmission")

text(pfit, use.n=TRUE, all=TRUE, cex=.8)

post(pfit, file = "c:/ptree.ps",

  title = "Pruned Classification Tree for Readmission")
```

Appendix C – Acronyms used in the Paper

ACA	Affordable Care Act
AIC	Akaike Information Criteria
AMA	Against Medical Advice
AMI	Acute Myocardial Infarction
AUC	Area Under The Curve
CABG	Coronary Artery Bypass Grafting
CART	Classification and Regression Trees
CCS	Clinical Classification Software
CDC	Centers for Disease Control
CHF	Congestive Heart Failure
CMS	Centers for Medicare and Medicaid Services
EMR	Electronic Medical Records
GDP	Gross Domestic Product
HCUP	Healthcare Cost and Utilization Project
LOOCV	Leave-One-Out Cross-Validation
MCC	Medical Complications
MS-DRG	Medical Severity Diagnosis Related Grouper
NCHS	National Center for Health Statistics
OECD	Organization for Economic Cooperation and Development
PCA	Principal Component Analysis
PTCA	Percutaneous Transluminal Coronary Angioplasty
ROC	Receiver Operating Characteristic
WHO	World Health Organization
YNHHSC	Yale New Haven Health Services Corporation

Notes

This work was granted approval by the Institutional Review Board (IRB) of the index hospital on October 28, 2013. Additionally, approval was granted by the Office of Research Integrity and Outreach at USM on October 31, 2013. Both review boards provided a waiver of review. For confidentiality purposes no patient health information (PHI) appears in any form within this paper.

Websites were recently revisited to verify they still existed.

We utilized SAS Enterprise Guide version 5.1. SAS Enterprise Guide is copyright© 2012 by SAS Institute Inc., Cary, NC., USA. Logs were not included in this paper due to length of the logs and for security reasons. The SAS Libname statements were altered prior to inclusion in the appendix for data security reasons. We utilized R version 2.15.3 (2013-03-01) , copyright © 2013 The R Foundation for Statistical Computing. The package RPART was used. Authors are Terry Therneau, Beth Atkinson, and Brian Ripley.
Visit http://cran.r-project.org/web/packages/rpart/rpart.pdf.

Other tools included Microsoft Office 2003 Professional, copyright © 1983-2003 Microsoft Corporation.

The system of documentation chosen was the author-date system as defined in the 15th edition of The Chicago Manual of Style.

References

Agrawal, Rakesh, Tomasz Imielinski, and Arun Swami. 1993. *Mining Association Rules between Sets of Items in Large Databases.* Proceeding SIGMOD '93 Proceedings of the 1993 ACM SIGMOD international conference on Management of data.

Agresti, Alan. 2007. *An Introduction to Categorical Data Analysis 2nd Edition.* New Jersey: John Wiley & Son.

Berkman, Barbara, Ruth Abrams. 1986. "Factors related to hospital readmissions of elderly cardiac patients". *Social Work.* 31(2):99-103.

Bernheim, Susannah, Zhenqui Lin, Jacqueline Grady, Kanchana Bhat, Haiyan Wang, Yongfei. Wang, Zameer Abedin, Mayur Desai, Shu-Xia Li, Smitha Vellanky, Elizabeth Drye, and Harlan Krumholz. 2011. *2011 Measures Maintenance Technical Report: Acute Myocardial Infarction, Heart Failure, and Pneumonia 30 Day Risk Standardized Readmission Measures.* Submitted By Yale New Haven Health Services Corporation / Center for Outcomes Research & Evaluation (YNHHSC/CORE) for CMS.

Breiman, Leo, Jerome Friedman, Richard Olshen, and Charles Stone. 1984. *Classification and Regression Trees.* New York: Chapman and Hall/CRC.

Centers for Disease Control and Prevention (CDC). 2011. Rising Health Care Costs are Unsustainable. Accessed February 3, 2014. http://www.cdc.gov/workplacehealthpromotion/businesscase/reasons/rising.html.

Centers for Disease Control and Prevention. 2014. Classification of Diseases, Functioning, and Disability. Accessed February 7, 2014. http://www.cdc.gov/nchs/icd/icd9cm.htm.

Centers for Medicare and Medicaid Services (CMS). 2009. National Health Expenditure Projections 2010-2020. Accessed February 3, 2014. https://www.cms.gov/Research-Statistics -Data-and-Systems/Statistics-Trends-and -Reports/NationalHealthExpendData/downloads/proj2010.pdf.

Centers for Medicare and Medicaid Services (CMS). 2013a. Readmission Reduction Program. accessed February 3, 2014. http://www.cms.gov/Medicare/Medicare-Fee-for-Service -Payment/AcuteInpatientPPS/Readmissions-Reduction-Program.html.

Centers for Medicare and Medicaid Services (CMS). 2013b. Lower Costs, Better Care: Reforming Our Health Care Delivery System.

Clinical Classifications Software (CCS) for ICD-9-CM. 2009. Healthcare Cost and Utilization Project (HCUP). Agency for Healthcare Research and Quality, Rockville, MD. Accessed February 7,2014. http://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp.

Cortes, Corinna, Vladamir Vapnik. 1995. "Support-Vector Networks". *Machine Learning.* 20(3):273-297.

Desai, Mayur, Brett Stauffer, Harm Feringa and Geoffrey Schreiner. 2009. "Statistical Models and Patient Predictors of Readmission for Acute Myocardial Infarction: A Systematic Review". *Circulation: Cardiovascular Quality and Outcomes.* 2:500-507.

Elixhauser, Anne, Claudia Steiner, Robert Harris, Rosanna Coffey. 1998. "Comorbidity Measures for Use with Administrative Data". *Medical Care.* 36 (1):8-27.

Fetter, Robert, John Thompson, Ronald Mills. 1976. "A System for Cost and Reimbursement Control in Hospitals". *Yale Journal of Biological Medicine.* 49(2): 123-136.

Hamel, Lutz. 2009. *Knowledge Discovery with Support Vector Machines*. New Jersey: John Wiley & Sons.

Harrell Jr., Frank. 2001. *Regression Modeling Strategies with Applications to Linear Models, Logistic Regression, and Survival Analysis*. New York: Springer.

Hastie, Trevor, Robert Tibshirani and Jerome Friedman. 2008. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. New York: Springer.

IMA Consulting, 2013, Reducing Hospital Readmissions, http://www.ima -consulting.com/uploads/0613.pdf (accessed February 3, 2014).

Krumholz, Harlan M., Yun Wang, Jennifer Mattera, Yongfei Wang, Lein Fang Han, Melvin Ingber, Sheila Roman, Sharon-Lise Normand. 2011. "An Administrative Claims Model Suitable for Profiling Hospital Performance Based on 30-Day Mortality Rates Among Patients With an Acute Myocardial Infarction", *Cardiovascular Quality Outcomes*. 4: 243-252.

Kutner, Michael, Christopher Nachtsheim, and John Neter. 2004. *Applied Linear Regression Models 4th Edition*, New York: McGraw-Hill/Irvin.

Lee, Kang In, John J.Koval. 1997. "Determination of the best significance level in forward stepwise logistic regression", *Communications in Statistics – Simulation and Computation*. 26(2): 559-575.

Maynard, Charles, Nathan Every and W. Douglas Weaver. 1997. "Factors associated with rehospitalization in patients with acute myocardial infarction". *American Journal of Cardiology*. 80(6): 777-779.

Murray, Christopher and Julio Frenk. 2010. "Ranking 37th — Measuring the Performance of the U.S. Health Care System". *New England Journal of Medicine*. 362: 98-99.

Organization for Economic Cooperation and Development (OECD). 2013. "Health: Key Tables from OECD". Accessed February 3, 2014. http://www.oecd-ilibrary.org/social-issues-migration -health/health-key-tables-from-oecd_20758480.

Ozer, Patrick. 2008. "Data Mining Algorithms for Classification". BSc Thesis Artificial Intelligence. Raboud University Nijmegen Netherlands.

Quan, Hude, Vijaya Sundararajan, Patricia Halfon, Andrew Fong, Bernard Burnand, Jean-Christophe Luthi, L. Duncan Saunders, Cynthia Beck, Thomas Feasby and William Ghali. 2005. "Coding Algorithms for Defining Comorbidities in ICD-9-CM and ICD-10 Administrative Data". *Medical Care*. 43:(11) 1130–1139.

Quinlan, J.Ross. 1993. *C4.5: Programs for Machine Learning*, New York: Morgan Kaufmann.

Refaeilzadeh, Payam, Lei Tang and Huan Liu. 2009. "*Cross-Validation.*", In *Encyclopedia of Database* Systems , eds. Ling Liu and M. Tamer Özsu, 532-538. New York: Springer.

Rosenblatt, Frank. 1958. "The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". *Psychological Review*. 65(6): 386–408.

Rodriguez, German. 2013. "Multinomial Response Models". Accessed March 31, 2014. http://data.princeton.edu/wws509/notes/c6.pdf.

Schapire, Robert, Yoav Freund. 2012. *Boosting Foundations and Algorithms*. Cambridge, Mass: MIT Press.

Shalizi, Cosma. 2009. "Classification and Regression Trees". Accessed February 5, 2014. http://www.stat.cmu.edu/~cshalizi/350/lectures/22/lecture-22.pdf.

Shi, Haijian. 2006. "Best-first Decision Tree Learning", Masters Thesis. University of Waikato, New Zealand
Shtatland, Ernest S., Emily Cain, and Mary B. Barton. 2001. "The perils of stepwise logistic regression and how to escape them using information criteria and the output delivery system". Accessed March 27, 2014. http://www2.sas.com/proceedings/sugi26/p222-26.pdf.

Steyerberg, Ewout, M.J.C. Eijkemans, Frank Harrell Jr, and J. Dik Habbema. 2000. "Prognostic modeling with logistic regression analysis: a comparison of selection and estimation methods in small data sets", *Statistics in Medicine*. 19(8): 1059-1079.

Steyerberg, Ewout, Frank Harrell Jr, Gerard Borsboom, M.J.C. Eijkemans, Yvonne Vergouwe and J. Dik Habbema. 2001. "Internal validation of predictive models: efficiency of some procedures for logistic regression analysis", *Journal of Clinical Epidemiology*. 54(8): 774-81.

Turner, Ken, Charles Burchill. 2006. "SAS code categorizing diagnosis codes and classifying into 1 or more of 31 available Elixhauser Comorbidity (ELX) groups available". Accessed February 3, 2014. http://mchp-appserv.cpe.umanitoba.ca/Upload/SAS/_ElixhauserICD9CM.sas.txt.

van Walraven, Carl, Peter Austin, Alison Jennings, Hude Quan and Alan Forster. 2009. "A Modification of the Elixhauser Comorbidity Measures Into a Point System for Hospital Death Using Administrative Data". *Medical Care*, 47(6): 626–633.

World Health Organization (WHO). 2010. "World Health Statistics 2010". Accessed February 3, 2014. http://www.who.int/whosis/whostat/EN_WHS10_Full.pdf.