PM 591 Final Project

Misha Khan

Spring 2023

Introduction

Healthcare Cost and Utilization Project, or HCUP, provided the National Inpatient Sample (NIS) dataset consisting of 200,000 patients starting in 1988. The original 175 features include date, comorbidity measures, number of chronic conditions and procedures, patient demographics (gender, age, form of insurance), hospital descriptions (location, ownership, bed size), and others. The raw dataset has 4,049,261 missing data.

In this case, length of stay is the outcome of interest. A patient's length of stay can affect staff, room, and resource availability. Also, research has shown that a longer, unnecessary length of stay can develop a greater risk to healthcare acquired infection alongside higher insurance costs. Predicting a patient's length of stay can provide insight on hospital usage and better determine under utilization of resources.

Methods

Preprocess

To reduce dimensionality of the dataset, I manually selected features that did not have multicollinearity and had a meaningful description in the data dictionary. From 175 features, the dataset was reduced to 46 features. The following features included are: AGE, APRDRG_Risk_Mortality, CM_AIDS … CM_WGHTLOSS, AMONTH, DIED, ELECTIVE, FEMALE, HCUP_ED, HOSPBRTH, NCHRONIC, NDX, NEOMAT, NPR, ORPROC, PAY1, PL_NCHS2006, RACE, and ZIPINC_QRTL. Next, categorical variables were factored. This generated the warning NAs introduced by coercion. As a result of factoring, there were some

variables that had both numbers and letters as their labels which created NAs. To handle this discrepancy, some features had to be manually releveled to remove inconsistent labeling. After this step, I used na.omit() to remove empty cells. As for the outcome variable, LOS was factored to 0 and 1 to create binary classification. Those who stayed for seven days or less were considered a short stay (0) while those who stayed longer than seven days were considered a long stay (1). Overall, this preprocessing step removed 15,968 rows which is roughly ~8% of the dataset. For my methods, a 70/30 training and testing split was used.

Method 1: Logistic Regression

A basic model interested me to see which method I could proceed with next. Because the outcome is binary, logistic regression is appropriate using glm() on all the features. In this model, these were a few of the features that had significant p value: CM_ANEMDEF1, CM_CHRNLUNG1, CM_DRUG1, DIED, ELECTIVE, NCHRONIC, NDX, NEOMAT.
Prior to calculating sensitivity and specificity of the model, I tuned the cutoff parameter using cutpointr() on the testing data. The optimal cutoff point was 0.951.

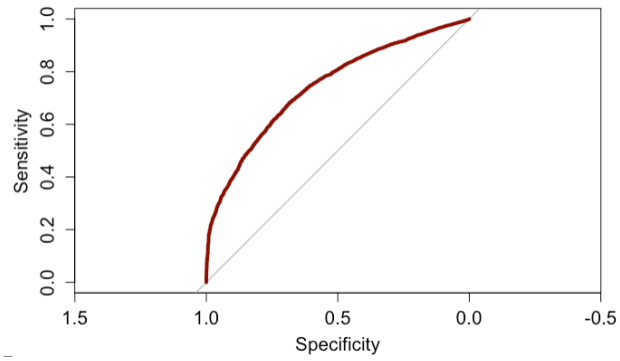
Below are the performance metrics of the model. The confusion matrix has a high number of misclassified outcomes.

```
       predicted
true    Short  Long
  Short  1654   766
  Long  16670 35860
```

```
classification_error         sensitivity         specificity
           0.3173066           0.6826575           0.6834711
```

The classification error is 31.7% while sensitivity and specificity are both 68.3%. The tradeoff of sensitivity and specificity is average but could use room for improvement especially in classification.

For classification, AUC is used as the performance metric. Here, the AUC is 0.743. The performance of the model is average but still has a notable classification error. The conservative sensitivity and specificity rate can possibly contribute to the misclassification of outcomes. Overall, this model is quite basic and does not entirely address bias variance tradeoff.



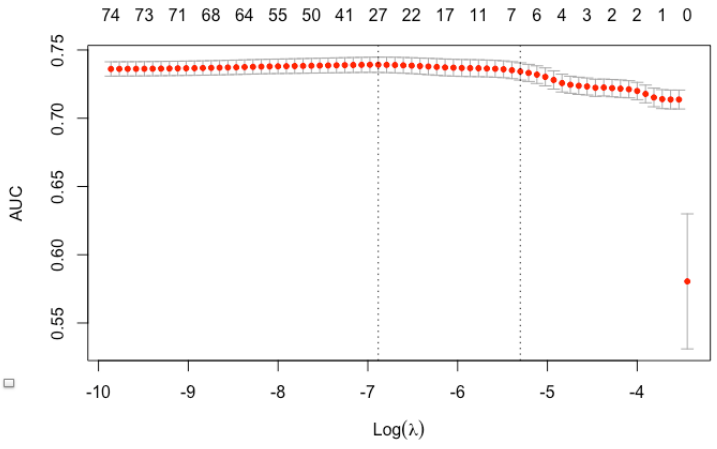
Method 2: LASSO & Elastic Net

Penalized regression adds a penalty term to reduce complexity and overfitting which can help determine an optimal balance of bias and variance. LASSO was utilized here due to the large dimensionality of the dataset and performance of feature selection. Because the outcome variable is a factor, fencoder() helped recoded the factors as dummy variables.

	Lambda	Index	Measure	SE	Nonzero
min	0.001024	38	0.7392	0.005614	27
1se	0.004979	21	0.7343	0.006320	7

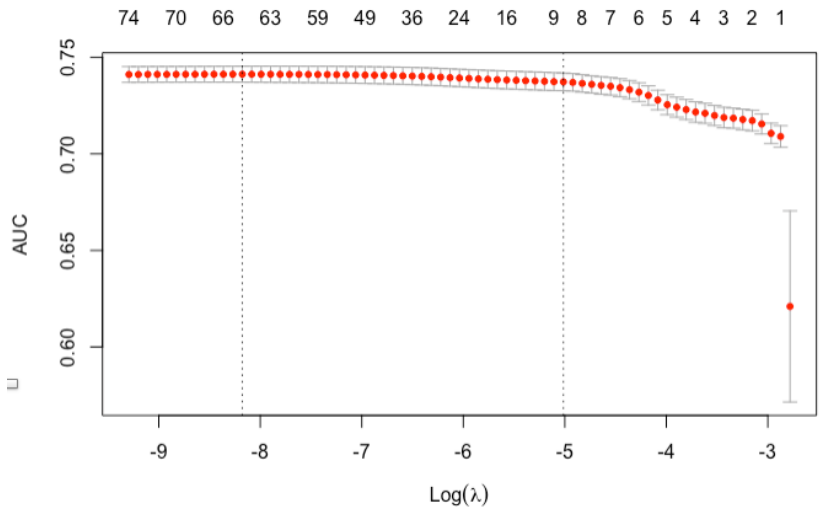
To tune the lambda parameter, cv_glmnet iterates over a grid of lambda values using cross validation (nfolds = 5) and selects the lambda with the lowest error.

On the right shows the lowest lambda values for LASSO. Optimal lambda is 0.001. For LASSO, the AUC is 0.734.



Elastic net can also control the mix of L1 and L2 penalties and additional tuning parameters. It also does not have the harsh penalization of variable selection. Optimal lambda value is relatively the same as LASSO's. The AUC for Elastic net is 0.739.

	Lambda	Index	Measure	SE	Nonzero
min	0.000281	59	0.7412	0.004143	66
1se	0.006643	25	0.7372	0.004577	9



Penalized regression gave slight improvement to feature selection and tradeoff of the model but still has an average AUC.

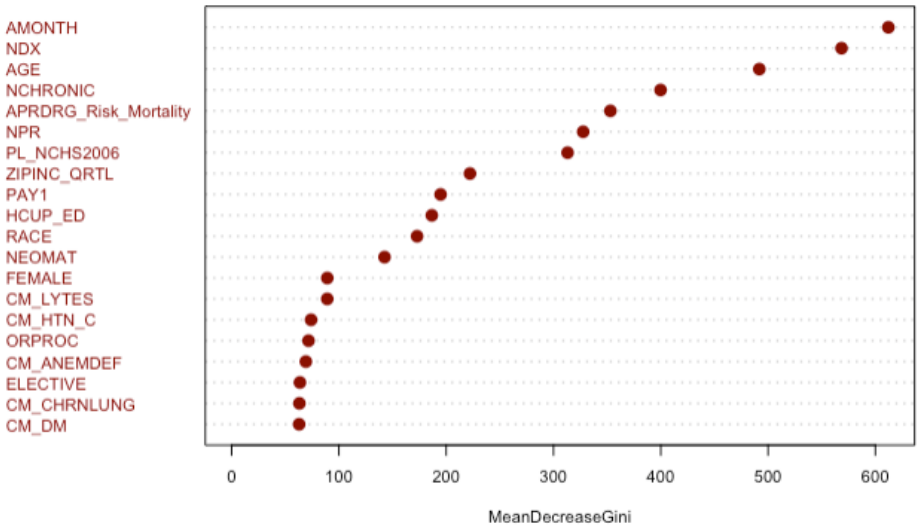
Method 3: Random Forest

Random forest can better address data that has nonlinear relationships. The split of the outcome was unbalanced: Short - 5646 and Long - 122,569. I ran two different ways of balancing the sample: a full set of the smaller class sample size and half of the smaller class sample size. By taking half of the smaller class size, it might have a better performance because using all the samples can cause the trees to be correlated. Because this is a classification problem, mtry equals the square root of the number of features is $\sqrt{46}$.

Balanced Full Set (5646)

```
 randomForest(formula = LOS ~ ., data = nis[train, ], mtry = sqrt(46),
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 20.14%
Confusion matrix:
      Short  Long class.error
Short  2906  2740   0.4852993
Long  23082 99487   0.1883184
```



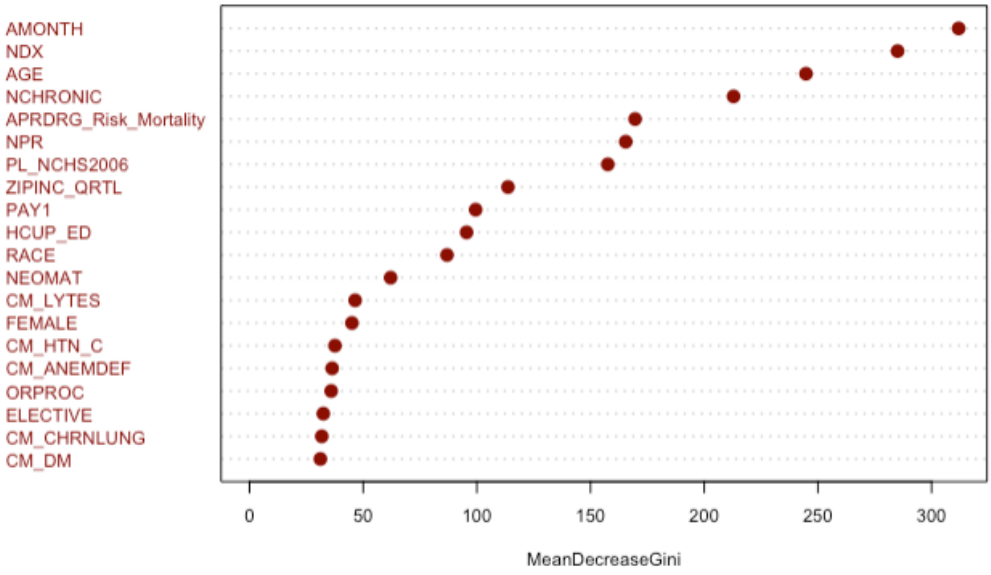
Balanced Full Set AUC: 0.763

The variable of importance in this random forest shows AMONTH, NDX, and AGE with the highest importance.

Balanced Half Set (2823)

```
 randomForest(formula = LOS ~ ., data = nis[train, ], mtry = sqrt(46),
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 25.09%
Confusion matrix:
      Short  Long class.error
Short  3452  2194   0.3885937
Long  29978 92591   0.2445806
```



Balanced Half Set AUC: 0.763

Although the AUC for both random forests are the same, there is a shift in importance for some of the lower variables when you compare it to the previous graph. The scale is lower and some of the other variables decreased in importance in comparison to the other variables like APRDRG_Risk_Mortality and NPR.

Conclusion

Method	AUC
Logistic Regression	0.743
LASSO	0.734
Elastic Net	0.739
Random Forest	0.763

To create a model, one must consider the bias variance tradeoff, optimal feature selection and tuning parameters, and high performance. Logistic regression provided a baseline understanding of the prediction problem at hand but was rudimentary and lacked complexity. Penalized regression added penalty terms to address bias variance tradeoff but did not improve performance in this case. Random forest averages predictions over multiple trees and reduces variance. By taking a balanced sample size of the smaller outcome class, this addressed misclassification error. Both ways of sampling generated the same hierarchy of variable importance of AMONTH, NDX, and AGE being the most important variables. Random forest has the highest AUC among the three methods.

Takeaways

From Random Forest's output, it makes sense as to why month, number of diagnoses, and age would be the most important variables to predict length of stay. The time of year creates varying peak times of hospital intake throughout the year therefore affecting a patient's length of stay. The number of diagnoses may show that patients with more diagnoses have a longer length of stay. Lastly, age can be important for the younger the patient, the quicker they recover and

therefore have a shorter stay. Random forest provided a sensible, high accuracy model to predict on length of stay.

In terms of the models, there are definitely areas of improvement. I would've liked for PCA to work. Unfortunately, there was minimal improvement of feature selection with PCA so it was removed. If time permitted, I would've also performed boosting because of its method to refit on the residuals. I think that would've improved the model performance.