Estimating Influenza Incidence from Diagnostic Codes

Jiayi Ji

Introduction

Background

Seasonal influenza affects 5-10% of adults worldwide annually. Health departments and governmental agencies are interested in surveillance of influenza incidence to plan and coordinate treatment activities and to limit infections through preventive measures. With the increased use of electronic medical records, large diagnostic and laboratory datasets are available in real-time and are a promising supplement to traditional surveillance activities. Diagnostic datasets of International Classification of Diseases (ICD) codes by themselves are not sensitive enough for tracking influenza incidence as influenza patients are often coded as influenza-like illnesses. Virologic sample data are more reliable but less abundant as all individuals suspected of influenza may not be tested.

Objective

In this study, we explore the application of machine learning methods using diagnostic and virologic data to get an estimate of influenza incidence. Specifically, I will choose a "best" statistical learning method for a subject with a specific combination of influenza-related ICD codes but with no laboratory confirmation, to be positive for influenza were s/he tested.

Methods

Data

Diagnostic and laboratory test data from the US Department of Defense Armed Forces Health Services Branch (DoD AFHSB) were obtained for a period spanning 12 years and covering over a 1000 treatment facilities. The diagnostic data are patient line records (~68 million) of military personnel and their dependents, coded with one or more of 15 influenza-related ICD codes. Laboratory test data (~0.5 million records) are available for a shorter duration with an overlap of 7 years with the diagnostic dataset.

Data Analysis

We identified a subset of 15 representative sites that accounted for ~10% of the diagnostic records for training. At these sites, we matched ICD codes with their tests using unique patient identifiers and diagnosis/test date ($\pm$ 3 days). A random sample of subjects with matching diagnosis/test were used to train different statistical learning methods including logistics regression, linear discriminant analysis, decision tree, bagging, random forest, gradient boosting and support vector machine with radial kernel. Five-fold cross-validated area under the curve (AUC) of the receiver operating characteristic (ROC) curves were used to compare the performance of the methods.[1] For certain methods involving the selection of tuning parameters, 5-fold cross validation were used.

Results

Tuning Parameter

Decision trees are highly susceptible to overfitting due to its natural complexity. Cost complexity pruning is one predominant method for achieving this goal, which uses a tuning parameter to selectively prune branches that do not contribute significant predictive accuracy. From Figure 1, we can see that the minimum cross-validated test error rate is achieved at 5 terminal nodes. Therefore, theres's no need for pruning.

For gradient boosting the tuning parameters include 1) The number of trees. 2) The number of split in each tree. From Figure 2, it seems that gradient boosting with depth equals to 4 and number of trees equals to 3374 has the lowest test classification error.

For support vector machine with radial kernel, from Figure 3, the optimal cost parameter occurs at cost equals to 1 when the lowest cross-validated error rate was achieved. Therefore I used that model for prediction.

Method Comparison

Figure 4 shows the result of 5-fold cross-validated AUC of the ROC curves with different statistical learning methods. Gradient boosting and logistics regression perform best among all the methods. The difference between gradient boosting (the best method) and decison tree (the worst method) is about 0.05 in terms of cross-validated AUC.

Discussions

Although not the best method, random forest is better than bagging in terms of cross-validated AUC. This is expected since random forests will only consider a random sample

of the total possible predictors while bagging will always use the same predictor for the first split and therefore cause correlation among the trees. Averaging across a set of correlated trees will not substantially reduce variance, at least not as much as if the trees were uncorrelated when random forest is adopted.

Gradient boosting is another approach to improve upon the result of a single decision tree. Instead of creating multiple independent decision trees through a bootstrapping process, gradient boosting grows trees sequentially, using information from the previously grown trees. From Figure 2, we can see that using too many trees will overfit the data and therefore the selection of tunning parameter is necessary.

A number of questions are not neatly answered through my analysis. For instance, I choose the best methods from a random sample of 15 representative sites. We could further apply logistics regression and gradient boosting to other sites to see whether the methods I selected are robust and can accommodate differences across treatment facilities. Besides, some patients might be misclassified no matter what methods I use. Therefore, we could address edge cases in the future.

Acknowledgement

This work is mentored by Sasikiran Kandula and Jeffrey Shaman from Department of Environmental Health Sciences. I wish to express my appreciation for the feedback and help they get me during each meeting. My thanks also go to Min Qian, who open the door of machine learning for me. I am forever grateful to the extraordinary classmates that I have met, with whom I have had the privilege of sharing and learning simultaneously.

Finally, my enjoyment of this project is immeasurably enhanced by the love and support of my father, Ping Ji, and mother, Huiqin Shi. Their presence is hidden in every line of R code for this work, representing the true gift of life.

References

1. James G, Witten D, Hastie T, Tibishirani R. An Introduction to Statistical Learning. 2013.

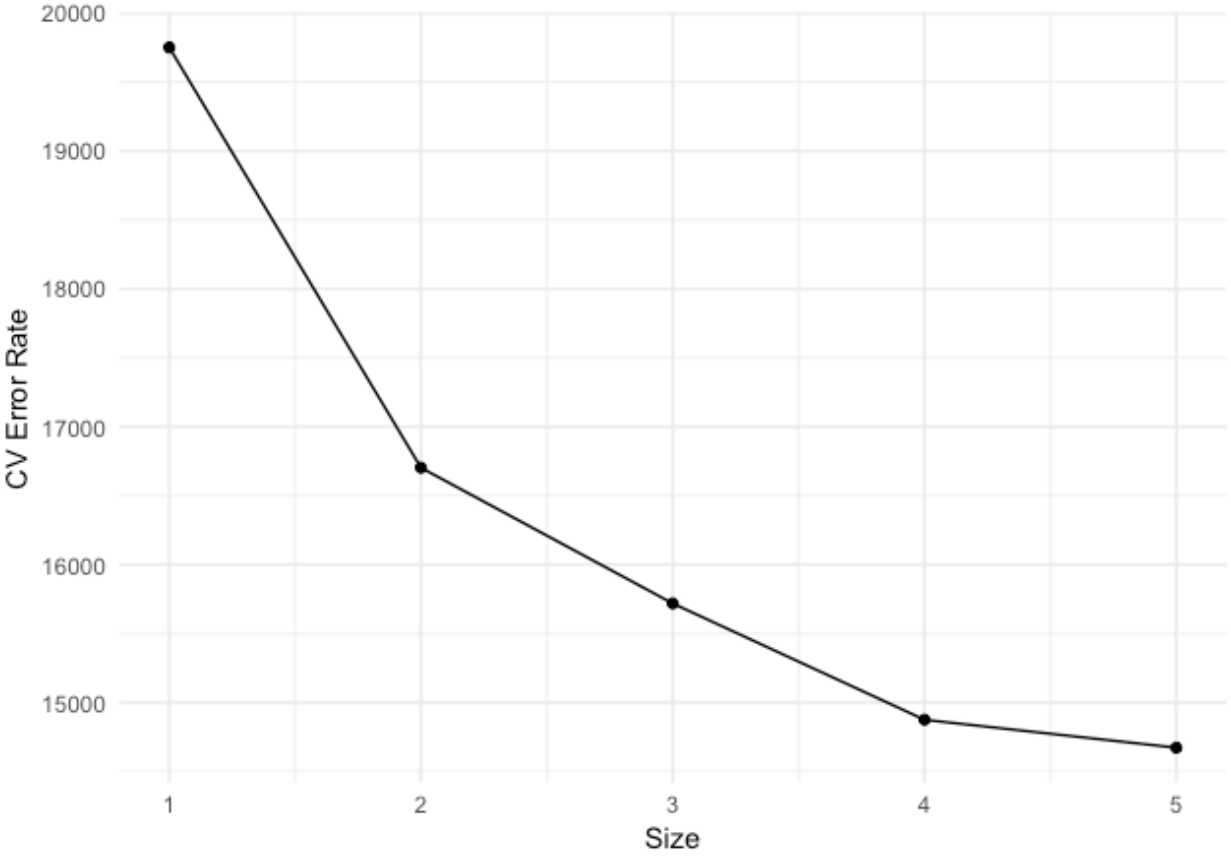

Figure 1: Pruning

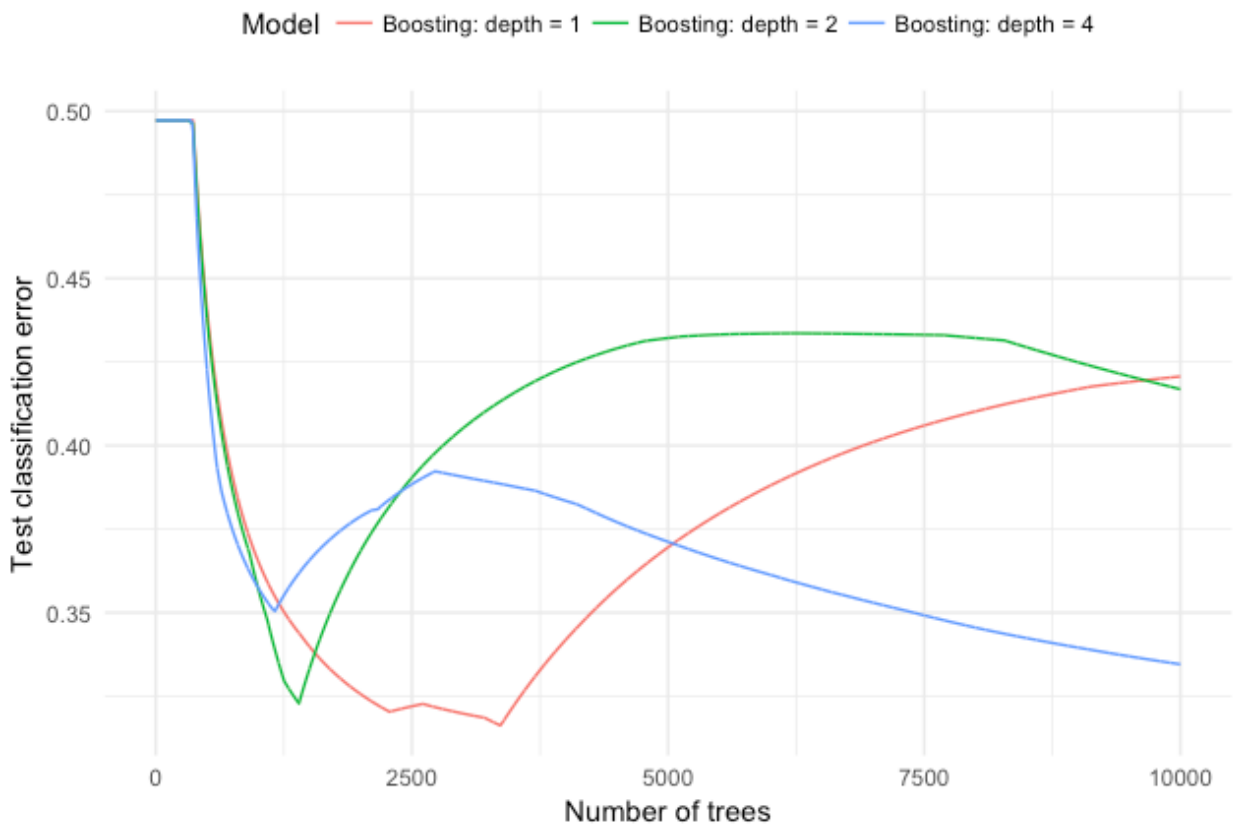

Figure 2: Gradient Boosting

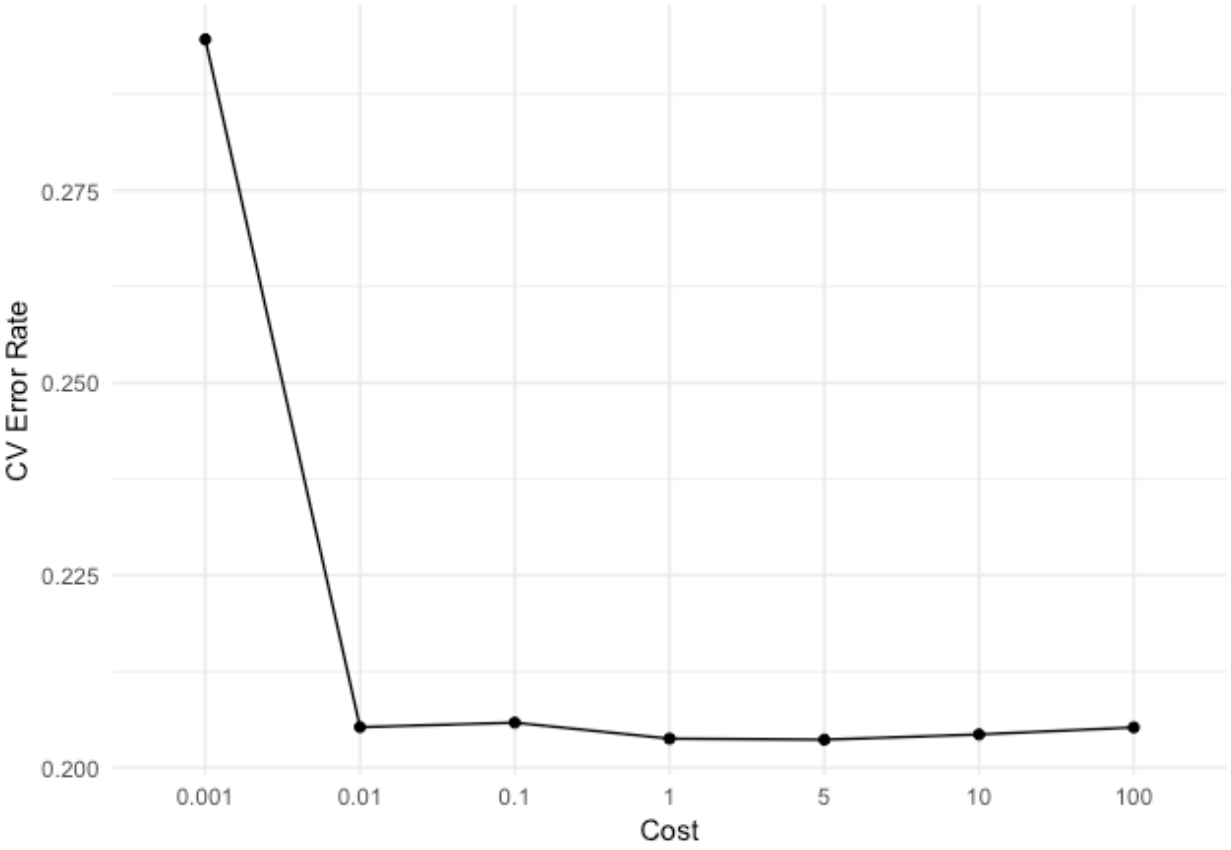

Figure 3: Support Vector Machine with Radial Kernel

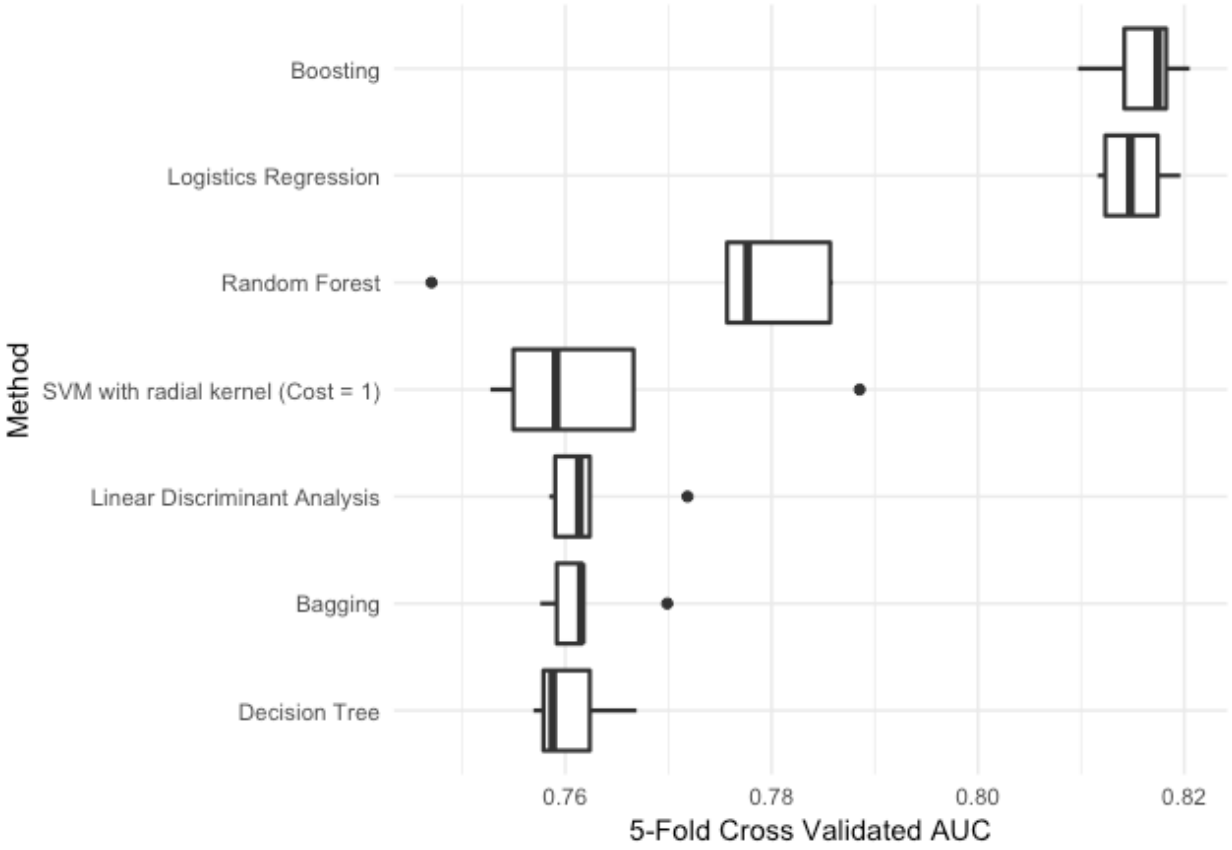

Figure 4: Model Comparison using 5-Fold Cross Validated AUC