Modeling Hospitalization Outcomes with Random Decision Trees and Bayesian Feature Selection

Thomson Van Nguyen[†]
Centre for Mathematical Sciences
University of Cambridge
Wilburforce Road
Cambridge CB3 0WA, UK
Email: tn248@cam.ac.uk

Bhubaneswar Mishra[†]
Courant Institute of Mathematical Sciences
New York University
715 Broadway, 10th Floor
New York, NY 10003, USA
Email: mishra@cs.nyu.edu

Abstract—**We propose several serial and highly parallelized approaches to modeling causality between hospitalization and healthcare observations, using data from the Heritage Health Prize competition. As any set of predictors culled from the raw dataset will be very prone to overfitting, we propose some feature selection methods to shrink to a subset of predictors that best represent the data available. We then compare the effectiveness of all our approaches, first against a self-designated test subset of the data, and then against the contest data used for evaluation of ranking and prizes. Our best implementation approach with a RMSLE (*root mean squared log error*) score of 0.462678 represents a linear blend of 20 random decision tree models with feature selection. This RMSLE score is 0.00552 away from the current leading team.**

I. Introduction

The Heritage Health Prize is a data-mining competition sponsored by the Heritage Provider Network, a physician network in Southern California, and administered by Kaggle, a company specializing in administering data-mining competitions. The goal of the prize is to develop a mathematical model that accurately predicts the number of days a patient will be hospitalized, given three years' worth of anonymized patient data. Once known, the hopeful goal is that health care providers can develop new care plans and strategies to reach patients before emergencies occur, thereby reducing the number of unnecessary hospitalizations and reducing overall administrative costs.

The United States has been slow to adopt Electronic Health Records until 2009, when it saw a surge in EHR usage among healthcare providers in the US as a result of the Health Information Technology for Economic and Clinical Health (HITECH) Act, incentivizing EHR adoption as part of the economic stimulus package passed by US congress. Ultimately, these longitudinal records from these datasets will allow insight into the health of large populations across their lifespan, thus allowing one to intuit not only observations on patterns of activity, but likely causes of these patterns.

In this paper, we introduce the Heritage Health Prize as a data-mining competition and outline various regression approaches. We describe the datasets given in detail and the problem formulation in Section II. In Section III, we outline three different approaches to the problem: an ordinary least squares linear regression with three simple predictors, a decision tree model created through recursive partitioning, and a random ensemble classifier (random forests). The latter model can be run in a highly parallelized environment using GPUs for modeling, and this is briefly described in Section III-C1. Section IV describes Bayesian feature selection as a method to increase model accuracy and reduce overfitting. All of these model approaches are evaluated in Section V, and future work is outlined in Section VI.

This work was supported by NYU SEED Grant xxxxxxx.

II. Data

The data provided by the Heritage Provider Network includes the following for all three years:

- A list of 120k members in the database, sorted by a unique, anonymized MemberID, gender, and age,
- A claims table containing 1.4 million medical claims made by the members which includes data on the primary diagnosis, physician specialty, Charlson Co-morbidity Index, and anonymized IDs for their primary care physician, vendor (company issuing the bill), and service provider,
- A labs table containing the number of lab tests performed,
- A drug prescription table containing the number of prescriptions filled by members, and
- A table of hospitalization days for members in year $1, 2$, and 3, with the goal of being able to predict year 4's hospitalization days. This table is right censored at 15, meaning the only values in this table are in $[0, 15]$.

Using an approach to preprocessing the data in consensus with the community of contestants on Kaggle, we formatted the data in a matrix X_A consisting of $78,049$ patient rows with claims in year 1 and observed values for hospitalization in year 2. The columns were individual counts for each specialist and general practitioner visits, primary condition groups, comorbidity index scores, and various composite predictors created from covariate analyses of the count predictors. The outcome vector y_A contained the number of hospitalization days in year 2 for each patient in X_A. The count predictors were normalized by a rank-preserving Box-Cox transformation to alleviate heteroscedasticity:

$$y_i^{(\lambda)} \begin{cases} (y_i^\lambda - 1) & \text{when} \quad \lambda \neq 0 \\ \log(y_i) & \text{when} \quad \lambda = 0. \end{cases}$$

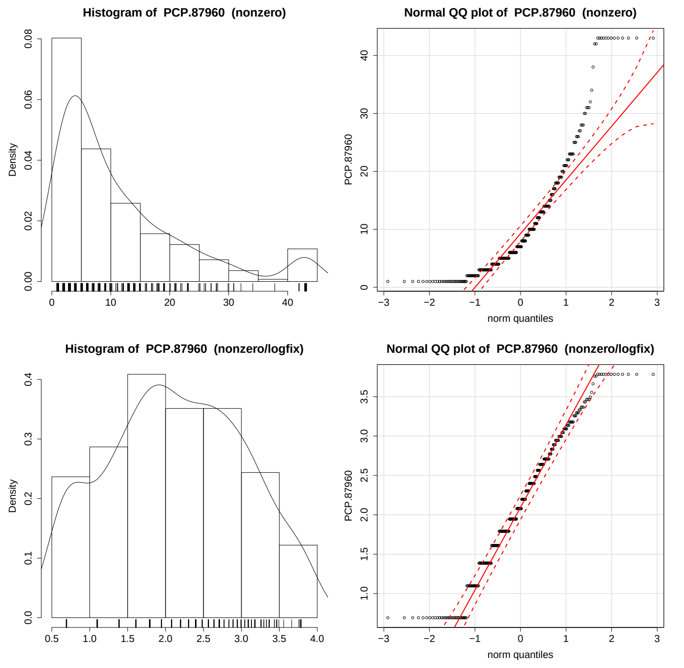

Fig. 1. Q-Q plot for the Primary Care Physician 87960 count predictor. The top right plot shows the Q-Q distribution before Box-Cox transformations have been applied, while the bottom right plot depicts the Q-Q distribution after.

Figure 1 shows the normality difference before and after such a transformation is applied. After normalization, we made model regressions on this dataset and cross-validated the model with a completely different predictor dataset X_B consisting of patients with claims in year 2 and observed values for hospitalization in year 3 (y_B). Internal model evaluation was dependent on RMSLE and Gini analysis. Submissions to the Heritage Health Prize were finally made by running the model on a last dataset X_C consisting of patients with claims in year 3, with the intent of predicting hospitalization days for year 4.

III. Approaches

We describe here a couple of approaches to the Heritage Health Prize, all with various degrees of success, complexity, and runtime.

A. Linear regression

We can run a simple ordinary least squares regression to create a weighted linear model, predicting hospitalization days for each patient j using a count predictor matrix X and regressed coefficients β:

$$y_j = X_j\beta + \epsilon, \tag{1}$$

where ϵ is the error between observed and predicted hospitalization values. An OLS regression simply attempts to minimize ϵ by finding the local minimum sum of squared euclidean distances between observed and predicted responses in the data. That is, for some estimated coefficient $\hat{\beta}_i$ we have

$$\hat{\beta}_i = (X'X)^{-1}X'y = (\frac{1}{n}\sum x_i x_i')^{-1}(\frac{1}{n}\sum x_i y_i). \tag{2}$$

B. K-nn classifier

Alternatively, we can move our linearization further downstream if we can first classify people who are ‘hospitalized’ or ‘non-hospitalized’. In our classifier, our outcome vector $\hat{y}$ is a binary variable specifying whether a patient has spent time in the hospital. Our classifier is a simple k-nearest neighbor algorithm that classifies by a majority vote of the class most common amongst its k nearest neighbors. Here, by nearest we mean the closest Hamming distance between predictors, as our count variables are discrete.

Those who have been classified as ‘non-hospitalized’ automatically get a prediction of zero days, while patients classified as ‘hospitalized’ will be run under the linear regression model described above to predict the number of hospitalization days.

The choice of an optimal k is a difficult one. In general, larger values of k will result in a more accurate classifier, but what remains critically important is minimizing the amount of noise in our predictors. For our approach, we tried various odd values for k to eliminate the occurrence of tie votes.

C. Random decision trees

Regression analysis through full recursive partitioning attempts to create a decision tree that classifies members of a population based on our feature set of predictors. While usually more accurate than linear regressions, one pitfall of full recursive partitioning is that it is prone to overfitting. To mitigate this, we create multiple random decision trees and combine the average score of all trees created to generate a prediction.

The algorithm is as follows:

1) We sample m items our feature set X_n.
2) The training set is bootstrap sampled n times for our decision tree. The rest of the the training set is used to calculate residual error.
3) A decision tree is generated from just these m variables and n training cases. This tree is fully grown and is not pruned or optimized.
4) Steps (1-3) is t times to create our desired number of trees.
5) To make a prediction, each patient is pushed down all t trees and the median vote of all trees is the final prediction..

Figure 2 is a simplified example of a decision tree with four predictors: Gender, and booleans for whether the patient had prostate cancer (male), or delivered a baby in hospital as an in-patient (female, pregnant).

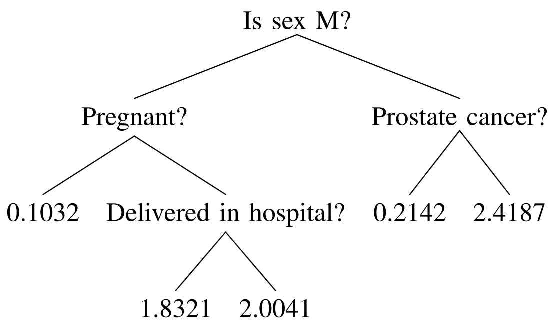

Fig. 2. Sample decision tree with four predictors. Here, a right traversal means 'yes' answers to the questions.

1) Parallel implementation: Because we are generating on the order of hundreds of thousands of trees on a large training set, this model is very computationally expensive to run on typical desktop computers with 1-4 processing cores. For example, on a PC with four cores on two 64-bit x86-64 processors (Intel i7) and 2GB RAM, a 250k RDT model took on the order of days to run. We have since moved all model computations to a 32-core, eight processor (Intel Xeon E5507) computer with 234GB RAM. With the extra computing cores, we can parallelize our random decision tree model by having each core calculate 7.8k trees, and combining all these constituent jobs at the end to create a 250k RDT model.

We are currently working on writing an implementation using Nvidia's Compute Unified Device Architecture, or CUDA. This is a C library that allows for algorithms to execute on an Nvidia Graphics Processing Unit (GPU). Parallel computation on a GPU would provide orders of magnitude difference in performance gains and runtime, as its throughput architecture emphasizes many concurrent threads at a time slowly rather than one serial thread quickly. Relevant to our implementation, this would result in much larger random decision tree models with better accuracy.

IV. Overfitting

One problem in regression analysis with having a large number of predictors is the tendency for models to describe random error moreso than any inferential rules. This 'curse of dimensionality' phenomenon is known as overfitting. We attempt to minimize the effects of overfitting by running our models with a subset of all predictors, selecting only predictors that best explain the data. Statistical methods such as bootstrapping and repeated cross-validation are also used to the same effect.

A. Feature Selection

Given n predictors, one could get an accurate assessment of which subset of predictors best describes the data by taking 2^n models of every possible subset. This becomes computationally impossible for large values of n or number of patients in our training dataset. Instead, we can perform some feature selection according to the following algorithm:

1) Train the model on the training set using all predictors.
2) Calculate model performance against the test set, using RMSLE as our performance metric:
$$\log(\sum_i Y_i - a - b_1 X_{1i} - \cdots - b_p X_{pi})^2 / n) \quad (3)$$
3) Calculate the individual variable importance and rank them.
4) For each subset size S_i, $i \in \{1, \ldots, S\}$,
    a) Keep the S_i most important variables.
    b) Retrain the model with only S_i predictors.
    c) Recalculate individual variable importance and rerank.
5) Determine which S_i yielded the most optimal RMSLE–call this S_f.
6) Fit model to S_f predictors and rank individual variables.

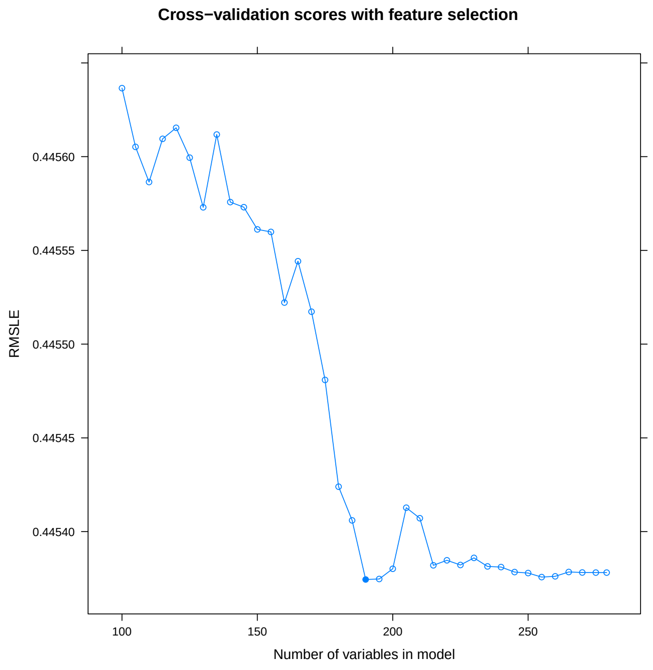

Fig. 3. RMSLE of 250k random decision tree models with varying subset sizes of predictors. The ideal number of predictors (190) was determined by Gibbs sampling and backward feature elimination.

Figure 3 depicts a sample implementation of this backwards elimination process, performing 250k random decision tree models at each subset value. Here, the ideal number of features was optimized at 190, with a best RMSLE of 0.44521.

B. Repeated cross-validation

To reduce the effects of full dataset overfitting, we perform k-fold cross-validation, repeated 25 times. In a k-fold cross-validation, we take the dataset and randomly partition them into k sub-datasets. The kth sub-dataset is used as the validation set, and $k-1$ models are trained and then cross-validated against k. This method ensures that every sample in our dataset is used exactly once.

However, one pitfall with k-fold cross-validation is that for any k you force a particular proportion of the dataset into training/testing pigeonholes. To mitigate this, we repeat k-fold cross-validation an arbitrary number of times for various values of k, and then take a linear average of the results. For our purposes, we are performing k-fold cross-validation repeated 25 times, for $k \in \{2, \dots 15\}$.

V. Evaluation

We ran several models with the various approaches described in Section III and evaluate their effectiveness. Table V contains a full comparison of all model approaches, with differing model parameters. The RMSLE score is the actual competition RMSLE given by Kaggle.

We note that a simple linear regression (LM3P) with only three predictors (age, gender, maximum Charlson co-morbidity score) achieved a 0.478246 RMSLE, which is better than half of the competitors currently in the contest. Adding every predictor in Section II gave us a 'kitchen sink' linear model (LMKITCHEN), and resulted in improvements. Lastly, we ran backwards feature elimination on the 'kitchen sink' model and was able to shrink the linear predictors to 182, or half of the feature set. This linear model (LMGIBBS) was the best linear model.

K-nearest neighbor classification and linear regression did not prove to be a fruitful approach: all three K-nearest neighbor models for varying values of $k = \{3, 5, 9\}$ (KNNK3, KNNK5, KNNK9) performed worse than the simple three predictor linear regression. The real performance gains came from running random decision trees on all predictors with increasing number of trees created (RDTKITCHEN50k, RDTKITCHEN250k, RDTKITCHEN500k). Further work in this area is needed–as our CUDA architecture is implemented for highly-parallel GPU processing, we can increase the number of trees to $1m$, $2m$, etc.

Applying backwards feature elimination on random decision tree regression netted further performance gains (RDTGIBBS50k, RDTGIBBS250k, RDTGIBBS500k), but the surprising aspect was blending the variable importance scores from the random decision tree models as linear weights for an OLS regression. Adding all features (LMRDTKITCHENBLEND) was better than either pure linear or pure RDT with all predictors, but the absolute best performance was achieved when backward feature elimination was applied to both RDT and linear models, and then blended. Figure 4 shows the variable importance generated from one RDT model at 500k trees.

VI. Future strategies

Further work is required to optimize our competition RMSLE to both remain competitive and surpass the prediction error threshold of 0.400000. Blending and averaging approaches have yielded the best results so far–it remains to be seen whether further blending of random decision trees, linear, and k-nearest neighbor models would result in a net improvement. Other classifiers have yet to be tested on the training dataset. A hybrid model in which we are able to accurately bin patients into 'hospitalized' and 'non-hospitalized' and then run a regression on the 'hospitalized' patients is one that we believe is worth finding.

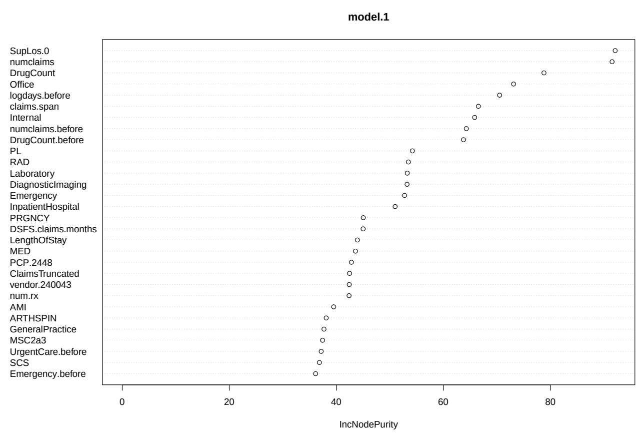

Fig. 4. Variable importance plot from

Model Approach	Best Competition RMSLE	Rank
LM3P	0.478246	11th
LMKITCHEN	0.468168	10th
LMGIBBS	0.465785	6th
KNNK3	0.484351	14th
KNNK5	0.479140	13th
KNNK9	0.478542	12th
RDTKITCHEN50k	0.467934	9th
RDTKITCHEN250k	0.466820	8th
RDTKITCHEN500k	0.465833	7th
RDTGIBBS50k	0.463438	4th
RDTGIBBS250k	0.463356	3rd
RDTGIBBS500k	0.462929	2nd
LMRDTKITCBLEND	0.464547	5th
LMRDTGIBBSBLEND	**0.462678**	**1st**

Fig. 5. Comparison of model approaches and its competition RMSLE. Our current best implementation is a linear blend of LMGIBBS and RDTGIBBS500k regressions.

Acknowledgment

The authors would like to thank Fabian Menges and Giuseppe Narzisi for their extensive assistance in maintaining the model simulation cluster at the Bioinformatics Lab at Courant. The NYU High Performance Computing cluster was also helpful in providing CPU hours for auxiliary/exploratory analyses. Kaggle and the Heritage Provider Network supplied the raw datasets. This work was funded by NYU SEED grant XXXXXX.

References

[1] Revolution Analytics. *foreach: Foreach looping construct for R*, 2011. R package version 1.3.2.

[2] W.L. Beaver, K. Wasserman, and B.J. Whipp. Improved detection of lactate threshold during exercise using a log-log transformation. *Journal of Applied Physiology*, 59(6):1936, 1985.

[3] M.E. Charlson, P. Pompei, K.L. Ales, and C.R. MacKenzie. A new method of classifying prognostic comorbidity in longitudinal studies: Development and validation* 1. *Journal of chronic diseases*, 40(5):373–383, 1987.

[4] H. Chipman. Bayesian variable selection with related predictors. *Canadian Journal of Statistics*, 24(1):17–36, 1996.

[5] Nvidia Corporation. Compute unified device architecture programming guide. *NVIDIA: Santa Clara, CA*, 83:129, 2007.

[6] G.J. Escobar, J.D. Greene, P. Scheirer, M.N. Gardner, D. Draper, and P. Kipnis. Risk-adjusting hospital inpatient mortality using automated inpatient, outpatient, and laboratory databases. *Medical care*, 46(3):232, 2008.

[7] E.I. George and R.E. McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, pages 881–889, 1993.

[8] H. Grahn, N. Lavesson, M.H. Lapajne, and D. Slat. A cuda implementation of random forests-early results. In *Third Swedish Workshop on Multi-core Computing*. Chalmers Institute of Technology, 2010.

[9] R. Groth. *Data mining: building competitive advantage*. Prentice Hall PTR, 2000.

[10] M. Kuhn. Building predictive models in r using the caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[11] K.H. Lee, S. Chakraborty, and J. Sun. Bayesian variable selection in semiparametric proportional hazards model for high dimensional survival data. *The International Journal of Biostatistics*, 7(1):21, 2011.

[12] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[13] K. Lunetta, L.B. Hayward, J. Segal, and P. Van Eerdewegh. Screening large-scale association study data: exploiting interactions using random forests. *BMC genetics*, 5(1):32, 2004.

[14] H. Quan, V. Sundararajan, P. Halfon, A. Fong, B. Burnand, J.C. Luthi, L.D. Saunders, C.A. Beck, T.E. Feasby, and W.A. Ghali. Coding algorithms for defining comorbidities in icd-9-cm and icd-10 administrative data. *Medical Care*, pages 1130–1139, 2005.

[15] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[16] T. Sharp. Implementing decision trees and forests on a gpu. *Computer Vision–ECCV 2008*, pages 595–608, 2008.

[17] L. Torgo. *Data Mining with R: Learning with Case Studies*. Chapman & Hall/CRC, 2010.

[18] S. Urbanek. multicore: Parallel processing of r code on machines with multiple cores or cpus. *R package version 0.1-3, URL http://www. rforge. net/multicore*, 2009.

[19] Hadley Wickham. The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1–29, 2011.

[20] S.L. Zeger and M.R. Karim. Generalized linear models with random effects; a gibbs sampling approach. *Journal of the American statistical association*, pages 79–86, 1991.