Heritage Provider Network Health Prize

Round 1 Milestone Prize

How We Did It – Team 'Market Makers'

Phil Brierley
Tiberius Data Mining
philb@tiberius.biz

David Vogel
Voloridge Investment Management
dvogel@voloridge.com

Randy Axelrod
Healthcare Consultant
rcaxelrod@cox.net

Overview

The Heritage Provider Network Health Prize is a competition to develop a predictive algorithm that, based on historical claims, will help identify those patients most likely to be admitted to hospital. The competition will run for two years, with milestone prizes awarded every six months. Team 'Market Makers' were top of the leaderboard on the 1st September 2011, the deadline date for the first milestone award.

This document describes the methodology used in generating the *Market Makers* submission. In the appendices we include details of all the relevant models built and the data used and scripts to produce a model capable of a leaderboard position in the top 10% of teams.

Broad Outline

There were four distinct steps in creating the solution;

1. *Manipulating the raw data into a suitable format for modelling*
   The data provided for use in the predictive models was each claim made by a patient, so there could be multiple records per patient. The objective is to predict the Days in Hospital, a single value for each patient per year. The claim level data was aggregated to a patient level to generate modelling data sets.

2. *Predictive Modelling*
   Predictive models were built utilising the data sets created in Step 1. Numerous mathematical techniques were used to generate a set of candidate solutions.

3. *Ensembling*
   The individual solutions produced in Step 2 were combined to create a single solution that was more accurate than any of its components.

4. *Future Proofing*
   The prediction data is one year ahead of the data provided to build the models. Events such as medical advances or policy changes might have subsequently occurred that may affect the model. Techniques can be used to help ensure the model works as efficiently as possible in the 'future'.

Data Manipulation

Data was provided for patients who made claims in a particular year, along with the number of Days in Hospital they spent in the subsequent year. Data from two contiguous years was provided on which to create the models, with the objective to use the claims data from a third contiguous year to make the predictions.

Two distinct modelling data sets were generated;

1. Only data from the previous year was used. This means that if a patient claimed in both years, they would appear twice, each year counting as a separate record. This is referred to as Dataset 1 inAppendix A.

2. All claim history from the previous two years was aggregated, meaning there would only be one record per patient. This gave a longer claim history, but lost resolution on recency. This is referred to as Dataset 2 in Appendix A.

The majority of the variables created were purely data driven, with no regard for context. An admission risk score was developed around the Primary Condition Group that was based on medical experience only (Dr Axelrod is a physician of 29 years with 14 years clinical practice and 22 years clinical informatics experience). This score was a 1-5 ranking for each PCG, and split by age band. The admission risks used are in Appendix A.

Model Building

The objective function for the model to minimise is:

$$\sqrt{\frac{1}{n}\sum_{i}^{n}[\log(pred_i+1)-\log(act_i+1)]^2}$$

where:

1. *i* is a member;
2. *n* is the total number of members;
3. *pred* is the predicted number of days spent in hospital for member i in the test period;
4. *act* is the actual number of days spent in hospital for member i in the test period.

If the actual number of days spent in hospital is transformed to the log scale,

$$act1_i = \log(act_i+1)$$

then the function to minimise becomes:

$$\sqrt{\frac{1}{n}\sum_{i}^{n}[(pred_{1i})-(act_{1i})]^2}$$

- the root of the mean squared error, or RMSE. This is convenient, as there are many existing algorithms designed to minimise this function. Hence we can utilise these algorithms by transforming the target variable and then reversing the transformation on the predicted values before submitting to the leaderboard:

$$pred\_submit_i = \exp(pred1_i) - 1$$

There were four underlying algorithms used in our models, all of which are freely available in the R language for statistical computing. Online references for each algorithm are given in the hyperlinks below.

1. *Gradient Boosting Machines*
   Greedy Function Approximation: A Gradient Boosting Machine, Jerome H. Friedman
   GBM package in R

2. *Neural Networks*
   Back Propagation Weight Update Rule
   Neural Network Source Code

3. *Bagged Trees*
   Random Forests Papers
   randomForest package in R

4. *Linear Models*
   lm function in the R Stats Package
   Least Squares

Multiple models were built on the two data sets using various parameter settings and variable subsets. Gradient Boosting Machines were the most powerful individual algorithm, with a leaderboard score around 0.461 being consistently achievable. They also gave the best individual model of 0.460 when used in conjunction with the data set containing only one year of history. Bagged Trees and Neural Network ensembles gave leaderboard errors of the order of 0.463, with linear regression the poorest individual performer at 0.466.

Truncation of the predictions was found to be useful in certain models. The predictions were already capped at 0 and 15 days, as these were the limits of possible values. 'Further capping' is to test what happens if rather than choose zero as the cap, other values such as 0.01, 0.02 etc. are investigated. We can use the cross validation sets to test what the best value to cap at is, in order to result in the lowest RMSE (in-fact cross validation sets are not required to determine if an algorithm is performing inefficiently at the extremes, the actual training results from the training data can be used to see if capping improves the training RMSE). Fig 0 shows how this further capping can improve the models, but also demonstrates that capping too far can be unwise.

Not all our models were further capped, and those that were we chose a value around 0.02 rather than determine it 'on the fly' for each model. A more systematic approach should improve the accuracy of our base models. We saw small improvements in the leaderboard score of the order of 0.000015 by employing further capping.

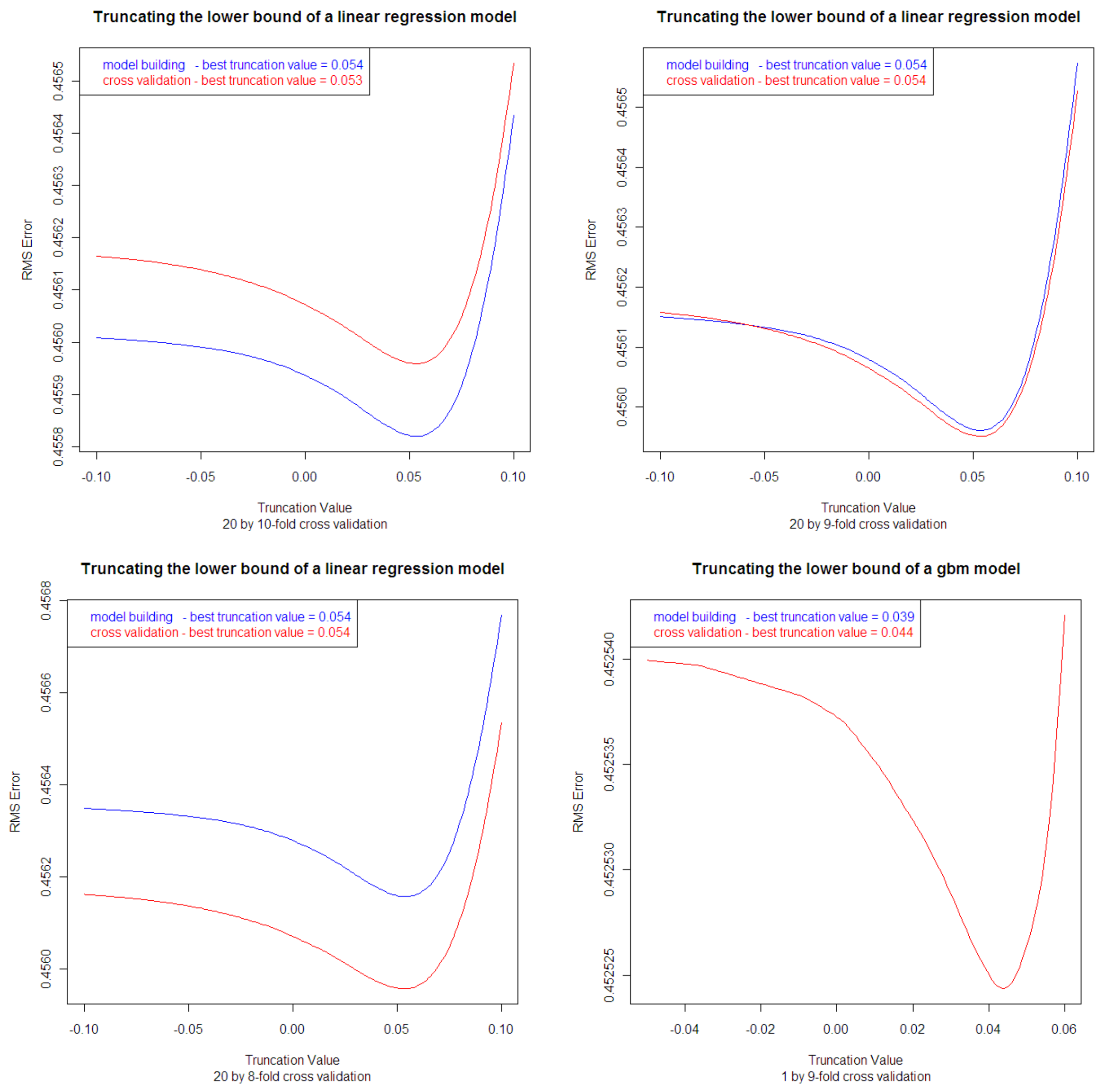


Fig 0 - Capping of the models lower bound beyond zero can be seen to be beneficial in both linear regression and GBM models. The best value to cap at is very consistent between the model building and cross validation sets, being 0.054 (on the log scale) for the linear regression model. It can also be seen that in the linear models, the number of folds chosen does have an impact.

Ensembling

The final model was based on a linear combination of the candidate models.

Once the modelling data sets were finalised, the largest incremental gain was not achieved by fine tuning the training parameters of an individual algorithm, but by combining predictions from multiple algorithms. Fig 1 and Fig 2 show that different algorithms can arrive at their solutions by different paths – they look at the problem from different perspectives. When these algorithms are combined there is resulting synergy.

Another way to view this is that different algorithms will get it wrong in different ways and in different places. Combining predictions is a way of protecting against this over (or under) fitting and should be viewed as a technique to prevent certain predictions from being very wrong. This defensive technique (don't put all your eggs in one basket) has the added benefit of improving the overall accuracy.

During the model generation process, repeated n-fold cross-validation was used to essentially generate out of sample prediction sets for the training data. These were then used in a linear regression model to establish the weighting of each candidate in the final solution.

Repeated n-fold cross validation is where you do n-fold cross validation not just once, but multiple times. The final cross validation set is then just an average. Predominantly we used two values of n, 2 and 12. When 2 was use we repeated multiple times until the cross validation error converged. The reason for using repeated 2-fold cross validation was mainly to overcome computer memory issues (the training data set is half the size of the complete data set) and to decrease processing time for each pass of the algorithm, rather than any specific mathematical benefits. Twelve fold cross validation was used when 12 processors were available so each fold could be computed simultaneously. The exact number of folds and hence the number of records presented to the algorithm does appear impact the model performance, as demonstrated in fig 0. Further work is required to determine the optimal number of folds.

As two different data sets were used with only some overlapping records, this weighting technique could not be based on the entire data sets. Hence subsets of the common records between the two data sets were used so that solutions could be combined using a common linear regression formula.

In order to protect against overfitting (seen by large coefficients in the linear regression model), we built many linear regression models on a randomly a selected 50% of the variables (or candidate base models), with the final overall coefficients for each base model being just an average of the coefficients of each linear regression (where the coefficient is zero if the base model is not selected). See http://ausdm09.freeforums.org/improving-generalisation-by-ensembling-t13.html for further details and R code to achieve this. We then calculated the variable importance of each base model in the linear ensemble (see Appendix A for details of the method) to remove the base models that did not contribute. We determined 20 base models were sufficient, repeating the process with these 20 models to arrive at the final weightings. The models and weightings are given in Appendix B.

An alternative to weighting by model was to just take the median prediction for each patient from a series of models. This is a very simple but powerful technique that does not rely on a hold-out or cross-validation generated set to base the model weightings on.

By way of an example, during the development of one of the data sets, models were iteratively built as new data was added and the algorithm settings refined. Any single model using the final data and algorithm reached a plateau score on the leaderboard of 0.4607. By simply taking the median of over 60 models built during this development process and not having to worry about how good or poor they were, a leaderboard score of 0.4603 was achieved. This might seem a small improvement, but it was real.

Alternatively, just taking the median of 9 models (each scoring no better than .4603) built using the differing data sets and algorithms gave a score of 0.4590, good enough for 9th place on the leaderboard.

The technique of blending algorithm dependent predictions with algorithm independent predictions was successful in a previous Kaggle competiton. The algorithm independent predictions are referring to the median models. For each record, the actual prediction could be from any of the algorithms in the median mix (whichever one gives the median score for that record).

The weightings were applied to the log scale version of the predicted values.

The final solution was an ensemble of approximately 20 models, although some of these models were essentially ensembles in themselves (such as the median based models).

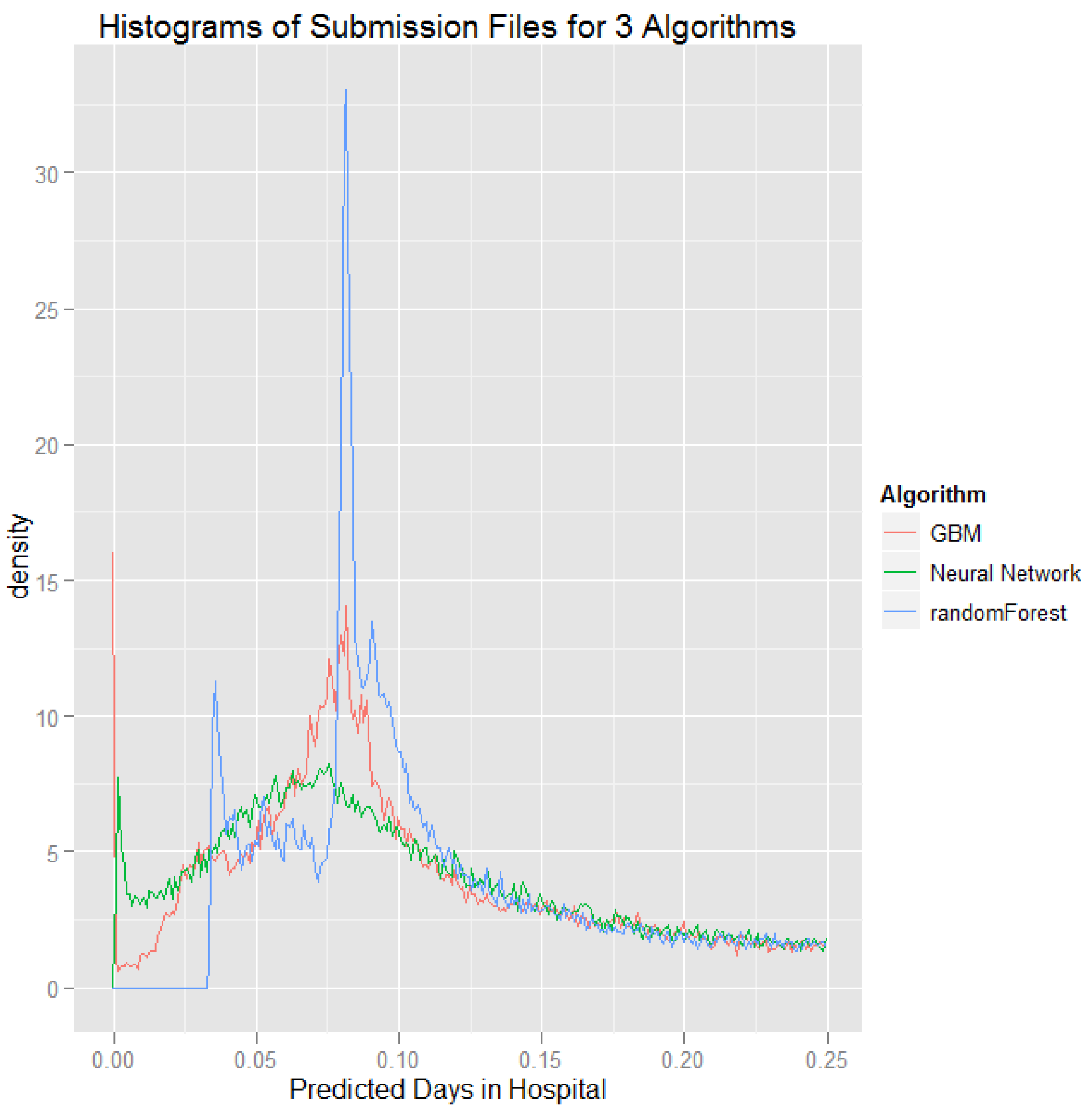


Fig 1 –*Different algorithms can produce markedly different models, even though the reported leaderboard scores may be very similar. These histograms show the randomForest model never resulted in a prediction below zero, where as the GBM and Neural Network did (which had to be truncated).*

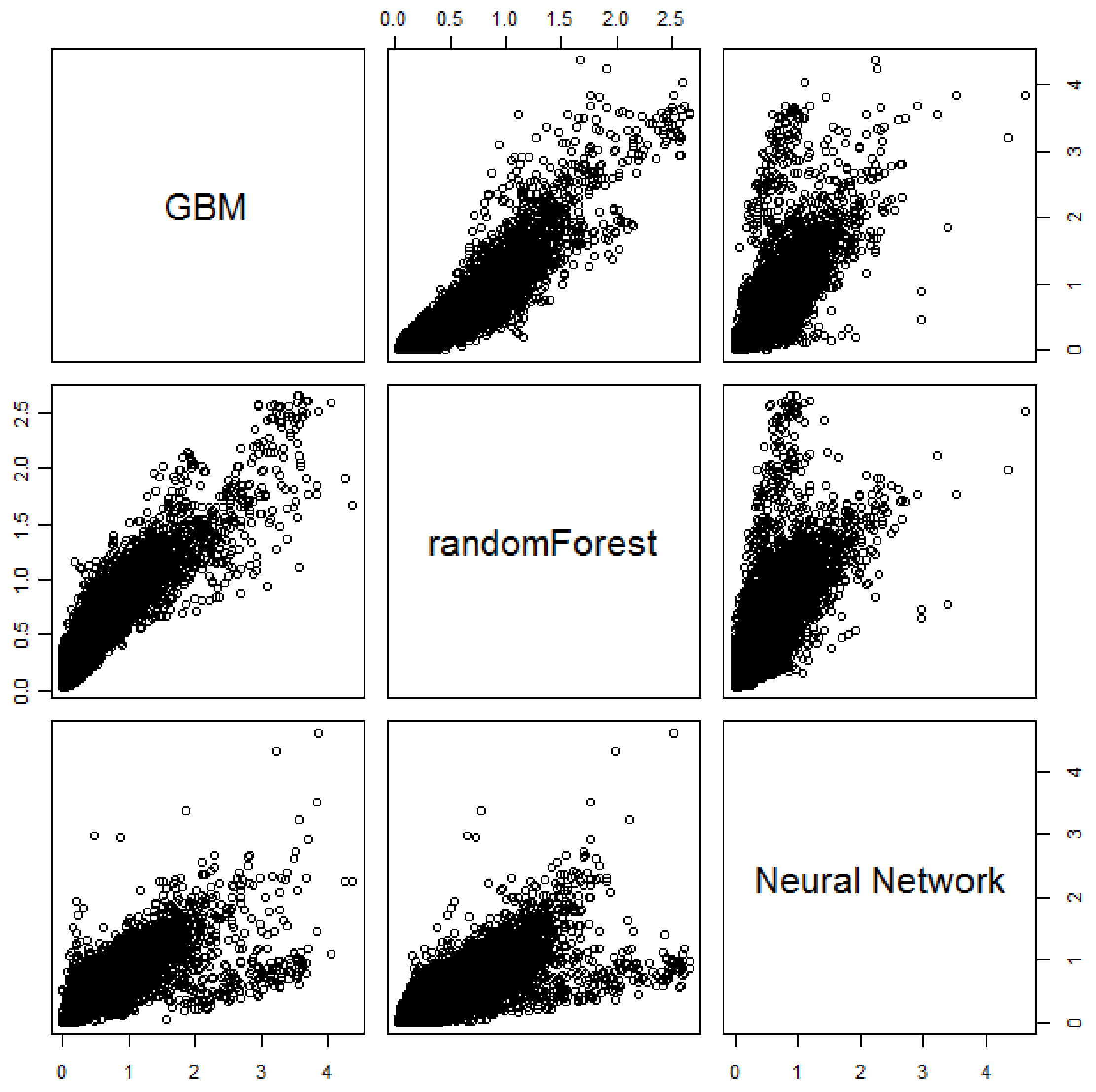


Fig 2 – *Scatter plots of the leaderboard predictions from three different algorithms. The lower the correlation then the better the synergy when combining. These visualisations are useful in identifying features or clusters of patients that the algorithms might be in disagreement about.*

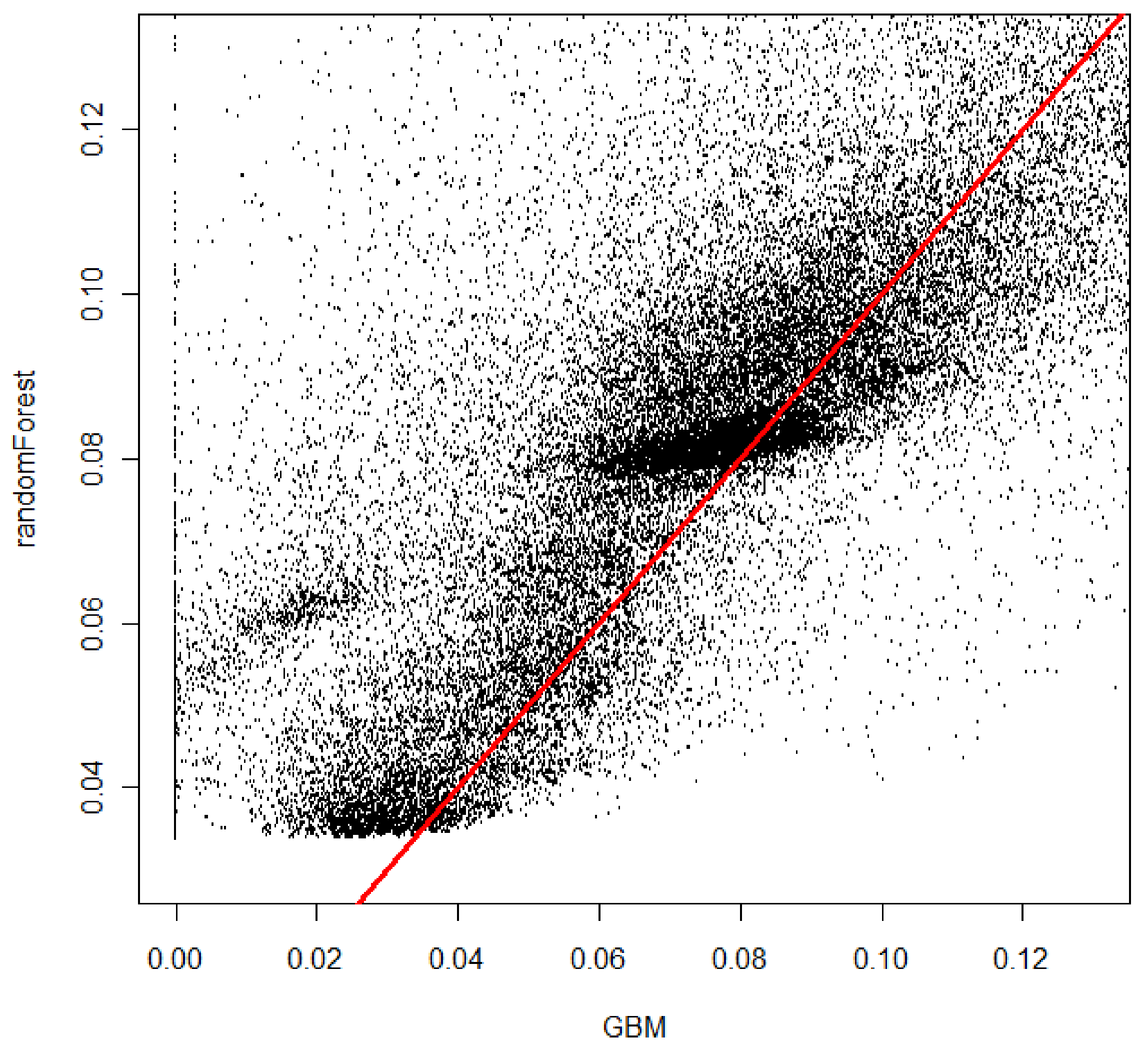


Fig 3 – *Three distinct patient clusters can be seen where both algorithms agree the patients are similar to each other.*

Future Proofing

In predictive modelling when we use historical events to predict the future, an underlying assumption is that the future will be the same as the past – but this is not always going to be the case.

For example;

- Medical advances could result in conditions that previously required a stay in hospital no longer requiring such (e.g. keyhole surgery).
- Clinicians, Vendors and Providers come and go through time. Each may have their own particular traits that affect hospital admissions during their tenure.

- Policies may change, e.g. the number of nights after giving birth that mothers are encouraged to stay in hospital.

In this particular task, we can detect certain changes that we can then guard against – forewarned is forearmed.

Any significant changes in predictor variable distributions can be identified as we have these predictor variables from the future time period. There are numerous ways to do this, such as:

- comparing means between the training and leaderboard data
- overlaying distributions and visualising the patterns
- calculating a univariate metric that quantifies the separation of two sets, such as the AUC, with the training data being one set and the leaderboard data the other
- build a classification model using all the variables and calculating the variable importance

Fig. 4 shows one such predictor variable that significantly changes through time. A derived variable was created which was the maximum PayDelay per patient per year. If we look at the distribution of this variable, then we see in Y1 a significant spike at 162 days. This reduces in the subsequent years, with the appearance of a spike at zero days in Y3, the data used for the leaderboard predictions. Y1 and Y2, the data used to build the models, did not have this phenomenon of a host of patients apparently being paid immediately.

We chose to not consider any variables based on Pay Delay as potential predictors as there has been some systematic change that will potentially alter the meaning of these variables in time. It is unknown how this affected the model performance as models were never built utilising this variable. All other variables were considered.

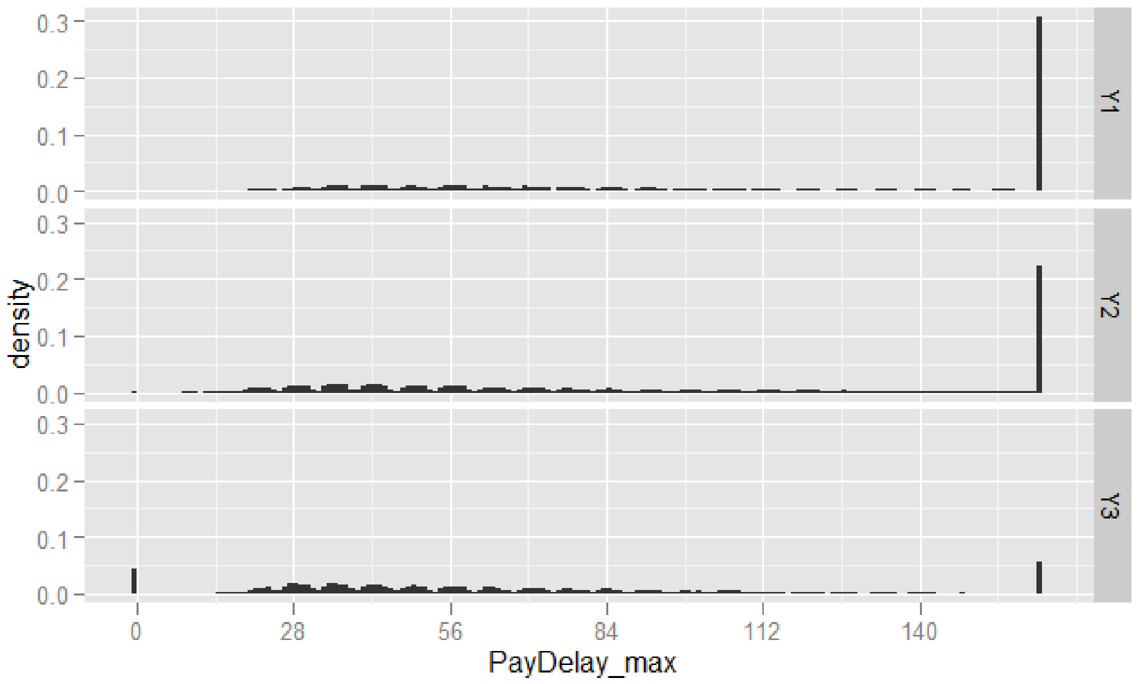


Fig 4 – *PayDelay can be seen to be significantly different in Y3 when compared with Y1 & Y2, with the appearance of a maximum PayDelay of zero. Any model built on Y1 & Y2 would not know how to deal with this value as it was not present in those years.*

In addition to the Y3 training data, there is also future information that can be extracted from the leaderboard itself:

- the forum post *The 'Optimized Constant Value' Benchmark* demonstrated how to extract insight from the leaderboard that can be used to calibrate the models to the future
- we blogged about another technique early on in the competition that indicated the Days in Hospital in Y4 looks more like Y2 than Y3

We found that calibrating the final predictions so that the overall average predicted Days in Hospital matched the optimized constant value benchmark gave a small improvement. Such 'insight to the future' is not really useable in a real life situation, but for the small difference it can make it is unavoidable to use this information in order to win this competition.

Appendix A - Variables used in the modelling

Appendix B contains SQL script to generate the majority of the variables used in the modelling data set 1. Here are descriptions of other variables that were created and tables for cross referencing to determine which variables were used in each base model.

Appearance Cohort flags – a set of binary flag for each patient to indicate those appearing in the same years (e.g. Y1 only, Y2 only, Y1,Y2 & Y3, etc.)

Counts for pairwise combinations of each PrimaryConditionGroup, Specialty and Procedure Group. This resulted in many fields, which were reduced by building classification models (using a binary target - admitted to hospital or not) and taking the most important variables in the resulting model.

The task here is to remove 'useless' variables in order to reduce the data set size – for example if there is a Specialty * PCG combination that only occurs for only one patient then including this combination as a variable if futile, will add little to the overall model accuracy but will contribute to overfitting.

There are numerous techniques commonly used in classification problems for variable elimination, such as stepwise logistic regression. To make the problem binary (0/1), we considered it as a 'did you go to hospital' prediction task (if DIH >= 1 then DIH = 1).

We considered the counts of each paring individually (PCG * Specialty, PCG * PG, Specialty * PG). For each we built a logistic regression model using all combinations (ie variables), and then calculated the variable importance to the model of each combination. If the least important did not affect the model accuracy it was removed, and the process started again. This was repeated until all combinations in the model suggested they were important.

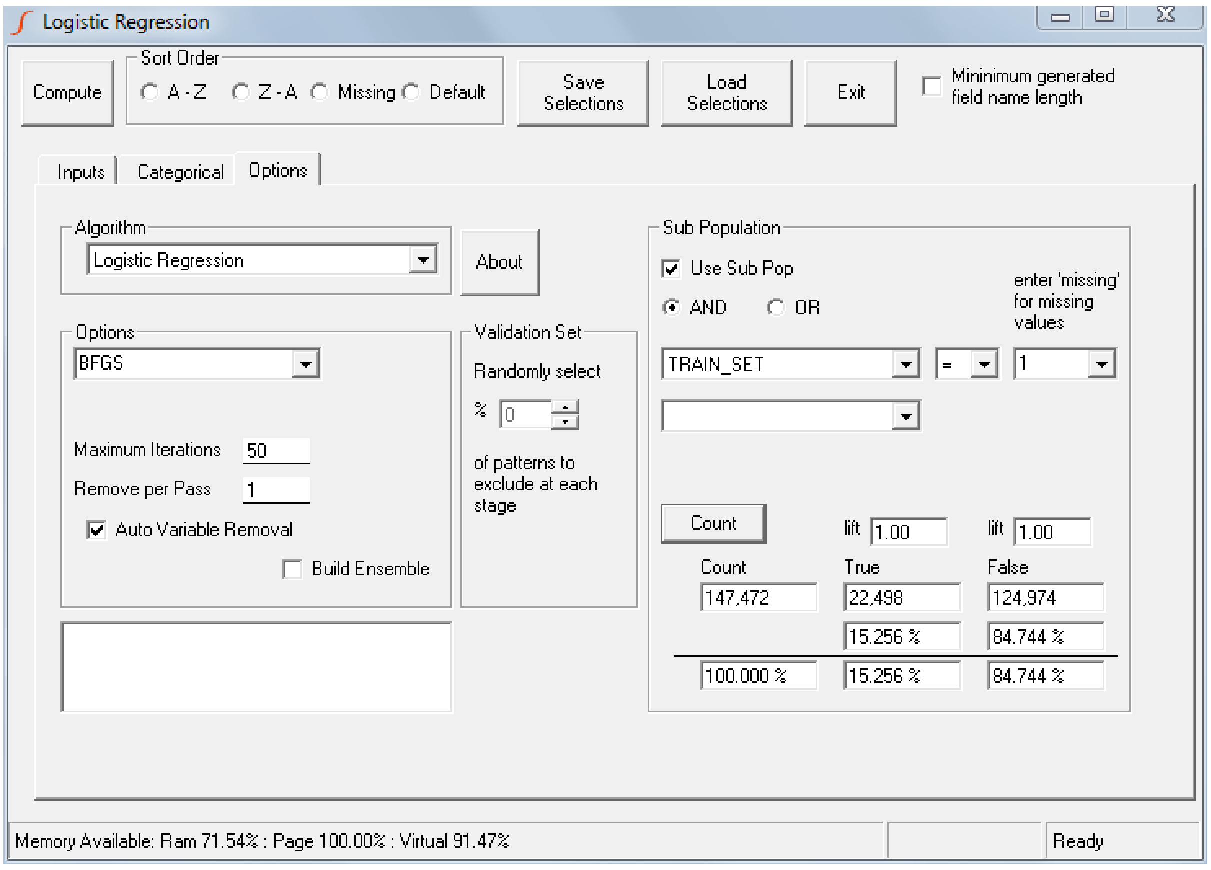
In order to calculate the model variable importance, each variable is in turn randomly permuted and the model accuracy (AUC/Gini) recalculated. This permutation is repeated several times and an average resulting accuracy taken. If the resulting model accuracy with the permuted variable is not significantly diminished, then this variable can be safely removed in the knowledge that it has little effect in the overall model.

The number of times the permutation is repeated (sweeps) for each variable should be as many as possible but at least two. This repetition improves the accuracy of the results but is computationally expensive for large data sets with many variables, especially when only one variable is being removed per pass. We used three sweeps.

In order to determine if a variable is significant, the data is initially randomly split in half and the permuted accuracy calculated for each half and compared with the base accuracy for each half. If the permuted accuracy is worse on both halves, then the variable is kept, otherwise it is a candidate for removal.

Due to the random permutation, repeating this process will not always result in the same subset of selected variables, but they should result in producing the same model accuracy. Hence we do not expect you to be able to exactly replicate the variables we ended up with, and if we repeated the process we ourselves would end up with a different subset in our final modelling data set, but this would be of little concern, as the important variables would be there, but maybe just not the same variables of lesser importance

In order to perform the variable reduction process, the logistic regression module was used from the Tiberius Data Mining software v7.0.3, where the process is automated.



Logistic Regression

Compute | Sort Order: ○ A - Z ○ Z - A ○ Missing ○ Default | Save Selections | Load Selections | Exit | ☐ Mininimum generated field name length

Inputs | Categorical | Options

Algorithm: Logistic Regression | About

Options: BFGS

Maximum Iterations: 50

Remove per Pass: 1

☑ Auto Variable Removal

☐ Build Ensemble

Validation Set: Randomly select % 0 of patterns to exclude at each stage

Sub Population

☑ Use Sub Pop

(●) AND ○ OR

enter 'missing' for missing values

TRAIN_SET = 1

Count | lift 1.00 | lift 1.00

Count	True	False
147,472	22,498	124,974
	15.256 %	84.744 %
100.000 %	15.256 %	84.744 %

Memory Available: Ram 71.54% : Page 100.00% : Virtual 91.47% | Ready

For each Primary Care Physician (PCP), Vendor and Provider, a value was calculated that was the probability that a patient associated with the entity would visit hospital. Each patient was then allocated the highest probability of all the PCPs (Vendors or Providers) that they were associated with, generating 3 fields in total. Only those PCPs (Vendors or Providers) were considered that were associated with a ‘critical mass’ of distinct patients in Y2 and Y3, with all others grouped together as ‘other’. This critical mass was chosen to give a total number of entities of around 200, being 100 for PCP, 200 for Vendor and 250 for Provider.

Lab & Drug Velocity: For lab and drug counts, a field was created that was the difference between the count in the earliest DSFS interval and the count in the latest DSFS interval.

Data Set 1 – only data from a single year used

Variable	SQLProvided	Subset	Description
Cohort Flags	N	1	appearance cohort flags
ClaimsTruncated	Y	2	raw value
age_05	Y	2	binary flag
age_15	Y	2	binary flag
age_25	Y	2	binary flag
age_35	Y	2	binary flag
age_45	Y	2	binary flag
age_55	Y	2	binary flag
age_65	Y	2	binary flag
age_75	Y	2	binary flag
age_85	Y	2	binary flag
age_MISS	Y	2	binary flag
sexMALE	Y	2	binary flag
sexFEMALE	Y	2	binary flag
sexMISS	Y	2	binary flag
no_Claims	Y	3	counts
no_Providers	Y	3	counts
no_Vendors	Y	3	counts
no_PCPs	Y	3	counts
no_PlaceSvcs	Y	3	counts
no_Specialities	Y	3	counts
no_PrimaryConditionGroups	Y	3	counts
no_ProcedureGroups	Y	3	counts
LOS_max	Y	2	see SQL
LOS_min	Y	2	see SQL
LOS_ave	Y	2	see SQL
LOS_stdev	Y	2	see SQL
LOS_TOT_UNKNOWN	Y	2	see SQL
LOS_TOT_SUPRESSED	Y	2	see SQL
LOS_TOT_KNOWN	Y	2	see SQL
dsfs_max	Y	2	see SQL
dsfs_min	Y	2	see SQL
dsfs_range	Y	2	see SQL
dsfs_ave	Y	2	see SQL
dsfs_stdev	Y	2	see SQL
CharlsonIndexI_max	Y	2	see SQL
CharlsonIndexI_min	Y	2	see SQL
CharlsonIndexI_ave	Y	2	see SQL
CharlsonIndexI_range	Y	2	see SQL
CharlsonIndexI_stdev	Y	2	see SQL
pcg1–pcg46	Y	2	Counts- 46 fields, one for each pcg
sp1–sp13	Y	2	Counts- 13 fields, one for each sp
pg1–pg18	Y	2	Counts- 18 fields, one for each pg
ps1-ps9	Y	2	Counts- 9 fields, one for each ps
V	N	4	PG * SPECIALTY counts(21 variables/pairs)
W	N	4	PCG * SPECIALTY counts (63 variables/pairs)

X	N	4	PCG* PG counts (28 variables/pairs)
pid1–pid46	N	5	Counts for 46 most predictive provider IDs
vid1–vid42	N	5	Counts for 42 most predictive vendor IDs
pcp_prob	N	6	as described in documentation
vendor_prob	N	6	as described in documentation
providerid_prob	N	6	as described in documentation
labCount_max	Y	7	
labCount_min	Y	7	
labCount_ave	Y	7	
labcount_months	Y	7	
labCount_velocity	N	7	as described in documentation
labCount_range	N	7	
labNull	Y	7	
drugCount_max	Y	7	
drugCount_min	Y	7	
drugCount_ave	Y	7	
drugcount_months	Y	7	
drugCount_velocity	N	7	as described in documentation
drugCount_range	N	7	
drugNull	Y	7	
MaxAdmissionRiskL70	N	8	Max of AdmissionRiskL70 over all claims
AveAdmissionRiskL70	N	8	Ave of AdmissionRiskL70 over all claims
MaxAdmissionRisG70	N	8	Max of AdmissionRiskG70 over all claims
AveAdmissionRiskG70	n	8	Ave of AdmissionRiskG70 over all claims

Data Set 2 – two years combined

Variable	Subset	Description
ClaimCount	1	Total number of claims in Yr1 and Yr2
ClaimCount2	1	Total number of claims in Yr2
rxCount	1	Total number of rx claims in Yr1 and Yr2
rxCount2	1	Total number of rx claims in Yr2
lbClaimCount	1	Total number of records for that member in the lab table
lbClaimCount2	1	Total number of Yr2 records for that member in the lab table
lbCount	1	Sum of LabCount for all records for that member in the lab table
lbCount2	1	Sum of LabCount for all Yr2 records for that member in the lab table
LOS_SUM	1	Sum of LOS across all claims
LOS_MAX	1	Max value of LOS across all claims
LOS_SUM2	1	Sum of LOS across Yr2 claims
LOS_ MAX2	1	Max value of LOS across Yr2 claims
CharlesonIndex_Num	1	=IF(CharlsonIndex = “0”,0,IF(CharlsonIndex = “1-2”,1,IF(CharlsonIndex = “3-4”,3,5)))
CharlesonIndex_MAX	1	Max value of CharlesonIndex_Num across all claims
CharlesonIndex_MAX2	1	Max value of CharlesonIndex_Num across Yr2 claims
DSFS_max	1	Max value of dsfs_months across all claims
DSFS_CountG<x>	1	Count of claims where dsfs_months exceeds "x"
rxDSFS_max	1	Max value of dsfs_months across all rx claims
rxDSFS_CountG<x>	1	Count of rx claims where dsfs_months eaxceeds "x"
lbDSFS_max	2	Max value of dsfs_months across all labs
lbDSFS_CountG<x>	2	Count of labs where dsfs_months eaxceeds "x"
Count_<ProcX>	1	Count of claims where procedure group = "ProcX"
Count_<SpecialtyX>	1	Count of claims where specialty = "SpecialtyX"
Count_<PlaceX>	1	Count of claims where placesvc = "PlaceX"
Ratio_ER	1	Count_emergency / Claimcount
Count_<PCGX>	1	Count of claims where PrimaryConditionGroup = "PCGX"
SupLOSCount	1	Sum of SupLOS across all claims
AgeApprox: based on the following lookup table derived from AgeAtFirstClaim		
AgeAtFirstClaim		AgeApprox
<null>	1	80
0-9	1	5
10-19	1	15
20-29	1	25
30-39	1	35
40-49	1	45
50-59	1	55
60-69	1	65
70-79	1	75
80+	1	85
Male	1	=IF(SEX = "M",1,0)
Female	1	=IF(SEX = "F",1,0)
NoGender	1	=IF(SEX = "",1,0)
SupLOSCount2	1	Sum of SupLOS across Yr2 claims
SumAdmissionRiskL70	3	Sum of AdmissionRiskL70 over all claims

SumAdmissionRiskL70_2	3	Sum of AdmissionRiskL70 over Yr2 claims
SumAdmissionRiskG70	3	Sum of AdmissionRiskG70 over all claims
SumAdmissionRiskG70_2	3	Sum of AdmissionRiskG70 over Yr2 claims
MaxAdmissionRiskL70	3	Max of AdmissionRiskL70 over all claims
MaxAdmissionRiskL70_2	3	Max of AdmissionRiskL70 over Yr2 claims
MaxAdmissionRiskG70	3	Max of AdmissionRiskG70 over all claims
MaxAdmissionRiskG70_2	3	Max of AdmissionRiskG70 over Yr2 claims
ProviderCount	1	Number of distinct providers over all claims
VendorCount	1	Number of distinct vendors over all claims
PCPCount	1	Number of distinct PCPs over all claims
SpecialtyCount	1	Number of distinct specialties over all claims
PlaceSvcCount	1	Number of distinct PlaceSvc over all claims
PCGCount	1	Number of distinct PCGs over all claims
ProcCount	1	Number of claims

Admission Risk Values

PCG	**AdmissionRiskL70 (age < 70)**	**AdmissionRiskG70 (age>70)**
AMI	5	5
APPCHOL	1	1
ARTHSPIN	1	2
CANCRA	3	4
CANCRB	3	4
CANCRM	4	3
CATAST	3	4
CHF	5	5
COPD	5	5
FLaELEC	1	1
FXDISLC	1	1
GIBLEED	4	3
GIOBSENT	3	2
GYNEC1	1	1
GYNECA	4	3
HEART2	4	3
HEART4	1	2
HEMTOL	3	2
HIPFX	1	4
INFEC4	1	1
LIVERDZ	3	3
METAB1	3	4
METAB3	1	1
MISCHRT	2	3
MISCL1	1	1
MISCL5	1	1
MSC2a3	1	1
NEUMENT	2	1
ODaBNCA	1	1
PERINTL	1	1
PERVALV	4	5
PNCRDZ	2	3
PNEUM	1	2
PRGNCY	1	1
RENAL1	2	3
RENAL2	4	3
RENAL3	2	2
RESPR4	1	1
ROAMI	3	4
SEIZURE	2	2
SEPSIS	2	5
SKNAUT	2	1
STROKE	5	5
TRAUMA	1	2
UTI	1	2

Appendix B – the models

Models in the final Ensemble

For the Data Set and Data Subsets refer to the two data tables in Appendix A

Model	Algorithm	Data Set	Data Subsets	Leaderboard Score	Ensemble Weight	Model Parameters	CV Folds	CV Repeats
1	Median	NA		0.4590	0.060	See next table		
2	BT	1	2,3,4,5,6,7,8	0.4625	-0.029	K = 2000 N=10	12	1
3	BT	1	1,2,3,4,5,6,7	0.4633	-0.494	K = 10000 N = 100	2	12
4	ENS	1	1,2,3,4,5,6,8	0.4609	0.394	K = 800 N = 10 | S=0.05 T=550 D=2 M=100	2	59
5	ENS	1	2,3,4,5,6	0.4610	0.371	K = 800 N = 100 | S=0.05 T=550 D=4 M=100	2	5
6	ENS	1	1,2,4,5,6	0.4612	-0.123	K = 400 N = 50 | S= 0.05 T=550 D= 3 M= 100	2	35
7	ENS	1	1,2,3,4,5,6,7,8	0.4608	0.078	K = 1000 N = 80 | S=0.05 T=550 D=4 M=100	2	21
8	GBM	1	1,2,3,4,5,6,7,8	0.4599	0.396	S=0.002 T=8000 F=0.9 D=7 M=100	12	1
9	GBM	2	1	0.4626	0.319	S=0.01 T = 2000 F = 0.9 D = 5 M = 50	12	1
10	GBM	1	2,3,4,5,6,7	0.4603	0.071	S=0.002 T=8000 F=0.9 D=7 M=100	12	1
11	GBM	2	1,2,3	0.4614	0.033	S = 0.005 T = 4500 F = 0.95 D = 5 M = 30	12	1
12	GBM	1	2,3,4,5,6,7,8	0.4603	0.010	S = 0.002 T = 10000 F = 0.9 D = 6 M = 100	12	1
13	LM	1	2,3,4	0.4679	-0.231		2	100
14	LM	1	2,3,4,5,6	0.4673	-0.229		2	200
15	LM	1	2,3,4,5,6,7,8	0.4668	0.181		2	300
16	NN	1	1,2,3,4,5,6,7	0.4622	0.191	L= 0.007 E = 500 H = 4 M = 50	2	50
17	GBM	2	2	0.4902	0.098	S=0.01 T = 1000, F = 0.9 D = 2 M = 50	1	12
18	NN	1	2,3	0.4692	-0.097	L= 0.007 E = 500 H = 4 M = 2	2	1
19	ENS	1	2,3,4,5,6,7	0.4614	0.021	K = 400 N = 50 | S=0.05 T=550 D=3 M=100	2	28
20	NN	1	2	0.4828	-0.018	L= 0.007 E = 500 H = 1 M = 2	2	1

Models in the Median Model

Model	Algorithm	Data Set	Data Subsets	Leaderboard Score	Model Parameters	Cv Folds	Cv Repeats
21	GBM	2	1,2,3	0.4619	S = 0.005 T = 3000 F = 0.95 D = 5 M = 50	12	1
11	GBM	2	1,2,3	0.4614	S = 0.005 T = 4500 F = 0.95 D = 5 M = 30	12	1
22	GBM	2	1	0.4625	S = 0.005 T = 3000F = 0.95 D = 5 M = 30	12	1
23	GBM	1	2,3,4,5,6,7,8	0.4603	S = 0.005 T = 5000 F = 1.0 D = 4 M = 100	12	1
24	GBM	1	2,3,4,5,6,7	0.4605	S = 0.005 T = 5000 F = 1.0 D = 4 M = 100	12	1
25	GBM	2	1	0.4621	S = 0.002 T = 5000 F = 0.9 D = 7 M = 50	12	1
6	ENS	1	1,2,4,5,6	0.4612	K = 400 N = 50 | S= 0.05 T=550 D= 3 M= 100	2	35
4	ENS	1	1,2,3,4,5,6,8	0.4609	K = 800 N = 10 | S=0.05 T=550 D=2 M=100	2	59
7	ENS	1	1,2,3,4,5,6,7,8	0.4608	K = 1000 N = 80 | S=0.05 T=550 D=4 M=100	2	21

GBM notation
S = shrinkage parameter
T = iterations
F = the fraction of the training set observations randomly selected to propose the next tree in the expansion
D = interaction depth
M = minimum number of observations in the trees terminal nodes

All used Gaussian distribution and default R parameters if not specified

Bagged Tree (BT) Notation
K = number of trees
N = minimum number of observations in the trees terminal nodes
Default R parameters if not specified

Neural Network (NN) Notation
L = learning rate
E = Number of training epochs
H = number of hidden neurons
M = number of models in ensemble

The hidden neurons had the hyperbolic tangent activation functions and the output neuron was linear.
Each model was built on a random 50% of the data and trained for the specified number of epochs. The weights from the epoch that gave the best RMSE on the held out 50% were used as the final model.
100 models were built with random weight initiation and the predictions averaged

Linear Models (LM) Models
The linear models were built with an offset

Ensemble Models (ENS)
The ensemble models were built using 3 algorithms simultaneously, GBMs, Bagged Trees and Linear Models. After each repeat, linear regression without an offset was used to weight the 3 models, with the weights then forced to sum to 1 by rescaling. Typical weightings were 0.2 for the linear models and 0.4 for both GBMs and Bagged Trees. The process was generally repeated until the cross validation error on the averaged predictions from all repeats was seen to converge, which was observed to be about 10-15 repeats. Otherwise they were just left to run continuously if time permitted.

Appendix C – Data Preparation SQL

SQL script to create a modelling data set. If this script is executed then the result will be a data set ready for use by the modelling code in Appendix C. Note that not all available data is utilised by this script.

```
/*****************************************************************
* SQL Code to create an example data set for the HHP
*
* Edit the path in the 'bulk insert' commands to locate
* the source data
* The end result is a table called 'modelling_set' which can
* THEN be used to build predictive models
*
* created in SQL server express - available for free from
* http://www.microsoft.com/sqlserver/en/us/editions/express.aspx
*****************************************************************/


/**************************
create a new database
**************************/
CREATE DATABASE HHP_comp
GO
USE HHP_comp


/**************************
load in the raw data
**************************/

--claims
CREATE TABLE Claims
(
        MemberID        VARCHAR(8)  --integers starting with 0, could be text!
,       ProviderID      VARCHAR(7)  --integers starting with 0, could be text!
,       Vendor  VARCHAR(6)  --integers starting with 0, could be text!
,       PCP     VARCHAR(5)  --integers starting with 0, could be text!
,       Year    VARCHAR(2)
,       Specialty       VARCHAR(25)
,       PlaceSvc        VARCHAR(19)
,       PayDelay        VARCHAR(4)
,       LengthOfStay    VARCHAR(10)
,       DSFS    VARCHAR(12)
,       PrimaryConditionGroup  VARCHAR(8)
,       CharlsonIndex   VARCHAR(3)
,       ProcedureGroup VARCHAR(4)
,       SupLOS  TINYINT
)

BULK INSERT Claims
FROM 'E:\comps\hhp\raw data\HHP_release2\Claims.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


--members
CREATE TABLE Members
(
        MemberID_M VARCHAR(8)  --integers starting with 0, could be text!
,       AgeAtFirstClaim        VARCHAR(5)
,       Sex     VARCHAR(1)
)

BULK INSERT Members
FROM 'E:\comps\hhp\raw data\HHP_release2\Members.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


-- drug count
CREATE TABLE DrugCount
(
        MemberID        INT
,       Year    VARCHAR(2)
,       DSFS    VARCHAR(12)
,       DrugCount       VARCHAR(2)
)

BULK INSERT DrugCount
FROM 'E:\comps\hhp\raw data\HHP_release3\DrugCount.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


-- Lab Count
CREATE TABLE LabCount
(
        MemberID        INT
,       Year    VARCHAR(2)
,       DSFS    VARCHAR(12)
,       LabCount        VARCHAR(3)
)


BULK INSERT LabCount
FROM 'E:\comps\hhp\raw data\HHP_release3\LabCount.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


--DaysInHospital_Y2
CREATE TABLE DaysInHospital_Y2
(
        MemberID        INT
,       ClaimsTruncated         TINYINT
,       DaysInHospital TINYINT
)


BULK INSERT DaysInHospital_Y2
FROM 'E:\comps\hhp\raw data\HHP_release2\DaysInHospital_Y2.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


-- DaysInHospital_Y3
CREATE TABLE DaysInHospital_Y3
(
        MemberID        INT
,       ClaimsTruncated         TINYINT
,       DaysInHospital TINYINT
)
```

```
BULK INSERT DaysInHospital_Y3
FROM 'E:\comps\hhp\raw data\HHP_release2\DaysInHospital_Y3.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)


-- Target
CREATE TABLE Target
(
        MemberID        INT
,       ClaimsTruncated         TINYINT
,       DaysInHospital TINYINT
)


BULK INSERT Target
FROM 'E:\comps\hhp\raw data\HHP_release2\Target.csv'
WITH
(
MAXERRORS = 0,
FIRSTROW = 2,
FIELDTERMINATOR = ',',
ROWTERMINATOR = '\n'
)



/*************************
adjust the claims data to
convert text to integers
**************************/


-- PayDelay
ALTER TABLE Claims
ADD PayDelayI integer
GO

UPDATE Claims
SET PayDelayI = CASE WHEN PayDelay = '162+' THEN 162 ELSE CAST(PayDelay AS integer) END


--dsfs
ALTER TABLE Claims
ADD dsfsI integer
GO

UPDATE Claims
SET dsfsI =
CASE
        WHEN dsfs = '0- 1 month' THEN 1
        WHEN dsfs = '1- 2 months' THEN 2
        WHEN dsfs = '2- 3 months' THEN 3
        WHEN dsfs = '3- 4 months' THEN 4
        WHEN dsfs = '4- 5 months' THEN 5
        WHEN dsfs = '5- 6 months' THEN 6
        WHEN dsfs = '6- 7 months' THEN 7
        WHEN dsfs = '7- 8 months' THEN 8
        WHEN dsfs = '8- 9 months' THEN 9
        WHEN dsfs = '9-10 months' THEN 10
        WHEN dsfs = '10-11 months' THEN 11
        WHEN dsfs = '11-12 months' THEN 12
        WHEN dsfs IS NULL THEN NULL
END


-- CharlsonIndex
ALTER TABLE Claims
ADD CharlsonIndexI INTEGER
GO

UPDATE Claims
```

```
SET CharlsonIndexI =
CASE
        WHEN CharlsonIndex = '0' THEN 0
        WHEN CharlsonIndex = '1-2' THEN 2
        WHEN CharlsonIndex = '3-4' THEN 4
        WHEN CharlsonIndex = '5+' THEN 6
END


-- LengthOfStay
ALTER TABLE Claims
ADD LengthOfStayI INTEGER
GO

UPDATE Claims
SET LengthOfStayI =
CASE
        WHEN LengthOfStay = '1 day'  THEN 1
        WHEN LengthOfStay = '2 days' THEN 2
        WHEN LengthOfStay = '3 days' THEN 3
        WHEN LengthOfStay = '4 days' THEN 4
        WHEN LengthOfStay = '5 days' THEN 5
        WHEN LengthOfStay = '6 days' THEN 6
        WHEN LengthOfStay = '1- 2 weeks' THEN 11
        WHEN LengthOfStay = '2- 4 weeks' THEN 21
        WHEN LengthOfStay = '4- 8 weeks' THEN 42
        WHEN LengthOfStay = '26+ weeks' THEN 180
        WHEN LengthOfStay IS NULL THEN null
END


/***************************
create a summary table
at the member/year level
****************************/
SELECT
        year
        ,Memberid

        ,COUNT(*) AS no_Claims
        ,COUNT(DISTINCT ProviderID) AS no_Providers
        ,COUNT(DISTINCT Vendor) AS no_Vendors
        ,COUNT(DISTINCT PCP) AS no_PCPs
        ,COUNT(DISTINCT PlaceSvc) AS no_PlaceSvcs
        ,COUNT(DISTINCT Specialty) AS no_Specialities
        ,COUNT(DISTINCT PrimaryConditionGroup) AS no_PrimaryConditionGroups
        ,COUNT(DISTINCT ProcedureGroup) AS no_ProcedureGroups

        ,MAX(PayDelayI) AS PayDelay_max
        ,MIN(PayDelayI) AS PayDelay_min
        ,AVG(PayDelayI) AS PayDelay_ave
        ,(CASE WHEN COUNT(*) = 1 THEN 0 ELSE STDEV(PayDelayI) END) AS PayDelay_stdev

        ,MAX(LengthOfStayI) AS LOS_max
        ,MIN(LengthOfStayI) AS LOS_min
        ,AVG(LengthOfStayI) AS LOS_ave
        ,(CASE WHEN COUNT(*) = 1 THEN 0 ELSE STDEV(LengthOfStayI) END) AS LOS_stdev

        ,SUM(CASE WHEN LENGTHOFSTAY IS NULL AND SUPLOS = 0 THEN 1 ELSE 0 END) AS
LOS_TOT_UNKNOWN
        ,SUM(CASE WHEN LENGTHOFSTAY IS NULL AND SUPLOS = 1 THEN 1 ELSE 0  END) AS
LOS_TOT_SUPRESSED
        ,SUM(CASE WHEN LENGTHOFSTAY IS NOT NULL THEN 1 ELSE 0 END) AS LOS_TOT_KNOWN

        ,MAX(dsfsI) AS dsfs_max
        ,MIN(dsfsI) AS dsfs_min
        ,MAX(dsfsI) - MIN(dsfsI) AS dsfs_range
        ,AVG(dsfsI) AS dsfs_ave
        ,(CASE WHEN COUNT(*) = 1 THEN 0 ELSE STDEV(dsfsI) END) AS dsfs_stdev

        ,MAX(CharlsonIndexI) AS CharlsonIndexI_max
        ,MIN(CharlsonIndexI) AS CharlsonIndexI_min
        ,AVG(CharlsonIndexI) AS CharlsonIndexI_ave
        ,MAX(CharlsonIndexI) - MIN(CharlsonIndexI) AS CharlsonIndexI_range
        ,(CASE WHEN COUNT(*) = 1 THEN 0 ELSE STDEV(CharlsonIndexI) END) AS
CharlsonIndexI_stdev
```

```
,SUM(CASE WHEN PrimaryConditionGroup = 'MSC2a3' THEN 1 ELSE 0 END) AS pcg1
,SUM(CASE WHEN PrimaryConditionGroup = 'METAB3' THEN 1 ELSE 0 END) AS pcg2
,SUM(CASE WHEN PrimaryConditionGroup = 'ARTHSPIN' THEN 1 ELSE 0 END) AS pcg3
,SUM(CASE WHEN PrimaryConditionGroup = 'NEUMENT' THEN 1 ELSE 0 END) AS pcg4
,SUM(CASE WHEN PrimaryConditionGroup = 'RESPR4' THEN 1 ELSE 0 END) AS pcg5
,SUM(CASE WHEN PrimaryConditionGroup = 'MISCHRT' THEN 1 ELSE 0 END) AS pcg6
,SUM(CASE WHEN PrimaryConditionGroup = 'SKNAUT' THEN 1 ELSE 0 END) AS pcg7
,SUM(CASE WHEN PrimaryConditionGroup = 'GIBLEED' THEN 1 ELSE 0 END) AS pcg8
,SUM(CASE WHEN PrimaryConditionGroup = 'INFEC4' THEN 1 ELSE 0 END) AS pcg9
,SUM(CASE WHEN PrimaryConditionGroup = 'TRAUMA' THEN 1 ELSE 0 END) AS pcg10
,SUM(CASE WHEN PrimaryConditionGroup = 'HEART2' THEN 1 ELSE 0 END) AS pcg11
,SUM(CASE WHEN PrimaryConditionGroup = 'RENAL3' THEN 1 ELSE 0 END) AS pcg12
,SUM(CASE WHEN PrimaryConditionGroup = 'ROAMI' THEN 1 ELSE 0 END) AS pcg13
,SUM(CASE WHEN PrimaryConditionGroup = 'MISCL5' THEN 1 ELSE 0 END) AS pcg14
,SUM(CASE WHEN PrimaryConditionGroup = 'ODaBNCA' THEN 1 ELSE 0 END) AS pcg15
,SUM(CASE WHEN PrimaryConditionGroup = 'UTI' THEN 1 ELSE 0 END) AS pcg16
,SUM(CASE WHEN PrimaryConditionGroup = 'COPD' THEN 1 ELSE 0 END) AS pcg17
,SUM(CASE WHEN PrimaryConditionGroup = 'GYNEC1' THEN 1 ELSE 0 END) AS pcg18
,SUM(CASE WHEN PrimaryConditionGroup = 'CANCRB' THEN 1 ELSE 0 END) AS pcg19
,SUM(CASE WHEN PrimaryConditionGroup = 'FXDISLC' THEN 1 ELSE 0 END) AS pcg20
,SUM(CASE WHEN PrimaryConditionGroup = 'AMI' THEN 1 ELSE 0 END) AS pcg21
,SUM(CASE WHEN PrimaryConditionGroup = 'PRGNCY' THEN 1 ELSE 0 END) AS pcg22
,SUM(CASE WHEN PrimaryConditionGroup = 'HEMTOL' THEN 1 ELSE 0 END) AS pcg23
,SUM(CASE WHEN PrimaryConditionGroup = 'HEART4' THEN 1 ELSE 0 END) AS pcg24
,SUM(CASE WHEN PrimaryConditionGroup = 'SEIZURE' THEN 1 ELSE 0 END) AS pcg25
,SUM(CASE WHEN PrimaryConditionGroup = 'APPCHOL' THEN 1 ELSE 0 END) AS pcg26
,SUM(CASE WHEN PrimaryConditionGroup = 'CHF' THEN 1 ELSE 0 END) AS pcg27
,SUM(CASE WHEN PrimaryConditionGroup = 'GYNECA' THEN 1 ELSE 0 END) AS pcg28
,SUM(CASE WHEN PrimaryConditionGroup IS NULL  THEN 1 ELSE 0 END) AS pcg29
,SUM(CASE WHEN PrimaryConditionGroup = 'PNEUM' THEN 1 ELSE 0 END) AS pcg30
,SUM(CASE WHEN PrimaryConditionGroup = 'RENAL2' THEN 1 ELSE 0 END) AS pcg31
,SUM(CASE WHEN PrimaryConditionGroup = 'GIOBSENT' THEN 1 ELSE 0 END) AS pcg32
,SUM(CASE WHEN PrimaryConditionGroup = 'STROKE' THEN 1 ELSE 0 END) AS pcg33
,SUM(CASE WHEN PrimaryConditionGroup = 'CANCRA' THEN 1 ELSE 0 END) AS pcg34
,SUM(CASE WHEN PrimaryConditionGroup = 'FLaELEC' THEN 1 ELSE 0 END) AS pcg35
,SUM(CASE WHEN PrimaryConditionGroup = 'MISCL1' THEN 1 ELSE 0 END) AS pcg36
,SUM(CASE WHEN PrimaryConditionGroup = 'HIPFX' THEN 1 ELSE 0 END) AS pcg37
,SUM(CASE WHEN PrimaryConditionGroup = 'METAB1' THEN 1 ELSE 0 END) AS pcg38
,SUM(CASE WHEN PrimaryConditionGroup = 'PERVALV' THEN 1 ELSE 0 END) AS pcg39
,SUM(CASE WHEN PrimaryConditionGroup = 'LIVERDZ' THEN 1 ELSE 0 END) AS pcg40
,SUM(CASE WHEN PrimaryConditionGroup = 'CATAST' THEN 1 ELSE 0 END) AS pcg41
,SUM(CASE WHEN PrimaryConditionGroup = 'CANCRM' THEN 1 ELSE 0 END) AS pcg42
,SUM(CASE WHEN PrimaryConditionGroup = 'PERINTL' THEN 1 ELSE 0 END) AS pcg43
,SUM(CASE WHEN PrimaryConditionGroup = 'PNCRDZ' THEN 1 ELSE 0 END) AS pcg44
,SUM(CASE WHEN PrimaryConditionGroup = 'RENAL1' THEN 1 ELSE 0 END) AS pcg45
,SUM(CASE WHEN PrimaryConditionGroup = 'SEPSIS' THEN 1 ELSE 0 END) AS pcg46

,SUM(CASE WHEN Specialty = 'Internal' THEN 1 ELSE 0 END) AS sp1
,SUM(CASE WHEN Specialty = 'Laboratory' THEN 1 ELSE 0 END) AS sp2
,SUM(CASE WHEN Specialty = 'General Practice' THEN 1 ELSE 0 END) AS sp3
,SUM(CASE WHEN Specialty = 'Surgery' THEN 1 ELSE 0 END) AS sp4
,SUM(CASE WHEN Specialty = 'Diagnostic Imaging' THEN 1 ELSE 0 END) AS sp5
,SUM(CASE WHEN Specialty = 'Emergency' THEN 1 ELSE 0 END) AS sp6
,SUM(CASE WHEN Specialty = 'Other' THEN 1 ELSE 0 END) AS sp7
,SUM(CASE WHEN Specialty = 'Pediatrics' THEN 1 ELSE 0 END) AS sp8
,SUM(CASE WHEN Specialty = 'Rehabilitation' THEN 1 ELSE 0 END) AS sp9
,SUM(CASE WHEN Specialty = 'Obstetrics and Gynecology' THEN 1 ELSE 0 END) AS sp10
,SUM(CASE WHEN Specialty = 'Anesthesiology' THEN 1 ELSE 0 END) AS sp11
,SUM(CASE WHEN Specialty = 'Pathology' THEN 1 ELSE 0 END) AS sp12
,SUM(CASE WHEN Specialty IS NULL THEN 1 ELSE 0 END) AS sp13

,SUM(CASE WHEN ProcedureGroup = 'EM' THEN 1 ELSE 0 END  )  AS  pg1
,SUM(CASE WHEN ProcedureGroup = 'PL' THEN 1 ELSE 0 END  )  AS  pg2
,SUM(CASE WHEN ProcedureGroup = 'MED' THEN 1 ELSE 0 END  )  AS  pg3
,SUM(CASE WHEN ProcedureGroup = 'SCS' THEN 1 ELSE 0 END  )  AS  pg4
,SUM(CASE WHEN ProcedureGroup = 'RAD' THEN 1 ELSE 0 END  )  AS  pg5
,SUM(CASE WHEN ProcedureGroup = 'SDS' THEN 1 ELSE 0 END  )  AS  pg6
,SUM(CASE WHEN ProcedureGroup = 'SIS' THEN 1 ELSE 0 END  )  AS  pg7
,SUM(CASE WHEN ProcedureGroup = 'SMS' THEN 1 ELSE 0 END  )  AS  pg8
,SUM(CASE WHEN ProcedureGroup = 'ANES' THEN 1 ELSE 0 END  )  AS  pg9
,SUM(CASE WHEN ProcedureGroup = 'SGS' THEN 1 ELSE 0 END  )  AS  pg10
,SUM(CASE WHEN ProcedureGroup = 'SEOA' THEN 1 ELSE 0 END  )  AS  pg11
,SUM(CASE WHEN ProcedureGroup = 'SRS' THEN 1 ELSE 0 END  )  AS  pg12
,SUM(CASE WHEN ProcedureGroup = 'SNS' THEN 1 ELSE 0 END  )  AS  pg13
,SUM(CASE WHEN ProcedureGroup = 'SAS' THEN 1 ELSE 0 END  )  AS  pg14
,SUM(CASE WHEN ProcedureGroup = 'SUS' THEN 1 ELSE 0 END  )  AS  pg15
```

```
        ,SUM(CASE WHEN ProcedureGroup IS NULL  THEN 1 ELSE 0 END  )  AS  pg16
        ,SUM(CASE WHEN ProcedureGroup = 'SMCD' THEN 1 ELSE 0 END  )  AS  pg17
        ,SUM(CASE WHEN ProcedureGroup = 'SO' THEN 1 ELSE 0 END  )  AS  pg18

        ,SUM(CASE WHEN PlaceSvc = 'Office' THEN 1 ELSE 0 END) AS ps1
        ,SUM(CASE WHEN PlaceSvc = 'Independent Lab' THEN 1 ELSE 0 END) AS ps2
        ,SUM(CASE WHEN PlaceSvc = 'Urgent Care' THEN 1 ELSE 0 END) AS ps3
        ,SUM(CASE WHEN PlaceSvc = 'Outpatient Hospital' THEN 1 ELSE 0 END) AS ps4
        ,SUM(CASE WHEN PlaceSvc = 'Inpatient Hospital' THEN 1 ELSE 0 END) AS ps5
        ,SUM(CASE WHEN PlaceSvc = 'Ambulance' THEN 1 ELSE 0 END) AS ps6
        ,SUM(CASE WHEN PlaceSvc = 'Other' THEN 1 ELSE 0 END) AS ps7
        ,SUM(CASE WHEN PlaceSvc = 'Home' THEN 1 ELSE 0 END) AS ps8
        ,SUM(CASE WHEN PlaceSvc IS NULL THEN 1 ELSE 0 END) AS ps9

INTO claims_per_member
FROM Claims
GROUP BY  year,Memberid

-- remove some nulls
        UPDATE claims_per_member
        SET LOS_max = 0 WHERE LOS_max IS NULL

        UPDATE claims_per_member
        SET LOS_min = 0 WHERE LOS_min IS NULL

        UPDATE claims_per_member
        SET LOS_ave = 0 WHERE LOS_ave IS NULL

        UPDATE claims_per_member
        SET LOS_stdev = -1 WHERE LOS_stdev IS NULL

        UPDATE claims_per_member
        SET dsfs_max = 0 WHERE dsfs_max IS NULL

        UPDATE claims_per_member
        SET dsfs_min = 0 WHERE dsfs_min IS NULL

        UPDATE claims_per_member
        SET dsfs_ave = 0 WHERE dsfs_ave IS NULL

        UPDATE claims_per_member
        SET dsfs_stdev = -1 WHERE dsfs_stdev IS NULL

        UPDATE claims_per_member
        SET dsfs_range = -1 WHERE dsfs_range IS NULL

        UPDATE claims_per_member
        SET CharlsonIndexI_range = -1 WHERE CharlsonIndexI_range IS NULL


/************************************
Members
************************************/

-- create binary flags for age
ALTER TABLE Members ADD age_05 INT
ALTER TABLE Members ADD age_15 INT
ALTER TABLE Members ADD age_25 INT
ALTER TABLE Members ADD age_35 INT
ALTER TABLE Members ADD age_45 INT
ALTER TABLE Members ADD age_55 INT
ALTER TABLE Members ADD age_65 INT
ALTER TABLE Members ADD age_75 INT
ALTER TABLE Members ADD age_85 INT
ALTER TABLE Members ADD age_MISS INT

GO

UPDATE Members SET age_05 = CASE WHEN ageATfirstclaim = '0-9' THEN 1 ELSE 0 END
UPDATE Members SET age_15 = CASE WHEN ageATfirstclaim = '10-19' THEN 1 ELSE 0 END
UPDATE Members SET age_25 = CASE WHEN ageATfirstclaim = '20-29' THEN 1 ELSE 0 END
UPDATE Members SET age_35 = CASE WHEN ageATfirstclaim = '30-39' THEN 1 ELSE 0 END
UPDATE Members SET age_45 = CASE WHEN ageATfirstclaim = '40-49' THEN 1 ELSE 0 END
UPDATE Members SET age_55 = CASE WHEN ageATfirstclaim = '50-59' THEN 1 ELSE 0 END
UPDATE Members SET age_65 = CASE WHEN ageATfirstclaim = '60-69' THEN 1 ELSE 0 END
UPDATE Members SET age_75 = CASE WHEN ageATfirstclaim = '70-79' THEN 1 ELSE 0 END
UPDATE Members SET age_85 = CASE WHEN ageATfirstclaim = '80+' THEN 1 ELSE 0 END
UPDATE Members SET age_MISS = CASE WHEN ageATfirstclaim IS NULL THEN 1 ELSE 0 END


--create binary flags for sex
ALTER TABLE Members
ADD sexMALE INT
GO

UPDATE Members
SET SexMALE =
CASE
        WHEN Sex = 'M' THEN 1 ELSE 0
END


ALTER TABLE Members
ADD sexFEMALE INT
GO

UPDATE Members
SET SexFEMALE =
CASE
        WHEN Sex = 'F' THEN 1 ELSE 0
END


ALTER TABLE Members
ADD sexMISS INT
GO

UPDATE Members
SET SexMISS =
CASE
        WHEN Sex IS NULL THEN 1 ELSE 0
END



/******************
DRUG COUNTS
******************/

-- convert to integers
ALTER TABLE drugcount ADD DrugCountI INT
GO
UPDATE DRUGCOUNT
SET DrugCountI =
CASE WHEN DrugCount = '7+' THEN 7 ELSE DrugCount END


SELECT
memberID AS memberID_dc
,Year AS YEAR_dc
,MAX(drugcountI) AS drugCount_max
,MIN(drugcountI) AS drugCount_min
,AVG(drugcountI * 1.0) AS drugCount_ave
,COUNT(*) AS drugcount_months
INTO DRUGCOUNT_SUMMARY
FROM
drugcount
GROUP BY
memberID
,Year



/******************
LAB COUNTS
******************/

-- convert to integers
ALTER TABLE LabCount ADD LabCountI INT
GO
UPDATE LabCount
SET LabCountI =
CASE WHEN LabCount = '10+' THEN 10 ELSE LabCount END
```

```
SELECT
memberID AS memberID_lc
,Year AS YEAR_lc
,MAX(labcountI) AS labCount_max
,MIN(labcountI) AS labCount_min
,AVG(labcountI * 1.0) AS labCount_ave
,COUNT(*) AS labcount_months
INTO LABCOUNT_SUMMARY
FROM
labcount
GROUP BY
memberID
,Year


/*********************************
Targets
*********************************/

SELECT *
INTO DIH
FROM
(
SELECT
MemberID AS MemberID_t
,'Y1' AS YEAR_t
,ClaimsTruncated
,DaysInHospital
,1 AS trainset
FROM DaysInHospital_Y2

UNION ALL

SELECT
MemberID AS MemberID_t
,'Y2' AS YEAR_t
,ClaimsTruncated
,DaysInHospital
,1 AS trainset
FROM DaysInHospital_Y3

UNION ALL

SELECT
MemberID AS MemberID_t
,'Y3' AS YEAR_t
,ClaimsTruncated
,null AS DaysInHospital
,0 AS trainset
FROM Target
) a



/******************************
Now merge them all together to
create the modeling data set
******************************/
SELECT a.*,b.*
INTO #temp1
FROM
DIH a
LEFT OUTER JOIN
members b
on a.MemberID_t = B.Memberid_M

ALTER TABLE #temp1 DROP COLUMN Memberid_M
ALTER TABLE #temp1 DROP COLUMN AgeAtFirstClaim
ALTER TABLE #temp1 DROP COLUMN Sex
GO



SELECT a.*,b.*
INTO #temp2
FROM
```

```
        #temp1 a
LEFT OUTER JOIN
        claims_per_member b
on a.MemberID_t = B.Memberid
AND a.YEAR_t = b.year

ALTER TABLE #temp2 DROP COLUMN Memberid
ALTER TABLE #temp2 DROP COLUMN year
GO


SELECT a.*,b.*
INTO #temp3
FROM
        #temp2 a
LEFT OUTER JOIN
        DRUGCOUNT_SUMMARY b
on a.MemberID_t = B.Memberid_dc
AND a.YEAR_t = b.YEAR_dc

ALTER TABLE #temp3 DROP COLUMN Memberid_dc
ALTER TABLE #temp3 DROP COLUMN YEAR_dc
GO



SELECT a.*,b.*
INTO #temp4
FROM
        #temp3 a
LEFT OUTER JOIN
        LABCOUNT_SUMMARY b
on a.MemberID_t = B.Memberid_lc
AND a.YEAR_t = b.YEAR_lc

ALTER TABLE #temp4 DROP COLUMN Memberid_lc
ALTER TABLE #temp4 DROP COLUMN YEAR_lc
GO



-- removel nulls for those who had
-- no lab or drug information
ALTER TABLE #temp4 ADD labNull INT
ALTER TABLE #temp4 ADD drugNull INT
GO

UPDATE #temp4 SET labNull = 0
UPDATE #temp4 SET labNull = 1 WHERE labCount_max IS NULL

UPDATE #temp4 SET drugNull = 0
UPDATE #temp4 SET drugNull = 1 WHERE drugCount_max IS NULL

UPDATE #temp4 SET labCount_max = 0 WHERE labCount_max IS NULL
UPDATE #temp4 SET labCount_min = 0 WHERE labCount_min IS NULL
UPDATE #temp4 SET labCount_ave = 0 WHERE labCount_ave IS NULL
UPDATE #temp4 SET labcount_months = 0 WHERE labcount_months IS NULL

UPDATE #temp4 SET drugCount_max = 0 WHERE drugCount_max IS NULL
UPDATE #temp4 SET drugCount_min = 0 WHERE drugCount_min IS NULL
UPDATE #temp4 SET drugCount_ave = 0 WHERE drugCount_ave IS NULL
UPDATE #temp4 SET drugcount_months = 0 WHERE drugcount_months IS NULL


SELECT *
INTO modelling_set
FROM #temp4
```

Appendix C – R code for GBM

R script to build a model. If this script is run then the result is a file ready to submit to the leaderboard.

```
########################################
# Example R code
# GBM model for HHP
# scores ~ 0.4635 on leaderboard
# which would be 52nd position of 499
# at the first milestone cut off date
#
# Requires the data having been prepared
# using the SQL supplied
########################################


########################################
#load the data
########################################
library(RODBC)

#set a connection to the database
conn <- odbcDriverConnect("driver=SQL Server;database=HHP_comp;server=servernamehere;")

#or this method involves setting up a DSN (Data Source Name) called HHP_compDSN
#conn <- odbcConnect("HHP_compDSN")

alldata <- sqlQuery(conn,"select * from modelling_set")


########################################
# arrange the data
########################################

#identify train and leaderboard data
trainrows <- which(alldata$trainset == 1)
scorerows <- which(alldata$trainset == 0)

#sanity check the size of each set
length(trainrows)
length(scorerows)

#display the column names
colnames(alldata)

#memberid is required as key for submission set
memberid <- alldata[scorerows,'MemberID_t']

#remove redundant fields
alldata$MemberID_t <- NULL
alldata$YEAR_t <- NULL
alldata$trainset <- NULL

#target - what we are predicting
theTarget <- 'DaysInHospital'

#put the target on the log scale
alldata[trainrows,theTarget] <- log1p(alldata[trainrows,theTarget])

#find the position of the target
```

```r
targindex <-  which(names(alldata)==theTarget)


##########################################
# build the model
##########################################

#GBM model settings, these can be varied
GBM_NTREES = 500
GBM_SHRINKAGE = 0.05
GBM_DEPTH = 4
GBM_MINOBS = 50

#build the GBM model
library(gbm)
GBM_model <- gbm.fit(
 x = alldata[trainrows,-targindex]
,y = alldata[trainrows,targindex]
,distribution = "gaussian"
,n.trees = GBM_NTREES
,shrinkage = GBM_SHRINKAGE
,interaction.depth = GBM_DEPTH
,n.minobsinnode = GBM_MINOBS
,verbose = TRUE)

#list variable importance
summary(GBM_model,GBM_NTREES)

#predict for the leaderboard data
prediction <- predict.gbm(object = GBM_model
,newdata = alldata[scorerows,-targindex]
,GBM_NTREES)

#put on correct scale and cap
prediction <- expm1(prediction)
prediction <- pmin(15,prediction)
prediction <- pmax(0,prediction)

#plot the submission distribution
hist(prediction, breaks=500)


##########################################
#write the submission to file
##########################################
submission <- cbind(memberid,prediction)
colnames(submission) <- c("MemberID","DaysInHospital")
fnname <- "C:\\GBM_demo.csv"
write.csv(submission, file=fnname, row.names = FALSE)


cat("\nFinished")
```