Classification Machine Learning Predicting Liver Disease

Dave Downing

5/6/2020

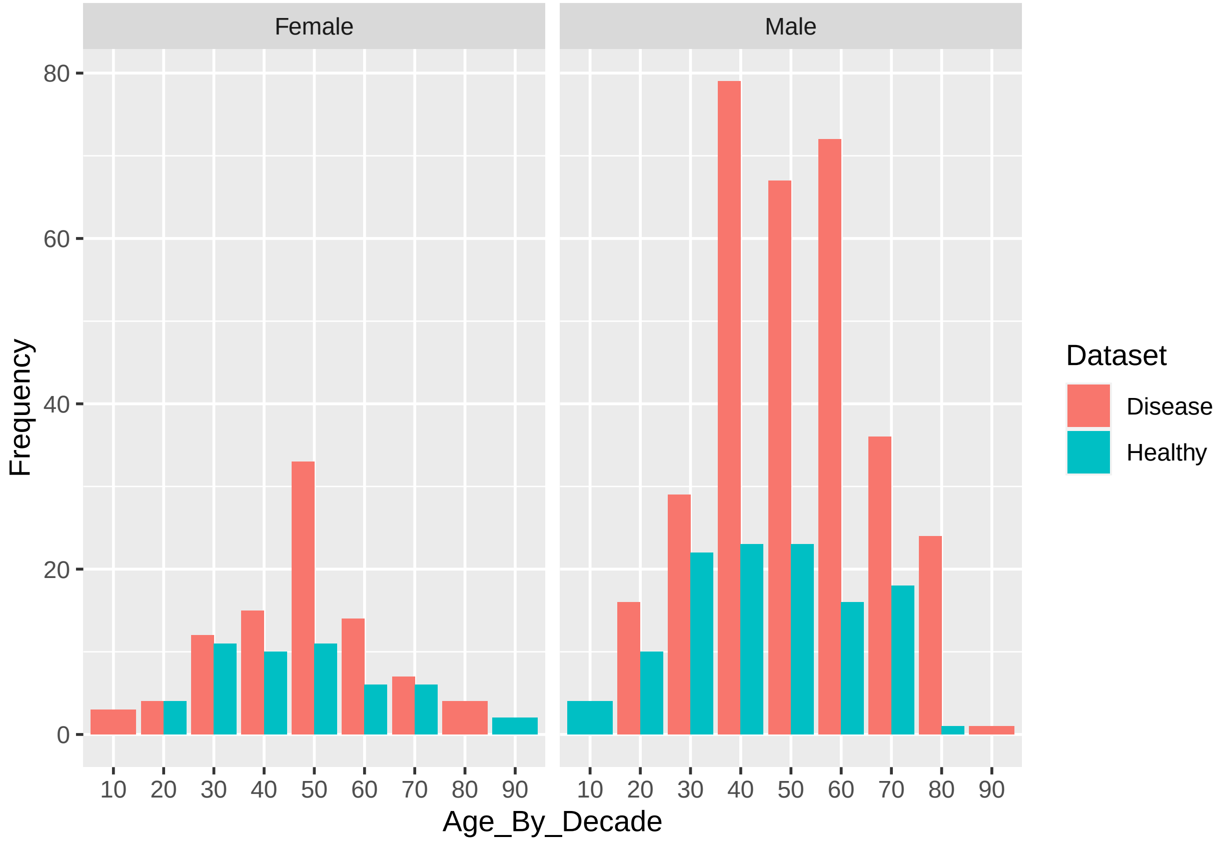
Contents

1 Introduction

The purpose of this project is to predict the presence of liver disease from a given set of data. The data set is referred to as the Indian Liver Patient data set made available from the UCI Machine Learning Repository at https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset. There are a total of 583 rows in the data set and these 11 columns:

Columns
Age
Gender
Total_Bilirubin
Direct_Bilirubin
Alkaline_Phosphotase
Alamine_Aminotransferase
Aspartate_Aminotransferase
Total_Protiens
Albumin
Albumin_and_Globulin_Ratio
Dataset

The first 10 columns are the variables used for predicting whether or not a patient has liver disease. They include age, gender, and a list of key liver enzymes and proteins. The Dataset column denotes the outcome and indicates whether or not the patient has liver disease, with 1 indicating liver disease and 2 indicating no liver disease. For the below graph they will be labeled disease and healthy:



This data set is imbalanced containing 416 patient records with liver disease and only 167 non liver disease patient records. The goal of this project is to see if there is predictive power in these available variables so that it would be possible to provide better health outcomes for these patients by being able to identify early signs of liver disease. To measure the success of the models the metrics overall accuracy will be used to determine the success. We will however look at other metrics such as AUPRC to determine the overall effectiveness of the models picking up on the smaller class of non liver disease patients.

The key steps taken in pursuit of this goal were to inspect and then clean and transform the data and then run it through a series of machine learning algorithms to try to improve the accuracy of liver disease prediction. The algorithms were also combined together in an ensemble method to see if combining them would help further. Since the data set is more imbalanced towards liver disease patients, four sampling methods (up-sampling, down-sampling, ROSE and SMOTE) were also completed to see if that would help with the prediction accuracy.

Ultimately the size of the data set and the imbalance towards patients with the disease made it very difficult to build a sufficient model. Generating overall accuracy greater than predicting everyone had the disease was very difficult. While some success was achieved with some models having an AUPRC greater than **0.85** along with much improved specificity, this was achieved at the the overall expense of the sensitivity metric which caused lower accuracy. Given the importance of detecting the disease this is not an ideal result. However, multiple models did agree on the importance of total bilirubin. Expanding the data set significantly, including specifically more of those without liver disease, and re-running the models may prove to be a very significant benefit in improving the models and determining if total bilirubin and the other predictors can be of aid in screening for potential liver disease.

2 Methods / Analysis

In order to avoid over-fitting our model, the data was split into a training set, **train**, and a test set, **test**. Only the training set was used to create the models. The test set was used exclusively for validation of the model results as a control for the predictions.

2.1 Data Processing

2.1.1 Duplicates

In data science duplicates are a very common problem as they can arise for a whole multitude of reasons. There are sometimes issues with the computer systems populating the data and there are also potential issues with humans duplicating data on accident as well. Examining our data we can see we do have a small number of duplicates in our data set:

duplicated(liver)	Count
FALSE	570
TRUE	13

Since these records should not be counted more than once as they can adversely affect our models, they must be removed from the data set.

```
liver <- unique(liver)
```

2.1.2 Check for Missing Values

It is also very common in data science to have fields with NULL values. Searching all of of columns for missing values returns the following results:

	Missing Values
Age	0
Gender	0
Total_Bilirubin	0
Direct_Bilirubin	0
Alkaline_Phosphotase	0
Alamine_Aminotransferase	0
Aspartate_Aminotransferase	0
Total_Protiens	0
Albumin	0
Albumin_and_Globulin_Ratio	4
Dataset	0

The data set is in very good shape with only the Albumin_and_Globulin_Ratio column missing any values at all, and only 4 values missing at that. Since it is such a small volume of records these records will be removed so that the data set is fully accurate without any statistical replacement.

```
liver <- na.omit(liver)
```

2.1.3 Data Types

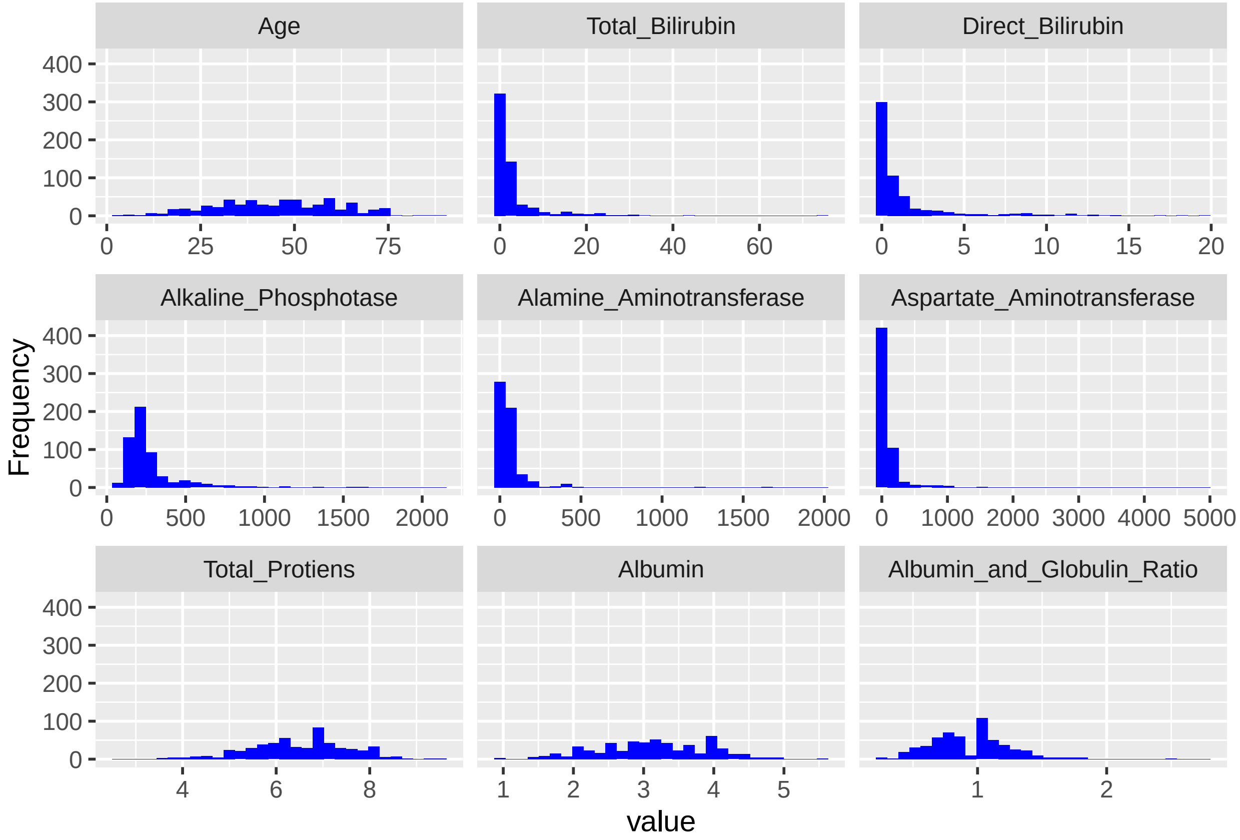
As discussed above, in this data set a 1 indicates the patient does have liver disease, but a 2 is the value to denote there is not disease. Converting this to a 0 makes much more sense as we can turn this into a binary.

Likewise, there is a column for Gender which has values of Male and Female. This value was also hot-encoded into a binary to run through the various machine learning algorithms. Male is set 1 and Female is set to 0.

```
liver$Dataset <- factor(ifelse(liver$Dataset==2,0,1), levels = c(1,0))

liver$Gender <- factor(ifelse(liver$Gender=="Male",1,0), levels = c(1,0))
```

2.1.4 Log Transformation

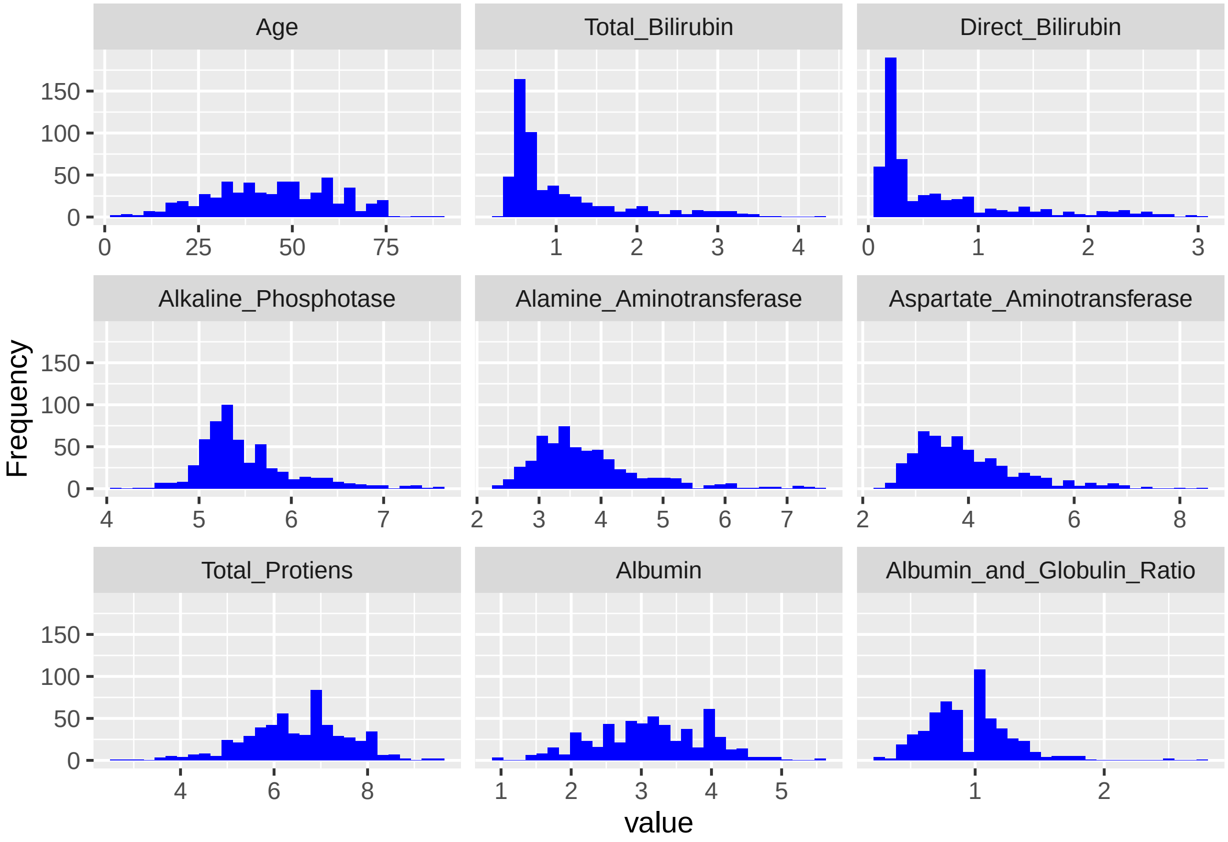
One of the other common problems in data science is that often the distributions of certain variables are highly skewed. When this occurs a simple log transformation can be applied to reduce the range of these predictors to limit the effect on the model. The below is a plot of all of the non-binary predictor variables so that the distributions can be examined:



The plot shows that Total_Billirubin, Direct_Billirubin, Alkaline_Phospotase, Alamine_Aminotransferase and Aspartate_Aminotransferase will all benefit from a log transformation.

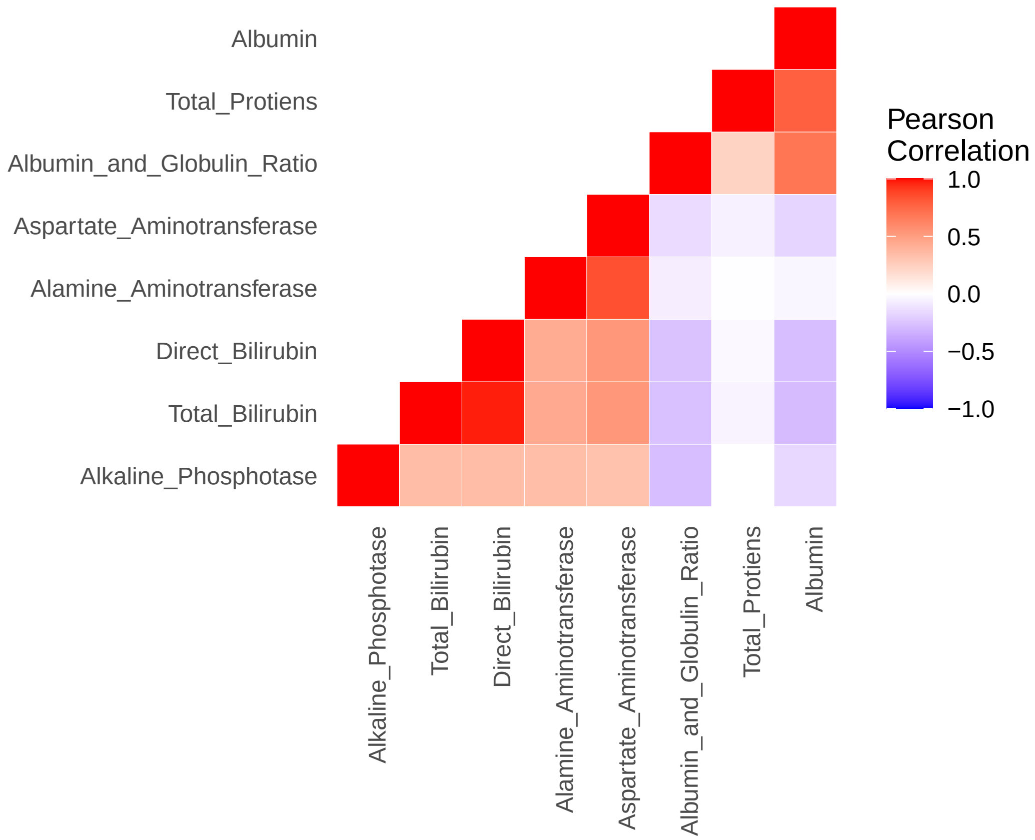
```
livertrans <- liver

livertrans$Total_Bilirubin <- log(livertrans$Total_Bilirubin +1)
livertrans$Direct_Bilirubin <- log(livertrans$Direct_Bilirubin +1)
livertrans$Alkaline_Phosphotase <- log(livertrans$Alkaline_Phosphotase +1)
livertrans$Alamine_Aminotransferase <- log(livertrans$Alamine_Aminotransferase +1)
livertrans$Aspartate_Aminotransferase <- log(livertrans$Aspartate_Aminotransferase +1)
```

After making the change, the histograms can be re-plotted to see the impact:


Age
Total_Bilirubin
Direct_Bilirubin
Alkaline_Phosphotase
Alamine_Aminotransferase
Aspartate_Aminotransferase
Total_Protiens
Albumin
Albumin_and_Globulin_Ratio
Frequency
0
50
100
150
0
25
50
75
1
2
3
4
0
1
2
3
4
5
6
7
2
3
4
5
6
7
2
4
6
8
4
6
8
1
2
3
4
5
1
2
value

2.1.5 Correlation

Lastly before we partition the data we can look at the correlations between all of the predictor variables to see if there are any highly correlated predictors. Here is a heat map of the correlations:



The map makes it very easy to see that total and direct bilirubin are very highly correlated. Looking at the just those two values we can see just how correlated they are:

	Total_Bilirubin	Direct_Bilirubin
Total_Bilirubin	1.0000000	0.9748306
Direct_Bilirubin	0.9748306	1.0000000

As they are pretty close to being perfectly correlated one of these variables should be removed from the algorithms. Here is the average of how highly correlated each variable is with the others. Since having too much correlation between predictors is not ideal, the variable with less total correlation will be kept and the other will be removed:

	x
Total_Bilirubin	0.3355673
Direct_Bilirubin	0.3375647

Both are very close but direct bilirubin will be removed as a predictor from the machine learning models since it is just a touch more highly correlated with the other variables than total bilirubin.

```
livertrans <- select (livertrans,-c(Direct_Bilirubin))
```

2.2 Modeling

Now that the data has been processed for the machine learning algorithms the data can be split into a training set (80%) and a test set (20%) which will be used for validation purposes only.

```
set.seed(1147, sample.kind = "Rounding")
test_index <- createDataPartition(y = livertrans$Dataset, times = 1,
                                  p = 0.2, list = FALSE)
train <- livertrans[-test_index,]
test <- livertrans[test_index,]
```

2.2.1 Metrics

To determine the performance of each machine learning algorithm, the following metrics were used.

Accuracy

This is just a measure of how many of the algorithm predictions were correct, whether true or false.

Sensitivity

This is a measure of the ability to correctly identify those with the disease (true positives).

Specificity

The next metric included is specificity, which is a measure of the ability to correctly identify those without the disease.

F1 Score

F1 Score is defined as the harmonic average of precision and sensitivity. Precision is the ratio of correctly predicted positive observations of the total predicted positive observations. Said more simply, recall is simply a measure of everyone that was predicted to have the disease, how many of them actually have it? The harmonic average of the recall and sensitivity generate the F1 score. This is a good metric since it is accounting for both false positives and false negatives.

AUC

AUC stands for area under the cover. It provides an measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

AUPRC

AUPRC stands for area under the precision recall curve. PR curves are used to examine the trade-off between the proportion of positively labeled examples that are truly positive (precision) as a function of the proportion of correctly classified positives (sensitivity also known as recall). In particular, PR analysis is preferred to ROC AUC analysis when there is a large imbalance in the data.

2.2.2 Naive Guessing Everyone

Since this particular data set is skewed toward people with the disease, the simplest way to predict if a patient has liver disease is to simply guess all patients are positive for the disease as a control for the machine learning algorithms. By doing so below are the results:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1	0	0.8307692	0.5	0

Since 71% of the data set has the disease, of course the accuracy is 71%. The sensitivity is perfect because we guessed 1 for every single record. Correspondingly, the specificity is 0 since we didn't guess a single person to not have liver disease. These will be the benchmarks for each of the machine models to compare against to see how they perform versus this baseline.

2.2.3 Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest predicts out a class prediction and the class with the most votes becomes the model's prediction. Due to these features and the algorithms popularity this was the first algorithm selected to train the data. Ten fold cross validation was repeated ten times and preprocessing was used to center and scale the data.

Here are the results:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325

The random forest was able to generate a AUPRC of **0.712.** The overall accuracy however is actually lower than just guessing that everyone has the disease. The sensitivity did improve from the 0 generated from the guessing model but it is still quite low at **0.121**.

2.2.3.1 Over (Up) Sampling One of the great features of random forests is that we also have the option to add a sampling component to help control for unbalanced data sets. With over-sampling, we randomly duplicate samples from the class with fewer records so that the number of samples in each class match. While losing information is avoided by taking this approach, the risk of over-fitting the model is possible. Therefore cross-validation is necessary on each fold independently to get an honest estimate of the model performance. Below our the results of this approach:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441

With the over-sampling approach, the overall AUPRC is less than the original random forest at **0.647.** The effect of the sampling can be clearly seen looking at the sensitivity and specificity. The specificity markedly increases from **0.121** to **0.364**, but this was at the expense of the sensitivity which correspondingly decreased from **0.877** to **0.815.**

2.2.3.2 Under (Down) Sampling With under-sampling, a subset of samples from the class with more instances are randomly selected to match the number of samples coming from the class. The main concern with under-sampling is that we lose potentially lose out on information from the removed samples. Just like with oversampling, cross-validation is thus necessary on each fold independently to control for this. Below our the results of this method:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386

Under-sampling really shows the effect that sampling can have on an unbalanced data set, as the specificity skyrockets all the way up to **0.758.** Unfortunately, the sensitivity falls down to **0.667.** The AUPRC also falls considerably to **0.599.**

2.2.3.3 ROSE Sampling The ROSE sampling approach is a hybrid method. Artificial balanced samples are generated according to a smoothed bootstrap approach to help deal with unbalanced data. Below are the results from this method:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263

The ROSE sampling approach had an even stronger effect on increasing the specificity **(0.818**), but also had the strongest effect on decreasing the sensitivity to **0.543.** The overall AUPRC **(0.621)** did increase back up versus the under-sampling method, but is still considerably less than over-sampling and original random forest models.

2.2.3.4 SMOTE Sampling The SMOTE sampling approach is another hybrid method. A combination of of over-sampling the minority class and under-sampling the majority class is sued to try to achieve a better classifier performance in AUC space. Over-sampling the minority class involves creating synthetic minority class examples in order to achieve this. Here are the results from the random forest with SMOTE sampling:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409

The SMOTE sampling approach generated the highest accuracy of the random forest approaches at **0.702**, but unfortunately the AUPRC was the second lowest at **0.610.**

2.2.4 Logistic Regression

Since we have a binary outcome and not that many features compare to the size of the data set a logistic regression was a very obvious choice due to its well-known and long-standing reputation. The following output shows the performance:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409
GLM	0.6929825	0.8888889	0.2121212	0.8044693	0.4494949	0.6664767

The logistic regression generates an AUPRC higher than all but our original random forest model, however it did best that model in overall accuracy.

2.2.5 K-Nearest Neighbors

The KNN was chosen as it is also very widely recognized and its approach differs from some of the others chosen so far. This algorithm assumes that similar things exist in close proximity to each other. In other words, similar things are near to each other. A tune grid with seq(1,100,4) was used to optimize for k. Below are the results of how KNN performed on this data set:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409
GLM	0.6929825	0.8888889	0.2121212	0.8044693	0.4494949	0.6664767
KNN	0.7105263	0.9753086	0.0606061	0.8272251	0.4820426	0.6867960

The KNN model bettered the AUC from the logistic regression, moving into second place behind the original random forest in AUPRC at **0.687.** It's accuracy of **0.711** is the best of the machine learning models, however this just ties it with naively guessing everyone has the disease. This is because this model guesses very strongly that most have the disease, with a sensitivity of **0.975** but a specificity not much better than our naive model at **0.061.**

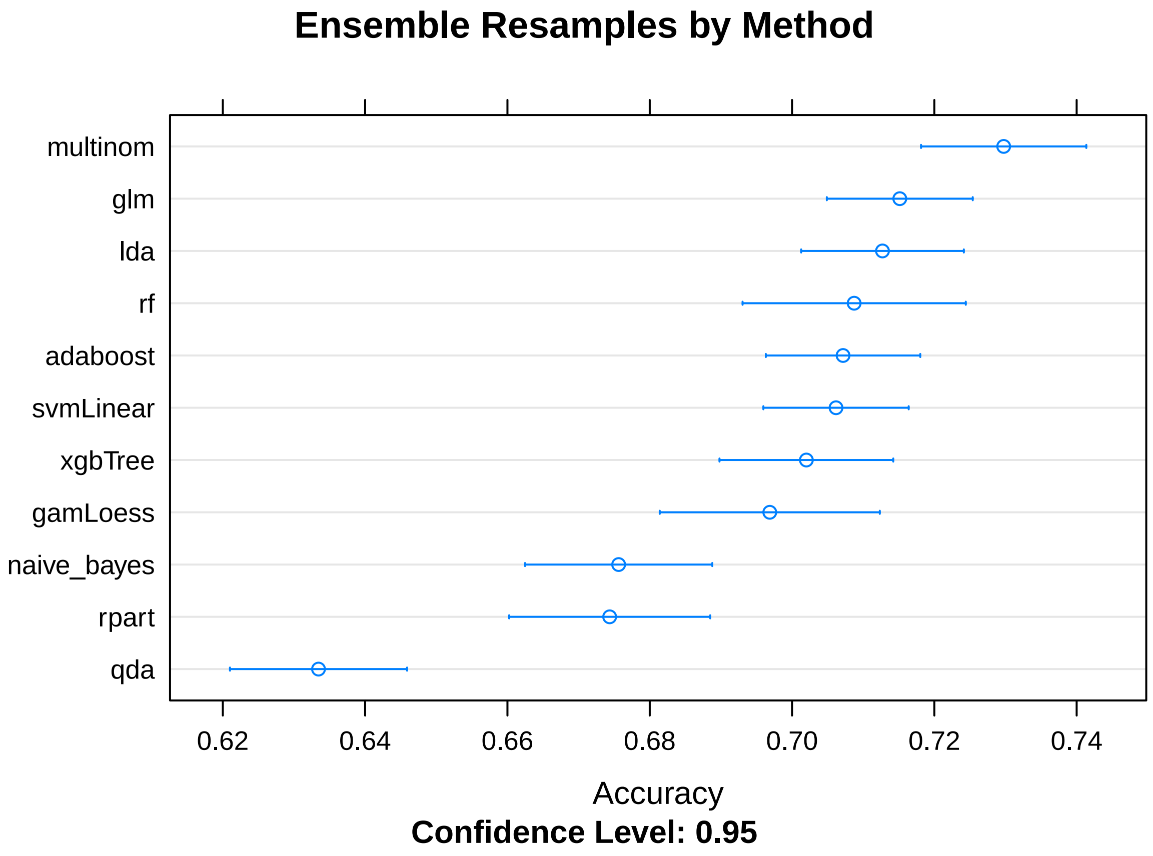
2.2.6 An Ensemble Approach

To see if significant improvement would be be possible by taking an ensemble approach, a multitude of new models were selected and run through a function to see which performed best. LDA, QDA, Naive Bayes, SVM, Loess, Rpart, Multinom and ADABoost were all selected in addition to Random Forest, XGBoost and GLM from above. The default caret tuning will be applied for these models. After running the models, 25 resamples will be taken to avoid over-fitting and the 95% confidence interval of accuracy of each models resamples will be measured. This will give a very nice visual to see which models performed the best.

Here are the metrics for all of the models:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409
GLM	0.6929825	0.8888889	0.2121212	0.8044693	0.4494949	0.6664767
KNN	0.7105263	0.9753086	0.0606061	0.8272251	0.4820426	0.6867960
Multinom	0.6929825	0.9012346	0.1818182	0.8066298	0.5415264	0.7289970
LDA	0.7017544	0.9259259	0.1515152	0.8152174	0.5387205	0.7274810
Adaboost	0.7017544	0.9012346	0.2121212	0.8111111	0.5566779	0.7359866
SvmLinear	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
Rpart	0.7017544	0.7901235	0.4848485	0.7901235	0.6374860	0.7808262
GamLoess	0.6929825	0.8888889	0.2121212	0.8044693	0.5505051	0.7332844
QDA	0.6578947	0.5679012	0.8787879	0.7022901	0.7233446	0.8625964
XGBTree	0.7368421	0.9506173	0.2121212	0.8369565	0.5813692	0.7466291
Naive Bayes	0.7192982	0.6913580	0.7878788	0.7777778	0.7396184	0.8559947

There is some very strong performance from some of the models with Naive Bayes and QDA producing AUPRC numbers over **0.855.** However, these improvements come at the expense of the sensitivity metrics which is not good when the focus is on trying to detect disease. There is some better performance on our overall accuracy, and while it would be tempting to only select these best models for the ensemble, care needs to be taken to avoid over-fitting. Instead, 25 resamples from each of the models can be taken to see how the accuracy compares by method to avoid this danger of over-fitting.



We can see when looking at the resamples that multinom actually does very well, with the lower end of its 95% confidence interval outperforming the median of the 95% confidence interval of all of the other model’s resamples.

Now these top six performing models based on the accuracy confidence intervals (glm, lda, multinom, svmLinear, adaboost and gamLoess) can be used to create the ensemble. First, the models are examined to see if they are very highly correlated with each other as using very highly correlated models is not advised:

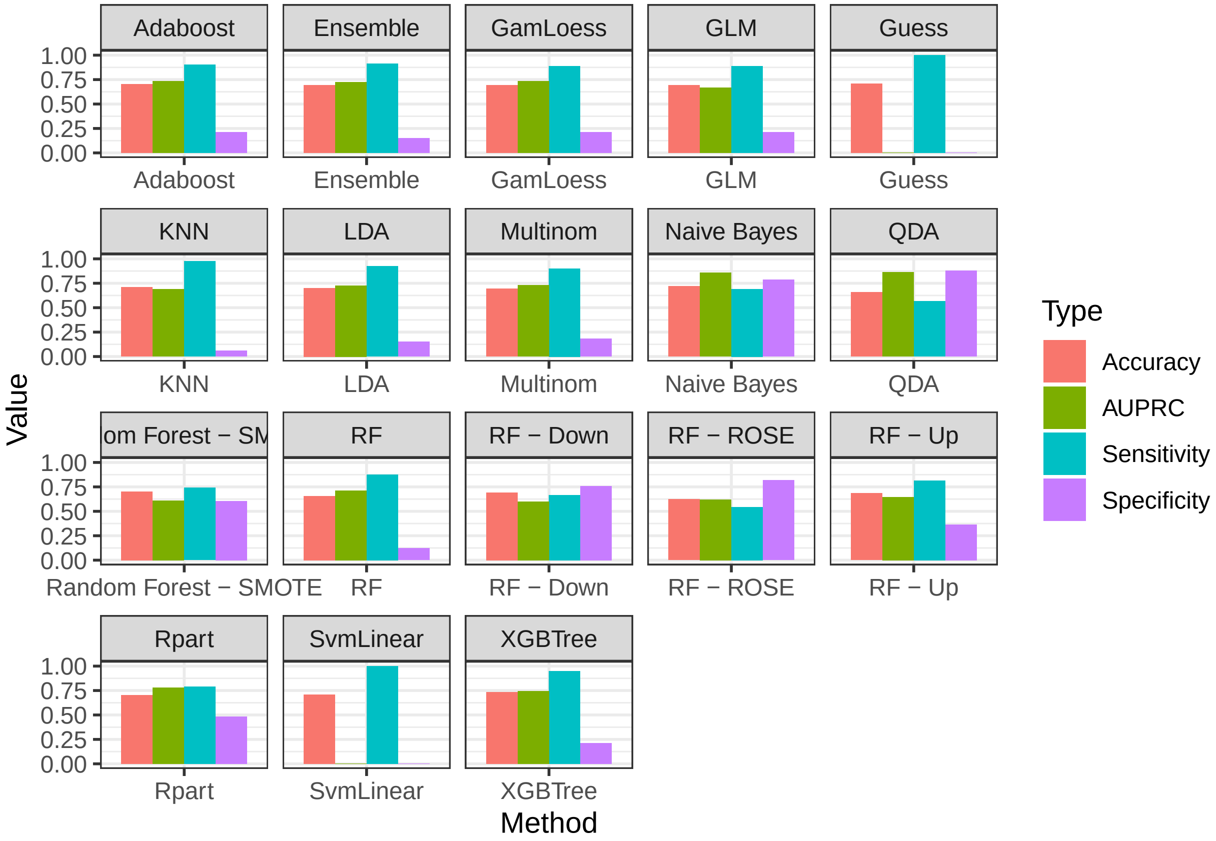
	multinom	glm	lda	rf	adaboost	svmLinear
multinom	1.0000000	-0.2241441	0.0163289	-0.4354081	0.0834399	0.3648580
glm	-0.2241441	1.0000000	0.2095561	-0.0910896	0.2968096	-0.4696080
lda	0.0163289	0.2095561	1.0000000	0.1538365	-0.1532272	0.0720027
rf	-0.4354081	-0.0910896	0.1538365	1.0000000	0.1041044	-0.0347789
adaboost	0.0834399	0.2968096	-0.1532272	0.1041044	1.0000000	-0.1493339
svmLinear	0.3648580	-0.4696080	0.0720027	-0.0347789	-0.1493339	1.0000000

Since there are no concerns with high correlation, an ensemble model can be created and the results are below:

Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409
GLM	0.6929825	0.8888889	0.2121212	0.8044693	0.4494949	0.6664767
KNN	0.7105263	0.9753086	0.0606061	0.8272251	0.4820426	0.6867960
Multinom	0.6929825	0.9012346	0.1818182	0.8066298	0.5415264	0.7289970
LDA	0.7017544	0.9259259	0.1515152	0.8152174	0.5387205	0.7274810
Adaboost	0.7017544	0.9012346	0.2121212	0.8111111	0.5566779	0.7359866
SvmLinear	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
Rpart	0.7017544	0.7901235	0.4848485	0.7901235	0.6374860	0.7808262
GamLoess	0.6929825	0.8888889	0.2121212	0.8044693	0.5505051	0.7332844
QDA	0.6578947	0.5679012	0.8787879	0.7022901	0.7233446	0.8625964
XGBTree	0.7368421	0.9506173	0.2121212	0.8369565	0.5813692	0.7466291
Naive Bayes	0.7192982	0.6913580	0.7878788	0.7777778	0.7396184	0.8559947
Ensemble	0.6929825	0.9135802	0.1515152	0.8087432	0.5325477	0.7248201

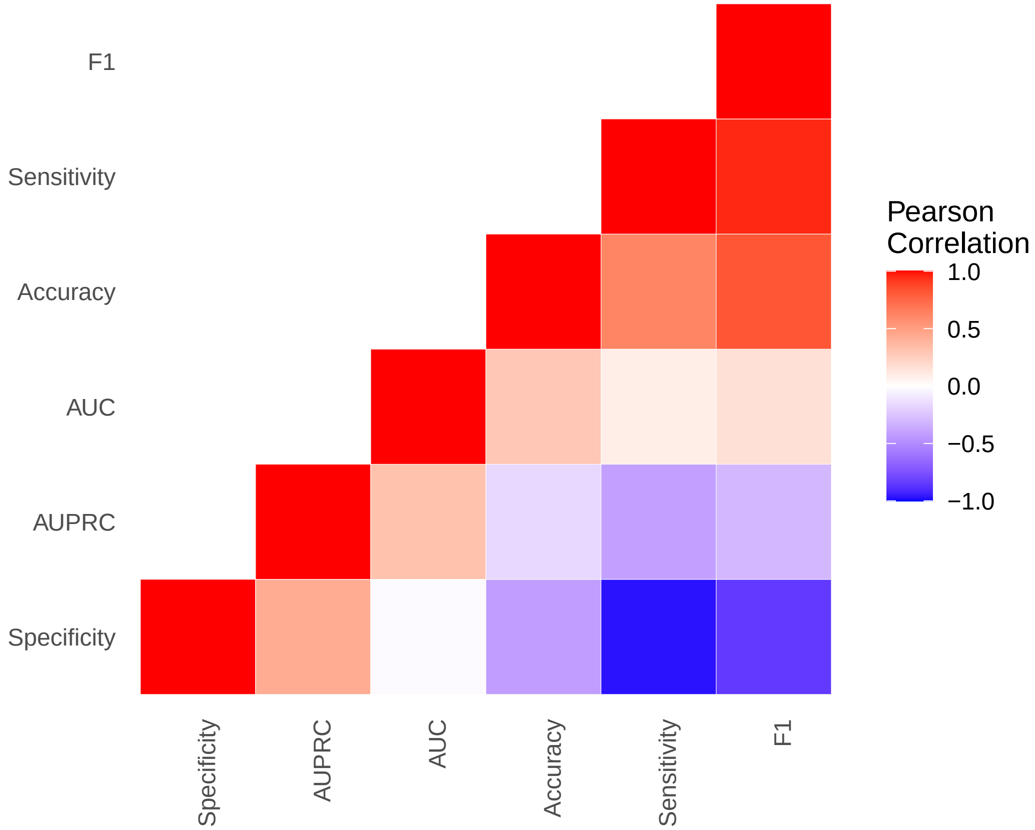
3 Results

Here are the results of the metrics for each of the models we examined:

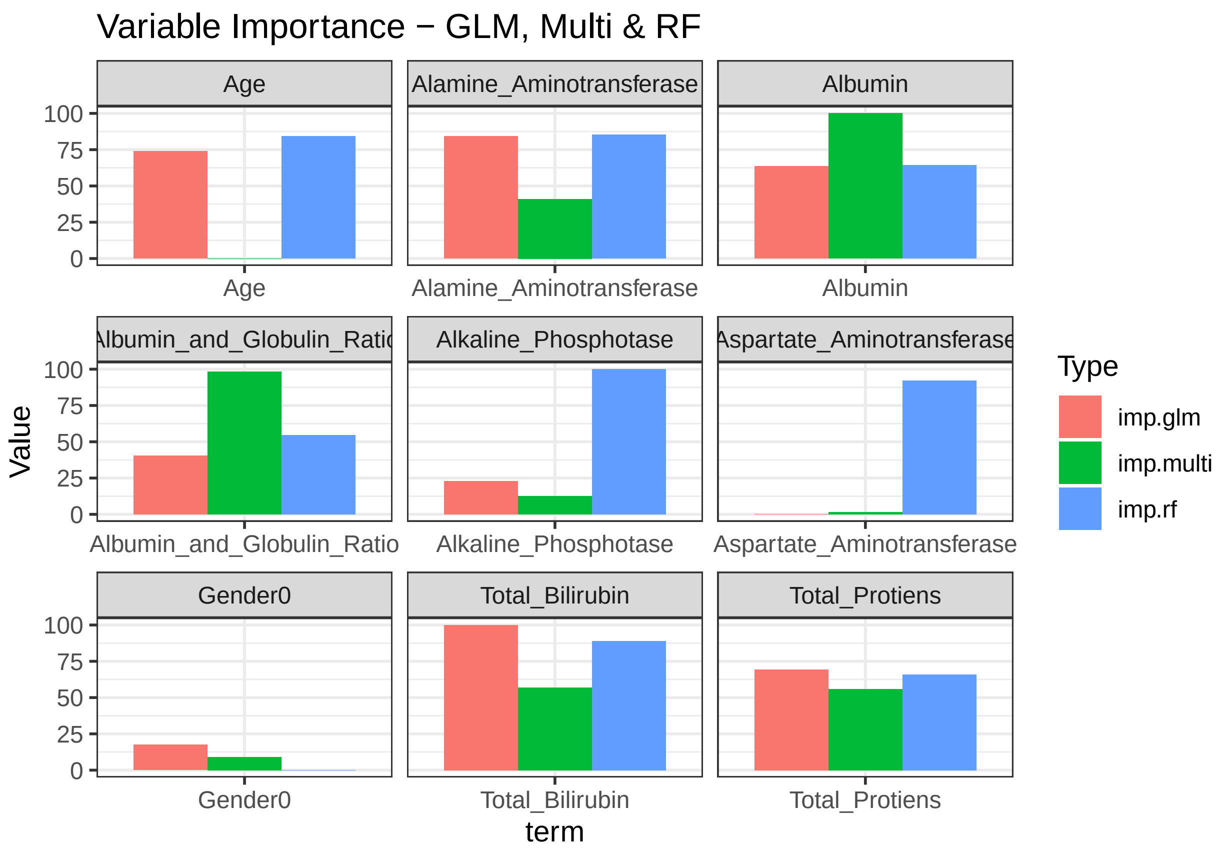


Method	Accuracy	Sensitivity	Specificity	F1	AUC	AUPRC
Guess	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
RF	0.6578947	0.8765432	0.1212121	0.7845304	0.5011223	0.7116325
RF - Up	0.6842105	0.8148148	0.3636364	0.7857143	0.4107744	0.6470441
RF - Down	0.6929825	0.6666667	0.7575758	0.7552448	0.2878788	0.5993386
RF - ROSE	0.6228070	0.5432099	0.8181818	0.6717557	0.3193042	0.6209263
Random Forest - SMOTE	0.7017544	0.7407407	0.6060606	0.7792208	0.3265993	0.6098409
GLM	0.6929825	0.8888889	0.2121212	0.8044693	0.4494949	0.6664767
KNN	0.7105263	0.9753086	0.0606061	0.8272251	0.4820426	0.6867960
Multinom	0.6929825	0.9012346	0.1818182	0.8066298	0.5415264	0.7289970
LDA	0.7017544	0.9259259	0.1515152	0.8152174	0.5387205	0.7274810
Adaboost	0.7017544	0.9012346	0.2121212	0.8111111	0.5566779	0.7359866
SvmLinear	0.7105263	1.0000000	0.0000000	0.8307692	0.5000000	0.0000000
Rpart	0.7017544	0.7901235	0.4848485	0.7901235	0.6374860	0.7808262
GamLoess	0.6929825	0.8888889	0.2121212	0.8044693	0.5505051	0.7332844
QDA	0.6578947	0.5679012	0.8787879	0.7022901	0.7233446	0.8625964
XGBTree	0.7368421	0.9506173	0.2121212	0.8369565	0.5813692	0.7466291
Naive Bayes	0.7192982	0.6913580	0.7878788	0.7777778	0.7396184	0.8559947
Ensemble	0.6929825	0.9135802	0.1515152	0.8087432	0.5325477	0.7248201

The difficulty with the data can easily be seen looking at the combined accuracy results. The control of just assigning everyone to have a positive liver disease status was about as effective as any of our model attempts. The svmLinear model ended up just guessing everyone has the disease just like the naive control! There are some bright spots, with some significant improvements in specificity and AUPRC by using some of the methods discussed to counter the imbalance in the data set. However, since the purpose of the model was to help identify those with liver disease, sensitivity is of vital importance. You can really see the trade off in metrics by examining the correlation between each of the metrics used for all of the models.



The difficulties with the size and balance of our data can further be seen by looking at variable importance. Take a look at this graph showing the variable importance of our models from multinom, random forest and logistic regression (glm):



Unfortunately this clearly shows that some of the predictors that are most important to some of our models are not very important at all to some of the others. However, there are some very interesting positives, most notably total bilirubin, which is deemed to be very important across all three of these models. This could be a very important predictor and further data is needed to verify this possibility.

4 Conclusion

As the results show, it is very difficult to use any of the models in their current state. However, the data set contains many more patients positive with liver disease than those without. Since there are many more individuals that do not have liver disease than those that do in the world population, it is very possible to greatly expand this data set and make it imbalanced in the opposite direction. This approach could prove highly effective in determining the most important variables in detecting liver disease and perhaps be used as a screening method to identify high risk patients. With the high degree of importance placed on total bilirubin across multiple models, expanding this data set in this manner to further investigate is highly advised.