Predicting Liver Disease in an Indian Sub-Population

Simon Piper

07/05/2020

Introduction

Overview Liver disease is a major cause of mortality in both the developed and developing regions of the world, accounting for approximately two million deaths globally each year. Obesity and diabetes are considered risk factors, while other common causes of liver disease include chronic hepatitis B and C and alcohol. In India, prevalence of liver disease has been increasing with the rise of excessive alcohol consumption, as well as an increase in harmful gases and contaminated food. This burden of disease places increasing pressure on healthcare systems and healthcare professionals to intervene before the condition has too far advanced.

This report uses a data set containing 583 patient records from North East of Andhra Pradesh, India, in seeking to develop a prediction algorithmn to identify whether an individual has liver disease or not. The original data set contains the following variables:

- ***Age***: The age of the individual, with those older than 89 listed as age 90.
- ***Gender***: Either Male of Female
- ***Total_Bilirubin***: Bilirubin is a yellow pigment found in the body's normal catabolic pathway, produced as a waste product from the breakdown of heme. This measure is derived from a liver function test, where high levels may be indicative of liver disease. It is measured in mg/dL.
- ***Direct_Bilirubin***: Also known as conjugated bilirubin. It is a water-soluble form of bilirubin. It is measured in mg/dL.
- ***Alkaline_Phosphatase*** (ALP): An enzyme found in various forms in the body involved in lipid transposition as well as calcification of bones, with high amounts found in both bone and liver. Elevated ALP levels may indicate liver or bone disorders. It is measured in U/L.
- ***Alamine_Aminotransferase*** (ALT): Found mainly in the liver, raised levels can be due to liver damage or inflammation, and such a test can be used as an early detection of liver disease. It is measured in U/L.
- ***Aspartate_Aminotransferase*** (AST): Found in a wider range of organs including the liver, brain, kidneys, heart and muscles, making it a relatively weaker indicator of liver disease compared to ALT. Alongside other tests it can be useful in detecting liver disease. It is measured in U/L.
- ***Total_Proteins***: This measures the total amount of albumin and globulin in the blood. It is measured in g/dL.
- ***Albumin***: A protein made specifically by the liver which functions to prevent fluid from leaking out of blood vessels. It is the main consituent of total protein, making up approximately 50%. Low levels of albumin can be indicative of liver disease. It is measured in g/dL.
- ***Albumin_and_Globulin_Ratio***: Comparing the amount of albumin to globulin, where globulin is considered the other main constituent of total protein. Globulin is produced in the liver by the immune system, and plays an important role in liver function, blood clotting and fighting infection. Low levels of globulin can be a sign of liver disease. It is measured in g/dL.
- ***Dataset***: A class label used to divide the individuals into those with liver disease ('1'), and those without ('2').

The model performance and degree to which it can correctly classify whether a patient has liver disease or not, will be measured according to several metrics which can be drawn from the following confusion matrix:

	Actual	
	Positive	Negative
Predicted		
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Such metrics include:

- **Accuracy**: The proportion of observations which are classified correctly. It is not robust when there is class imbalance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall**: The proportion of actual positive outcomes correctly identified. Also known as the true positive rate (TPR) or sensitivity.

$$Recall = \frac{TP}{TP + FN}$$

- **Specificity**: The proportion of actual negative outcomes correctly identified. Also known as the true negative rate (TNR).

$$Specificity = \frac{TN}{N}$$

- **Precision**: The total number of correctly classified positive outcomes out of all positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

- **F1**: A harmonic mean of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

- **Matthew's Correlation Coefficient (MCC)**: A correlation coefficient between the observed and predicted classifications. It takes into account true and false positives and negatives, and regarded as a balanced measure when classes are of different frequencies.
  - MCC = 1 represents perfect positive correlation - the classifier is perfect ($FP = FN = 0$)
  - MCC = 0 represent no correlation - the classifier randomly classifies
  - MCC = -1 represents perfect negative correlation - the classifier always missclassifies ($TP = TN = 0$)

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **ROC & AUC**: The receiver operating characteristic curve (ROC) is a graph of TPR vs FPR showing the performance of a classification model at different threshold values. Where $0 \leq threshold \leq 1$ and p is the probability that a patient has liver disease, then they are classified as having the disease if $p > threshold$. The 'Area Under the ROC Curve' (**AUC**), provides a very useful aggregate measure of the performance model across all possible classification thresholds. AUC ranges between 0 and 1, where a model with an excellent measure of separability has an AUC of 1, and 0.5 being equivalent to random classification.

Method & Analysis

Data Wrangling Columns are first renamed and response variable altered for data manipulation purposes:

```
rawdata <- rawdata %>% rename(TB = Total_Bilirubin, DB = Direct_Bilirubin,
                              ALP = Alkaline_Phosphotase,
                              ALT = Alamine_Aminotransferase,
                              AST = Aspartate_Aminotransferase,
                      TP = Total_Protiens, A = Albumin,
                      AG_Ratio = Albumin_and_Globulin_Ratio,
                      Disease = Dataset)

rawdata$Disease <- replace(rawdata$Disease, rawdata$Disease==2, "No")
rawdata$Disease <- replace(rawdata$Disease, rawdata$Disease==1, "Yes")
```

Male is created as a dummy variable for modelling purposes, taking the value 1 where male and 0 if female:

```
rawdata$Male <- as.character(rawdata$Gender)
rawdata$Male <- replace(rawdata$Male, rawdata$Male == "Male", 1)
rawdata$Male <- replace(rawdata$Male, rawdata$Male =="Female", 0)
rawdata$Male <- as.numeric(rawdata$Male)
```

The data is then checked for the presence of any missing values:

```
colSums(is.na(rawdata))
```

```
##      Age   Gender       TB       DB      ALP      ALT      AST       TP
##        0        0        0        0        0        0        0        0
##        A AG_Ratio  Disease     Male
##        0        4        0        0
```

The relevant rows are identified in the *AG_Ratio* column and missing values are replaced with mean values. As the data set is relatively small, this method of adjustment is preferred over removal of entire rows:

```
which(is.na(rawdata$AG_Ratio))
```

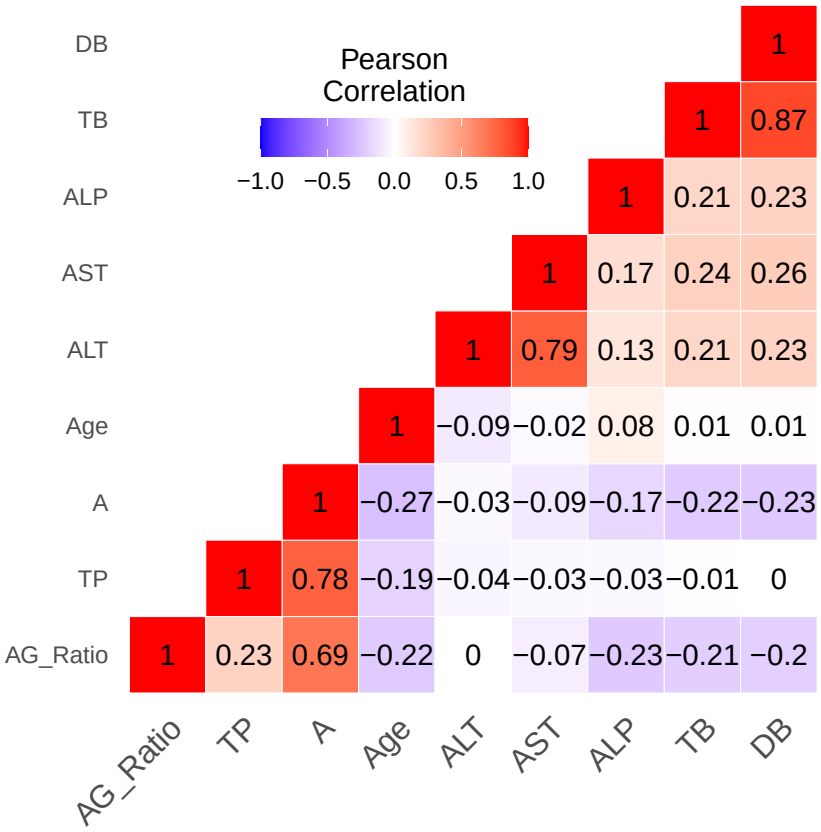
```
## [1] 210 242 254 313
```

```
rawdata$AG_Ratio <- replace(rawdata$AG_Ratio, rawdata$AG_Ratio=="", NA)
rawdata$AG_Ratio <- as.numeric(rawdata$AG_Ratio)

mean(rawdata$AG_Ratio, na.rm = TRUE)
```

```
## [1] 0.9470639
```

```
rawdata$AG_Ratio[is.na(rawdata$AG_Ratio)] <-
  mean(rawdata$AG_Ratio, na.rm = TRUE)
```

The predictor variables are then assessed for multicollinearity, to ensure that there is no strong correlation between variables as this would lead to inflated standard errors and overfitting, and therefore affecting performance of the machine learning model:



Pearson's correlation coefficient is a statistical measure to assess if a linear relationship exists between any two variables, where $-1 \leq r \leq 1$ and $0.6 \leq r$ indicates 'strong' positive correlation.

As perhaps expected, there is strong correlation between *TB* and *DB*, and so a new variable *DB_TB* is created representing the DB/TB ratio.

Furthermore, *A* appears strongly correlated with *AG_Ratio*, and so *G* is calculated and created as a separate variable:

```
rawdata <- rawdata %>% mutate(DB_TB = DB / TB, G = A / AG_Ratio)
```

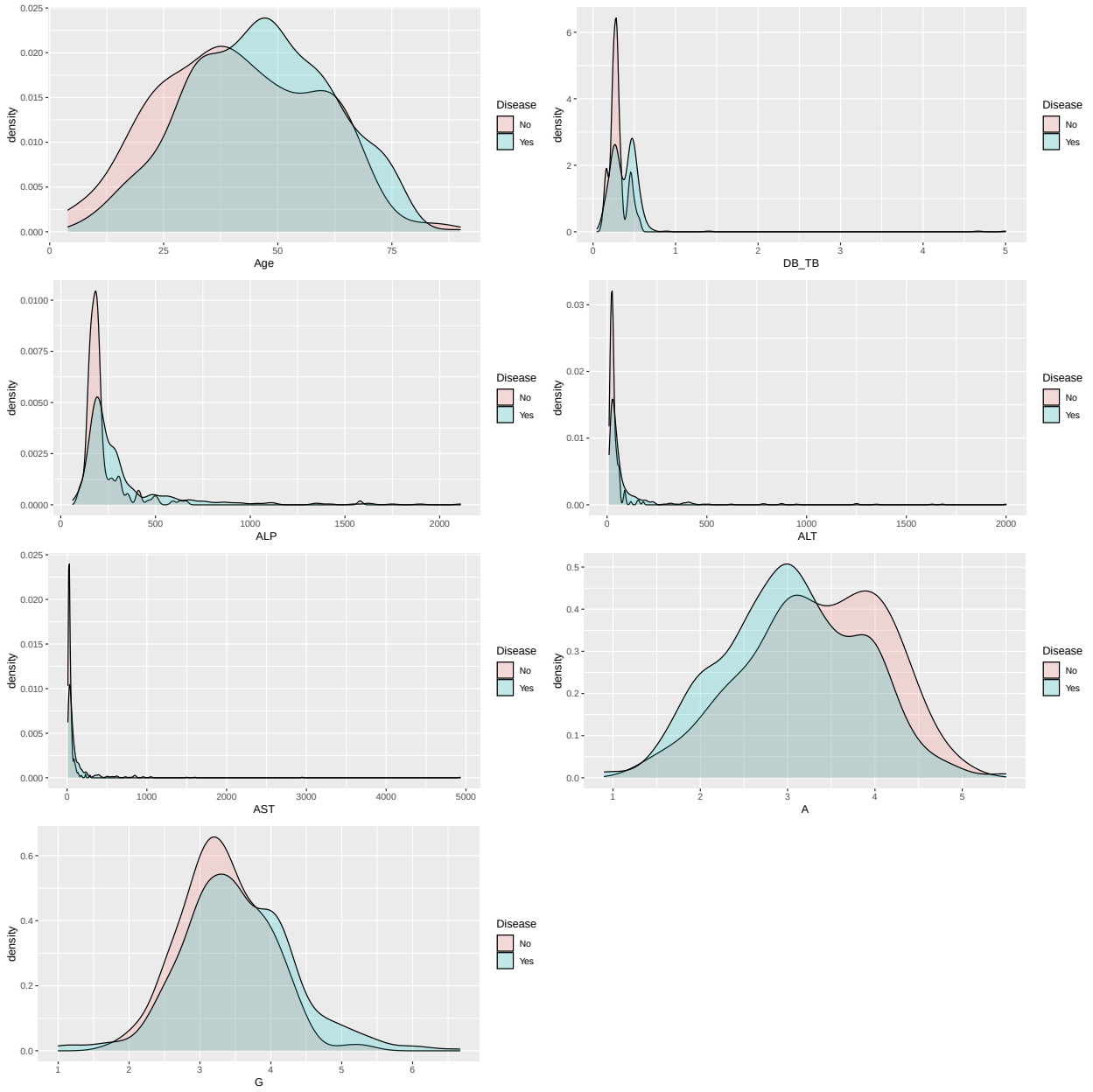
Correlation can then be checked between *A*, *G* and *TP*:

```
##        A   TP    G
## A   1.00 0.78 0.03
## TP  0.78 1.00 0.57
## G   0.03 0.57 1.00
```

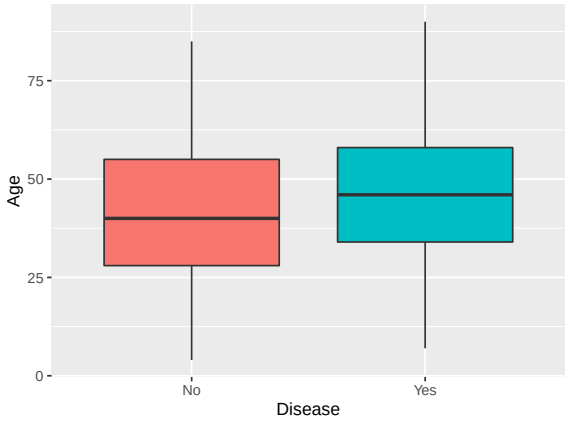
As *A* is strongly correlated with *TP* but not *G*, and *TP* is largely composed of albumin and globulin, *TP* can be dropped (in addition to *TB*,*DB* and *AG_Ratio*).

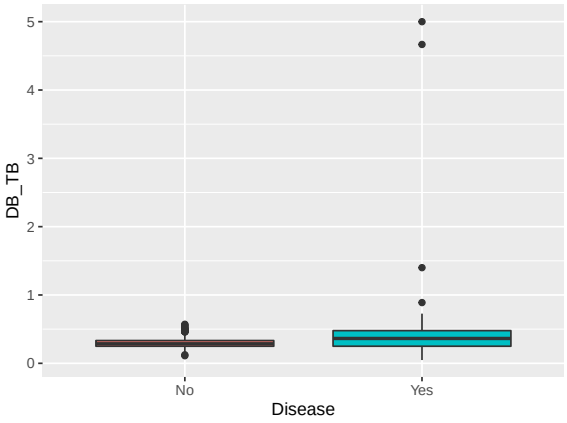
AST and *ALT* show strong correlation, however, there is concern that combining them to form an AST/ALT ratio results in a feature of much reduced significance, while omitting entirely would weaken the predictive power of the subsequent model. Thus, they are left as separate predictors.

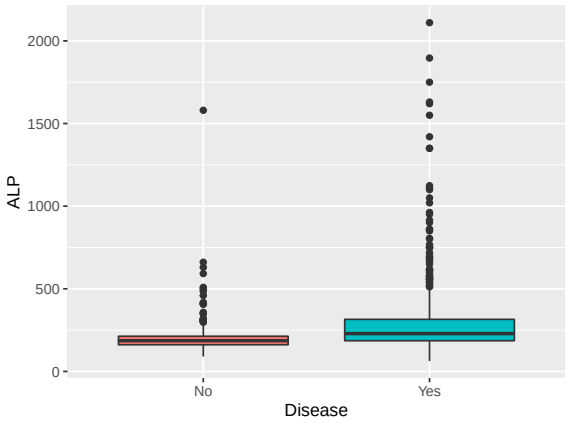
Exploratory Data Analysis

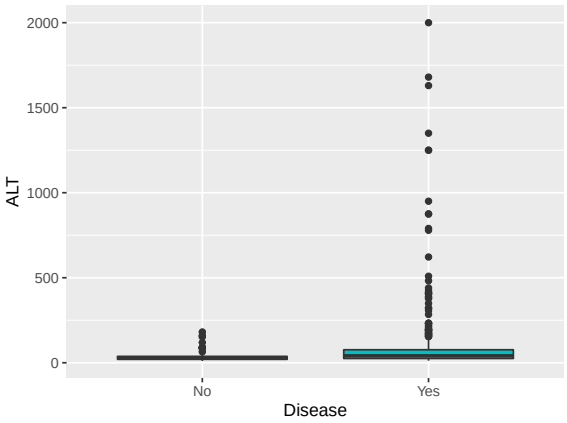


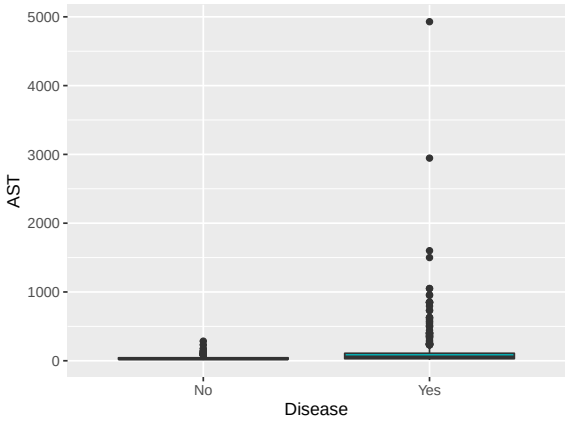
Density plots seem to display a slightly higher average age for patients with liver disease compared to those without. They also show lower albumin (A) levels on average for those with liver disease relative to those without. The plots for *ALT*, *AST*, *ALP* and *DB_TB* show a significantly skewed distribution, characteristic of outliers. These can be further analysed with boxplots:

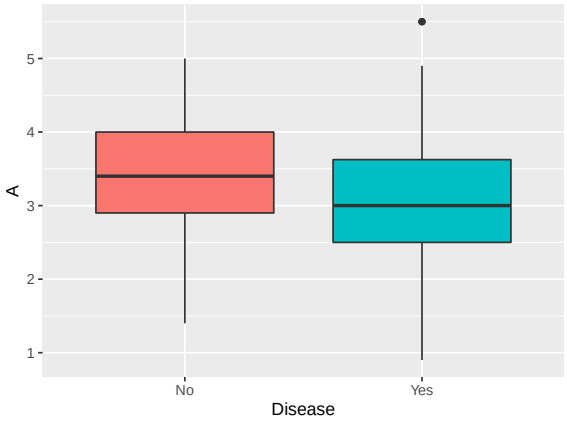

Age
75
50
25
0
No
Yes
Disease

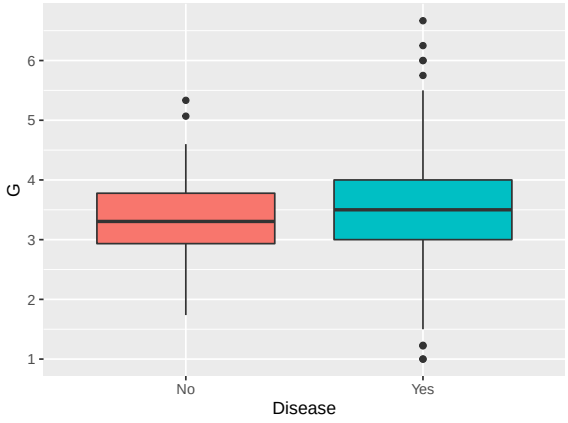

DB_TB
5
4
3
2
1
0
No
Yes
Disease


ALP
2000
1500
1000
500
0
No
Yes
Disease


ALT
2000
1500
1000
500
0
No
Yes
Disease

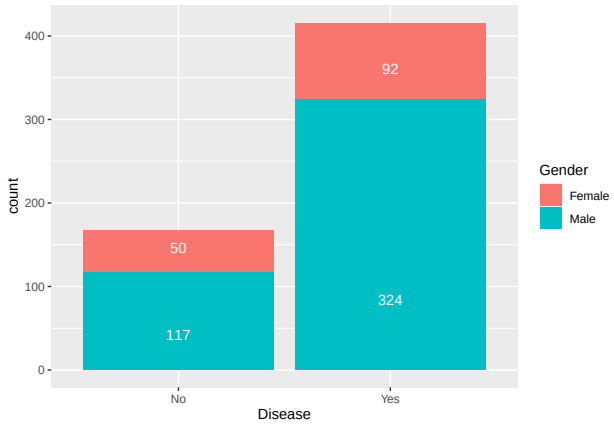

AST
5000
4000
3000
2000
1000
0
No
Yes
Disease


A
5
4
3
2
1
No
Yes
Disease


G
6
5
4
3
2
1
No
Yes
Disease

As *ALP*, *ALT* and *AST* have a higher proportion of outliers compared to the other features, these are handled using log transformation during modelling.

The following plot illustrates the level of imbalance which exists in the data, not just between males and females, but more significantly between patients with liver disease and those without:




```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  583
##
##
##              | Disease
##       Gender |        No |       Yes | Row Total |
## -------------|-----------|-----------|-----------|
##       Female |        50 |        92 |       142 |
##              |     0.352 |     0.648 |     0.244 |
##              |     0.299 |     0.221 |           |
## -------------|-----------|-----------|-----------|
##         Male |       117 |       324 |       441 |
##              |     0.265 |     0.735 |     0.756 |
##              |     0.701 |     0.779 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       167 |       416 |       583 |
##              |     0.286 |     0.714 |           |
## -------------|-----------|-----------|-----------|
##
##
```

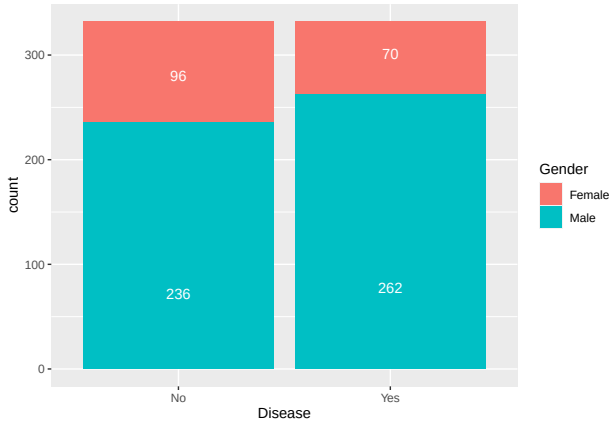
In order to develop the model and conduct the analysis, the data set is split into a training set and test set. Only the training set is used for tuning, with predictions and analysis made against the test set. Best practice would be to create another validation set reserved for analysis only with the final model, however the small sample size of the data set makes this unnecessary.

```
set.seed(755, sample.kind = "Rounding")
test_index <- createDataPartition(y = rawdata$Disease, times = 1, p = 0.2, list = FALSE)
train_set <- rawdata[-test_index,]
test_set <- rawdata[test_index,]
```

As class imbalance can have a negative impact on model fitting, subsampling can be applied to the training data set in order to avoid any such issues. Using a smaller data set, up-sampling is preferred to down-sampling, whereby the minority class is randomly sampled (with replacement) in order that it is the same size as the majority class:

```
'%ni%' <- Negate('%in%')
options(scipen = 999)
set.seed(100, sample.kind ="Rounding")
up_train <- upSample(x = train_set[, colnames(train_set) %ni% "Class"],
                     y = factor(train_set$Disease))
```

The following plot illustrates the class balance for each gender following upsampling:



The up-sampled training data set is then structured as follows:

```
str(up_train)
```

```
## 'data.frame':    664 obs. of  10 variables:
##  $ Age    : int  25 33 20 84 57 38 38 17 62 42 ...
##  $ ALP    : int  183 165 128 188 187 410 410 145 160 630 ...
##  $ ALT    : int  91 15 20 13 19 59 59 18 42 25 ...
##  $ AST    : int  53 23 30 21 23 57 57 36 110 47 ...
##  $ A      : num  2.3 3.5 1.9 3.2 2.9 3 3 3.9 2.6 2.3 ...
##  $ Disease: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Male   : num  1 1 1 0 1 0 0 0 1 1 ...
##  $ DB_TB  : num  0.167 0.312 0.455 0.286 0.3 ...
##  $ G      : num  3.29 3.8 2 2.91 2.42 ...
##  $ Class  : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Model 1 - Logistic Regression A logistic regression is a binary response model which contrains the conditional probability, $Pr(Y = 1|X = x)$, to between 0 and 1. It uses Maximum Liklihood Estimation (MLE), making use of the logistic transformation:

$$G(p) = \log \frac{p}{1-p}$$

whereby the conditional probability that a patient has liver disease can be modelled with:

$$Pr(Y = 1|X = x) = G(\beta_0 + \beta_1 x_1 + ... + \beta_k x_k) = G(\beta_0 + x\beta),$$

where G is a function taking on values strictly between zero and one: $0 < G(z) < 1$, for all real numbers z.

```
set.seed(1, sample.kind = "Rounding")
fit_glm <- glm(Disease ~ Age + A + G + DB_TB + log(ALP) + log(ALT) + log(AST) + Male,
               family = "binomial", data = up_train)
summary(fit_glm)
```

```
##
## Call:
## glm(formula = Disease ~ Age + A + G + DB_TB + log(ALP) + log(ALT) +
##     log(AST) + Male, family = "binomial", data = up_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5511  -0.9236  -0.2709   0.9537   2.2548
##
## Coefficients:
##              Estimate Std. Error z value        Pr(>|z|)
## (Intercept) -9.312590   1.404510  -6.630 0.0000000000335 ***
## Age          0.018141   0.005536   3.277         0.00105 **
## A           -0.176785   0.123199  -1.435         0.15130
## G            0.371931   0.139761   2.661         0.00779 **
## DB_TB        2.447085   0.774512   3.160         0.00158 **
## log(ALP)     0.598568   0.243077   2.462         0.01380 *
## log(ALT)     0.644969   0.213762   3.017         0.00255 **
## log(AST)     0.378871   0.185619   2.041         0.04124 *
## Male         0.102888   0.211979   0.485         0.62741
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 920.50  on 663  degrees of freedom
## Residual deviance: 749.04  on 655  degrees of freedom
## AIC: 767.04
##
## Number of Fisher Scoring iterations: 5
```

An advantage of the logistic model is it's interpretability, where for example each additional year to a patient's age corresponds to a 0.018 increase in the log odds of disease. Additionally, an increase in one unit of log ALT is associated with a 0.645 increase in the log odds of disease.

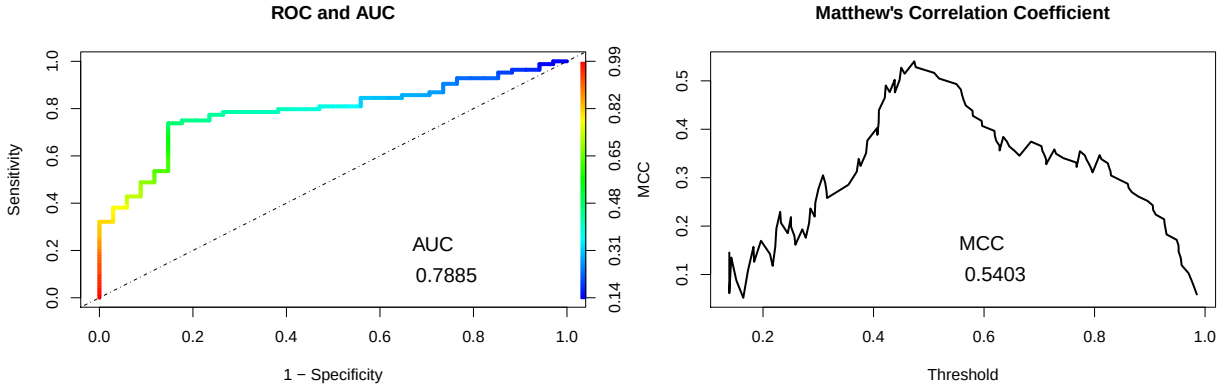
Each of the coefficient estimates shows statistical significance to at least the 5% level, except for that of *A* and *Male*. However, as theory and evidence strongly suggests that each of these parameters holds clinical significance in predicting liver disease, and there is concern that their omission could mean misspecification of the model, they are included in subsequent models within this anlaysis.

The *caret* package can then be used to perform k-fold cross validation with the training set, where it is split into $k = 5$ non-overlapping sets. The average MSE can then be calculated across k sets, after calculating for each, in selecting the optimal parameters which minimise the MSE. K-fold cross validation ($k = 5$) is used across all subsequent models.

```
ctrl_glm <- trainControl(method = "cv", number = 5)
up_train$Disease <- as.factor(up_train$Disease)
```

```
set.seed(1, sample.kind = "Rounding")
fit_glmk <- train(Disease ~ Age + A + G + DB_TB + Male + log(ALP) + log(ALT) + log(AST),
                  method = "glm", family = "binomial", data = up_train,
                  trControl = ctrl_glm)
```

The ROC can then be plotted and AUC calculated:



The ROC curve displays the tradeoff between the true positive rate (sensitivity) and false positive rate (1 - specificity) for different probability thresholds, showing its diagnostic ability. ROC curves which approach closer to the top-left corner indicate better performance, being able to correctly classify a positive outcome - in this case a patient with disease. A perfect classifier is able to discriminate between those with and without disease with 100% sensitivity and 100% specificity, characterised by an ROC curve which moves through (0,1).

The diagonal line indicates a random classifier, while any ROC curve below this would indicate performance worse than a random classifier. In order to compare performance of different classifiers, a common metric is to measure the area under the ROC curve - the AUC.

While accuracy is a useful metric in measuring the proportion of correct classifications, it is for a single threshold value. The ROC curve however, considers all thresholds and can be considered more informative to a certain extent.

The confusion matrix can then be calculated after making predictions against the test set, where $p > 0.5$ classifies a patient as having liver disease:

```
y_hat_glmk <- predict(fit_glmk, newdata = test_set)
cm_glmk <-confusionMatrix(data = factor(y_hat_glmk), reference = factor(test_set$Disease),
                  positive = "Yes", mode="everything")
cm_glmk
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  29  24
##        Yes  5  60
##
##                Accuracy : 0.7542
##                  95% CI : (0.6665, 0.8288)
##     No Information Rate : 0.7119
##     P-Value [Acc > NIR] : 0.1807814
##
```

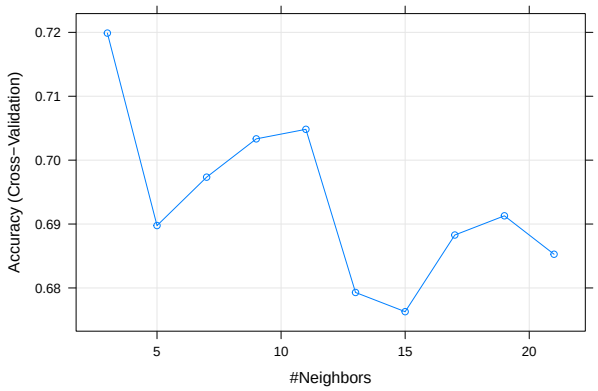
```
##                  Kappa : 0.4863
##
##  Mcnemar's Test P-Value : 0.0008302
##
##             Sensitivity : 0.7143
##             Specificity : 0.8529
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.5472
##               Precision : 0.9231
##                  Recall : 0.7143
##                      F1 : 0.8054
##              Prevalence : 0.7119
##          Detection Rate : 0.5085
##    Detection Prevalence : 0.5508
##       Balanced Accuracy : 0.7836
##
##        'Positive' Class : Yes
##
```

Interestingly, the results show that a logistic regression, perhaps considered a baseline algorithmn, has a higher specificity (0.8529) than sensitivity (0.7143), meaning it is better at detecting negative outcomes or those without disease than those with. Given the up-sampling performed on the training data set to create balanced classes of patients with and without disease, the accuracy measure of 0.7542 can be considered a valid metric for inference.

Model 2 - K-Nearest Neighbours (kNN) The K-Nearest Neighbours algorithmn exploits the distance or proximity of similar data points, where for any object (x_1, x_2), $p(x_1, x_2)$ can be estimated using the k nearest points and taking an average of the 0's and 1's associated. Large values of k can potentially result in over-smoothing, while a very low value of k tends to lead to over-training. Therefore, k can be selected to maximise accuracy:

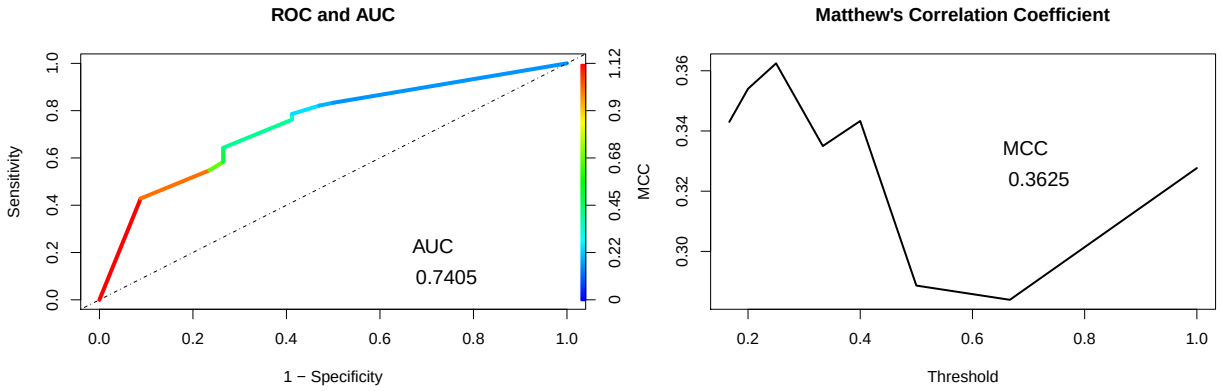
```
ctrl_k <- trainControl(method = "cv", number = 5)
set.seed(1, sample.kind = "Rounding")
fit_knn <- train(Disease ~ Age + A + G + DB_TB + Male + log(ALP) + log(ALT) + log(AST),
                 method = "knn", data = up_train, tuneGrid = data.frame(k = seq(3,21,2)),
                trControl = ctrl_k)
fit_knn$bestTune
```

```
##   k
## 1 3
```



```
## ROC curve variable importance
##
##       Importance
## AST       100.00
## ALT        92.39
## ALP        83.26
## DB_TB      69.20
## G          32.57
## A          27.49
## Age        16.85
## Male        0.00
```

While the tuning parameter which maximises accuracy is where k = 3, variable importance can then also be calculated, evaluating how frequently a predictor is used. This shows that the predictor making the greatest contribution to the model is *AST*, while the dummy variable *Male* has a (scaled) value of 0.



Both AUC and MCC values are lower than that of the logistic regression, while the maximal MCC (0.3625) appears to lie at a much lower threshold value approaching 0.25.

```
y_hat_knn <- predict(fit_knn, newdata = test_set)
cm_knn <-confusionMatrix(data = factor(y_hat_knn), reference = factor(test_set$Disease),
                         positive = "Yes", mode="everything")
cm_knn
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction No Yes
##        No  25  37
##        Yes  9  47
##
##               Accuracy : 0.6102
##                 95% CI : (0.5161, 0.6986)
##    No Information Rate : 0.7119
##    P-Value [Acc > NIR] : 0.9934
##
##                  Kappa : 0.2368
##
##  Mcnemar's Test P-Value : 0.00006865
##
##             Sensitivity : 0.5595
```

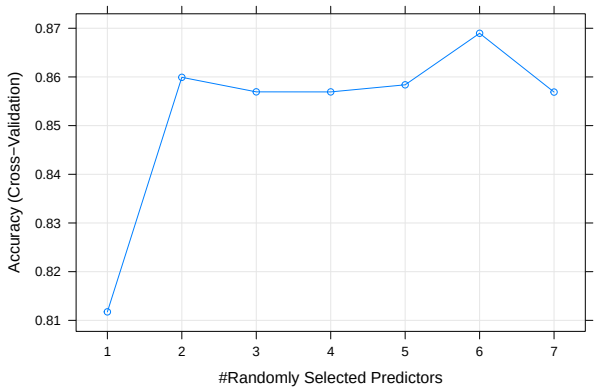
```
##             Specificity : 0.7353
##          Pos Pred Value : 0.8393
##          Neg Pred Value : 0.4032
##               Precision : 0.8393
##                  Recall : 0.5595
##                      F1 : 0.6714
##              Prevalence : 0.7119
##          Detection Rate : 0.3983
##    Detection Prevalence : 0.4746
##       Balanced Accuracy : 0.6474
##
##        'Positive' Class : Yes
##
```

For a threshold value of 0.5, the confusion matrix reveals lower metrics in every respect relative to the initial logistic regression, revealing a slightly weaker classifier.

Model 3 - Random Forest Random forest models improve prediction performance and stability by averaging multiple decision trees. Classification trees generate many predictors, with a final prediction then based on the average prediction of all trees.

```
ctrl_rf <- trainControl(method = "cv", number = 5)
set.seed(1, sample.kind = "Rounding")
fit_rf <- train(Disease ~ Age + A + G + DB_TB + Male + log(ALP) + log(ALT) + log(AST),
                method = "rf", data = up_train, trControl = ctrl_rf,
                tuneGrid = data.frame(mtry = seq(1,7,1)))
fit_rf$bestTune
```

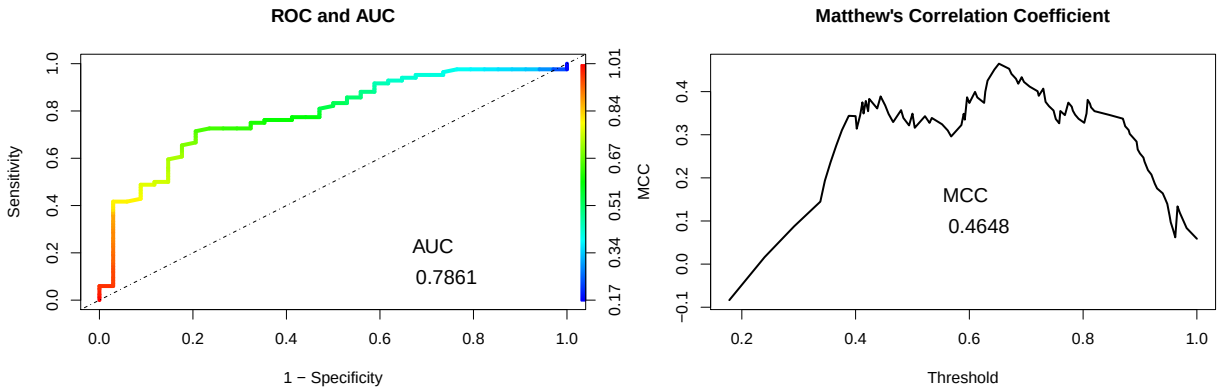
```
##   mtry
## 6    6
```




```
## rf variable importance
##
##          Overall
## log(ALP)  100.00
## log(ALT)   80.30
## log(AST)   79.41
## DB_TB      67.80
## G          63.98
## Age        63.34
## A          37.61
```

```
## Male        0.00
```

Tuning is optimised where mtry = 6, and similar to the previous model, the parameters with the highest variable importance are *log(ALP), log(ALT), log(AST)* and *DB_TB*.




```
y_hat_rf <- predict(fit_rf, newdata = test_set)
cm_rf <- confusionMatrix(data = factor(y_hat_rf), reference = factor(test_set$Disease),
                         positive = "Yes", mode="everything")
cm_rf
```

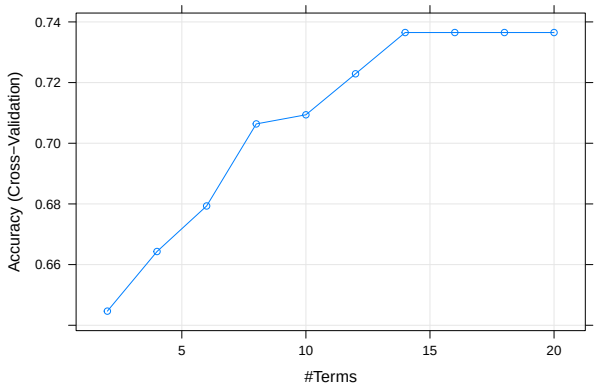
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  16  13
##        Yes 18  71
##
##                Accuracy : 0.7373
##                  95% CI : (0.6483, 0.814)
##     No Information Rate : 0.7119
##     P-Value [Acc > NIR] : 0.3095
##
##                   Kappa : 0.3303
##
##  Mcnemar's Test P-Value : 0.4725
##
##             Sensitivity : 0.8452
##             Specificity : 0.4706
##          Pos Pred Value : 0.7978
##          Neg Pred Value : 0.5517
##               Precision : 0.7978
##                  Recall : 0.8452
##                      F1 : 0.8208
##              Prevalence : 0.7119
##          Detection Rate : 0.6017
##    Detection Prevalence : 0.7542
##       Balanced Accuracy : 0.6579
##
##        'Positive' Class : Yes
##
```

The random forest model shows very high sensitivity/recall (0.8452) relative to the previous two models, although this seems to come at the expense of a much reduced specificity (0.4706). Further confirming its ability to better classify positive outcomes, the random forest model also yields a higher F1 value of 0.8202, as a harmonic mean of recall and precision.

Model 4 - Earth Utilising the *earth* package, a multivariate adaptive regression splines (MARS) model is estimated. This is considered an extended non-parametric regression with a greater level of flexibility, automatically identifying any non-linearities in the data in order to maximise predictive accuracy.

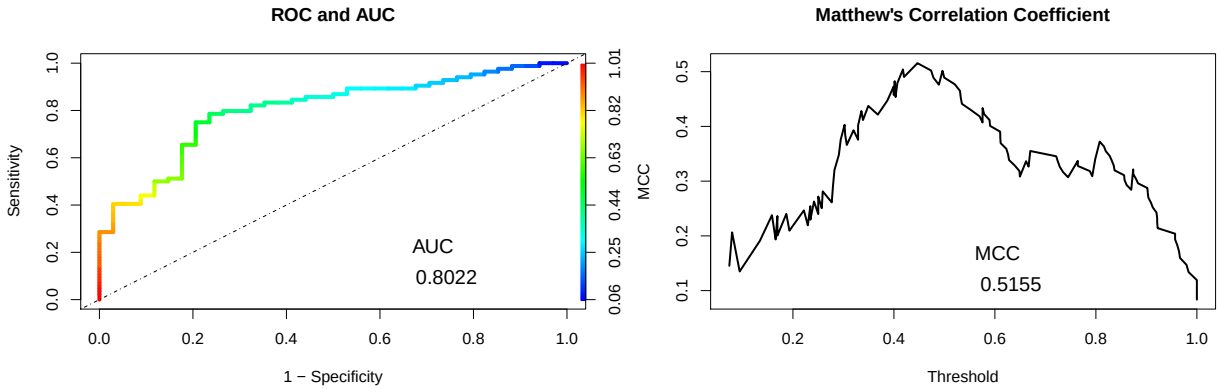
```
ctrl_e <- trainControl(method = "cv", number = 5)
set.seed(1, sample.kind = "Rounding")
fit_e <- train(Disease ~ Age + A + G + DB_TB + Male + log(ALP) + log(ALT) + log(AST),
               method = "earth", data = up_train, trControl = ctrl_e,
              tuneGrid = expand.grid(.degree=1, .nprune=(1:10)*2))

fit_e$bestTune
```

```
##   nprune degree
## 7     14      1
```




```
## earth variable importance
##
##          Overall
## log(ALP)  100.00
## DB_TB      87.20
## log(ALT)   87.20
## Age        61.30
## G          11.68
## log(AST)    0.00
## Male        0.00
## A           0.00
```

Similar to the random forest model, the variable of greatest (scaled) importance is *log(ALP)*, while in addition to *Male*, *log(AST)* and *A* also have a value of 0.



The earth model shows the highest AUC (0.8022) compared to the other three previous models, indicating a stronger classifier across different probability threshold values.

```
y_hat_e <- predict(fit_e, newdata = test_set)
cm_e <- confusionMatrix(data = factor(y_hat_e), reference = factor(test_set$Disease),
                        positive = "Yes", mode="everything")

cm_e
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  27  23
##        Yes  7  61
##
##                Accuracy : 0.7458
##                  95% CI : (0.6574, 0.8214)
##     No Information Rate : 0.7119
##     P-Value [Acc > NIR] : 0.24070
##
##                   Kappa : 0.4564
##
##  Mcnemar's Test P-Value : 0.00617
##
##             Sensitivity : 0.7262
##             Specificity : 0.7941
##          Pos Pred Value : 0.8971
##          Neg Pred Value : 0.5400
##               Precision : 0.8971
##                  Recall : 0.7262
##                      F1 : 0.8026
##              Prevalence : 0.7119
##          Detection Rate : 0.5169
##    Detection Prevalence : 0.5763
##       Balanced Accuracy : 0.7602
##
##        'Positive' Class : Yes
##
```

Model 5 - Ensemble (Final Model) In order to further improve performance, machine learning algorithmns can be combined to form an ensemble, or meta-model, with an optimal predictive accuracy, better capturing the underlying distribution of the data. The *caretEnsemble* package is used to create a simple robust linear blend of two models, where tuning parameters are selected which maximise accuracy. As in previous models, k-fold cross validation is performed ($k = 5$):

```
ctrl_en <- trainControl(method = "cv", number = 5, savePredictions = "final",
                        classProbs = TRUE)
models <- c("knn", "earth")
set.seed(1, sample.kind = "Rounding")
fit_en <- caretList(Disease ~ Age + A + G + DB_TB + Male + log(ALP) + log(ALT) + log(AST),
                    data = up_train, methodList = models,
                    trControl = ctrl_en)

stackControl <- trainControl(method = "cv", number = 5, classProbs = TRUE,
                             summaryFunction = twoClassSummary)
greedy_ensemble <- caretEnsemble(fit_en, trControl = stackControl, metric="ROC")
greedy_ensemble$models
```

```
## $knn
## k-Nearest Neighbors
##
## 664 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 531, 530, 532, 532, 531
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.6747478  0.3492794
##   7  0.6897859  0.3794159
##   9  0.6943536  0.3885857
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
##
## $earth
## Multivariate Adaptive Regression Spline
##
## 664 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 531, 530, 532, 532, 531
## Resampling results across tuning parameters:
##
##   nprune  Accuracy   Kappa
##    2      0.6401371  0.2802538
##    8      0.6837712  0.3673113
##   14      0.6911995  0.3821837
```

```
##
## Tuning parameter 'degree' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nprune = 14 and degree = 1.
##
## attr(,"class")
## [1] "caretList"
```

```
varImp(greedy_ensemble)
```

```
##            overall       knn     earth
## Male      0.000000  0.000000  0.000000
## log(AST)  0.000000  0.000000  0.000000
## A         3.046029  6.519186  0.000000
## G         5.399490  7.721879  3.362708
## ALP       9.223849 19.741107  0.000000
## ALT      10.234994 21.905184  0.000000
## AST      11.078515 23.710507  0.000000
## Age      11.267365  3.994429 17.645878
## log(ALT) 13.373717  0.000000 25.102739
## log(ALP) 15.335974  0.000000 28.785935
## DB_TB    21.040067 16.407709 25.102739
```

Evaluating variable importance shows *DB_TB* to be the predictor most frequently used while both *Male* and *log(AST)* have a (scaled) value of 0.

The kNN and earth algorithmns are selected for the ensemble as they both have low-moderate correlation compared to the other models. Lower correlation between the predictions of the ensembled models is important in increasing the error-correcting capability of the ensemble, effectively combining the different 'skills' in forming a stronger model:

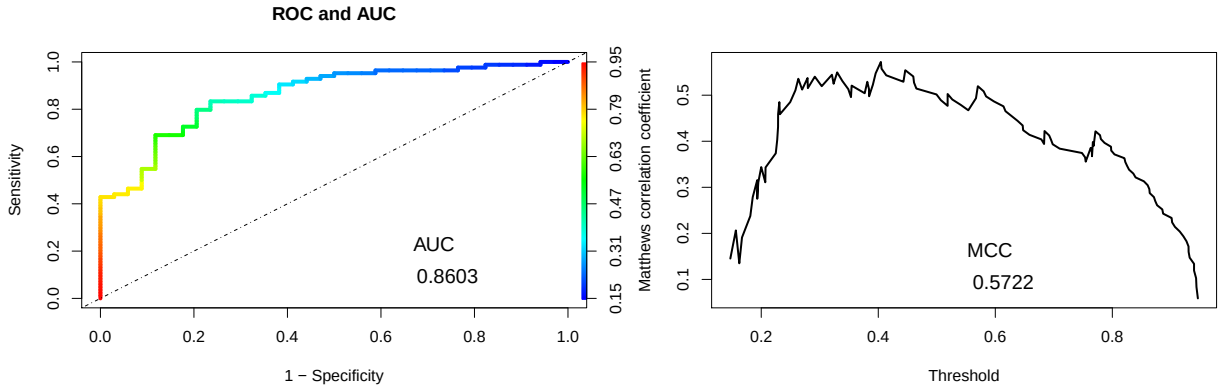
```
modelCor(resamples(fit_en))
```

```
##             knn     earth
## knn   1.0000000 0.3856034
## earth 0.3856034 1.0000000
```

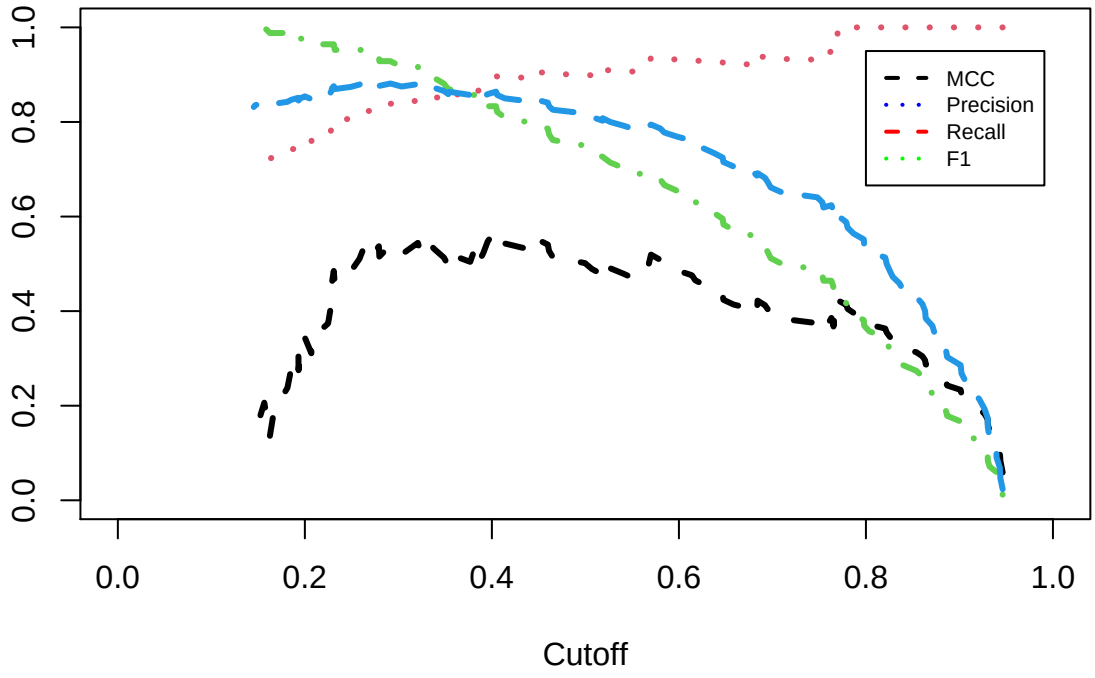
The AUC can then be reported comparing each model alongside the combined ensemble:

```
caTools::colAUC(pred, test_set$Disease)
```

```
##                  knn     earth  ensemble
## No vs. Yes 0.8179272 0.8021709 0.8602941
```



The graphs shows an increased AUC (0.8603) and MCC (0.5722) of the ensemble relative to the previous individual classifiers.



The graph displays the precision, recall, F1 and MCC for different probabilistic threshold values. If the same weight of importance is given to precision as recall in classifying a patient, then the intersection implies a better threshold value would be one at ~0.4, where MCC also appears to be maximised. In this case, if $p > 0.4$ then the patient would be classified as having liver disease.

As with previous models, a confusion matrix can be created assuming a threshold value of 0.5 forming predictions against the test set:

```
y_hat_ensemble <- predict(greedy_ensemble, newdata = test_set)
cm_ensemble <- confusionMatrix(data = factor(y_hat_ensemble),
                               reference = factor(test_set$Disease),
                               positive = "Yes", mode="everything")
cm_ensemble
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  27  22
##        Yes  7  62
##
##                Accuracy : 0.7542
##                  95% CI : (0.6665, 0.8288)
##     No Information Rate : 0.7119
##     P-Value [Acc > NIR] : 0.18078
```

```
##
##                   Kappa : 0.4704
##
##  Mcnemar's Test P-Value : 0.00933
##
##             Sensitivity : 0.7381
##             Specificity : 0.7941
##          Pos Pred Value : 0.8986
##          Neg Pred Value : 0.5510
##               Precision : 0.8986
##                  Recall : 0.7381
##                      F1 : 0.8105
##              Prevalence : 0.7119
##          Detection Rate : 0.5254
##    Detection Prevalence : 0.5847
##       Balanced Accuracy : 0.7661
##
##        'Positive' Class : Yes
##
```

The values for key metrics within the confusion matrix are either the same or greater than the previous individual earth model, indicating an improved predictive algorithmn.

Results

The below table displays a summary of the results with corresponding metrics for each of the different models tested:

	Logistic	kNN	Random Forest	Earth	Ensemble
AUC	0.7885000	0.7405000	0.7861000	0.8022000	0.8603000
MCC	0.5403000	0.3625000	0.4648000	0.5155000	0.5722000
Sensitivity	0.7142857	0.5595238	0.8452381	0.7261905	0.7380952
Specificity	0.8529412	0.7352941	0.4705882	0.7941176	0.7941176
Pos Pred Value	0.9230769	0.8392857	0.7977528	0.8970588	0.8985507
Neg Pred Value	0.5471698	0.4032258	0.5517241	0.5400000	0.5510204
Precision	0.9230769	0.8392857	0.7977528	0.8970588	0.8985507
Recall	0.7142857	0.5595238	0.8452381	0.7261905	0.7380952
F1	0.8053691	0.6714286	0.8208092	0.8026316	0.8104575
Prevalence	0.7118644	0.7118644	0.7118644	0.7118644	0.7118644
Detection Rate	0.5084746	0.3983051	0.6016949	0.5169492	0.5254237
Detection Prevalence	0.5508475	0.4745763	0.7542373	0.5762712	0.5847458
Balanced Accuracy	0.7836134	0.6474090	0.6579132	0.7601541	0.7661064

Considering the individual classifiers, the logistic regression reveals a higher accuracy (0.7542), specificity (0.8529) and precision (0.9231), although this comes with a slightly lower relative sensitivity (0.7143). If measuring only on sensitivity, then the random forest model shows the greatest ability to correctly classify actual positive outcomes with a value of 0.8452. Combining kNN and earth models to form an ensemble shows the highest measure concerning AUC (0.8603), MCC (0.5722), as well as accuracy (0.7542), demonstrating an improved classifier relative to both individual models.

Conclusion

The objective of this report is to develop an effective machine learning model to predict liver disease. A data set from an Indian sub-population of both patients with and without the disease is used, containing a number of predictive features. The data is first prepared for analysis, to then be partitioned into both

training and test sets, where tuning parameters can be selected using the training set and predictions then made against the separate test set. The performance of each model can then be assessed using a number of different metrics derived from the confusion matrix.

With the purpose of detecting disease in a patient population, of greater concern is being able to identify those with the disease, an actual positive outcome in this setting, compared to those without the disease, an actual negative outcome. If patients are misdiagnosed, they will go untreated, which poses the greater risk compared to falsely diagnosing a healthy individual. Considered a rather more informative measure, evaluating classification performance concerning true positive rate (TPR) versus false positive rate (FPR) for different probabilistic thresholds, the highest AUC value (0.8603) was given when combining kNN and earth models to create an ensemble. In addition, the ensemble delivered the highest MMC value (0.5722), which is regarded a robust metric, taking into account true and false positives and negatives. Regarding sensitivity, the ensemble is only inferior to the random forest model, however this model performs too poorly when attempting to classify negative outcomes. As misdiagnosing a healthy individual still carries a significant cost to the healthcare system in treating a healthy patient, overall accuracy and specificity are still of concern. In this instance, the ensemble displays the (joint) highest accuracy (0.7542), and a specificity which is only inferior to the logistic regression (0.7941).

Furthermore, the final ensemble model reveals that if the same weight of importance is given to recall and precision, their intersection together with the F1 curve, implies an optimal cutoff ~0.4 as opposed to the default of 0.5. It is at this lower cutoff where MCC also appears to be maximised. In this case, if $p > 0.4$ then the algorithmn would predict liver disease in a patient. Intuitively, lowering the threshold increases the liklihood of capturing more positive outcomes, that is patients with disease, although not too far as to incur the greater cost of increased false positives.

Interpretation of coefficient estimates from the logistic regression as well as variable importance measures of subsequent models, delivered insights as to the strength of particular predictive features. Within this analysis, *Male* seemed to carry no predictive power of whether a patient had liver disease, while *log(AST)* (aspartate aminotransferase) also had a (scaled) variable importance value of 0 in the final ensemble model.

While this report presents interesting findings, handling a relatively small sub-population sample of individuals may pose a limitation on any insights drawn. Although considered a valid technique, the use of up-sampling to address class imbalance in the original data set by randomly sampling the minority class, can pose a risk of over-fitting the model. This up-sampling method however, is only applied to the training set to prevent any 'leakage' to the test set, and any downside can be considered outweighed by the risk of a biased classifier from class imbalance in the original data set (71.4% liver disease, 28.6% healthy).

With the burden of chronic illnesses such as liver disease only increasing at a global level, developing a predictive algorithmn to identify patients requiring treatment is of increasing significance. A model such as this, using mostly clinical blood test results has a wide degree of application and allows for extension to other areas of work in a clinical setting. Further models could be developed using blood or enzyme markers, to predict for example organ failure or the onset of various different cancers. Given the available data, such predictive features could be used to facilitate early intervention in correctly diagnosing patients with disease, while also alleviating the strain on healthcare systems.

References

1. Asrani, S.K. et al. 'Burden of Liver Diseases in the World'. *Journal of Hepatology.* 2019, 70(1), pp.151-171.
2. Krishnan, S. 'Liver Diseases - An Overview'. *World Journal of Pharmacy and Pharmaceutical Sciences.* 2019, 8(1), pp.1385-1395.