Predicting Liver Disease

Martin J Page

15/06/2020

EXECUTIVE SUMMARY

The liver is an essential organ for digesting food and ridding the body of toxic substances. Liver disease is an increasing medical problem. This relates to factors such as excessive consumption of alcohol, inhalation of harmful gases, intake of contaminated food, pickles and drugs. Although there are many causes of liver disease and treatment will depend on the cause, quicker and earlier identification of liver dysfunction will improve the delivery of healthcare to patients and reduce the burden on doctors.

The Data

The data for this project come from the Indian Liver Patient Dataset curated on Kaggle and now stored on this project's GitHub repository for convenient access. The dataset contains 416 liver patient records and 167 non-liver patient records collected from North East of Andhra Pradesh, India. The data include two demographic variables - the (1) age and (2) gender of the subjects - and 8 medical measurements relating to liver function - (1) total bilirubin, (2) direct bilirubin, (3) alkaline phosphotase, (4) alamine aminotransferase, (5) aspartate aminotransferase, (6) total protiens, (7) albumin, and (8) albumin and globulin ratio. Any patient older than 89 year is recorded as being age 90.

Project Goal

This project aims to build a prediction algorithm that can classify subjects as liver disease patients or non-liver disease patients based on the features in this dataset.

Approach

After formatting the data, the a validation dataset is taken out and the remaining data are split into a training and a testing set. Exploratory data analysis is performed on the training set, after which we build and test 7 classification algorithms and an ensemble model. First, we consider three linear-based models: a logistic regression model, a generalised additive model (GAM) and linear discriminant analysis (LDA). Next we consider K-nearest neighbour (KNN). We then test three decision tree models: Rpart, random forest (RF) and gradient boosted regression trees (GBM). Finally, we aggregate the predictions of the 7 models by a majority vote in an ensemble model.

METHODS

Loading the Data

The dataset is loaded from a CSV file stored on GitHub.

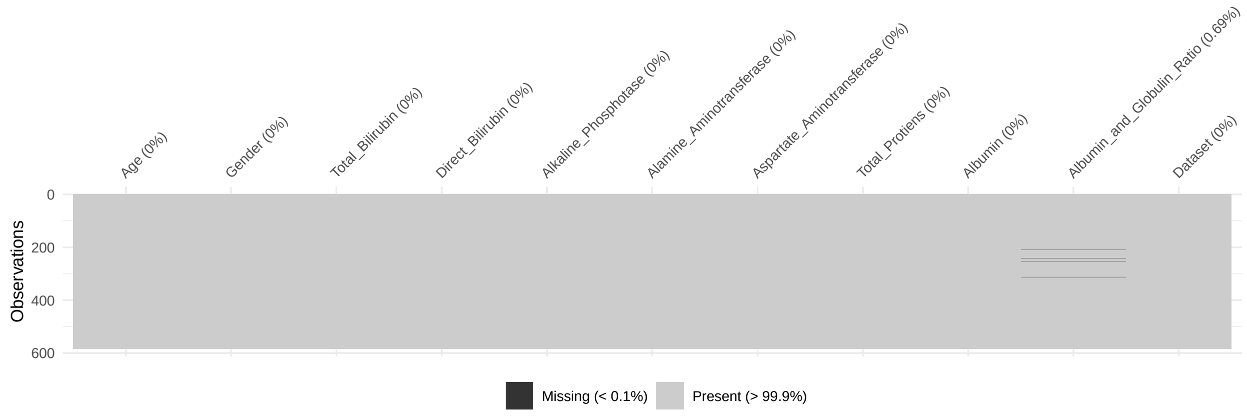
```r
url <- "https://raw.githubusercontent.com/martinjpage/HarvardCapstone/master/indian_liver_patient.csv"
liver <- read.csv(url)
```

Formatting the Data

The gender variable and the outcome (patient or control) are coerced to factors as they are categorical variables. The function `vis_miss()` from the `visdat` package is used to visualise any missing data. The rows containing missing values are removed from the dataset. As only a very small amount of the data are missing, removing the data is not expected to introduce any bias. Therefore, no imputation strategy was applied.

```r
liver$Gender <- factor(liver$Gender)
liver$Dataset <- factor(liver$Dataset, levels = c(1,2), labels = c("Patient", "Control"))

vis_miss(liver)
```



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

Partitioning the Data

A `validation` dataset is created by removing 10% of the observations from the main data frame (`liver`). The remaining data (`modelling`) are partitioned into a `training` (90%) and `testing` (10%) set. The training set is used to build the models and the testing set is used for intermediate evaluation of the models, which informs decisions and tuning for model building. The validation set is used only once to evaluate the final models. The dataset is a modest size, so removing 10% for validation and 10% for testing while leaving 80% for model building was deemed appropriate and follows the 80/20 split convention.

```r
set.seed(1)
inValid <- createDataPartition(y = liver$Dataset, times = 1, p = 0.1, list = FALSE)
validation <- liver[inValid,]
modelling <- liver[-inValid,]
set.seed(2)
InTrain <- createDataPartition(y = modelling$Dataset, times = 1, p = 0.9, list = FALSE)
training <- modelling[InTrain,]
testing <- modelling[-InTrain,]
dim(validation); dim(training); dim(testing)
```

```
[1] 59 11
```

```
[1] 469  11
```

```
[1] 51 11
```

Exploratory Data Analysis

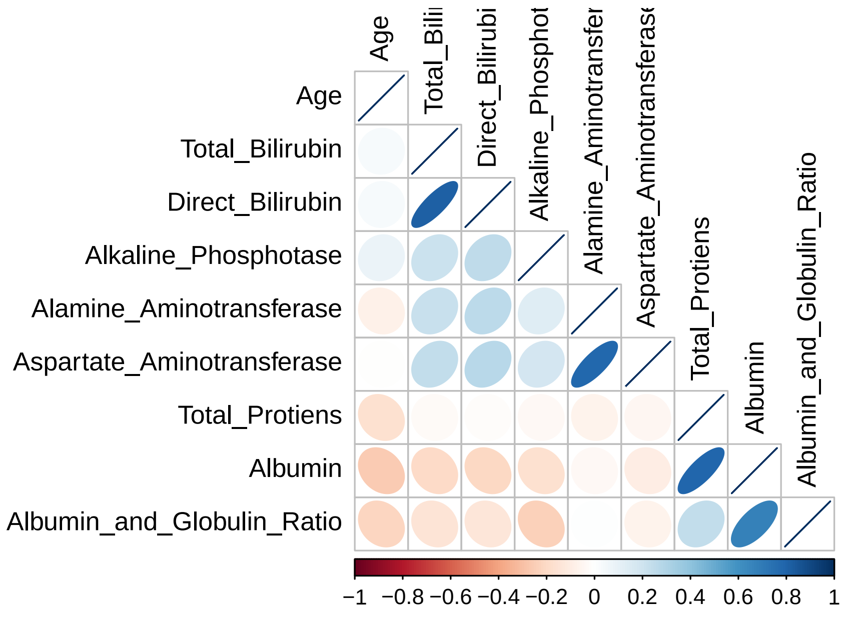
Variables with Low Variance Variables with low variability will not make useful features for prediction. We evaluate whether any of the features in the training dataset exhibit near zero variance. None of the variables is returned, so we keep all the variables in the dataset.

```
nsv <- nearZeroVar(training, saveMetrics = TRUE)
sum(nsv$nzv == TRUE)
```

```
[1] 0
```

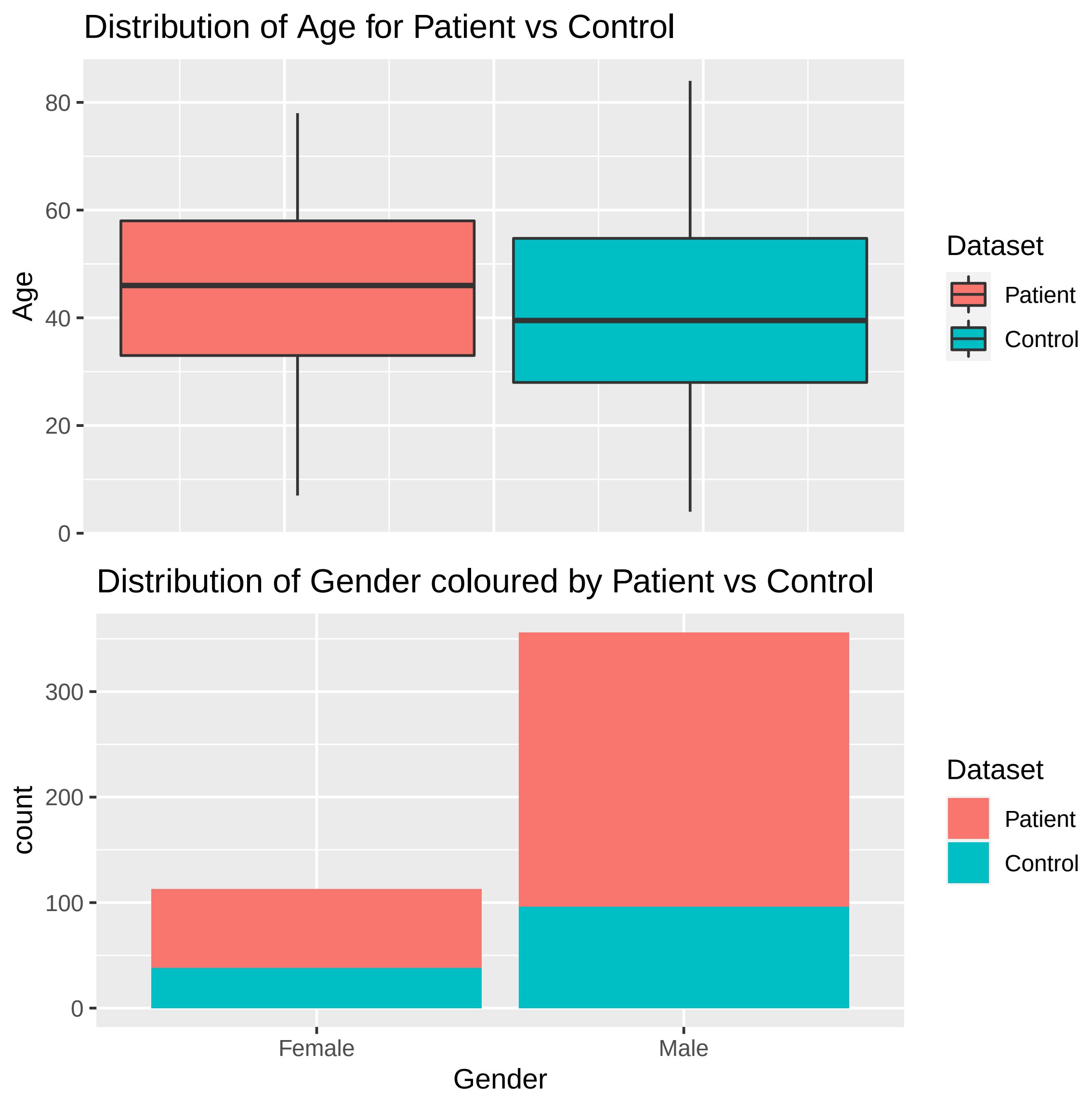
Correlation between predictors The function `corrplot()` from the `corrplot` package is used plot the correlation matrix of all the numerical variables in the dataset to get a sense of how the features are related to each other.

```
as.matrix(cor(training[,c(-2,-11)])) %>%
    corrplot(method = "ellipse", tl.col = "black", type = "lower")
```



Demographics: Age and Gender Both outcome groups appear to have similar age distributions. Having age-matched groups can be important for biological experiments as age can affect biological functioning. On the other hand, the data are not balanced for sex. There are more males included in the study. This imbalance should be consider when evaluating the model, as male-specific values might be overrepresented and influence the prediction.

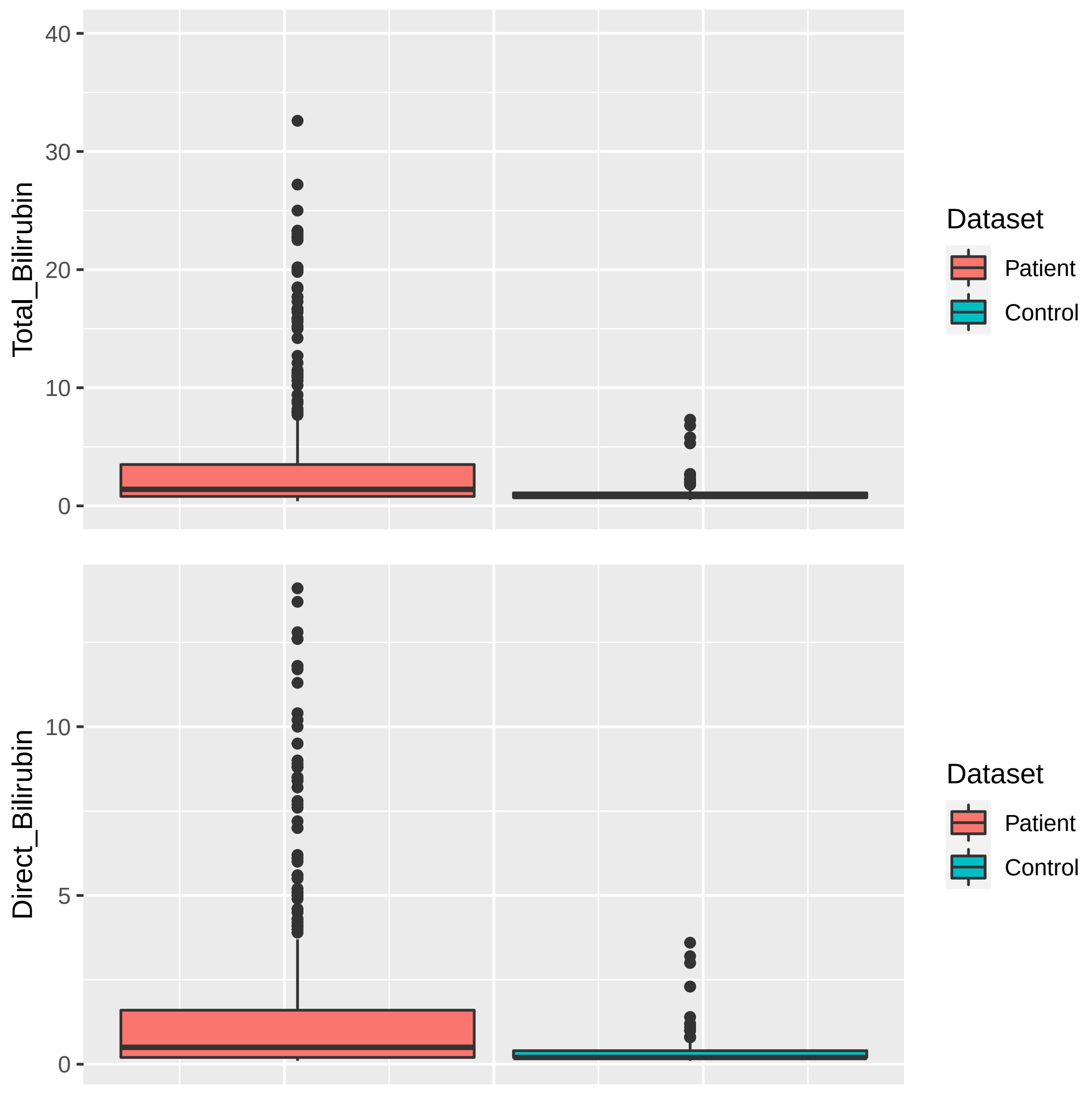
```
p1 <- training %>% ggplot(aes(y = Age, fill = Dataset)) + geom_boxplot() +
        theme(axis.title.x = element_blank(),
         axis.text.x = element_blank(), axis.ticks.x = element_blank()) +
        ggtitle("Distribution of Age for Patient vs Control")
p2 <- training %>% ggplot(aes(x = Gender, fill = Dataset)) + geom_bar() +
        ggtitle("Distribution of Gender coloured by Patient vs Control")
grid.arrange(p1, p2, ncol = 1)
```



Bilirubin Measurements Total and direct bilirubin appear to be elevated in the patient group relative to the control group. The patient group also contains more extreme values that lay outside 1.5× the interquartile range limit set for outliers. These measurements might be good metric on which to separate the two groups.

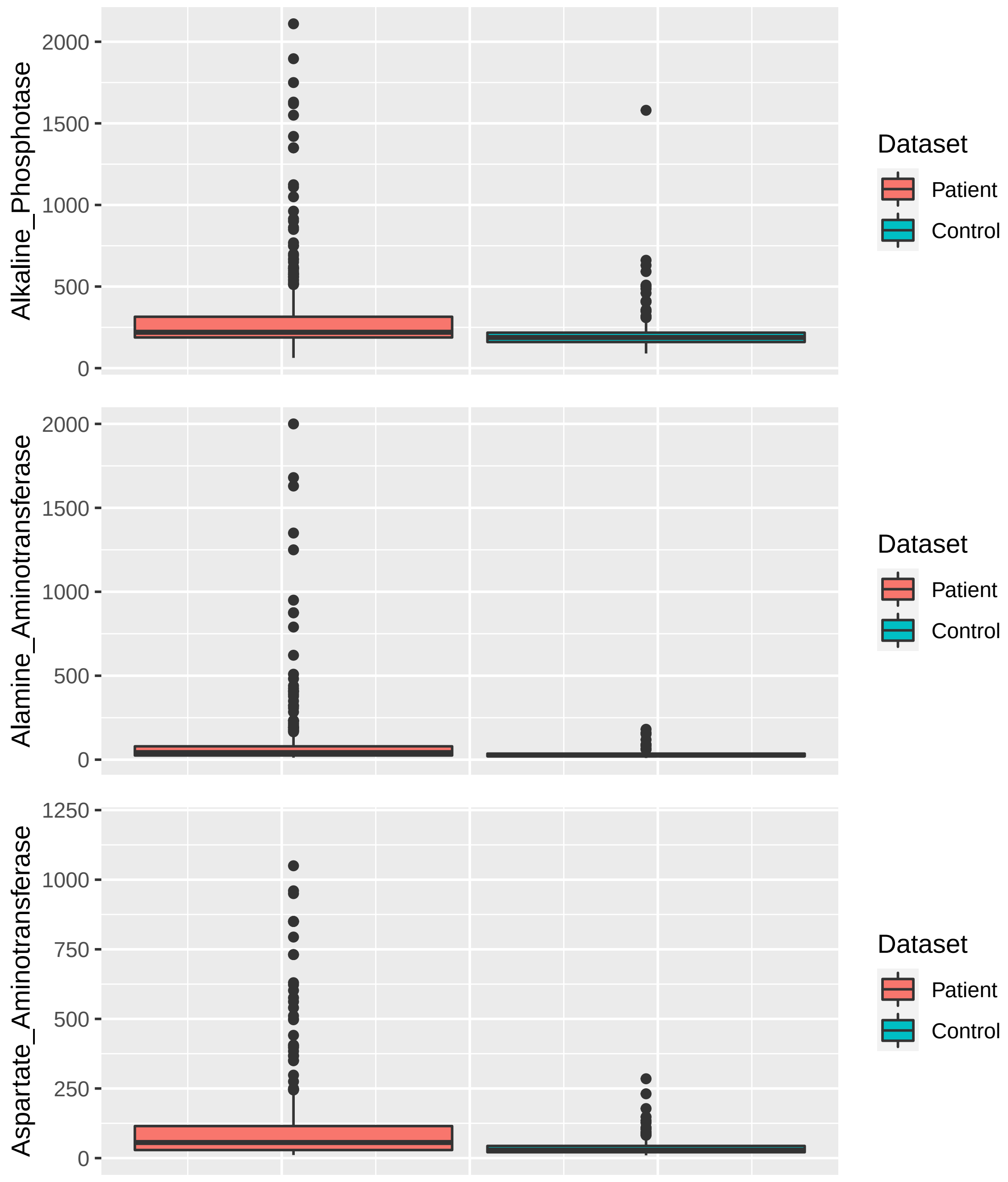
```
p3 <- training %>% ggplot(aes(y = Total_Bilirubin, fill = Dataset)) + geom_boxplot() +
        coord_cartesian(ylim = c(0, 40)) + theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank())
```

```
p4 <- training %>% ggplot(aes(y = Direct_Bilirubin, fill = Dataset)) + geom_boxplot() +
         theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
         axis.ticks.x = element_blank())
grid.arrange(p3, p4, ncol = 1)
```



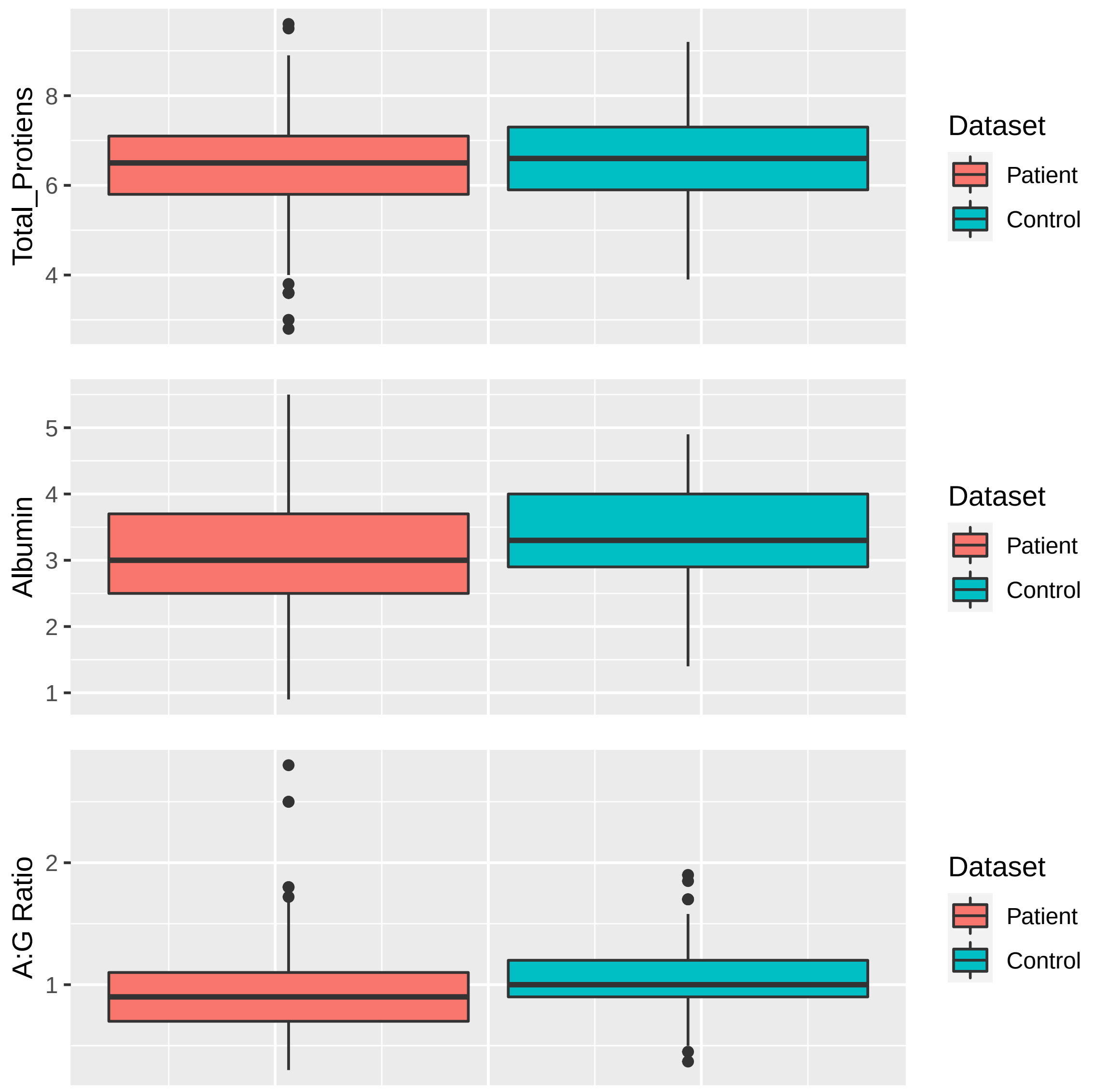
Enzyme Measurements The three enzymes appear to be elevated in the patient group, including having more extreme values (outliers). These values might also offer a good metric to separate the two groups.

```
p5 <- training %>% ggplot(aes(y = Alkaline_Phosphotase, fill = Dataset)) + geom_boxplot()+
        theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
p6 <- training %>% ggplot(aes(y = Alamine_Aminotransferase, fill = Dataset)) +
        geom_boxplot()+ theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank())
p7 <- training %>% ggplot(aes(y = Aspartate_Aminotransferase, fill = Dataset)) +
        geom_boxplot() + coord_cartesian(ylim = c(0, 1200))+ theme(axis.title.x =
        element_blank(), axis.text.x = element_blank(), axis.ticks.x = element_blank())
grid.arrange(p5, p6, p7, ncol = 1)
```



Protein Measurements The total protein content, the albumin protein level and the albumin to globulin protein ratio seem to overlap between the two groups and might not be strong metrics upon which to separate patients from control subjects.

```
p8 <- training %>% ggplot(aes(y = Total_Protiens, fill = Dataset)) + geom_boxplot() +
        theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
p9 <- training %>% ggplot(aes(y = Albumin, fill = Dataset)) + geom_boxplot() +
        theme(axis.title.x = element_blank(), axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
p10 <- training %>% ggplot(aes(y = Albumin_and_Globulin_Ratio, fill = Dataset)) +
        geom_boxplot() + ylab("A:G Ratio") + theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(), axis.ticks.x = element_blank())
grid.arrange(p8, p9, p10, ncol = 1)
```



Building Models

Models are built using 10-fold cross validation. This method randomly splits the training observations into 10 non-overlapping sets. The estimated error is calculated for each set and the average estimated error is obtained. The model selects the parameter that minimises the estimated error.

```
control <-  trainControl(method = "cv", number = 10, p = 0.9, allowParallel = TRUE)
```

Model 1: Logistic Regression Model Logistic regression in a generalised linear model for binary outcomes using the logit transformation. The coefficients of the model need to be exponentiated to appear on the original scale. Regression is a good baseline approach and has the advantage of being interpretable. However, this model cannot capture potential non-linearity in the data. The model was built using all the regressors and the intercept term is removed.

```
glmMod <-  train(Dataset ~ 0 + .,
                 data = training, method = "glm", family = "binomial")
p_glm <- predict(glmMod, newdata = testing)
glmAcc <- confusionMatrix(p_glm,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- tibble(Method = "Logistic Regression", Accuracy = round(glmAcc, 3))
```

Model 2: GAM GamLoess is a bascially a type of bin smoother. Essentially, loess fits a line locally in a specific window, but the overall fit (connecting the local lines) can take on non-linear shapes. Loess keeps the number of points used in the local fit the same. Different spans, the size of the neighbourhood as a proportion, are tried as a tuning parameter. Only lines (`degree = 1`) are tested.

```
set.seed(2)
gamMod <- train(Dataset ~ ., data = training, method = "gamLoess",
                trControl = control,
                tuneGrid = expand.grid(span = seq(0.15, 0.65, length = 10), degree = 1))
p_gam <- predict(gamMod, newdata = testing)
gamAcc <- confusionMatrix(p_gam,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "GAM Model", Accuracy = round(gamAcc,3)))
```

Model 3: LDA Linear discriminant analysis is a form of Naive Bayes with the additional assumption that all the predictors share the same standard deviations. Naive Bayes assumes that the predictor variables are independent of each other. LDA essentially draws lines through the covariate space assuming that the features have multivariate Gaussian distributions within each class and that the same covariate matrix can be used for every class. LDA, like simple regression is not able to capture non-linearity. LDA should only be used with a modest number of predictors.

```
ldaMod <- train(Dataset ~ ., data = training, method = "lda",
                trControl = control)
p_lda <- predict(ldaMod, newdata = testing)
ldaAcc <- confusionMatrix(p_lda,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "LDA Model", Accuracy = round(ldaAcc,3)))
```

Model 4: KNN K-nearest neighbours is similar to bin smoothing but can adapt better to multiple dimensions. It classifies objects based on closest training examples in the feature space. Different values of k, the number of nearest points, are tried as a tuning parameter.

```r
set.seed(4)
knnMod <- train(Dataset ~ ., data = training, method = "knn",
                trControl = control,
                tuneGrid = data.frame(k = seq(3,21,2)))

p_knn <- predict(knnMod, newdata = testing)
knnAcc <- confusionMatrix(p_knn,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "KNN Model", Accuracy = round(knnAcc,3)))
```

Model 5: Rpart Decision trees form predictions by calculating which class is the most common among the observations and partitioning the observations into subsets that have the same class (purity). A range of complexity parameters (cp), the minimum improvement in the model needed at each node, are tried

```r
rpartMod <- train(Dataset ~ ., data = training, method = "rpart",
                  trControl = control,
                  tuneGrid = data.frame(cp = seq(0, 0.1, 0.01)))
p_rpart <- predict(rpartMod, newdata = testing)
rpartAcc <- confusionMatrix(p_rpart,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "Rpart Model", Accuracy = round(rpartAcc,3)))
```

Model 6: Random Forest Random forest attempts to improve the prediction performance of decision trees and to reduce instability by averaging multiple decision trees. We set the model to randomly build 100 trees and test different mtry values, the number of variables randomly sampled as candidates at each split.

```r
set.seed(6)
rfMod <- train(Dataset ~ ., data = training, method = "rf",
               trControl = control, ntree = 100,
               tuneGrid = data.frame(mtry = seq(1, 200, 25)))
p_rf <- predict(rfMod, newdata = testing)
rfAcc <- confusionMatrix(p_rf,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "Random Forest Model", Accuracy = round(rfAcc,3)))
```

Model 7: GBM Boosting is a method that essentially builds an ensemble of weak prediction models. It fits decision trees in a iterative manner, where it weighs and adds up lots of possibly weak predictors to get a strong predictor. We test different numbers of trees to fit, number of splits in each tree, levels of regularisation by shrinkage (learning rate) and the number of observations in each leaf (sub-sampling).

```r
set.seed(8)
gbmMod <- train(Dataset ~ ., data = training, method = "gbm", verbose = FALSE,
                trControl = control,
                tuneGrid = data.frame(expand.grid(n.trees = c(50, 100, 250),
                interaction.depth = seq(1,10, length.out = 3), shrinkage = c(.01, .1, .3),
                n.minobsinnode = c(5, 10, 15))))
p_gbm <- predict(gbmMod, newdata = testing)
```

```r
gbmAcc <- confusionMatrix(p_gbm,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "GBM Model", Accuracy = round(gbmAcc,3)))
```

Model 8: Ensemble by Majority Vote The ensemble model takes the predictions of the 7 individual models and for each observation counts the number of predictions for the liver patient group and the number of predictions for the non-liver patient group. Based on a majority vote, a final prediction is chosen for each observation. This is a way to play on the advantages and disadvantages of each model by aggregating the predictions.

```r
p_all <- tibble(p_glm, p_gam, p_lda, p_knn, p_rpart, p_rf, p_gbm)
ens_list <- apply(p_all,1,table)
ens_vote <- sapply(ens_list, which.max)
p_ens <- factor(names(ens_vote))
ensAcc <- confusionMatrix(p_ens,
                          testing$Dataset)$overall["Accuracy"]
acc_results <- bind_rows(acc_results,
            tibble(Method = "Ensemble Model", Accuracy = round(ensAcc,3)))
```

Validation

The models are evaluated for accuracy using the validation set, which has not been used during any stage of model development.

```r
pv_glm <- predict(glmMod, newdata = validation)
pv_gam <- predict(gamMod, newdata = validation)
pv_lda <- predict(ldaMod, newdata = validation)
pv_knn <- predict(knnMod, newdata = validation)
pv_rpart <- predict(rpartMod, newdata = validation)
pv_rf <- predict(rfMod, newdata = validation)
pv_gbm <- predict(gbmMod, newdata = validation)
pv_all <- tibble(pv_glm, pv_gam, pv_lda, pv_knn, pv_rpart, pv_rf, pv_gbm)

ensv_list <- apply(pv_all, 1, table)
ensv_vote <- sapply(ensv_list, which.max)
pv_ens <- factor(names(ensv_vote))
ensvAcc <- confusionMatrix(pv_ens,
                          validation$Dataset)$overall["Accuracy"]
pv_all <- bind_cols(pv_all, tibble(pv_ens))
accv_results <- sapply(pv_all, function(p) {
    round(confusionMatrix(validation$Dataset, p)$overall["Accuracy"],3)})

final_results <- bind_cols(acc_results, tibble(Validation = unname(accv_results)))
```

RESULTS

```r
final_results %>% knitr::kable()
```

Method	Accuracy	Validation
Logistic Regression	0.706	0.780
GAM Model	0.667	0.746
LDA Model	0.745	0.729
KNN Model	0.667	0.712
Rpart Model	0.725	0.712
Random Forest Model	0.667	0.712
GBM Model	0.745	0.746
Ensemble Model	0.706	0.729

Evaluation on the testing set produced accuracy values ranging from 67% to 75%, with the GBM model ranked first. However, the testing set was also used to tune the model. Therefore, the validation set is used for the final evaluation of the performance of the models. Most of the models in fact performed better on the validation set. For the validation test, the logistic regression model performed the best with 78% accuracy. The GBM model also performed well on the validation set and tied in second rank with the GAM model. However, the GBM model is consistent between the testing and validation sets, whereas the GAM model performed poorly on the testing set. The ensemble model was build using 7 individual algorithms, an odd number so that the voting would always be based on an exact majority. However, the ensemble model provided only average performance.

Based on the random forest model, the importance of each variable is ranked below. Age as well as the measurements of the enzymes rank as the most important features in the random forest model.

```r
data.frame(rfMod$finalModel$importance) %>% arrange(desc(MeanDecreaseGini))
```

```
                           MeanDecreaseGini
Alamine_Aminotransferase          28.498329
Age                               27.739539
Alkaline_Phosphotase              27.445311
Aspartate_Aminotransferase        25.823954
Total_Protiens                    16.927521
Albumin                           16.807730
Total_Bilirubin                   16.730059
Direct_Bilirubin                  14.322287
Albumin_and_Globulin_Ratio        12.610480
GenderMale                         3.400719
```

Quickly looking at the sensitivity and specificity of the GBM and logistic regression models reveal that both models have high sensitivity with the logistic regression model outperforming the GBM model by about 5%. However, the specificity is quite low for both models.

```r
gbmMat <- confusionMatrix(pv_gbm, validation$Dataset)
glmMat <- confusionMatrix(pv_glm, validation$Dataset)

gbmMat$byClass[c(1,2)] %>% cbind(glmMat$byClass[c(1,2)]) %>%
     knitr::kable(col.names = c("GBM Model", "Logistic Regression Model"))
```

	GBM Model	Logistic Regression Model
Sensitivity	0.9285714	0.9761905
Specificity	0.2941176	0.2941176

CONCLUSION

7 models and an ensemble that aggregates the predictions of the individual models based on majority vote were built to classify patients into liver disease and non-liver disease groups. The highest performing model on the testing set, the GBM model performed similarly on the validation set with 75% accuracy. The logistic regression model was the best performing model on the final evaluation with the validation set at 78% accuracy. Considering the better interpretability of regression models, the logistic regression model should be favoured over the more complex GBM model even though the prediction accuracy is fairly similar. This project does not consider the sensitivity and specificity in detail; however, both the GBM and logistic regression models have high sensitivity and low specificity. This might be suitable for a screening method with the intention to refer as many positive cases as possible for further examination. However, the low specificity might impact the practical application of these models if too many patients are referred for further examination (based on the high catch-all sensitivity, which is good at identifying the true positives) compared to the number who actual have liver problems (based on the low specificity (true negative rate), which suggests many false positive referrals).

Potential Impact

One of the challenges with treating liver patients is that liver problems are not easily discovered in an early stage when damage is only partial. However, treatment during the early stage of disease typically results in better outcomes for the patients. Early diagnosis of liver problems will therefore increase the patient survival rate. In this context, automatic classification algorithms can assist doctors in making medical diagnoses, reducing the burden on doctors as well as enable earlier and effective treatment of patients.

Limitations

469 observations were used to train the models in the project. Although this is a fair number of records, especially for clinical patient data, more observations might improve the training of the algorithms. Also, liver disease is not homogeneous - there are many different causes. Patients will also be at different stages of the disease (early vs late) and have different severities (mild vs severe). Patients may also have co-morbidities that complicate their medical profiles. Not knowing the exact description of the clinical sample, such as the inclusion and exclusion criteria of the patients, makes describing the exact application of the model (such as on which type of patient it is calibrated and potential underlying influences on the data) difficult. Additionally, the gender imbalance in the records might influence the models. In terms of the model building approach, scaling the data before building the models might remove strongly biased predictors as well as predictors with high variability and therefore improve model performance.

Future Work

Future work should apply these models to similar datasets from different patient populations such as from different countries and ethnic backgrounds. Patients from different countries might have a difference in classifier performance, whether because of physiological differences, socio-cultural factors or different hospital standards and protocols. Future work could also consider whether other biomarkers of liver function are available that could offer additional features for prediction. Future work could also evaluate the model not based only on overall accuracy but also on sensitivity and specificity. For an early detection screening method, sensitivity (i.e., testing positive when the disease is positive) would generally be favoured over specificity. Lastly, the model could be expanded from predicting a binary outcome of patient and non-patient, to predicting potential severity of the disease if information on the clinical stage where added to the dataset.

References:
1. *https://archive.ics.uci.edu/ml/datasets/ILPD+(Indian+Liver+Patient+Dataset)*
2. *https://www.kaggle.com/uciml/indian-liver-patient-records*