Liver Disease Report

Spencer Fisher

12/20/2020

Introduction

I created a machine learning model that predicts whether or not an individual has liver disease based on ten different factors, age, gender, total bilirubin, direct bilirubin, alkaline phosphatase, alamine aminotransferase, aspartate aminotransferase, total proteins, albumin, and albumin and globulin ratio. The dataset is called Indian Liver Patient Record and is from Kaggle. The dataset has data on 579 subjects and eleven columns, ten representing the factors previously mentioned and one that represents whether or not the patient has liver disease.

```
head(dat)
```

```
##   Age Gender Total_Bilirubin Direct_Bilirubin Alkaline_Phosphotase
## 1  65 Female             0.7              0.1                  187
## 2  62   Male            10.9              5.5                  699
## 3  62   Male             7.3              4.1                  490
## 4  58   Male             1.0              0.4                  182
## 5  72   Male             3.9              2.0                  195
## 6  46   Male             1.8              0.7                  208
##   Alamine_Aminotransferase Aspartate_Aminotransferase Total_Protiens Albumin
## 1                       16                         18            6.8     3.3
## 2                       64                        100            7.5     3.2
## 3                       60                         68            7.0     3.3
## 4                       14                         20            6.8     3.4
## 5                       27                         59            7.3     2.4
## 6                       19                         14            7.6     4.4
##   Albumin_and_Globulin_Ratio Dataset
## 1                       0.90       1
## 2                       0.74       1
## 3                       0.89       1
## 4                       1.00       1
## 5                       0.40       1
## 6                       1.30       1
```

Method/Analysis

To make my model, before I could start, I had to first both download and understand the data. Downloading the data was somewhat of a daunting task since I was unable to download the data straight from the website in my r-script. With the Kaggle command line tool I downloaded this data to my harddrive and then uploaded it to my google drive. This way if anyone runs my R script it will download the data without having to have credentials to Kaggle. Once I had the data downloaded, I performed some data cleaning.

First, I had to remove all of the NAs in the data so that it didn't throw off my models. Second, I noticed that instead of using 1s and 0s to represent whether or not the patient had liver disease, the data used 1s and 2s, and since some different models need 0s and 1s, I replaced all of the 2s with 0s.

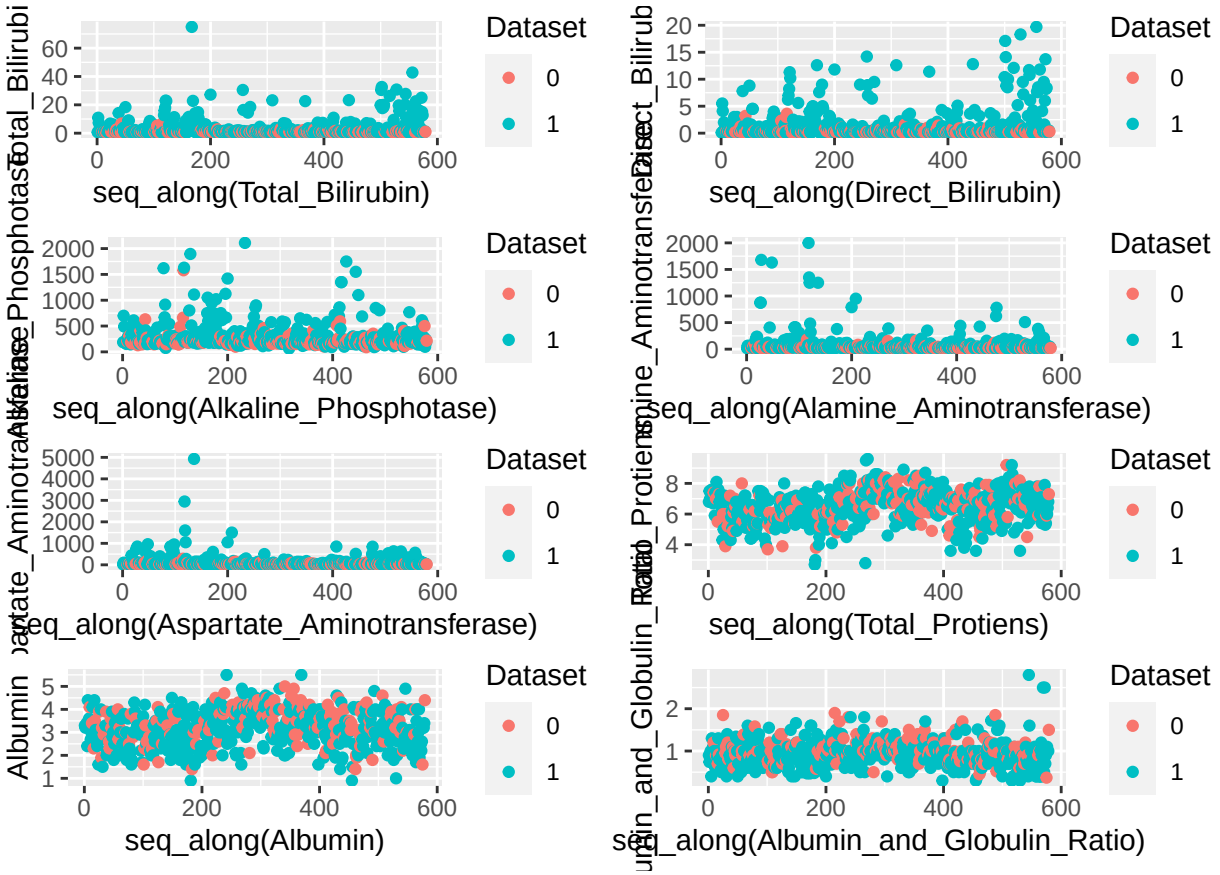
```
dat <- na.omit(dat)

dat$Dataset <- gsub(2,0,dat$Dataset)
```

The final step of preparing the data was creating test and train sets. Because we have a very small amount of data, having only around 600 patients, I decided to use a split of 90-10 for the train and test sets. This ensures that we have a large amount of data to train with. I used the createDataPartition function to split the data into the two sets.

```
y <- dat$Dataset
set.seed(1)
RNGkind(sample.kind = "Rejection")
test_index <- createDataPartition(y, times = 1, p = 0.1, list = FALSE)
test_set <- dat[test_index, ]
train_set <- dat[-test_index, ]
```

The next step was understanding what variables have obvious correlation with whether or not the patient is a liver patient, so I plotted each variable along its index, with the color representing whether the patient has liver problems. After plotting this for every variable, I noticed that there was a lot a variability and randomness to each graph. This meant that there is a lot of limitation to how well each model will be able to predict liver issues.

```
tb <- ggplot(dat, aes(seq_along(Total_Bilirubin), Total_Bilirubin, color=Dataset)) +
  geom_point()

db <- ggplot(dat, aes(seq_along(Direct_Bilirubin), Direct_Bilirubin, color=Dataset)) +
  geom_point()

ap <- ggplot(dat, aes(seq_along(Alkaline_Phosphotase), Alkaline_Phosphotase, color=Dataset)) +
  geom_point()

aa <- ggplot(dat, aes(seq_along(Alamine_Aminotransferase), Alamine_Aminotransferase, color=Dataset)) +
  geom_point()

asa <- ggplot(dat, aes(seq_along(Aspartate_Aminotransferase), Aspartate_Aminotransferase, color=Dataset
  geom_point()

tp <- ggplot(dat, aes(seq_along(Total_Protiens), Total_Protiens, color=Dataset)) +
  geom_point()

a <- ggplot(dat, aes(seq_along(Albumin), Albumin, color=Dataset)) +
  geom_point()

ag <- ggplot(dat, aes(seq_along(Albumin_and_Globulin_Ratio), Albumin_and_Globulin_Ratio , color=Dataset
  geom_point()
ggarrange(tb, db, ap, aa, asa, tp, a, ag,
          ncol = 2, nrow = 4)
```



Throughout the making of this model, I will be calculating the percentage of predictions that are correct to measure the accuracy of the different models. Since there are only two levels for the prediction, an accuracy of 0.50 would be guessing, so anything above this is better than guessing.

The first method that I used to predict liver disease was a generalized linear model (GLM). I started with a glm model because it is generally the simplest model to create. A generalized linear model uses the equation below:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon$$

Yi is the value representing whether or not the patient has liver disease, B0 y-intercept of the regression line, B1 is the coefficient for the x values, xi represents the explanatory variables, and epsilon represents independent errors. The glm functions finds the values for the slope and intercept that limit the average average error that each prediction has. It then uses this line to predict other patients' results. The accuracy for this method was 0.85.

```
train_glm <- train(y = train_set[,11], x = train_set[, -11], method = "glm")
glm_preds <- predict(train_glm, test_set[, -11])
acc_table <- data.frame(method = "glm", acc = mean(glm_preds == test_set$Dataset))
acc_table
```

```
##   method       acc
## 1    glm 0.8474576
```

The next method that I used to predict ratings was quadratic discriminant analysis (QDA). QDA creates a quadratic boundary using Bayes' Theorem to estimate results.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Bayes' Theorem is a probability theorem that explains that the probability of A given B is equal to the probability of B given A multiplied by the probability of A divided by the probability of B. QDA makes two assumptions, that the covariance for each response class is different and that the distribution of the explanatory variables is normal. The QDA method then uses Bayes' Theorem to make a quadratic equation to make predictions. The accuracy of these predictions was 0.64.

```
ctrl <- trainControl(method="repeatedcv",repeats = 3)
train_qda <- train(Dataset ~ ., data = train_set, method = "qda", trControl = ctrl, preProcess = c("cen
qda_preds <- predict(train_qda, test_set[, -11])
acc_table <- bind_rows(acc_table, data_frame(method="qda", acc = mean(qda_preds == test_set$Dataset)))
acc_table
```

```
##   method       acc
## 1    glm 0.8474576
## 2    qda 0.6440678
```

The third method that I used is called k-nearest neighbors (KNN). The KNN method uses the ideology that things that are close are similar, so it splits the graph into k different sections, k being the number of sections. Then depending on the section that a point is in, the model makes a prediction. The accuracy of this model was 0.73.

```
train_KNN <- train(Dataset ~ ., data = train_set, method = "knn", trControl = ctrl, preProcess = c("cen
knn_preds <- predict(train_KNN, test_set)
acc_table <- bind_rows(acc_table, data_frame(method="knn", acc = mean(knn_preds == test_set$Dataset)))
acc_table
```

```
##   method       acc
## 1    glm 0.8474576
## 2    qda 0.6440678
## 3    knn 0.7288136
```

The Fourth method that I used is linear discriminant analysis (LDA). LDA is very similar to QDA as it is also based on Bayes' Theorem, however, instead of making a quadratic equation, it makes a linear equation. The accuracy of this model was 0.71.

```
train_lda <- train(Dataset ~ ., data = train_set, method = "lda", trControl = ctrl, preProcess = c("cen
lda_preds <- predict(train_lda, test_set[, -11])
acc_table <- bind_rows(acc_table, data_frame(method="lda", acc = mean(lda_preds == test_set$Dataset)))
acc_table
```

```
##   method       acc
## 1    glm 0.8474576
## 2    qda 0.6440678
## 3    knn 0.7288136
## 4    lda 0.7118644
```

The fifth Method that I used was a generalized additive model (GAM). The GAM method creates linear models for each individual explanatory variable and combines the models to create a more complicated equation. This equation is then used to make predictions. The accuracy of this model was 0.80.

```
train_gam <- train(Dataset ~ ., data = train_set, method = "gamLoess", trControl = ctrl, preProcess = c
gam_preds <- predict(train_gam, test_set[, -11])
acc_table <- bind_rows(acc_table, data_frame(method="gamLoess", acc = mean(gam_preds == test_set$Dataset
acc_table
```

```
##     method       acc
## 1      glm 0.8474576
## 2      qda 0.6440678
## 3      knn 0.7288136
## 4      lda 0.7118644
## 5 gamLoess 0.7966102
```

The fifth method that I used is the Random Forest method. The random forest ensemble creates a number of decision trees. Then the model creates an ensemble of the trees in which the most popular result among the different trees is the one that is predicted. The accuracy of this model was 0.78.

```
train_rf <- train(Dataset ~ ., data = train_set,method = "rf", tuneGrid = data.frame(mtry = 3), importa
rf_preds <- predict(train_rf, test_set[, -11])
acc_table <- bind_rows(acc_table, data_frame(method="rf", acc = mean(rf_preds == test_set$Dataset)))
acc_table
```

```
##     method       acc
## 1      glm 0.8474576
## 2      qda 0.6440678
## 3      knn 0.7288136
## 4      lda 0.7118644
## 5 gamLoess 0.7966102
## 6       rf 0.7796610
```

The final model that I tested was an ensemble of the best performing models. An ensemble is when you take the predictions from several models and use the prediction that is the most common across the models. I chose to use all of the models except QDA since almost all of them performed similarly well. I created my ensemble by creating a table of the predictions from the five models. I had to replace the 0s with 2s because the mode function was not working properly before. After converting, I gathered the modes of the models, and converted back from 2s to 0s. Now that I had the final ensemble predictions, I calculated the accuracy which was 0.78.

```
ensemble <- data.frame(lda_preds, glm_preds, gam_preds, knn_preds, rf_preds)
modefunc <- function(x){
  tabresult <- tabulate(x)
  themode <- which(tabresult == max(tabresult))
  if(sum(tabresult == max(tabresult))>1) themode <- NA
  return(themode)
}
ensembleNum<- ifelse(ensemble == 0, 2, 1)
modes<-apply(ensembleNum, 1, modefunc)
ensemble$mode <- ifelse(modes == 2, 0, 1)
ensemble
```

```
##    lda_preds glm_preds gam_preds knn_preds rf_preds mode
## 1          1         1         1         1        1    1
```

```
## 2          1          1          1          1          1     1
## 3          1          1          1          1          1     1
## 4          1          1          1          1          1     1
## 5          1          1          1          1          1     1
## 6          1          1          1          1          1     1
## 7          1          1          1          0          1     1
## 8          1          1          1          1          1     1
## 9          1          1          1          1          1     1
## 10         1          1          1          1          1     1
## 11         1          1          1          1          1     1
## 12         1          1          1          1          1     1
## 13         1          1          1          1          1     1
## 14         1          1          1          1          1     1
## 15         1          0          0          1          1     1
## 16         1          1          1          1          1     1
## 17         1          1          1          1          1     1
## 18         1          1          1          1          1     1
## 19         1          1          1          1          1     1
## 20         1          1          1          1          1     1
## 21         1          1          1          1          1     1
## 22         1          1          0          1          1     1
## 23         1          0          0          1          0     0
## 24         1          1          1          1          1     1
## 25         1          1          1          1          1     1
## 26         1          1          1          1          1     1
## 27         1          0          0          0          1     0
## 28         1          1          1          1          1     1
## 29         1          1          1          1          1     1
## 30         1          1          1          1          1     1
## 31         1          1          1          1          1     1
## 32         1          1          0          1          0     1
## 33         1          1          1          0          1     1
## 34         1          0          0          0          1     0
## 35         1          1          1          1          1     1
## 36         1          1          1          1          1     1
## 37         1          0          0          1          1     1
## 38         0          0          0          1          0     0
## 39         1          1          1          1          0     1
## 40         1          1          1          1          1     1
## 41         1          1          1          1          1     1
## 42         1          1          1          1          1     1
## 43         1          0          1          0          1     1
## 44         0          1          1          1          1     1
## 45         1          1          1          1          1     1
## 46         1          1          1          1          1     1
## 47         1          1          1          1          1     1
## 48         1          1          1          1          1     1
## 49         1          1          1          1          1     1
## 50         1          1          1          1          1     1
## 51         1          1          1          1          1     1
## 52         1          1          1          1          1     1
## 53         1          1          1          1          1     1
## 54         1          1          0          1          1     1
## 55         1          1          1          1          1     1
```

```
## 56         1         1         0         1         1    1
## 57         1         0         0         1         1    1
## 58         1         1         1         1         1    1
## 59         1         1         1         1         1    1
```

```
acc_table <- bind_rows(acc_table, data_frame(method="ensemble", acc = mean(ensemble$mode == test_set$Da
acc_table
```

```
##     method       acc
## 1      glm 0.8474576
## 2      qda 0.6440678
## 3      knn 0.7288136
## 4      lda 0.7118644
## 5 gamLoess 0.7966102
## 6       rf 0.7796610
## 7 ensemble 0.7796610
```

Results

This is the entire table of accuracies that I created for each iteration of my model.

```
acc_table
```

```
##     method       acc
## 1      glm 0.8474576
## 2      qda 0.6440678
## 3      knn 0.7288136
## 4      lda 0.7118644
## 5 gamLoess 0.7966102
## 6       rf 0.7796610
## 7 ensemble 0.7796610
```

Going through each accuracy individually, the first accuracy is the GLM, and the accuracy is 0.8474576. This is surprisingly high considering that GLM is a simple model. The next accuracy was from the QDA model, and the accuracy is 0.6440678. This is very low considering usually, a quadratic fit works better for most datasets, however, this makes sense because some of the data does not seem to be very normally distributed, and QDA only works when data is normally distributed. This explains why the accuracy is almost as low of guessing. The next accuracy is from the KNN model. The KNN model's accuracy was 0.7288136, which is one of the highest accuracies. This makes sense since when looking at the plots of the data, there are lots of small clumps of each type of data. This explains why KNN was one of the higher models because KNN makes clusters of points near each other. Both LDA and RF had average accuracy. LDA had an accuracy of 0.7118644 and RF had an accuracy of 0.7796610. GAM had the second highest accuracy of 0.7966102, The final model was the ensemble. The model with the highest accuracy was the GLM, so in order for the accuracy to be higher than the GLM's accuracy, the other models would have to have correctly predicted the patients that the GLM model incorrectly predicted, however, the accuracy was lower, this means odds are that all of the models incorrectly predicted the same patients.

Conclusion

I started with downloading, cleaning, and examining the data. Once I was ready to make models, I started by using the simplest model that I could make, a GLM. This proved to bethe best model to predict liver

disease. The next few models were several iterations of models that each tried different common machine learning methods such as, LDA, QDA, GAM, etc. These got models ranged from barely above average to almost as good as the GLM. Finally, I created an ensemble of every model except for QDA, however, this model was also not as good as the GLM model.

This model performed mediocrely, but it still has lots of limitations. The main limitation is that this model is not the most accurate, definitely not accurate enough to be actually used to diagnose patients. This is mainly due to the fact that all of the variables that are measured do not explain whether the patient has liver disease to a high degree. I know this because when plotting each variable, the patients with liver disease all tend to be randomly scattered throughout the plot. This means that the patterns that each method can find are very limited.

This model could easily be transferred to either a different dataset, or another disease entirely. It is possible that there are other predictors that are significantly better at predicting liver disease, and other datasets could have them.

Overall, I thoroughly enjoyed this project because it finally felt like I could apply my learning throughout this entire program and apply it to a real-life situation, that could possibly help people if further developed.