Indian Liver Patient Predictions

Shriya Reddy P.

11/01/2020

files and dataset avaliable on github: https://github.com/shriyarp/IndianLiverPatient

Introduction

The goal of the project is to predict if a person is a liver patient based on a number of predictors including age, gender, total bilurubin, total proteins, etc.

The dataset consists of 583 observations. 75 percent of the observations will belong to a training dataset while the remaining 25 percent of the observations will belong to the test dataset.

Each observation contains values for the person's age, gender, total bilurubin, direct bilurubin, total proteins, albumin,alkaline phosphates, Alamine Aminotransferase, Aspartate Aminotransferase, Albumin_and_Globulin_Ratio and a value that classifies the person as a patient or not

The steps taken towards this goal are as follows:-

1. Load the data
2. Analyse the data
3. Transform the data
4. Analyze algorithms to create a model
5. Make Predictions

Analysis/Methods

Loading Data

The dataset was downloaded from: https://www.kaggle.com/uciml/indian-liver-patient-records

The dataset along with the files can be accessed on github: https://github.com/shriyarp/IndianLiverPatient

```
indian_liver_patient <- read.csv("indian_liver_patient.csv", header = TRUE)
```

The first 6 rows of the data loaded are given below:

Table 1: Example rows

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_A
65	Female	0.7	0.1	187	16	

Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_A
62	Male	10.9	5.5	699	64	
62	Male	7.3	4.1	490	60	
58	Male	1.0	0.4	182	14	
72	Male	3.9	2.0	195	27	
46	Male	1.8	0.7	208	19	

Analysing Data

The dimensions of the entire dataset is:

```
##    rows columns
##     583      11
```

The column names and types of the dataset are:

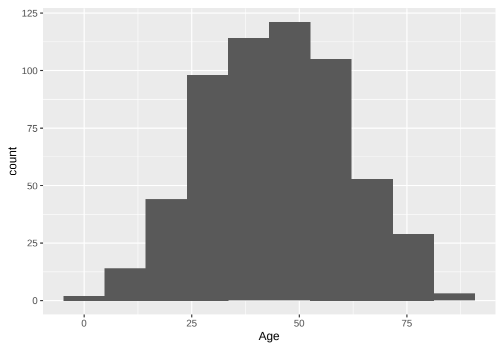
```
##                            colTypes
## Age                         integer
## Gender                       factor
## Total_Bilirubin             numeric
## Direct_Bilirubin            numeric
## Alkaline_Phosphotase        integer
## Alamine_Aminotransferase    integer
## Aspartate_Aminotransferase  integer
## Total_Protiens              numeric
## Albumin                     numeric
## Albumin_and_Globulin_Ratio  numeric
## Dataset                     integer
```

The number of missing values in each column is given below:-

```
##                            missingNumber
## 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 distribution of each column is as follows:-

Age

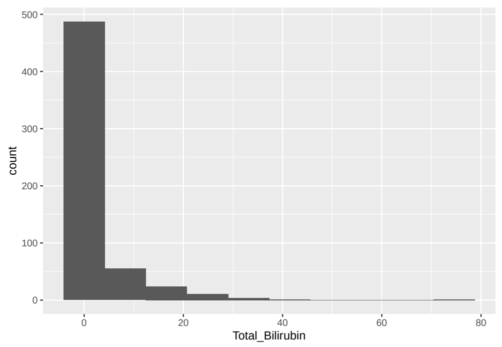


```
##      stdDev     mean
## 1: 16.18983 44.74614
```

Gender

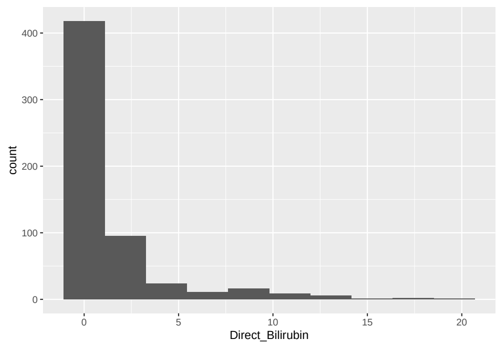
```
##    Gender percentage
## 1: Female         24
## 2:   Male         76
```

Total_Bilirubin



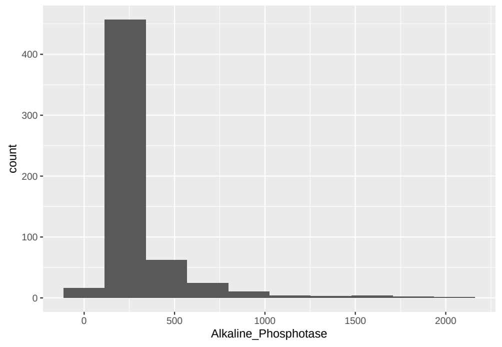
```
##      stdDev     mean
## 1: 6.209522 3.298799
```

Direct_Bilirubin



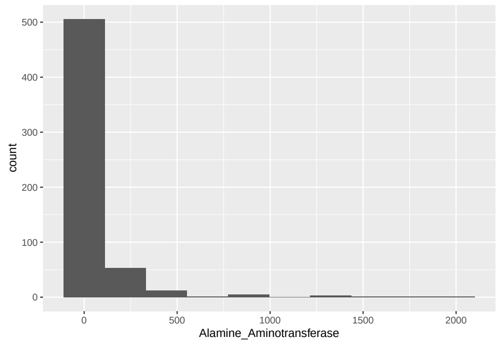
```
##      stdDev     mean
## 1: 2.808498 1.486106
```

Alkaline_Phosphotase



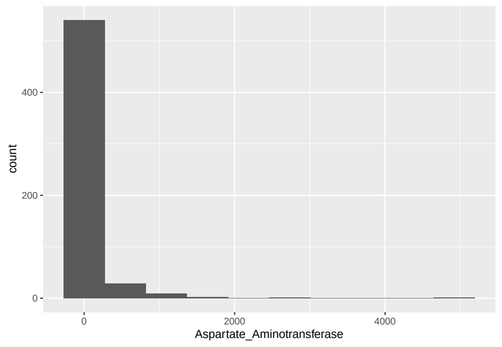
```
##     stdDev     mean
## 1: 242.938 290.5763
```

Alamine_Aminotransferase



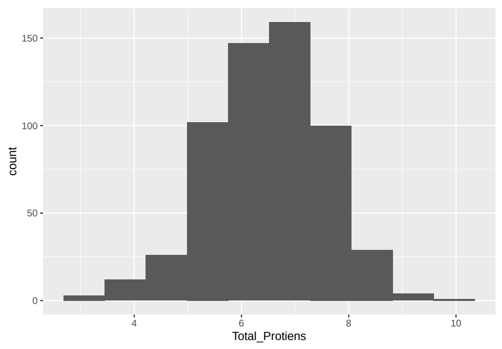
```
##      stdDev     mean
## 1: 182.6204 80.71355
```

Aspartate_Aminotransferase



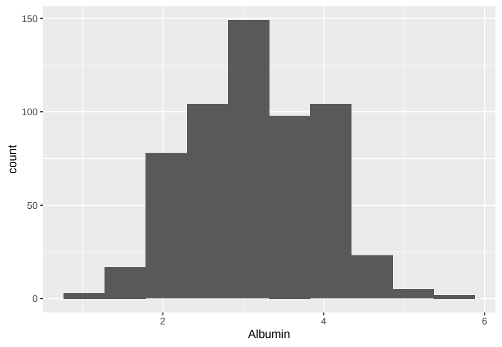
```
##      stdDev     mean
## 1: 288.9185 109.9108
```

Total_Protiens



```
##      stdDev    mean
## 1: 1.085451 6.48319
```

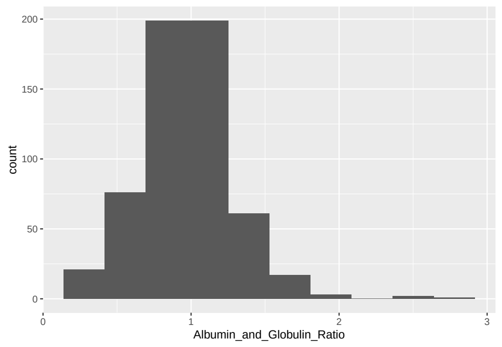
Albumin



```
##       stdDev     mean
## 1: 0.7955188 3.141852
```

Albumin_and_Globulin_Ratio

```
## Warning: Removed 4 rows containing non-finite values (stat_bin).
```



```
##       stdDev      mean
## 1: 0.3195921 0.9470639
```

isLiverPatient

```
##   Dataset percentage
## 1       1         71
## 2       2         29
```

Transform Data

From the above distributions and information, we observe that the Albumin_and_Globulin_Ratio column is missing 4 values and we also observe that the Dataset column(which indicates if a person is a liver patient) is stored as in integer despite being a categorical variable

Thus, we fill in the 4 missing values of the Albumin_and_Globulin_Ratio column with the average value of the column

```r
#identify missing idexes
indices <- which(is.na(indian_liver_patient$Albumin_and_Globulin_Ratio))
#calculate mean of the column
mean_agratio <- round(mean(indian_liver_patient$Albumin_and_Globulin_Ratio, na.rm = TRUE)*100)/100
#replace missing values with mean
for (i in indices) {
  indian_liver_patient$Albumin_and_Globulin_Ratio[i] <- mean_agratio
}
```

We also tranform the dataset column by changing its type from 'integer' to 'factor'

```r
indian_liver_patient <- indian_liver_patient %>% mutate(Dataset = factor(Dataset))
```

We now seperate out the training and test datasets

```r
set.seed(1)
#create test index
index <- sample(nrow(indian_liver_patient), round(0.25*nrow(indian_liver_patient)))

train <- indian_liver_patient[-index,]
test <- indian_liver_patient[index,]
```

Evaluate Algorithms to create a model

We first create a method for validating our data

```r
control <- trainControl(method="cv", number=10)
```

We evaluate using 8 different algorithms:

1. Quadratic Discriminant Analysis
2. AdaBoost Classification Trees
3. Naive Bayes
4. Linear Discriminant Analysis
5. CART
6. K-nearest neighbours
7. Support Vector Machines with Radial Basis Function Kernel
8. Randowm Forest

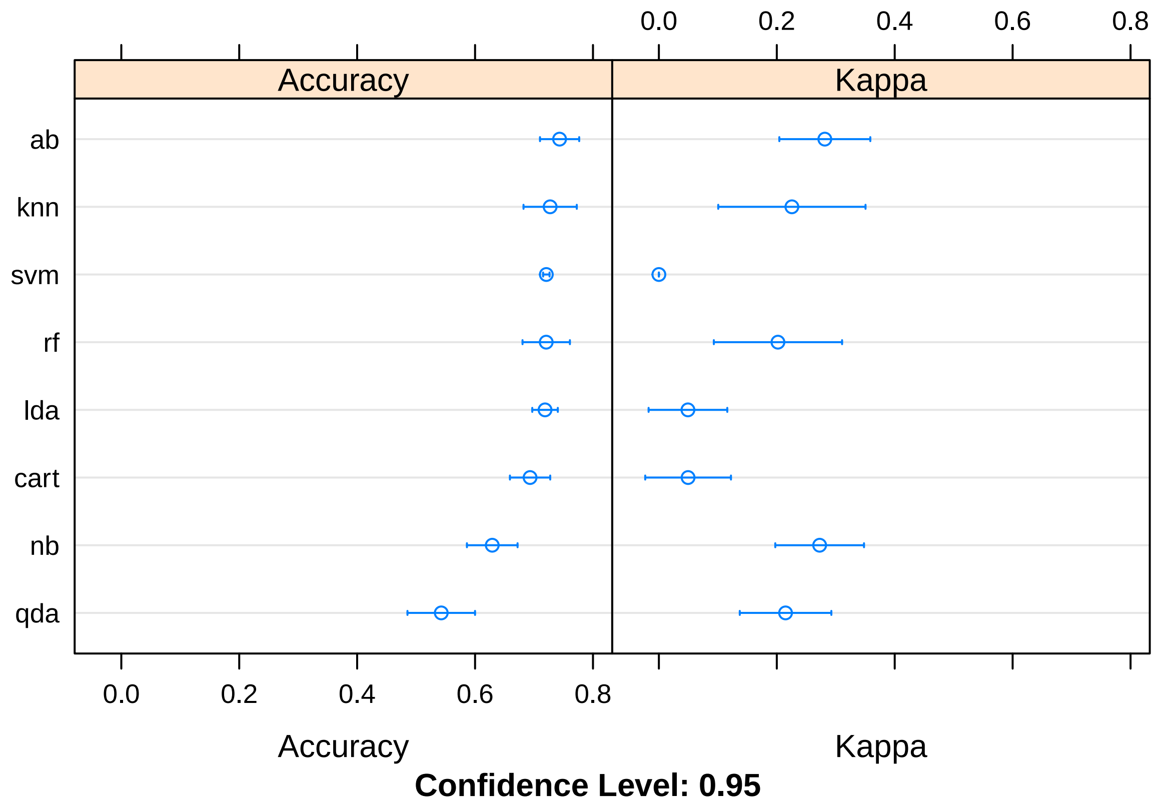
```r
#Quadratic Discriminant Analysis
set.seed(4)
fit.qda <- train(Dataset~., data=train, method="qda", trControl=control)
```

```r
#AdaBoost Classification Trees
set.seed(4)
fit.ab <- train(Dataset~., data=train, method="adaboost", trControl=control)

#Naive Bayes
set.seed(4)
fit.nb <- train(Dataset~., data=train, method="naive_bayes", trControl=control)

# Linear Discriminant Analysis
set.seed(4)
fit.lda <- train(Dataset~., data=train, method="lda",  trControl=control)

# CART
set.seed(4)
fit.cart <- train(Dataset~., data=train, method="rpart",  trControl=control)

# K-nearest neighbours
set.seed(4)
fit.knn <- train(Dataset~., data=train, method="knn",  trControl=control)

# Support Vector Machines with Radial Basis Function Kernel
set.seed(4)
fit.svm <- train(Dataset~., data=train, method="svmRadial",  trControl=control)

# Random Forest
set.seed(4)
fit.rf <- train(Dataset~., data=train, method="rf",  trControl=control)
```

Post Building the models, we now select the best model based on accuracy

```r
results <- resamples(list(qda = fit.qda, ab = fit.ab, nb=fit.nb, lda=fit.lda, cart=fit.cart, knn=fit.kn
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: qda, ab, nb, lda, cart, knn, svm, rf
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## qda  0.4444444 0.4630021 0.5465116 0.5426474 0.6186575 0.6363636    0
## ab   0.6511628 0.7225159 0.7527778 0.7433674 0.7714059 0.8181818    0
## nb   0.4883721 0.6125793 0.6437632 0.6291308 0.6741543 0.6818182    0
## lda  0.6666667 0.7086416 0.7209302 0.7186751 0.7256871 0.7727273    0
## cart 0.5813953 0.6819503 0.7127378 0.6933028 0.7256871 0.7272727    0
## knn  0.6046512 0.6928030 0.7209302 0.7273455 0.7897727 0.7954545    0
## svm  0.7045455 0.7209302 0.7209302 0.7208468 0.7272727 0.7272727    0
## rf   0.6279070 0.6976744 0.7241015 0.7206812 0.7500000 0.7954545    0
##
## Kappa
##               Min.    1st Qu.     Median       Mean    3rd Qu.      Max. NA's
## qda   0.056999162 0.13279635 0.20990669 0.21485516 0.31357700 0.3529412    0
## ab    0.110344828 0.19385382 0.32237119 0.28147053 0.35942069 0.4210526    0
## nb    0.026748971 0.22183427 0.30637095 0.27268206 0.35352475 0.3636364    0
## lda  -0.084507042 0.00000000 0.02960526 0.04929304 0.06859206 0.2253521    0
## cart -0.096317280 0.00000000 0.00000000 0.04953943 0.09639479 0.2691218    0
## knn  -0.064046579 0.12551760 0.20424865 0.22560578 0.32514487 0.4705882    0
## svm   0.000000000 0.00000000 0.00000000 0.00000000 0.00000000 0.0000000    0
## rf   -0.002785515 0.05379053 0.23005152 0.20197101 0.28503875 0.4406780    0
```



Based on the results, we now find the the best fit is AdaBoost Classification Trees The fit for AdaBoost Classification Trees along with accuracy metrics is:

```
## AdaBoost Classification Trees
##
## 437 samples
##  10 predictor
##   2 classes: '1', '2'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 394, 394, 394, 392, 393, 393, ...
## Resampling results across tuning parameters:
##
##   nIter  method         Accuracy   Kappa
##    50    Adaboost.M1    0.7093199  0.2563313
##     50    Real adaboost  0.7433674  0.2814705
##    100    Adaboost.M1    0.7158786  0.2718407
##    100    Real adaboost  0.7274583  0.2200980
##    150    Adaboost.M1    0.7227519  0.2777852
##    150    Real adaboost  0.7159866  0.1936912
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nIter = 50 and method = Real adaboost.
```

Make Predictions

We can now proceed to make predictions based on the AdaBoost Classification Trees

```
#predict
y_hat <- predict(fit.ab,test)
```

Result

The confusion matrix obtained is:

```
confusionMatrix(y_hat,test$Dataset)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 86 29
##          2 15 16
##
##                Accuracy : 0.6986
##                  95% CI : (0.6173, 0.7717)
##     No Information Rate : 0.6918
##     P-Value [Acc > NIR] : 0.46886
##
##                   Kappa : 0.2266
##
##  Mcnemar's Test P-Value : 0.05002
##
##             Sensitivity : 0.8515
##             Specificity : 0.3556
##          Pos Pred Value : 0.7478
##          Neg Pred Value : 0.5161
##              Prevalence : 0.6918
##          Detection Rate : 0.5890
##    Detection Prevalence : 0.7877
##       Balanced Accuracy : 0.6035
##
##        'Positive' Class : 1
##
```

from the confusion matrix and the other readings above, we find that:-

1. The accuracy of the predictions stands at 69.9% (approx. 70%)
2. The sensitivity of the data is 85.15%
3. The specificity of the data is 33.56%

Therefore, the prediction are correct 70% of the time. On recipt of a positive result, the probability of a correct result is 85%. The sensitivity is relatively high, which is a good things for medical cases. However, the specificity of the data is very low, meaning that, on receipt of a negative outcome, the chances that the person is not actually a liver patient is 33.56%. This also could mean that people who are actually liver patients could be ignored.

Conculsion

After, evaluating the dataset and performing the necessay tranformations and choosing appropriate algorithms, The best algorithm found in this case was the AdaBoost Classification Trees, giving an accuracy of approximately 70% and a TPR of 85%. However, models like k nearest neighbours, support vector machine and random forest were not far behind in terms of accuracy. Therefore, it is not possible to conclude that only the adaboost classification trees algorithm was the best algorithm to use out of the 8 chosen. Being in possession of more domain knowledge would probably be beneficial in order to determine the best kind of algorithm for the kind of data that has been provided. However despite, having a relatively low number of observations in the dataset, the predictions obtained were satisfactorily accurate.