Indian Liver Patients Prediction

Ankur

6/25/2021

As advised, visited Kaggle and downloaded dataset on Indian Liver Patients
https://www.kaggle.com/uciml/indian-liver-patient-records

Executive Summary:

Why ?

Liver is an important organ in the body. With the algorithm built we will try to assess whether person is liver patient or not. This will help humanity to proactively work on diseases before it goes worse.

What ?

We have data of 583 patients. Each patient is tested for some bio paremeters and last row tells whether person is liver patient or not. We will build algorithms on this data and check for most accurate method. We will build Logistics Regression and KNN Models and see which one is more accurate. Data will be spitted in Training & Testing in a ratio of 80% & 20%.

About Data:

Context: Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

Content: This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The “Dataset” column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age “90”.

Columns:

Age of the patient Gender of the patient Total Bilirubin Direct Bilirubin Alkaline Phosphotase Alamine Aminotransferase Aspartate Aminotransferase Total Protiens Albumin Albumin and Globulin Ratio Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

Lets Import the file before that lets get the required packages

```
# Loading the required packages
library(dummies)

## dummies-1.5.6 provided by Decision Patterns

library(boot) # For K fold
library(caret)

## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:boot':
##
##     melanoma

## Loading required package: ggplot2

library(ggplot2)
library(class) # For Knn
```

Importing File

```
f<- file.choose()
data <- read.csv(f)
```

Data Visualisation

```
summary(data)

##       Age           Gender          Total_Bilirubin  Direct_Bilirubin
##  Min.   : 4.00   Length:583         Min.   : 0.400   Min.   : 0.100
##  1st Qu.:33.00   Class :character   1st Qu.: 0.800   1st Qu.: 0.200
##  Median :45.00   Mode  :character   Median : 1.000   Median : 0.300
##  Mean   :44.75                      Mean   : 3.299   Mean   : 1.486
##  3rd Qu.:58.00                      3rd Qu.: 2.600   3rd Qu.: 1.300
##  Max.   :90.00                      Max.   :75.000   Max.   :19.700
##
##  Alkaline_Phosphotase Alamine_Aminotransferase Aspartate_Aminotransferase
##  Min.   :  63.0       Min.   :  10.00          Min.   :  10.0
##  1st Qu.: 175.5       1st Qu.:  23.00          1st Qu.:  25.0
##  Median : 208.0       Median :  35.00          Median :  42.0
##  Mean   : 290.6       Mean   :  80.71          Mean   : 109.9
##  3rd Qu.: 298.0       3rd Qu.:  60.50          3rd Qu.:  87.0
##  Max.   :2110.0       Max.   :2000.00          Max.   :4929.0
##
##  Total_Protiens     Albumin      Albumin_and_Globulin_Ratio    Dataset
##  Min.   :2.700   Min.   :0.900   Min.   :0.3000             Min.   :1.000
##  1st Qu.:5.800   1st Qu.:2.600   1st Qu.:0.7000             1st Qu.:1.000
##  Median :6.600   Median :3.100   Median :0.9300             Median :1.000
##  Mean   :6.483   Mean   :3.142   Mean   :0.9471             Mean   :1.286
##  3rd Qu.:7.200   3rd Qu.:3.800   3rd Qu.:1.1000             3rd Qu.:2.000
```

```
##  Max.   :9.600   Max.   :5.500   Max.   :2.8000            Max.   :2.000
##                                  NA's   :4
```

```
head(data)
```

```
##   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 Album
in
## 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
```

We can see age varies from 4 uptil 90 in Dataset column, 1 is Liver patient and 2 is not a liver patient.

Check for any missing values in the dataset

```
sum(is.na(data))
```

```
## [1] 4
```

There are 4 Missing Values in the data. Lets see in which column?

```
apply(data,2,function(x)sum(is.na(x)))
```

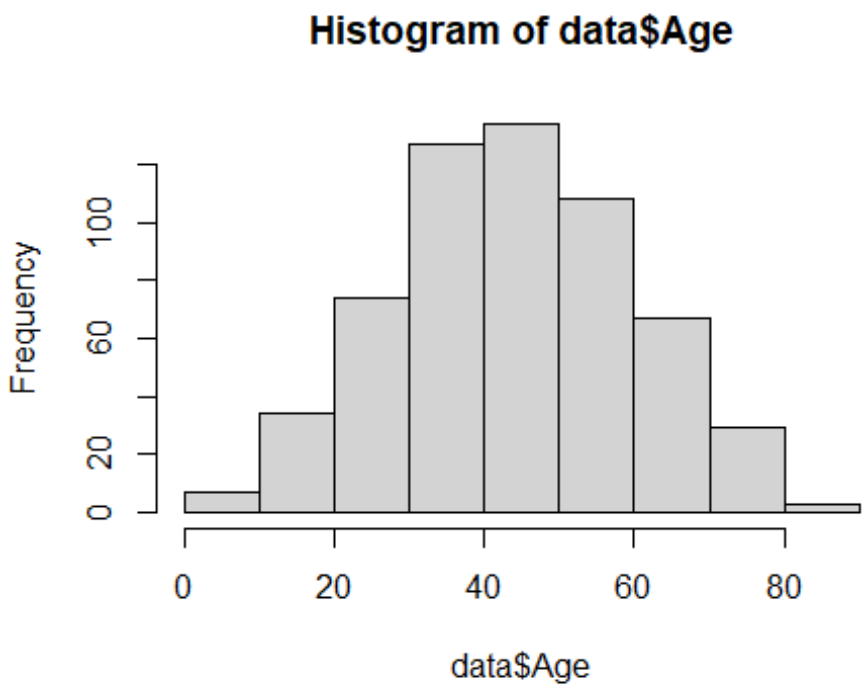
```
##                         Age                      Gender
##                           0                           0
##            Total_Bilirubin           Direct_Bilirubin
##                           0                           0
##       Alkaline_Phosphotase   Alamine_Aminotransferase
```

```
##                           0                           0
## Aspartate_Aminotransferase             Total_Protiens
##                           0                           0
##                     Albumin Albumin_and_Globulin_Ratio
##                           0                           4
##                     Dataset
##                           0
```

THere are 4 Missing columns in Column Named Albumin & Glucose Ration

Age Histogram

```
hist(data$Age)
```



No of Males & Females in dataset

```
table(data$Gender)
```

```
##
## Female   Male
##    142    441
```

There are 441 Male & 142 Female

No of Patients in Data Set

```
table(data$Dataset)
```

```
##
##   1   2
## 416 167
```

416 are having liver disease

Lets Tabulate

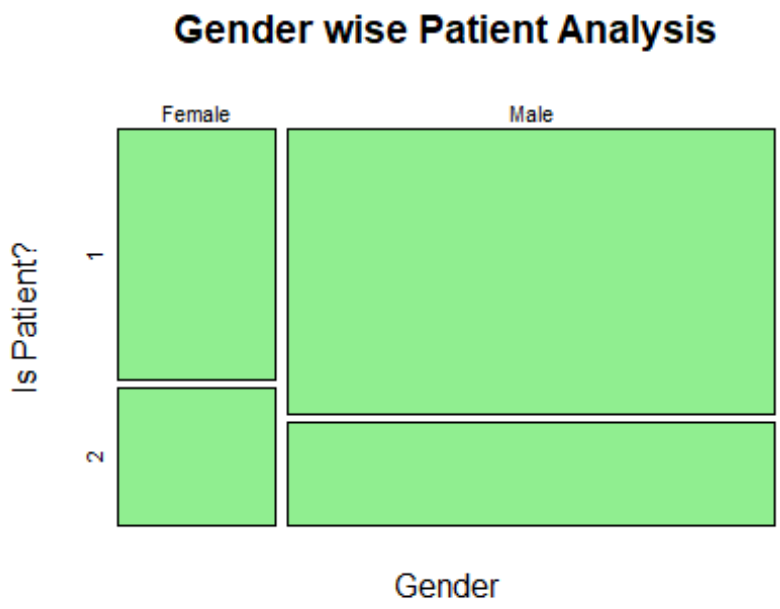
```
table(data$Gender,data$Dataset)
```

```
##
##            1   2
##   Female  92  50
##   Male   324 117
```

324 males & 92 Females are patient

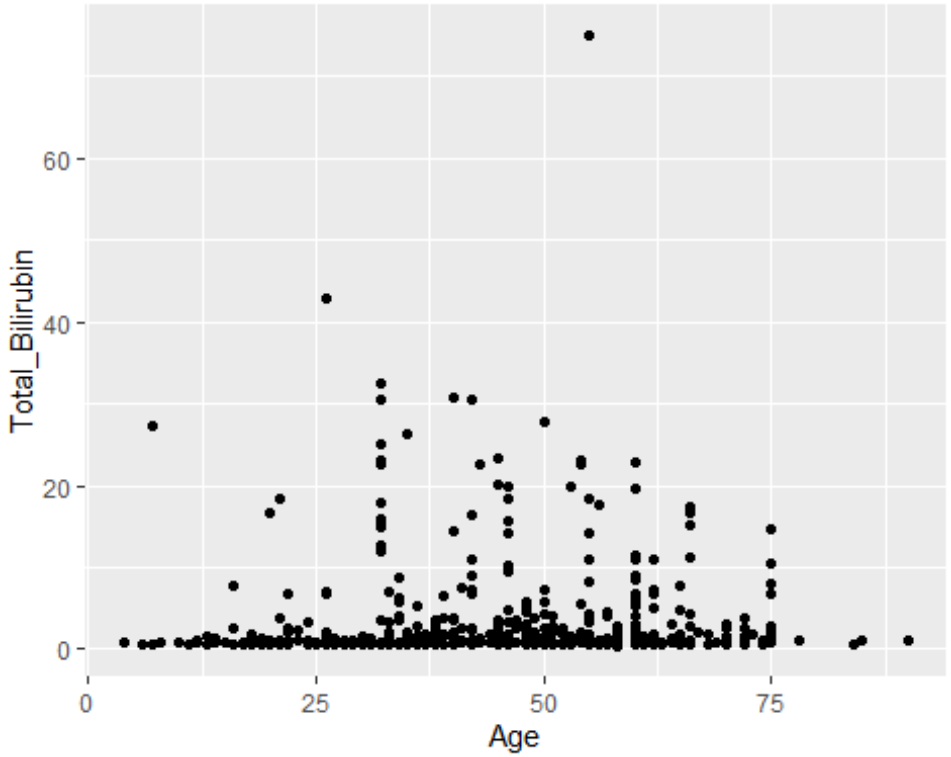
Plotting the Table

```
plot(table(data$Gender,data$Dataset),
     ylab = "Is Patient?",
     xlab = "Gender",
     main = "Gender wise Patient Analysis",
     col = c("lightgreen"))
```



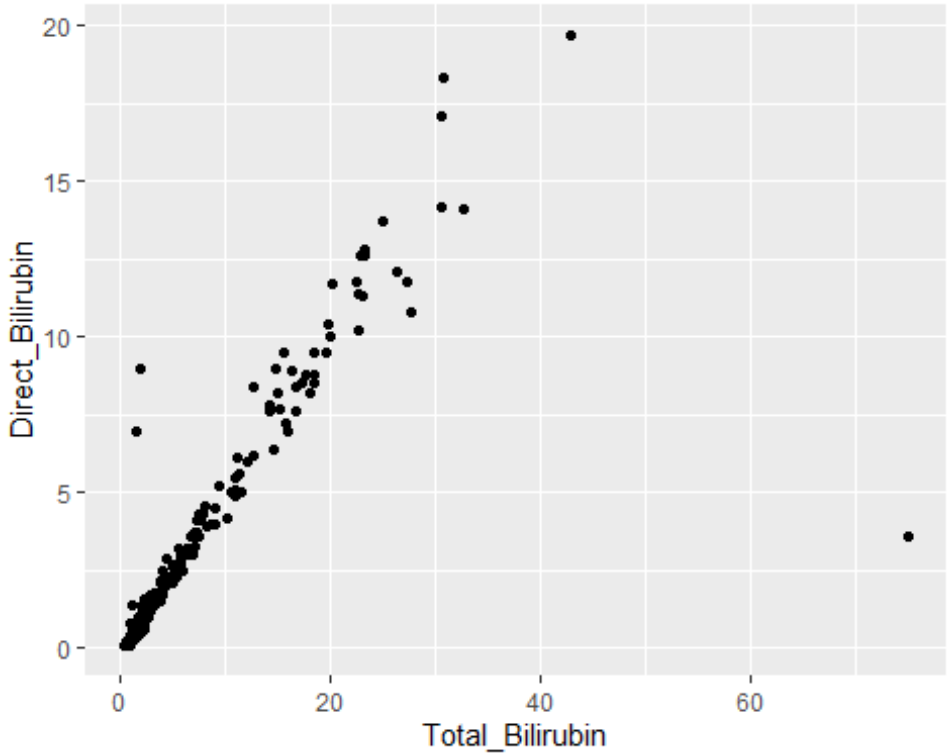
Is there is connect between Age & Total Bilirubin

```
p<- ggplot(data= data)
p<- p+ geom_point(aes(Age,Total_Bilirubin))
p
```



Relationship between Total Bilirubin & Direct Bilirubin

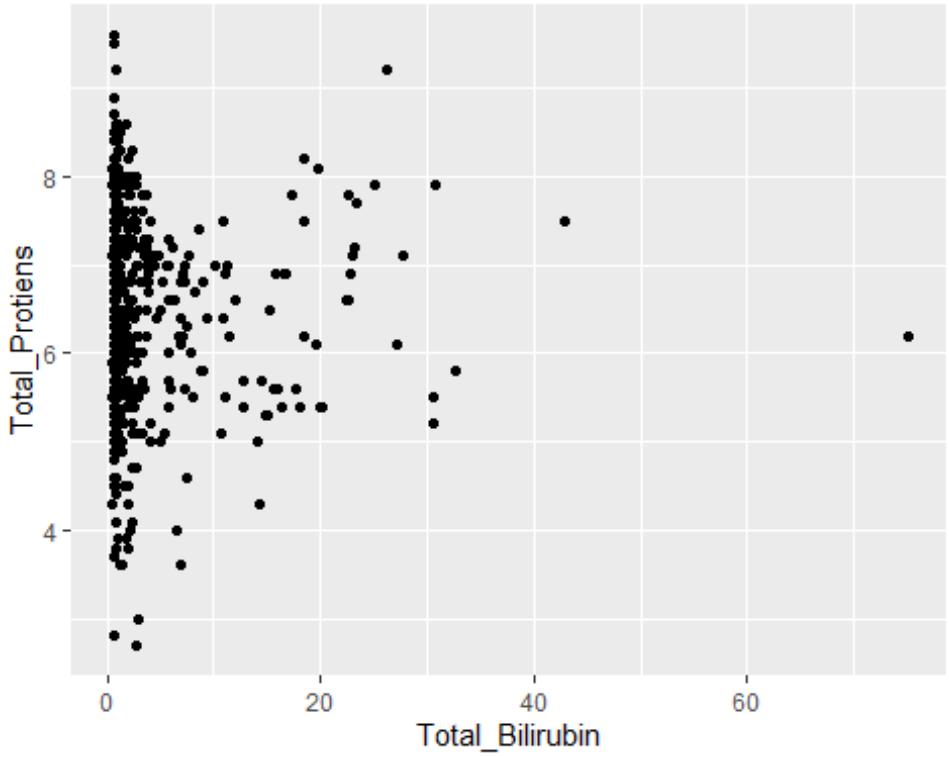
```
p1<- ggplot(data= data)
p1<- p1+ geom_point(aes(Total_Bilirubin, Direct_Bilirubin))
p1
```



Total Bilirubin & Direct Bilirubin are proportional

Relationship between Total Bilirubin & Direct Bilirubin

```
p2<- ggplot(data= data)
p2<- p2+ geom_point(aes(Total_Bilirubin, Total_Protiens))
p2
```



Data Wrangling

Lets encode Dataset as 0 and 1.

```
data$Dataset = ifelse(data$Dataset == 2, 0 ,1)
str(data)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : chr  "Female" "Male" "Male" "Male" ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202 29
0 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ..
.
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.
4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ..
.
##  $ Dataset                   : num  1 1 1 1 1 1 1 1 0 1 ...
```

Now we will factorize Dataset variable in the data

```
data$Dataset = as.factor(data$Dataset)
contrasts(data$Dataset)
```

```
##   1
## 0 0
## 1 1
```

```
str(data)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : chr  "Female" "Male" "Male" "Male" ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 1.8 0.9 0.9 0.9 0.7
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.7 0.2 0.3 0.3 0.2
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 208 154 202 202 29
0 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 19 16 14 22 53 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 14 12 11 19 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7.6 7 6.7 7.4 6.8 ..
.
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 4.4 3.5 3.6 4.1 3.
4 ...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1.3 1 1.1 1.2 1 ..
.
##  $ Dataset                   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2
1 2 ...
```

Removing NAs from data

```
data_clean = data[complete.cases(data),]
```

Partition of Data

Now we will split the data into Train & test Set

```
index <-  createDataPartition(data_clean$Dataset,p= .8, times=1, list=F)

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

Modeling

Fitting a logistic regression model:

We will use logistic regression model on the training set which is 80% of the total data.

```
logistic<- glm(Dataset~., data=train, family='binomial')
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logistic)
```

```
##
## Call:
## glm(formula = Dataset ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0842  -1.0608   0.4122   0.9015   1.4807
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -2.1679602  1.3900477  -1.560  0.11885
## Age                         0.0183742  0.0071118   2.584  0.00978 **
## GenderMale                 -0.0089801  0.2610501  -0.034  0.97256
## Total_Bilirubin             0.0096763  0.0793014   0.122  0.90288
## Direct_Bilirubin            0.4431609  0.2429766   1.824  0.06817 .
## Alkaline_Phosphotase        0.0006793  0.0007483   0.908  0.36397
## Alamine_Aminotransferase    0.0129710  0.0055099   2.354  0.01857 *
## Aspartate_Aminotransferase  0.0016890  0.0034725   0.486  0.62670
## Total_Protiens              0.5877692  0.3889604   1.511  0.13076
## Albumin                    -1.1691421  0.7546981  -1.549  0.12135
## Albumin_and_Globulin_Ratio  1.0159298  1.1405290   0.891  0.37306
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 554.14  on 463  degrees of freedom
## Residual deviance: 460.66  on 453  degrees of freedom
## AIC: 482.66
##
## Number of Fisher Scoring iterations: 7
```

Pedicting on the Dataset

```
pred= predict(logistic, test, type='response')
pred
```

```
##         6         8        10        21        26        27        29
35
## 0.5127027 0.4232627 0.7209430 0.8476746 0.9999979 0.9999979 0.4640301 0.69
51085
##        36        42        46        52        67        71        73
78
## 0.8117604 0.6919943 0.7012182 0.4410335 0.7686927 0.8976175 0.7103593 0.93
93530
##        81        86        92        97       101       108       111
113
## 0.9101710 0.5317527 0.9997753 0.9506602 0.4991756 0.6439912 0.4801906 0.58
84162
```

```
##       125       127       133       136       138       140       146
152
## 0.4275775 0.9662155 0.4488504 1.0000000 0.6758400 0.4188229 0.6039365 0.64
10087
##       156       158       160       161       163       169       180
183
## 0.9430373 0.7733824 0.6065944 0.9959585 0.9443683 0.9992917 0.9740821 0.77
40497
##       186       187       190       197       202       203       206
214
## 0.7115313 0.9559590 0.6150776 0.9124088 0.6697183 0.5044713 0.6513444 0.50
42358
##       220       222       229       236       246       258       260
269
## 0.5463746 0.5333385 0.6581705 0.9829855 0.6128032 0.8723640 0.9995241 0.98
12205
##       274       276       278       287       295       301       304
321
## 0.5595581 0.5388179 0.7586832 0.4200320 0.5048131 0.5884862 0.5700601 0.45
64727
##       324       329       331       339       341       342       345
357
## 0.5078623 0.5745226 0.7073135 0.8876104 0.7560714 0.7755925 0.5188539 0.56
73150
##       364       374       377       388       389       398       406
407
## 0.5167813 0.4146746 0.5026056 0.5713355 0.5765170 0.7264095 0.4457390 0.50
75895
##       423       424       426       428       436       437       460
464
## 0.6352090 0.6872399 0.7329869 0.7574268 0.3477944 0.5584787 0.8757723 0.45
84872
##       469       470       471       477       480       488       491
500
## 0.7421777 0.6557173 0.6001217 0.5504003 0.9999944 0.8051486 0.7828675 0.87
01925
##       504       510       513       523       525       526       528
532
## 0.9989756 0.9991586 0.6551584 0.5905477 0.3840887 0.6481083 0.8624300 0.99
99815
##       538       541       543       546       550       557       559
568
## 0.4898980 0.5981914 0.6382043 0.6101992 0.7740791 0.5696359 0.9984299 0.86
50450
##       573       575       578
## 0.9928208 0.9800861 0.9797002

pred = ifelse(pred > 0.40, 1, 0)
pred= as.factor(pred)
```

Checking Accuracy with Confusion Matrix

```
matrix = confusionMatrix(pred,test$Dataset)
matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  2  0
##          1 31 82
##
##                Accuracy : 0.7304
##                  95% CI : (0.6397, 0.8089)
##     No Information Rate : 0.713
##     P-Value [Acc > NIR] : 0.3837
##
##                   Kappa : 0.0843
##
##  Mcnemar's Test P-Value : 7.118e-08
##
##             Sensitivity : 0.06061
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.72566
##              Prevalence : 0.28696
##          Detection Rate : 0.01739
##    Detection Prevalence : 0.01739
##       Balanced Accuracy : 0.53030
##
##        'Positive' Class : 0
##
```

Accuracy is 73.04%

K Nearest Neighbour

We will create new variable for interpretability

```
train_knn = train

test_knn= test
```

Let us create some dummy variable for Gender in both train and test

Train data set

```
gender = dummy(train_knn$Gender)
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = F
ALSE):
## non-list contrasts argument ignored
```

```
train_knn$Gender = gender[,2]
str(train_knn)
```

```
## 'data.frame':    464 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 26 17 57 72 64 ...
##  $ Gender                    : int  0 1 1 1 1 0 1 1 1 1 ...
##  $ Total_Bilirubin           : num  0.7 10.9 7.3 1 3.9 0.9 0.9 0.6 2.7 0.9
...
##  $ Direct_Bilirubin          : num  0.1 5.5 4.1 0.4 2 0.2 0.3 0.1 1.3 0.3
...
##  $ Alkaline_Phosphotase      : int  187 699 490 182 195 154 202 210 260 31
0 ...
##  $ Alamine_Aminotransferase  : int  16 64 60 14 27 16 22 51 31 61 ...
##  $ Aspartate_Aminotransferase: int  18 100 68 20 59 12 19 59 56 58 ...
##  $ Total_Protiens            : num  6.8 7.5 7 6.8 7.3 7 7.4 5.9 7.4 7 ...
##  $ Albumin                   : num  3.3 3.2 3.3 3.4 2.4 3.5 4.1 2.7 3 3.4
...
##  $ Albumin_and_Globulin_Ratio: num  0.9 0.74 0.89 1 0.4 1 1.2 0.8 0.6 0.9
...
##  $ Dataset                   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 2
2 1 ...
```

Test Data Set

```
gender=dummy(test_knn$Gender)
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = F
ALSE):
## non-list contrasts argument ignored
```

```
test_knn$Gender = gender[,2]
str(test_knn)
```

```
## 'data.frame':    115 obs. of  11 variables:
##  $ Age                       : int  46 29 55 51 34 34 20 38 30 62 ...
##  $ Gender                    : int  1 0 1 1 1 1 1 0 1 1 ...
##  $ Total_Bilirubin           : num  1.8 0.9 0.7 2.2 4.1 4.1 1.1 2.6 1.3 0.
6 ...
##  $ Direct_Bilirubin          : num  0.7 0.3 0.2 1 2 2 0.5 1.2 0.4 0.1 ...
##  $ Alkaline_Phosphotase      : int  208 202 290 610 289 289 128 410 482 16
0 ...
##  $ Alamine_Aminotransferase  : int  19 14 53 17 875 875 20 59 102 42 ...
##  $ Aspartate_Aminotransferase: int  14 11 58 28 731 731 30 57 80 110 ...
##  $ Total_Protiens            : num  7.6 6.7 6.8 7.3 5 5 3.9 5.6 6.9 4.9 ..
.
##  $ Albumin                   : num  4.4 3.6 3.4 2.6 2.7 2.7 1.9 3 3.3 2.6
...
##  $ Albumin_and_Globulin_Ratio: num  1.3 1.1 1 0.55 1.1 1.1 0.95 0.8 0.9 1.
1 ...
##  $ Dataset                   : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 1
2 1 ...
```

Let us do scaling for Knn

```
train_scale = scale(train_knn[,-11])
test_scale = scale(test_knn[,-11])
```

We are not sure how many neighbors to use so we will run a loop and record the metrics for each number of neighbors

```
Results_knn = matrix(0,nrow = 13, ncol = 4)
colnames(Results_knn) = c('Neighbors','01error','10error','Accuracy')

for(i in 1:13){
  pred_knn = knn(train = train_scale,
                 test = test_scale,
                 cl = train_knn[,11],
                 k = i + 2,
                 prob = TRUE)

  knn_con = confusionMatrix(data = pred_knn, reference = test_knn[,11])
  Results_knn[i,1] = i + 2
  Results_knn[i,2] = knn_con$table[2,1]
  Results_knn[i,3] = knn_con$table[1,2]
  Results_knn[i,4] = knn_con$overall[1]
}

Results_knn = as.data.frame(Results_knn)
Results_knn
```

```
##    Neighbors 01error 10error  Accuracy
## 1          3      18      15 0.7130435
## 2          4      18      16 0.7043478
## 3          5      21       9 0.7391304
## 4          6      22       8 0.7391304
## 5          7      21       9 0.7391304
## 6          8      26       6 0.7217391
## 7          9      25       6 0.7304348
## 8         10      25       5 0.7391304
## 9         11      25       5 0.7391304
## 10        12      26       5 0.7304348
## 11        13      26       5 0.7304348
## 12        14      24       8 0.7217391
## 13        15      28       9 0.6782609
```

Accuracy = 68.69%

##Conclusion: As advised, we have built two models.

We have ran two models as follows . Logistics Regression: 73.04% Knn: 68.69%