Liver Cancer Analysis

2022-09-20

Executive Summary

By training 4 machine learning algorithms on Indian Liver Cancer data set (583 obs/11 variables), this analysis is to determine if liver cancer (represented by the target column named "dataset") can be identified by 10 available predictors/features. If yes, how accurate the model is and what are the top 5 predictors for each model respectively.

```
##### Install packages ######

if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.2      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret))
  install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table))
  install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table
##
## Attaching package: 'data.table'
##
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
##
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
if(!require(rpart))
  install.packages("rpart", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: rpart
```

```
if(!require(matrixStats))
  install.packages("matrixStats", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: matrixStats
##
## Attaching package: 'matrixStats'
##
## The following object is masked from 'package:dplyr':
##
##     count
```

```
if(!require(dslabs))
  install.packages("dslabs", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dslabs
```

```
#if(!require(genefilter)) install.packages("genefilter", repos = "http://cran.us.r-project.org")
if(!require(gam))
  install.packages("gam", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gam
## Loading required package: splines
## Loading required package: foreach
##
## Attaching package: 'foreach'
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
## Loaded gam 1.20.2
```

```
if(!require(gridExtra))
  install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: gridExtra
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(randomForest))
  install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:gridExtra':
##
##     combine
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!require(tinytex))
  install.packages("tinytex", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tinytex
```

```
##### loading packages #####

library(tidyverse)
library(caret)
library(data.table)
library(rpart)
library(matrixStats)
library(dslabs)
#library(genefilter)
library(gam)
library(gridExtra)
library(randomForest)
library(tinytex)

#tinytex::install_tinytex()
#update.packages(ask = FALSE, checkBuilt = TRUE)
#update.packages("tidyverse")
#update.packages()
#installed.packages()
#remove.packages(pkgs=row.names(x=installed.packages(priority="NA")))
#old.packages()
```

Comment

1. Download dataset indian-liver-patient-records directly from url <- "www.kaggle.com/uciml/indian-liver-patient-records" Then push the dataset "indian-liver-patient-records" to github, finally, read it to R environment, and name it as "liver.df"
2. Albumin_and_Globulin_Ratio has 1% NA

```
#liver.df <- read.csv("C:/Users/yfu/Desktop/LiverPatients_download.csv")
liver.df <- read.csv(url(
           "https://raw.githubusercontent.com/yfu2021/liver_cancer/master/LiverPatients_download.csv"
              )
            )
summary(liver.df)
```

```
##       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
```

Comments

1. Rename the data element 'Dataset' to 'Diagnosis', and change this data element type to factor with levels of 1 and 0 respectively, 1 is liver cancer, 0 is non liver cancer
2. Clean the data - fill in the missing value of Albumin_and_Globulin_Ratio with median value of Albumin_and_Globulin_Ratio
3. Remove the column 'Dataset' from the liver.df

```
liver <- liver.df %>%
         mutate(Diagnosis = factor(ifelse(Dataset==1, 1, 0)),
                Gender = as.numeric(ifelse(Gender=="Female", 0, 1)),
                Albumin_and_Globulin_Ratio = ifelse(is.na(Albumin_and_Globulin_Ratio),
                                                    median(Albumin_and_Globulin_Ratio,na.rm=TRUE),
                                                    Albumin_and_Globulin_Ratio) # Replace NA with median value
              ) %>% select(-Dataset)
```

Comment

1. Check the distribution of columns/predictors.
2. Check if there are any data elements with very few non-unique values or close to zero variation

```
str(liver)
```

```
## 'data.frame':    583 obs. of  11 variables:
##  $ Age                       : int  65 62 62 58 72 46 26 29 17 55 ...
##  $ Gender                    : num  0 1 1 1 1 1 0 0 1 1 ...
##  $ 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  187 699 490 182 195 208 154 202 202 290 ...
##  $ 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 ...
##  $ Diagnosis                 : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 1 2 ...
```

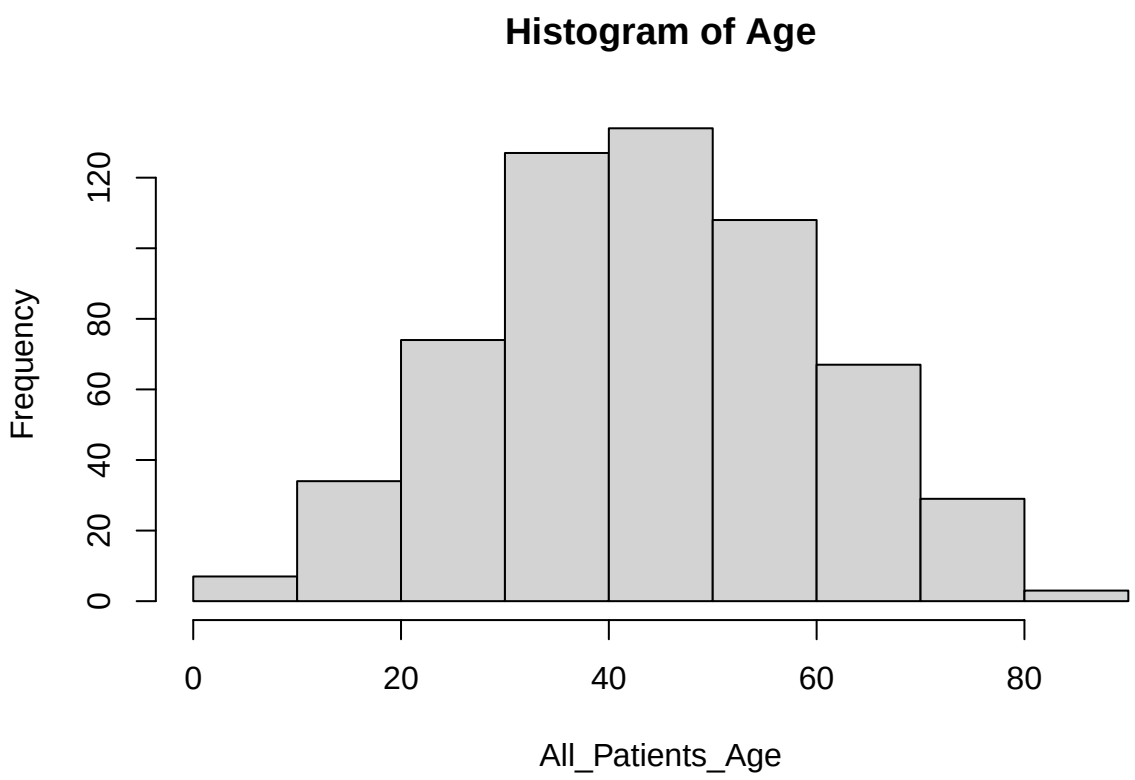
```r
sum(liver$Diagnosis==1)
```

```
## [1] 416
```

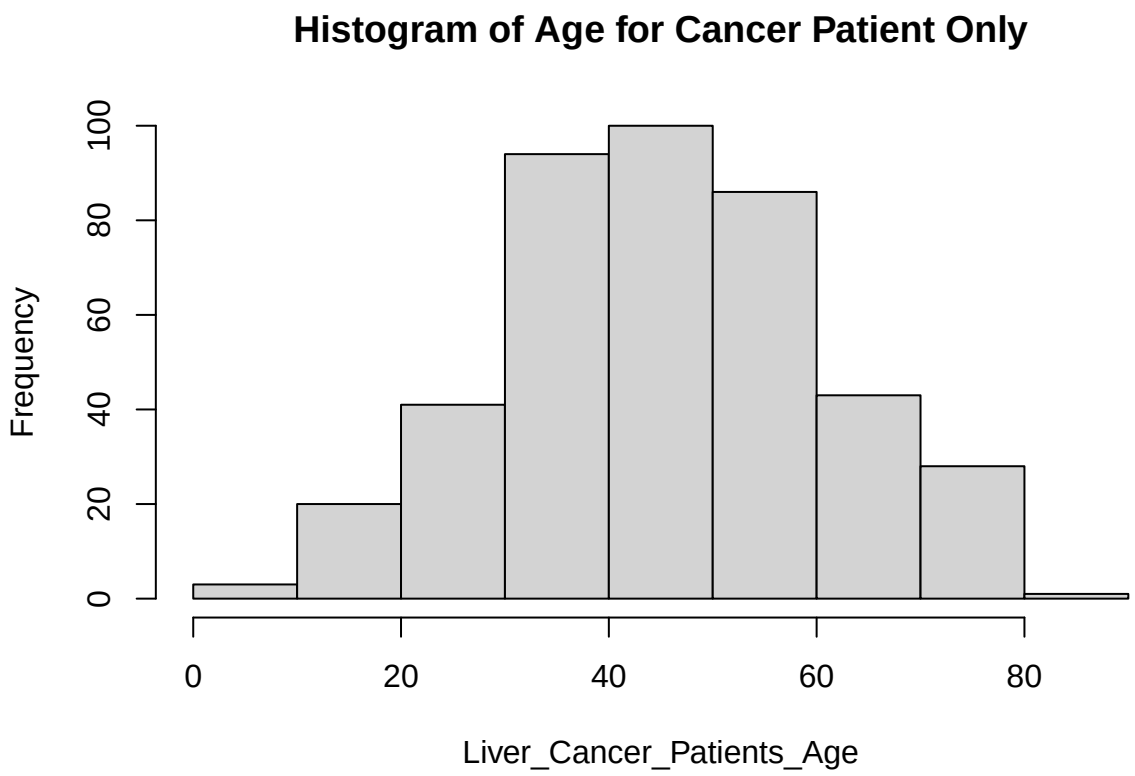
```r
sum(liver$Diagnosis==0)
```

```
## [1] 167
```

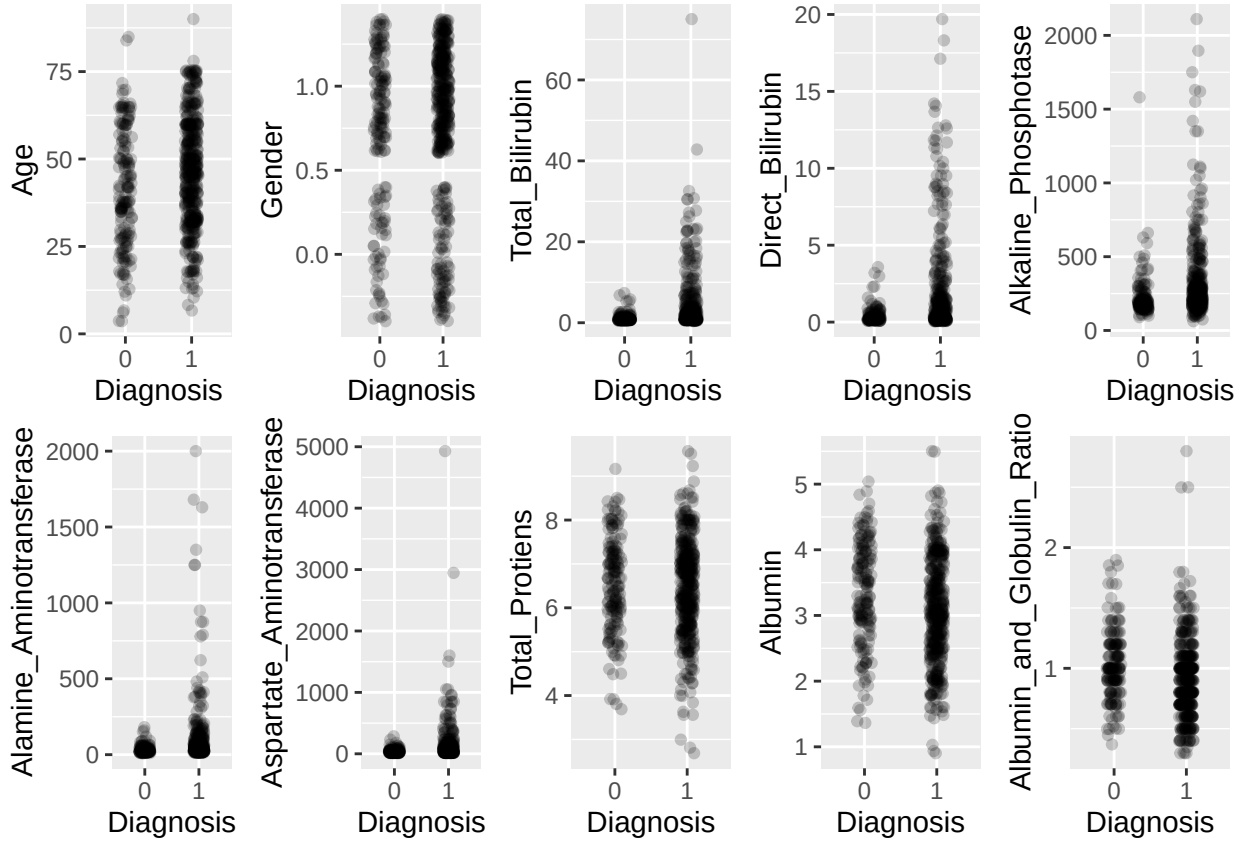
```r
All_Patients_Age <- liver$Age
hist(All_Patients_Age, main="Histogram of Age")
```




```r
Cancer <- liver %>% filter(Diagnosis==1)
Liver_Cancer_Patients_Age <- Cancer$Age
hist(Liver_Cancer_Patients_Age, main="Histogram of Age for Cancer Patient Only")
```




```
#Distribution of all predictors vs target value

p1 <- liver %>% ggplot(aes(Diagnosis, Age)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p2 <- liver %>% ggplot(aes(Diagnosis,Gender)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p3 <- liver %>% ggplot(aes(Diagnosis,Total_Bilirubin)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p4 <- liver %>% ggplot(aes(Diagnosis,Direct_Bilirubin)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p5 <- liver %>% ggplot(aes(Diagnosis,Alkaline_Phosphotase)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p6 <- liver %>% ggplot(aes(Diagnosis,Alamine_Aminotransferase)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p7 <- liver %>% ggplot(aes(Diagnosis,Aspartate_Aminotransferase)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p8 <- liver %>% ggplot(aes(Diagnosis,Total_Protiens)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p9 <- liver %>% ggplot(aes(Diagnosis,Albumin)) +
                geom_jitter(width = 0.1, alpha = 0.2)
p10 <- liver %>% ggplot(aes(Diagnosis,Albumin_and_Globulin_Ratio)) +
                geom_jitter(width = 0.1, alpha = 0.2)


grid.arrange(p1, p2,p3,p4,p5,p6,p7,p8,p9,p10, nrow=2, ncol = 5)
```




```
#Check if there is any data element with very few non-unique values or close to zero variation

nearZeroVar(liver)
```

```
## integer(0)
```

Finding

None of a data element has a few non-unique values and close to zero variation

Comment

Correlation analysis is performed on the target value and other 10 predictors on the original data set liver.df, to see if there are correlation exist between any of the predictors and the target value/column.

```
liver.df <- liver.df %>% mutate(Gender=as.numeric( ifelse(Gender=="Female", 0, 1) ))

cor(liver.df, use="pairwise.complete")
```

```
##                                    Age       Gender Total_Bilirubin
## Age                         1.000000000  0.056560251     0.011762651
## Gender                      0.056560251  1.000000000     0.089290824
## Total_Bilirubin             0.011762651  0.089290824     1.000000000
## Direct_Bilirubin            0.007529138  0.100436436     0.874617930
## Alkaline_Phosphotase        0.080424612 -0.027496175     0.206668795
## Alamine_Aminotransferase   -0.086882759  0.082332236     0.214064740
## Aspartate_Aminotransferase -0.019909857  0.080336244     0.237831323
## Total_Protiens             -0.187461261 -0.089121043    -0.008099343
## Albumin                    -0.265924361 -0.093799266    -0.222250406
## Albumin_and_Globulin_Ratio -0.216408346 -0.003424034    -0.206267186
## Dataset                    -0.137350627 -0.082415914    -0.220207565
```

```
##                            Direct_Bilirubin Alkaline_Phosphotase
## Age                          0.0075291381          0.08042461
## Gender                       0.1004364357         -0.02749618
## Total_Bilirubin              0.8746179301          0.20666880
## Direct_Bilirubin             1.0000000000          0.23493871
## Alkaline_Phosphotase         0.2349387058          1.00000000
## Alamine_Aminotransferase     0.2338940545          0.12567995
## Aspartate_Aminotransferase   0.2575439811          0.16719590
## Total_Protiens              -0.0001387414         -0.02851436
## Albumin                     -0.2285305729         -0.16545287
## Albumin_and_Globulin_Ratio  -0.2001246852         -0.23416650
## Dataset                     -0.2460463416         -0.18486561
##                            Alamine_Aminotransferase Aspartate_Aminotransferase
## Age                                     -0.08688276                -0.01990986
## Gender                                   0.08233224                 0.08033624
## Total_Bilirubin                          0.21406474                 0.23783132
## Direct_Bilirubin                         0.23389405                 0.25754398
## Alkaline_Phosphotase                     0.12567995                 0.16719590
## Alamine_Aminotransferase                 1.00000000                 0.79196568
## Aspartate_Aminotransferase               0.79196568                 1.00000000
## Total_Protiens                          -0.04251819                -0.02564537
## Albumin                                 -0.02974167                -0.08529030
## Albumin_and_Globulin_Ratio              -0.00237499                -0.07003983
## Dataset                                 -0.16341616                -0.15193375
##                            Total_Protiens     Albumin
## Age                        -0.1874612615 -0.26592436
## Gender                     -0.0891210427 -0.09379927
## Total_Bilirubin            -0.0080993434 -0.22225041
## Direct_Bilirubin           -0.0001387414 -0.22853057
## Alkaline_Phosphotase       -0.0285143556 -0.16545287
## Alamine_Aminotransferase   -0.0425181903 -0.02974167
## Aspartate_Aminotransferase -0.0256453651 -0.08529030
## Total_Protiens              1.0000000000  0.78405334
## Albumin                     0.7840533354  1.00000000
## Albumin_and_Globulin_Ratio  0.2348871811  0.68963234
## Dataset                     0.0350082358  0.16138782
##                            Albumin_and_Globulin_Ratio     Dataset
## Age                                      -0.216408346 -0.13735063
## Gender                                   -0.003424034 -0.08241591
## Total_Bilirubin                          -0.206267186 -0.22020756
## Direct_Bilirubin                         -0.200124685 -0.24604634
## Alkaline_Phosphotase                     -0.234166499 -0.18486561
## Alamine_Aminotransferase                 -0.002374990 -0.16341616
## Aspartate_Aminotransferase               -0.070039828 -0.15193375
## Total_Protiens                            0.234887181  0.03500824
## Albumin                                   0.689632342  0.16138782
## Albumin_and_Globulin_Ratio                1.000000000  0.16313136
## Dataset                                   0.163131363  1.00000000
```

Finding

1. The target value (named "Dataset" in liver.df) is not independent with all predictors, e.g, the target value 'Dataset' has 24.6% of coefficient with Direct_Bilirubin…
2. No further t-testing will be done on predictors to extract a subset for ML models training in order to improve accuracy based on a certain threshold of p-value.

Comment

1. From the summary statistic analysis of data set “liver.df, it is observed that the unit of each column is different, there exists big variance of the values among all predictors.
2. Will center and scale the columns of the predictors of the data set of ‘liver’.
3. set.seed(1, sample.kind = “Rounding”) # simulate R 3.5, there is warning message comes out. This is not a warning or a cause for alarm - it’s a confirmation that R is using the alternate seed generation method, and it should expect #to receive this message in your console.

```
# centering and scaling on all predictors

options(digits = 3)

#set.seed(1) # if using R 3.5 or earlier
set.seed(1, sample.kind = "Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# setp 1 - center and scale 10 predictors

Predictors <- liver %>%  select(-Diagnosis)
Diagnosis <- liver$Diagnosis  ###%>%  select(Diagnosis)

x_centered <- sweep(Predictors, 2, colMeans(Predictors))
x_scaled <- sweep(x_centered, 2, colSds(as.matrix(Predictors)), FUN = "/")
```

Comment

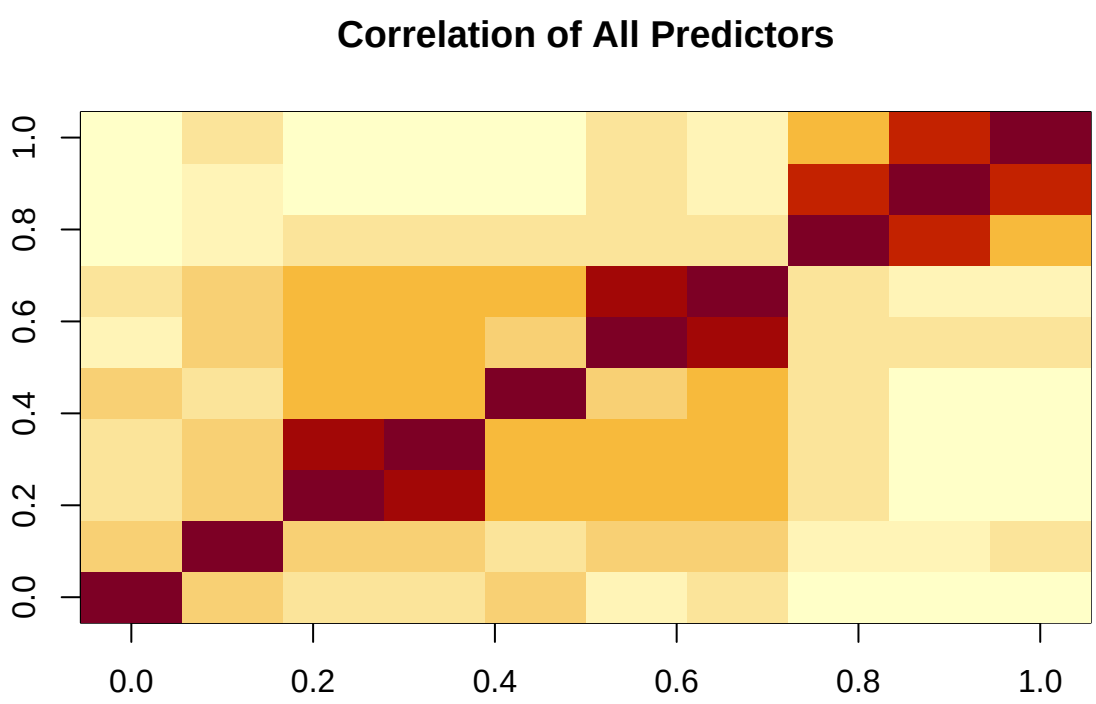
1. Next, correlation analysis will be performed on the scaled data set ‘x_scaled’.
2. Virtually present the correlation of the predictors/features of the scaled data set ‘x_scaled’

```
cor(x_scaled, use="pairwise.complete") ##%>% knitr::kable()
```

```
##                               Age   Gender Total_Bilirubin Direct_Bilirubin
## Age                       1.00000  0.05656          0.0118         0.007529
## Gender                    0.05656  1.00000          0.0893         0.100436
## Total_Bilirubin           0.01176  0.08929          1.0000         0.874618
## Direct_Bilirubin          0.00753  0.10044          0.8746         1.000000
## Alkaline_Phosphotase      0.08042 -0.02750          0.2067         0.234939
## Alamine_Aminotransferase -0.08688  0.08233          0.2141         0.233894
## Aspartate_Aminotransferase -0.01991  0.08034          0.2378         0.257544
## Total_Protiens            -0.18746 -0.08912         -0.0081        -0.000139
## Albumin                   -0.26592 -0.09380         -0.2223        -0.228531
## Albumin_and_Globulin_Ratio -0.21597 -0.00318         -0.2060        -0.199850
##                           Alkaline_Phosphotase Alamine_Aminotransferase
## Age                                     0.0804                 -0.08688
## Gender                                 -0.0275                  0.08233
## Total_Bilirubin                         0.2067                  0.21406
## Direct_Bilirubin                        0.2349                  0.23389
## Alkaline_Phosphotase                    1.0000                  0.12568
## Alamine_Aminotransferase                0.1257                  1.00000
## Aspartate_Aminotransferase              0.1672                  0.79197
## Total_Protiens                         -0.0285                 -0.04252
## Albumin                                -0.1655                 -0.02974
## Albumin_and_Globulin_Ratio             -0.2338                 -0.00225
##                           Aspartate_Aminotransferase Total_Protiens Albumin
## Age                                           -0.0199      -0.187461 -0.2659
## Gender                                         0.0803      -0.089121 -0.0938
```

```
## Total_Bilirubin                          0.2378      -0.008099 -0.2223
## Direct_Bilirubin                         0.2575      -0.000139 -0.2285
## Alkaline_Phosphotase                     0.1672      -0.028514 -0.1655
## Alamine_Aminotransferase                 0.7920      -0.042518 -0.0297
## Aspartate_Aminotransferase               1.0000      -0.025645 -0.0853
## Total_Protiens                          -0.0256       1.000000  0.7841
## Albumin                                 -0.0853       0.784053  1.0000
## Albumin_and_Globulin_Ratio              -0.0699       0.233828  0.6861
##                           Albumin_and_Globulin_Ratio
## Age                                        -0.21597
## Gender                                     -0.00318
## Total_Bilirubin                            -0.20602
## Direct_Bilirubin                           -0.19985
## Alkaline_Phosphotase                       -0.23378
## Alamine_Aminotransferase                   -0.00225
## Aspartate_Aminotransferase                 -0.06993
## Total_Protiens                              0.23383
## Albumin                                     0.68609
## Albumin_and_Globulin_Ratio                  1.00000
```

```
image(as.matrix(cor(x_scaled, use="pairwise.complete")), axes = TRUE,
      main = "Correlation of All Predictors")
```



Observation

1. Total_Bilirubin and Direct_Bilirubin is highly correlated (correlated coefficient 0.8746);
2. Alamine_Aminotransferase and Aspartate_Aminotransferase is highly correlated (correlated coefficient 0.7920);
3. Total_Protiens is highly correlated with |Albumin (correlated coefficient 0.784053)
4. Albumin_and_Globulin_Ratio is highly correlated with Albumin (correlated coefficient 0.6861)

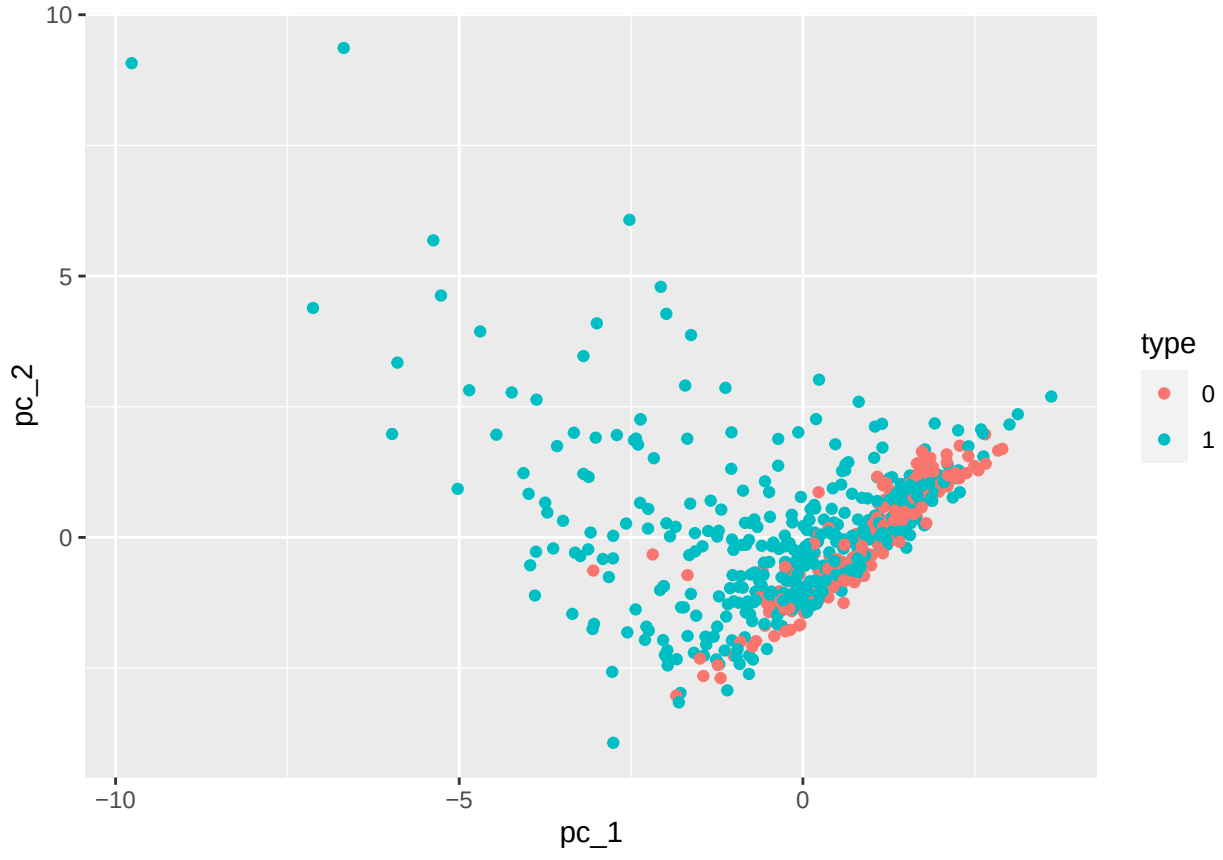
```r
# principal component analysis

pca <- prcomp(x_scaled)
summary(pca)
```

```
## Importance of components:
##                           PC1   PC2   PC3   PC4   PC5    PC6    PC7    PC8
## Standard deviation     1.666 1.424 1.170 1.028 0.959 0.8957 0.8145 0.4509
## Proportion of Variance 0.278 0.203 0.137 0.106 0.092 0.0802 0.0664 0.0203
## Cumulative Proportion  0.278 0.480 0.617 0.723 0.815 0.8951 0.9615 0.9818
##                           PC9    PC10
## Standard deviation     0.3544 0.23707
## Proportion of Variance 0.0126 0.00562
## Cumulative Proportion  0.9944 1.00000
```

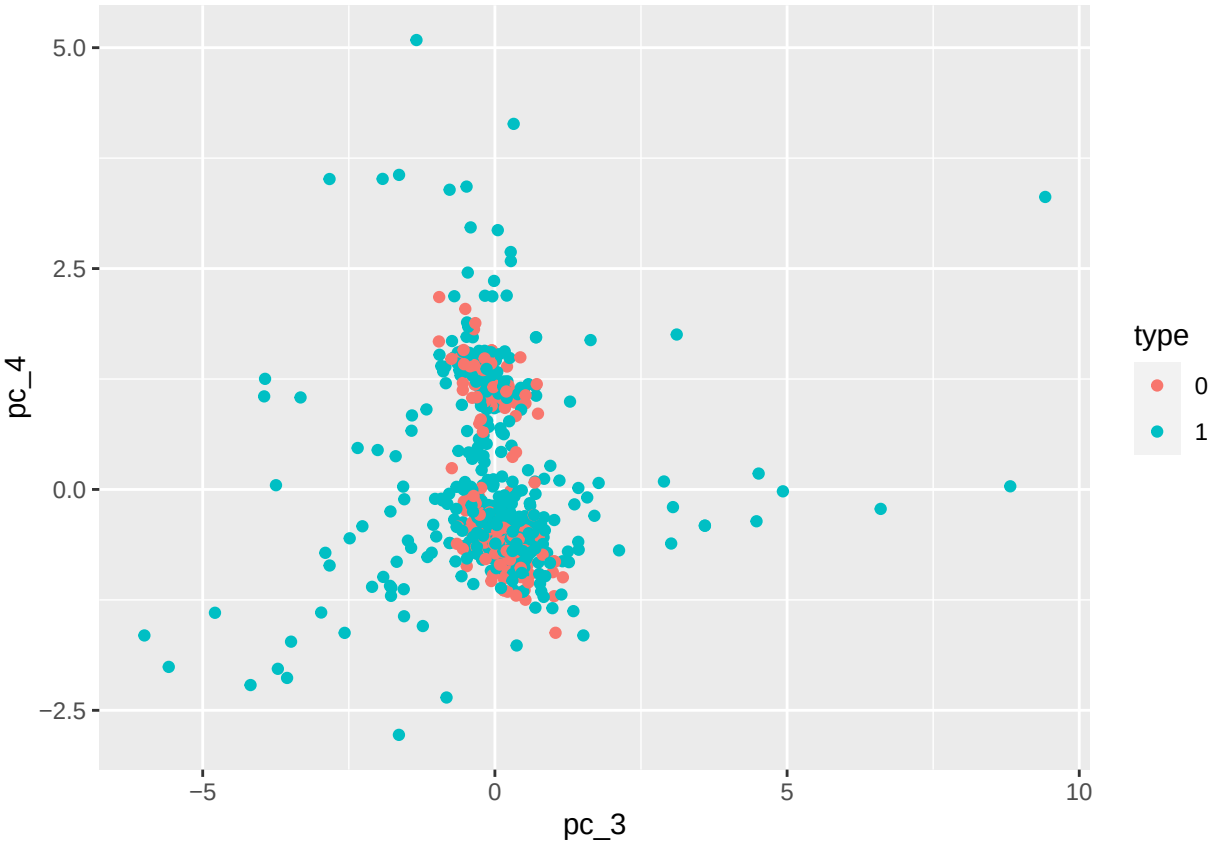
Observations

1. PC1 & PC2 cumulative proportion only accounts for 48% of the variance
2. 6 principal components are needed to explain about 89% of the variance.
3. 7 principal components are needed to explain about 95% of the variance.
4. 8 principal components are needed to explain about 98% of the variance.

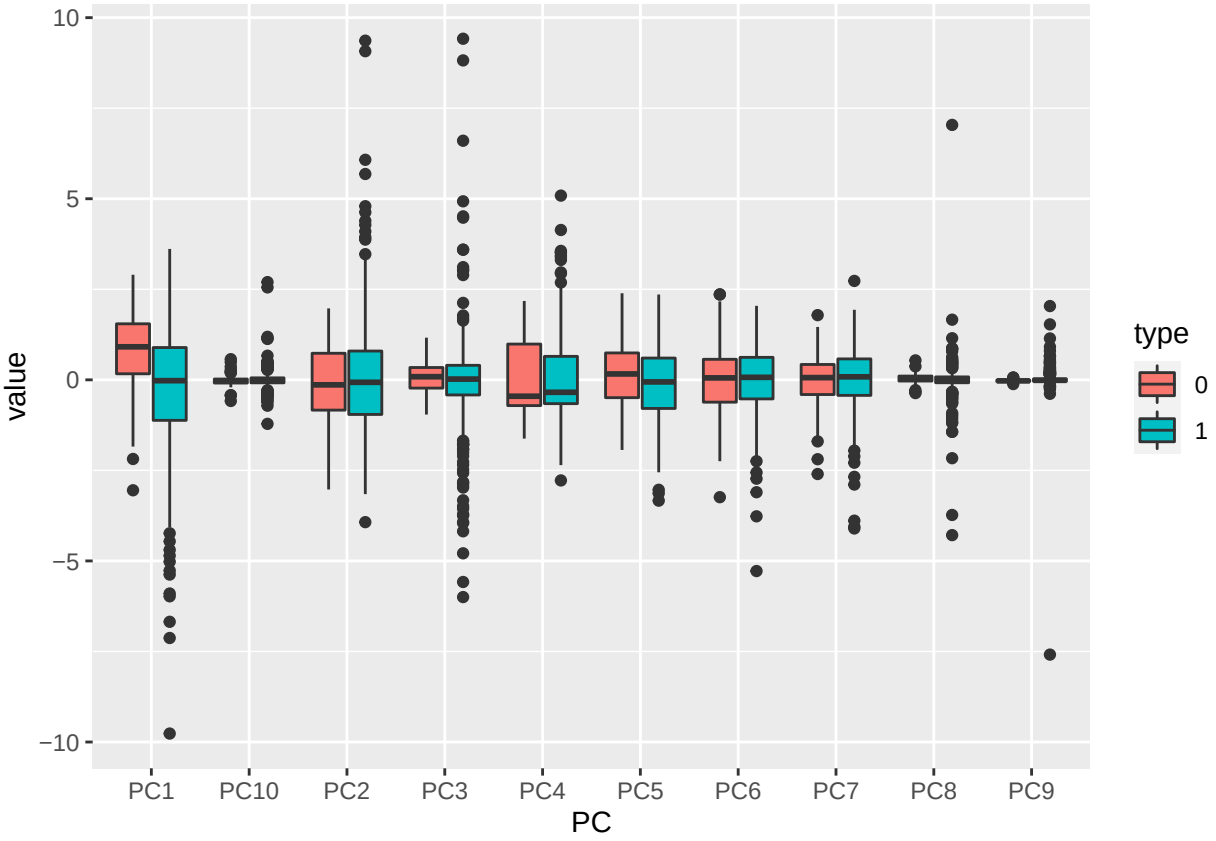
```r
# Some basic analysis on principal component analysis

# 1. By looking at the distribution of PC1 and PC2, Liver cancer patients
#    tend to have larger values of PC2 than non-liver cancer patient

data.frame(pc_1 = pca$x[,1], pc_2 = pca$x[,2],
           type=Diagnosis ) %>%
  ggplot(aes(pc_1, pc_2, color = type)) +
  geom_point()
```



```
#2. Liver cancer patients tend to have larger variance of PC3
#   than non-liver cancer patient.

data.frame(pc_3 = pca$x[,3], pc_4 = pca$x[,4],
           type=Diagnosis ) %>%
  ggplot(aes(pc_3, pc_4, color = type)) +
  geom_point()
```




```
#3. Distribution of IQRs from PC 1 through PC 10

data.frame(type = Diagnosis, pca$x[,1:10]) %>%
  gather(key = "PC", value = "value", -type) %>%
  ggplot(aes(PC, value, fill = type)) +
  geom_boxplot()
```



Findings

1. All IQRs of PC1 - PC10 overlapped categorized by the liver cancer (1, 0)
2. PC1 has biggest difference of IQR categorized by the liver cancer (1, 0), but not significant enough to predict the liver cancer.
3. All features/predictors will be included for 4 training ML models

```
#Split the Scaled data to train set and test set

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

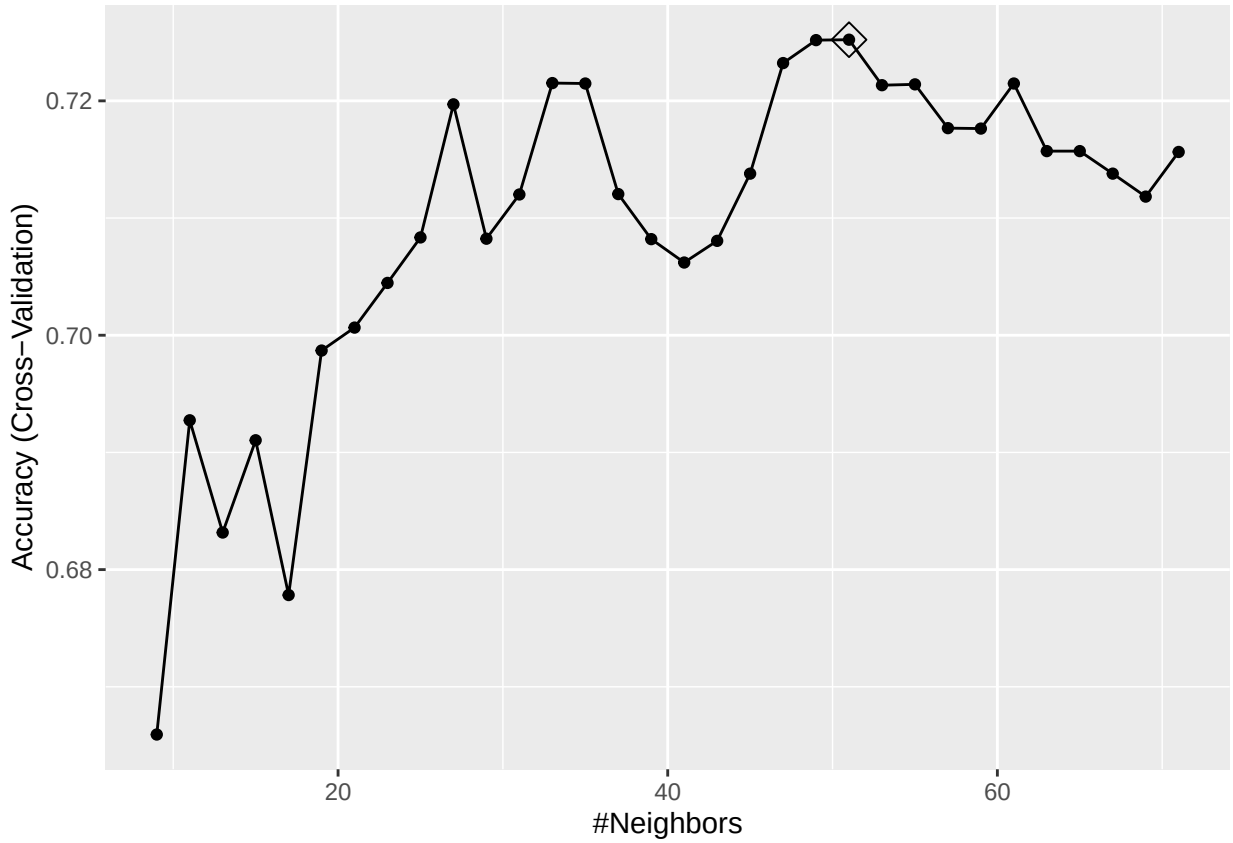
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(Diagnosis, times = 1, p = 0.1, list = FALSE)
test_x <- x_scaled[test_index,]   ## str(test_x)
test_y <- Diagnosis[test_index]    ## length(test_y)

train_x <- x_scaled[-test_index,]   ## str(train_x)
train_y <- Diagnosis[-test_index]   ## length(train_y)
```

```
#Logistic regression model

# set.seed(1) if using R 3.5 or earlier
set.seed(1, sample.kind = "Rounding") # if using R 3.6 or later
train_glm <- train(train_x, train_y, method = "glm")
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
glm_pred <- predict(train_glm, test_x)
logistic_Accuracy2 <-  mean(glm_pred == test_y)
```

```
glm_sensitivity <- rbind(sensitivity(glm_pred, test_y), specificity(glm_pred, test_y))
rownames( glm_sensitivity )  <-  c("sensitivity","specificity")
colnames( glm_sensitivity )  <-  c("Logistic Regression")

# Top 5 important Predictors

a <- varImp(train_glm)[[1]]
varImp_df_glm <- data.frame(matrix(c(rownames(a), as.numeric(a[,1])), nrow=10, ncol=2,
                                   dimnames=list(c(seq(1:10)), c("Predictor", "Value"))))
varImp_df_glm <- varImp_df_glm %>% mutate(Value=as.numeric(Value)) %>% arrange(desc(Value))
rownames(varImp_df_glm) <- seq(1:10)
Top_5_Glm_Predictors <- varImp_df_glm[1:5,1]
Top_5_GLM_predictors <- data.frame(Top_5_Glm_Predictors)
```

```
#K-nearest neighbors model K-nearest neighbors model

# set.seed(1)
set.seed(1, sample.kind = "Rounding") # simulate R 3.5
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(train_x, train_y, method = "knn",
                   tuneGrid = data.frame(k = seq(9, 71, 2)),
                   trControl = control)
ggplot(train_knn, highlight=TRUE, main="k nearst neighbor Model accuracy vs k")
```

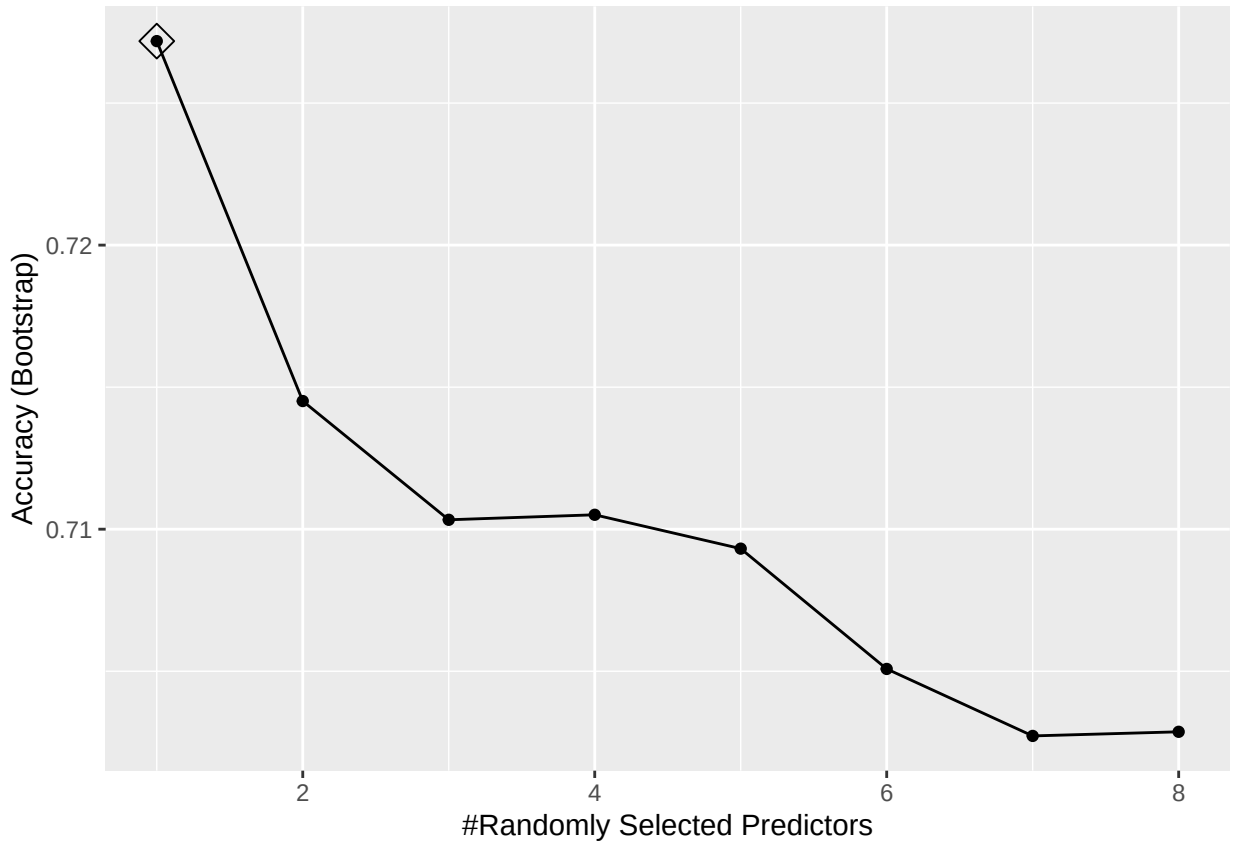



```
train_knn$bestTune
```

```
##     k
## 22 51
```

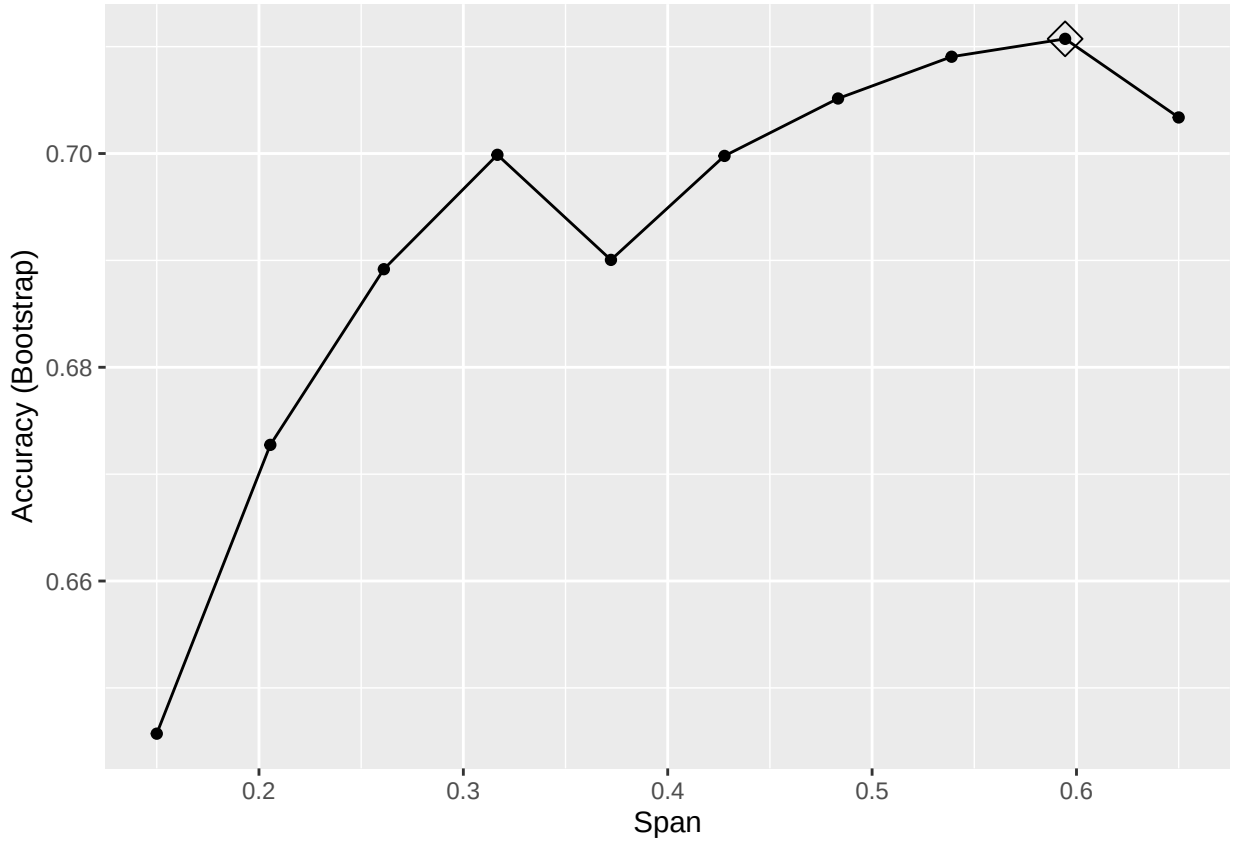
```r
knn_pred <- predict( train_knn, test_x)
K_nearst_Accuracy2 <- mean(knn_pred== test_y)

knn_sensitivity <- rbind(sensitivity(knn_pred, test_y), specificity(knn_pred, test_y))
rownames( knn_sensitivity )  <-  c("sensitivity","specificity")
colnames( knn_sensitivity )  <-  c("K_Nearst_Neighbor")

# Top 5 important Predictors
a <- varImp(train_knn)[[1]]
varImp_df_knn <- data.frame(matrix(c(rownames(a), as.numeric(a[,1])), nrow=10, ncol=2,
                                   dimnames=list(c(seq(1:10)), c("Predictor", "Value"))))
varImp_df_knn <- varImp_df_knn %>% mutate(Value=as.numeric(Value)) %>% arrange(desc(Value))
rownames(varImp_df_knn) <- seq(1:10)
Top_5_knn_Predictors <- varImp_df_knn[1:5,1]
Top_5_knn_predictors <- data.frame(Top_5_knn_Predictors)
```

```r
#random Forest model

# set.seed(1)
set.seed(1, sample.kind = "Rounding") # simulate R 3.5

tuning <- data.frame(mtry = c(1,2,3,4,5,6,7,8))
train_rf <- train(train_x, train_y,
                  method = "rf",
                  tuneGrid = tuning,
                  importance = TRUE)

ggplot(train_rf,highlight=TRUE, title="random Forest Model Acuracy distribution")
```



```
rf_pred <- predict(train_rf, test_x)
rf_Accuracy2 <- mean(rf_pred == test_y)

rf_sensitivity <- rbind(sensitivity(rf_pred, test_y), specificity(rf_pred, test_y))
rownames( rf_sensitivity )  <-  c("sensitivity","specificity")
colnames( rf_sensitivity )  <-  c("random Forest")

# Top 5 important Predictors
a <- varImp(train_rf)[[1]]
varImp_df_rf <- data.frame(matrix(c(rownames(a), as.numeric(a[,1])), nrow=10, ncol=2,
                                  dimnames=list(c(seq(1:10)), c("Predictor", "Value"))))
varImp_df_rf <- varImp_df_knn %>% mutate(Value=as.numeric(Value)) %>% arrange(desc(Value))
rownames(varImp_df_rf) <- seq(1:10)
Top_5_rf_Predictors <- varImp_df_rf[1:5,1]
Top_5_rf_predictors <- data.frame(Top_5_rf_Predictors)
```

```
#Local Polynomial Regression Model

# set.seed(1)
set.seed(1, sample.kind = "Rounding") # simulate R 3.5
grid <- expand.grid(span = seq(0.15, 0.65, len = 10), degree = 1)
train_loess <- train(train_x, train_y, method = "gamLoess", tuneGrid=grid)
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
ggplot(train_loess, highlight=TRUE)
```




```
loess_pred <- predict(train_loess, test_x)
Loess_Accuracy2 <- mean(loess_pred == test_y)

Loess_sensitivity <- rbind(sensitivity(loess_pred, test_y), specificity(loess_pred, test_y))
rownames( Loess_sensitivity )  <-  c("sensitivity","specificity")
```

```
colnames( Loess_sensitivity )  <-  c("Local Polynomial Regression")

#### Top 5 important Predictors ####
a <- varImp(train_loess)[[1]]
varImp_df_loess <- data.frame(matrix(c(rownames(a), as.numeric(a[,1])), nrow=10, ncol=2,
                                     dimnames=list(c(seq(1:10)), c("Predictor", "Value"))))
varImp_df_loess <- varImp_df_loess %>% mutate(Value=as.numeric(Value)) %>% arrange(desc(Value))
rownames(varImp_df_loess) <- seq(1:10)
Top_5_Loess_Predictors <- varImp_df_loess[1:5,1]
Top_5_Loess_predictors <- data.frame(Top_5_Loess_Predictors)
```

```
#Format/combine the results from all ML models

Accuracy_results <- data_frame(Method = "Logistic Regression Model", Accuracy = logistic_Accuracy2)
Accuracy_results <- bind_rows(Accuracy_results,
                              data_frame(Method="K nearst neighbors Model",
                                         Accuracy = K_nearst_Accuracy2 ))
Accuracy_results <- bind_rows(Accuracy_results,
                              data_frame(Method="Random Forest",
                                         Accuracy =  rf_Accuracy2 ))
Accuracy_results <- bind_rows(Accuracy_results,
                              data_frame(Method="Local Polynomial Regression Model",
                                         Accuracy =  Loess_Accuracy2 ))
Accuracy_results %>% knitr::kable(align='c')
```

Method	Accuracy
Logistic Regression Model	0.695
K nearst neighbors Model	0.712
Random Forest	0.746
Local Polynomial Regression Model	0.678

```
#Format Sensitivity and Specificity Results of all Models

cbind(glm_sensitivity , knn_sensitivity, rf_sensitivity, Loess_sensitivity) %>%
      knitr::kable(align='c')
```

	Logistic Regression	K_Nearst_Neighbor	random Forest	Local Polynomial Regression
sensitivity	0.294	0.059	0.353	0.353
specificity	0.857	0.976	0.905	0.810

```
#Format top 5 important predictors/features

cbind(Top_5_GLM_predictors, Top_5_knn_predictors, Top_5_rf_predictors, Top_5_Loess_predictors)%>%
      knitr::kable()
```

Top_5_Glm_Predictors	Top_5_knn_Predictors	Top_5_rf_Predictors	Top_5_Loess_Predictors
Age	Total_Bilirubin	Total_Bilirubin	Total_Protiens
Alamine_Aminotransferase	Aspartate_Aminotransferase	Aspartate_Aminotransferase	Alkaline_Phosphotase
Total_Protiens	Direct_Bilirubin	Direct_Bilirubin	Total_Bilirubin
Albumin	Alamine_Aminotransferase	Alamine_Aminotransferase	Age
Direct_Bilirubin	Alkaline_Phosphotase	Alkaline_Phosphotase	Direct_Bilirubin

Conclusion

1. The prediction on the test data by applying 4 machine learning models tend to have lower sensitivity but high specificity, which means the models tend to predict accurately on the true negative liver cancer patients, but not the true positive liver cancer patients
2. Currently, the data set is pretty small which only contains 583 instances. Larger data set is encouraged to be collected for training ML models. Giver the reason that in Indian, only around 3-5 instances of liver cancer per 100,000 persons which means the prevalence of liver cancer is very low in India. Larger data might can overcome the possible imbalanced small data sets.
3. It is also encouraged to collect other useful/critical information such as patient family cancer history, patient family liver cancer history; patient Hepatitis B/C history; alcohol intake history; weight change information, nutrition habit, etc. for comprehensive analysis on the liver cancer study.

Reference

1. HarvardX PH125.8x Data Science: Machine Learning – 7.1 Final Assessment: Breast Cancer Prediction Project (Verified Learners only)
2. Liver Cancer 101: What are the Five Gospel Truths about the Disease