Autism Spectrum Disorder Screening

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library('dplyr')
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library('tidyr')
library('ggplot2')
library('caret')
```

```
## Loading required package: lattice
```

```
library('e1071')
library('rpart')
library('neuralnet')
```

```
##
## Attaching package: 'neuralnet'
```

```
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
library('caretEnsemble')
```

```
##
## Attaching package: 'caretEnsemble'
## The following object is masked from ’package:ggplot2’:
##
##     autoplot
```

```
# reading the autism dataset
aut_data <- read.csv("/Users/sanjanagorlla/Desktop/Autism project/autism_screening.csv")
```

```
# visualising first few rows of the dataset
head(aut_data)
```

```
##   A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score
## 1        1        1        1        1        0        0        1        1
## 2        1        1        0        1        0        0        0        1
## 3        1        1        0        1        1        0        1        1
## 4        1        1        0        1        0        0        1        1
## 5        1        0        0        0        0        0        0        1
## 6        1        1        1        1        1        0        1        1
##   A9_Score A10_Score age gender      ethnicity jundice austim contry_of_res
## 1        0         0  26      f White-European      no     no United States
## 2        0         1  24      m         Latino      no    yes        Brazil
## 3        1         1  27      m         Latino     yes    yes         Spain
## 4        0         1  35      f White-European      no    yes United States
## 5        0         0  40      f              ?      no     no         Egypt
## 6        1         1  36      m         Others     yes     no United States
##   used_app_before result    age_desc relation Class.ASD
## 1              no      6 18 and more     Self        NO
## 2              no      5 18 and more     Self        NO
## 3              no      8 18 and more   Parent       YES
## 4              no      6 18 and more     Self        NO
## 5              no      2 18 and more        ?        NO
## 6              no      9 18 and more     Self       YES
```

```
# checking the dimension of the dataset
dim(aut_data)
```

```
## [1] 704  21
```

```
# the datset has 704 rows and 21 columns

# checking the type of all variables in the dataset
str(aut_data)
```

```
## ’data.frame’:    704 obs. of  21 variables:
##  $ A1_Score       : int  1 1 1 1 1 1 0 1 1 1 ...
##  $ A2_Score       : int  1 1 1 1 0 1 1 1 1 1 ...
##  $ A3_Score       : int  1 0 0 0 0 1 0 1 0 1 ...
##  $ A4_Score       : int  1 1 1 1 0 1 0 1 0 1 ...
##  $ A5_Score       : int  0 0 1 0 0 1 0 0 1 0 ...
##  $ A6_Score       : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ A7_Score       : int  1 0 1 1 0 1 0 0 0 1 ...
##  $ A8_Score       : int  1 1 1 1 1 1 1 0 1 1 ...
##  $ A9_Score       : int  0 0 1 0 0 1 0 1 1 1 ...
##  $ A10_Score      : int  0 1 1 1 0 1 0 0 1 0 ...
##  $ age            : num  26 24 27 35 40 36 17 64 29 17 ...
##  $ gender         : chr  "f" "m" "m" "f" ...
##  $ ethnicity      : chr  "White-European" "Latino" "Latino" "White-European" ...
##  $ jundice        : chr  "no" "no" "yes" "no" ...
##  $ austim         : chr  "no" "yes" "yes" "yes" ...
##  $ contry_of_res  : chr  "United States" "Brazil" "Spain" "United States" ...
##  $ used_app_before: chr  "no" "no" "no" "no" ...
##  $ result         : num  6 5 8 6 2 9 2 5 6 8 ...
##  $ age_desc       : chr  "18 and more" "18 and more" "18 and more" "18 and more" ...
##  $ relation       : chr  "Self" "Self" "Parent" "Self" ...
##  $ Class.ASD      : chr  "NO" "NO" "YES" "NO" ...
```

```
# statistical summary of all the variables
summary(aut_data)
```

```
##     A1_Score        A2_Score        A3_Score        A4_Score
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :0.0000   Median :0.0000   Median :0.0000
##  Mean   :0.7216   Mean   :0.4531   Mean   :0.4574   Mean   :0.4957
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##     A5_Score        A6_Score        A7_Score        A8_Score
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :1.0000
##  Mean   :0.4986   Mean   :0.2841   Mean   :0.4176   Mean   :0.6491
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##     A9_Score       A10_Score           age           gender
##  Min.   :0.0000   Min.   :0.0000   Min.   : 17.0   Length:704
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 21.0   Class :character
##  Median :0.0000   Median :1.0000   Median : 27.0   Mode  :character
##  Mean   :0.3239   Mean   :0.5739   Mean   : 29.7
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.: 35.0
##  Max.   :1.0000   Max.   :1.0000   Max.   :383.0
##                                    NA's   :2
##   ethnicity           jundice             austim            contry_of_res
##  Length:704         Length:704         Length:704         Length:704
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##  used_app_before       result         age_desc             relation
##  Length:704         Min.   : 0.000   Length:704         Length:704
##  Class :character   1st Qu.: 3.000   Class :character   Class :character
##  Mode  :character   Median : 4.000   Mode  :character   Mode  :character
##                     Mean   : 4.875
##                     3rd Qu.: 7.000
##                      Max.   :10.000
##
##   Class.ASD
##  Length:704
##  Class :character
##  Mode  :character
##
##
##
##
```

```
# the max value of age is 383 which is invalid
# therefore removing the row with age = 383
rownames(aut_data[aut_data$age == 383,])
```

```
## [1] "53"   "NA"   "NA.1"
```

```
aut_data <- aut_data[-53,]
max(aut_data$age, na.rm = TRUE) # now the maximum value for age is 64
```

```
## [1] 64
```

```
# count of numerical columns
length(select_if(aut_data,is.numeric))
```

```
## [1] 12
```

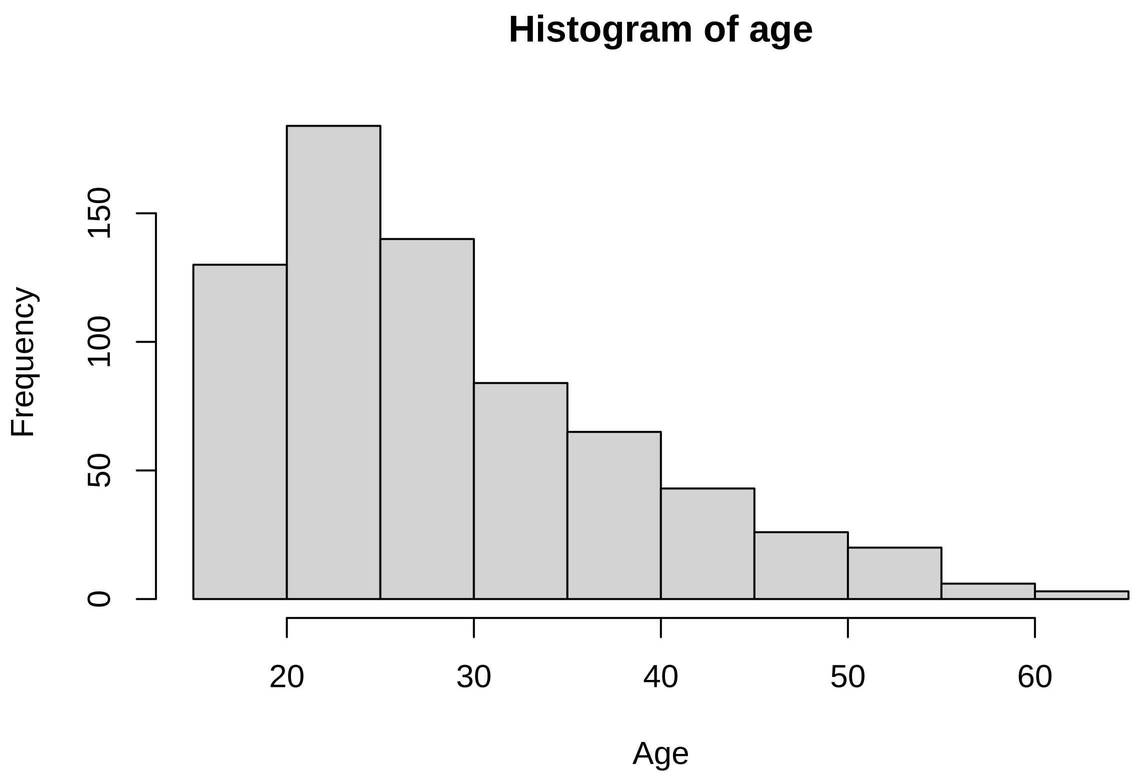
```
# there are 12 numerical columns in the dataset

# count of categorical columns
length(select_if(aut_data,is.character))
```

```
## [1] 9
```

```
# there are 9 categorical columns in the dataset

# checking for the distribution of all the variables
#library("psych")
#pairs.panels(aut_data)

# checking for missing values in the dataset
colSums(is.na(aut_data))
```

```
##        A1_Score        A2_Score        A3_Score        A4_Score        A5_Score
##               0               0               0               0               0
##        A6_Score        A7_Score        A8_Score        A9_Score       A10_Score
##               0               0               0               0               0
##             age          gender       ethnicity         jundice          austim
##               2               0               0               0               0
##   contry_of_res used_app_before          result        age_desc        relation
##               0               0               0               0               0
##       Class.ASD
##               0
```

```
# there are only 2 missing values for age in the entire dataset

# checking for the distribution of age column
hist(aut_data$age, main = 'Histogram of age', xlab = "Age")
```



```
# the distribution of age is positively skewed therefore we will impute the values
# using median

# imputing missing value with median age
aut_data$age[is.na(aut_data$age)] <- median(aut_data$age, na.rm = TRUE)
sum(is.na(aut_data$age))
```

```
## [1] 0
```

```
# selecting the continuos columns
num_aut_data <- select_if(aut_data, is.numeric)

# selecting the categorical columns
cat_aut_data <- select_if(aut_data, is.character)

# checking for count of unique values in categorical variables
cat_aut_data %>% summarise_all(n_distinct)
```

```
##   gender ethnicity jundice austim contry_of_res used_app_before age_desc
## 1      2        12       2      2            67               2        1
##   relation Class.ASD
## 1        6         2
```

```
# Below are the column names with the count of unique values:
# gender - 2
# etnicity - 12
# jundice - 2
# autism - 2
# country_of_res - 67
# used_app_before - 2
# age_desc - 1
# relation - 6
# Class.ASD - 2

# since age_desc has only one unique value it is of no use, so we can drop it
cat_aut_data <- cat_aut_data[,-7]

# further checking unique values in each column
unique(cat_aut_data$gender)
```

```
## [1] "f" "m"
```

```
unique(cat_aut_data$ethnicity)
```

```
##  [1] "White-European"  "Latino"           "?"                "Others"
##  [5] "Black"           "Asian"            "Middle Eastern " "Pasifika"
##  [9] "South Asian"     "Hispanic"         "Turkish"          "others"
```

```
# in ethnicity column there is a '?' that is an invalid value and 'Other' and 'others'
# are treated as different values, although they should be treated as same

# replacing '?' and 'others' with 'Others'
cat_aut_data$ethnicity[cat_aut_data$ethnicity == "?"] <- "Others"
cat_aut_data$ethnicity[cat_aut_data$ethnicity == "others"] <- "Others"
unique(cat_aut_data$ethnicity)
```

```
##  [1] "White-European"  "Latino"           "Others"           "Black"
##  [5] "Asian"           "Middle Eastern " "Pasifika"         "South Asian"
##  [9] "Hispanic"        "Turkish"
```

```
unique(cat_aut_data$jundice)
```

```
## [1] "no"  "yes"
```

```
unique(cat_aut_data$austim)
```

```
## [1] "no"  "yes"
```

```r
unique(cat_aut_data$contry_of_res)
```

```
##  [1] "United States"        "Brazil"               "Spain"
##  [4] "Egypt"                "New Zealand"          "Bahamas"
##  [7] "Burundi"              "Austria"              "Argentina"
## [10] "Jordan"               "Ireland"              "United Arab Emirates"
## [13] "Afghanistan"          "Lebanon"              "United Kingdom"
## [16] "South Africa"         "Italy"                "Pakistan"
## [19] "Bangladesh"           "Chile"                "France"
## [22] "China"                "Australia"            "Canada"
## [25] "Saudi Arabia"         "Netherlands"          "Romania"
## [28] "Sweden"               "Tonga"                "Oman"
## [31] "India"                "Philippines"          "Sri Lanka"
## [34] "Sierra Leone"         "Ethiopia"             "Viet Nam"
## [37] "Iran"                 "Costa Rica"           "Germany"
## [40] "Mexico"               "Russia"               "Armenia"
## [43] "Iceland"              "Nicaragua"            "Hong Kong"
## [46] "Japan"                "Ukraine"              "Kazakhstan"
## [49] "AmericanSamoa"        "Uruguay"              "Serbia"
## [52] "Portugal"             "Malaysia"             "Ecuador"
## [55] "Niger"                "Belgium"              "Bolivia"
## [58] "Aruba"                "Finland"              "Turkey"
## [61] "Nepal"                "Indonesia"            "Angola"
## [64] "Azerbaijan"           "Iraq"                 "Czech Republic"
## [67] "Cyprus"
```

```r
unique(cat_aut_data$used_app_before)
```

```
## [1] "no"  "yes"
```

```r
unique(cat_aut_data$relation)
```

```
## [1] "Self"                       "Parent"
## [3] "?"                          "Health care professional"
## [5] "Relative"                   "Others"
```

```r
# relation column also has am invalid value which is "?"
# replacing this "?" with "Others"
cat_aut_data$relation[cat_aut_data$relation == "?"] <- "Others"
unique(cat_aut_data$relation)
```

```
## [1] "Self"                       "Parent"
## [3] "Others"                     "Health care professional"
## [5] "Relative"
```

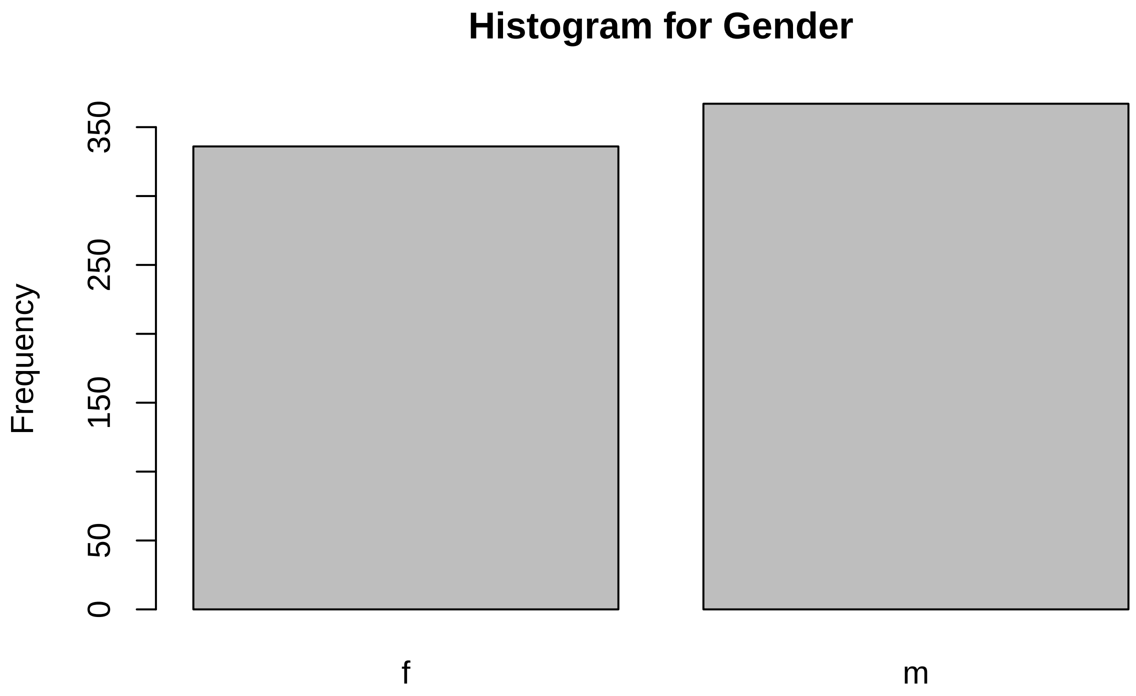
```r
unique(cat_aut_data$Class.ASD)
```

```
## [1] "NO"  "YES"
```

```
# checking the distribution of male and female in the data
table(cat_aut_data$gender)
```

```
##
##   f   m
## 336 367
```

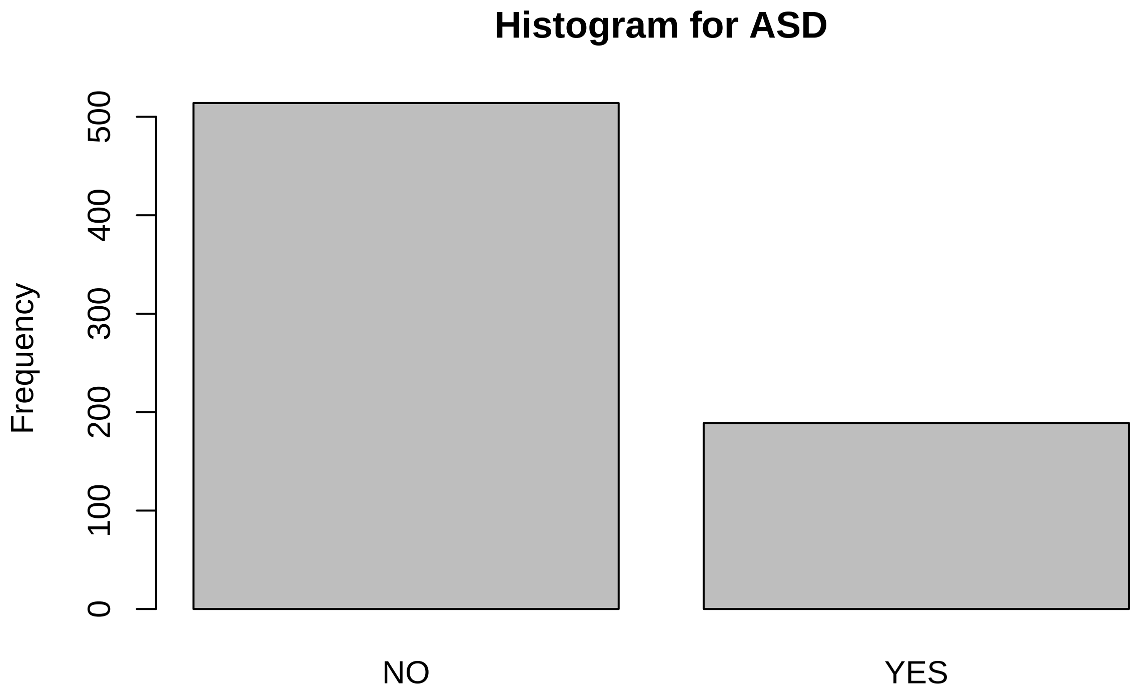
```
# there are 336 females and 367 males
# plotting the same on the histogram
barplot(table(cat_aut_data$gender), main = "Histogram for Gender", ylab = "Frequency")
```



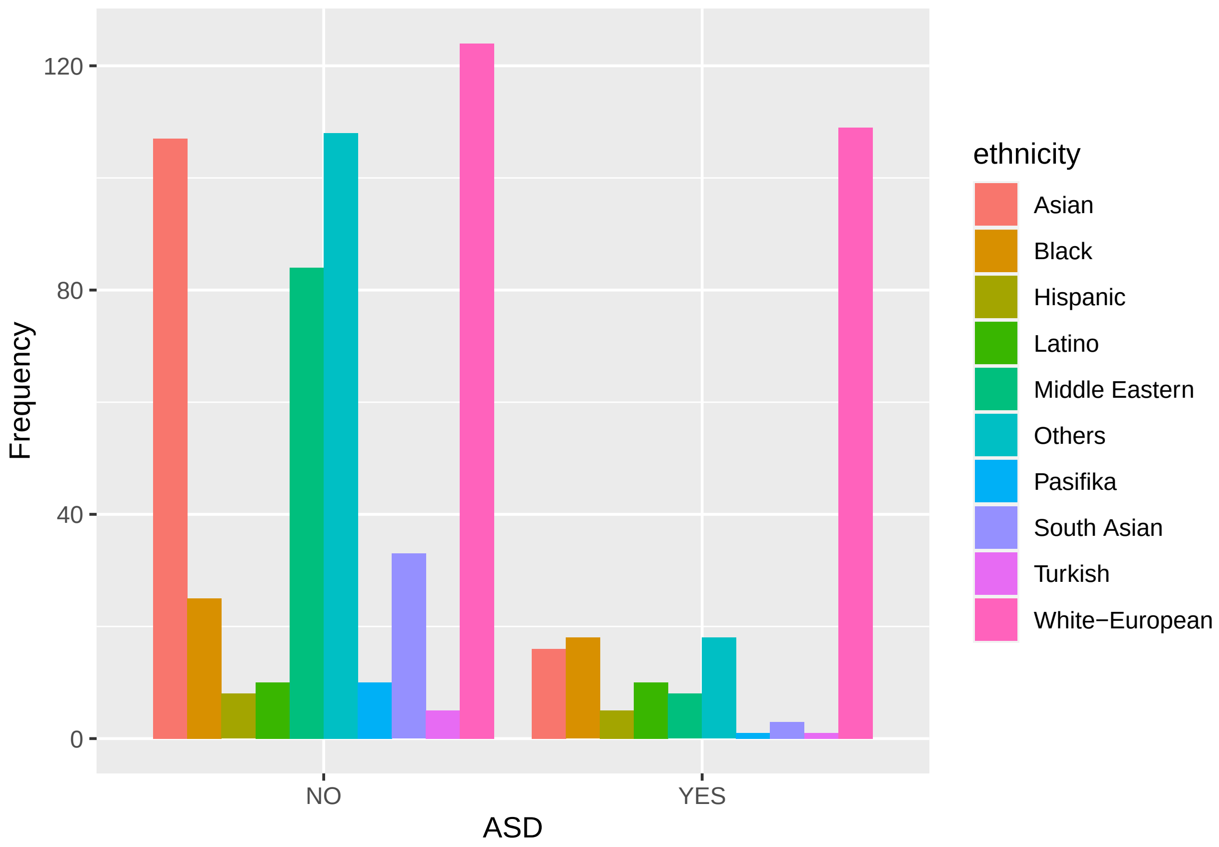
```
# checking for the count of Autism Spectrum Disorder (ASD)
table(cat_aut_data$Class.ASD)
```

```
##
##  NO YES
## 514 189
```

```
barplot(table(cat_aut_data$Class.ASD), main = "Histogram for ASD", ylab = "Frequency")
```



```r
# there are 189 ASD patients and 514 normal patients

# plotting distribution of ASD with ethnicity
tbl <- with(cat_aut_data, table(ethnicity, Class.ASD))
ggplot(as.data.frame(tbl), aes(factor(Class.ASD), Freq, fill = ethnicity)) +
  geom_col(position = 'dodge') + xlab("ASD") + ylab("Frequency")
```



```
# from the plot we can see that Pacifica and Turkish have the least ASD patients
# whereas White Europeans have maximum number of ASD patients
# On the other hand Turkish have least number of normal people and White Europeans
# have maximum number of normal people

# label encoding the binary categorical variables gender, jundice, autism,
# used_app_before, Class.ASD
cat_aut_data$gender <- ifelse(cat_aut_data$gender == "m", 1, 0)
cat_aut_data$jundice <- ifelse(cat_aut_data$jundice == "yes", 1, 0)
cat_aut_data$austim <- ifelse(cat_aut_data$austim == "yes", 1, 0)
cat_aut_data$used_app_before <- ifelse(cat_aut_data$used_app_before == "yes", 1, 0)
cat_aut_data$Class.ASD <- ifelse(cat_aut_data$Class.ASD == "YES", 1, 0)

# One hot encoding for rest of the categorical variables
dummy <- dummyVars(" ~ .", data = cat_aut_data, sep = "_")
cat_aut_data <- data.frame(predict(dummy, newdata = cat_aut_data))
head(cat_aut_data)
```

```
##   gender ethnicityAsian ethnicityBlack ethnicityHispanic ethnicityLatino
## 1      0              0              0                 0               0
## 2      1              0              0                 0               1
## 3      1              0              0                 0               1
## 4      0              0              0                 0               0
## 5      0              0              0                 0               0
## 6      1              0              0                 0               0
##   ethnicityMiddle.Eastern. ethnicityOthers ethnicityPasifika
## 1                        0               0                 0
## 2                        0               0                 0
## 3                        0               0                 0
## 4                        0               0                 0
## 5                        0               1                 0
## 6                        0               1                 0
##   ethnicitySouth.Asian ethnicityTurkish ethnicityWhite.European jundice austim
## 1                    0                0                       1       0      0
## 2                    0                0                       0       0      1
## 3                    0                0                       0       1      1
## 4                    0                0                       1       0      1
## 5                    0                0                       0       0      0
## 6                    0                0                       0       1      0
##   contry_of_resAfghanistan contry_of_resAmericanSamoa contry_of_resAngola
## 1                        0                          0                   0
## 2                        0                          0                   0
## 3                        0                          0                   0
## 4                        0                          0                   0
## 5                        0                          0                   0
## 6                        0                          0                   0
##   contry_of_resArgentina contry_of_resArmenia contry_of_resAruba
## 1                      0                    0                  0
## 2                      0                    0                  0
## 3                      0                    0                  0
## 4                      0                    0                  0
## 5                      0                    0                  0
## 6                      0                    0                  0
##   contry_of_resAustralia contry_of_resAustria contry_of_resAzerbaijan
## 1                      0                    0                       0
## 2                      0                    0                       0
## 3                      0                    0                       0
## 4                      0                    0                       0
## 5                      0                    0                       0
## 6                      0                    0                       0
##   contry_of_resBahamas contry_of_resBangladesh contry_of_resBelgium
## 1                    0                       0                    0
## 2                    0                       0                    0
## 3                    0                       0                    0
## 4                    0                       0                    0
## 5                    0                       0                    0
## 6                    0                       0                    0
##   contry_of_resBolivia contry_of_resBrazil contry_of_resBurundi
## 1                    0                   0                    0
## 2                    0                   1                    0
## 3                    0                   0                    0
## 4                    0                   0                    0
## 5                    0                   0                    0
## 6                    0                   0                    0
##   contry_of_resCanada contry_of_resChile contry_of_resChina
## 1                   0                  0                  0
## 2                   0                  0                  0
## 3                   0                  0                  0
## 4                   0                  0                  0
## 5                   0                  0                  0
## 6                  0                  0                  0
##   contry_of_resCosta.Rica contry_of_resCyprus contry_of_resCzech.Republic
## 1                       0                   0                           0
## 2                       0                   0                           0
## 3                       0                   0                           0
## 4                       0                   0                           0
## 5                       0                   0                           0
## 6                       0                   0                           0
##   contry_of_resEcuador contry_of_resEgypt contry_of_resEthiopia
## 1                    0                  0                     0
## 2                    0                  0                     0
## 3                    0                  0                     0
## 4                    0                  0                     0
## 5                    0                  1                     0
## 6                    0                  0                     0
##   contry_of_resFinland contry_of_resFrance contry_of_resGermany
## 1                    0                   0                    0
## 2                    0                   0                    0
## 3                    0                   0                    0
## 4                    0                   0                    0
## 5                    0                   0                    0
## 6                    0                   0                    0
##   contry_of_resHong.Kong contry_of_resIceland contry_of_resIndia
## 1                      0                    0                  0
## 2                      0                    0                  0
## 3                      0                    0                  0
## 4                      0                    0                  0
## 5                      0                    0                  0
## 6                      0                    0                  0
##   contry_of_resIndonesia contry_of_resIran contry_of_resIraq
## 1                      0                 0                 0
## 2                      0                 0                 0
## 3                      0                 0                 0
## 4                      0                 0                 0
## 5                      0                 0                 0
## 6                      0                 0                 0
##   contry_of_resIreland contry_of_resItaly contry_of_resJapan
## 1                    0                  0                  0
## 2                    0                  0                  0
## 3                    0                  0                  0
## 4                    0                  0                  0
## 5                    0                  0                  0
## 6                    0                  0                  0
##   contry_of_resJordan contry_of_resKazakhstan contry_of_resLebanon
## 1                   0                       0                    0
## 2                   0                       0                    0
## 3                   0                       0                    0
## 4                   0                       0                    0
## 5                   0                       0                    0
## 6                   0                       0                    0
##   contry_of_resMalaysia contry_of_resMexico contry_of_resNepal
## 1                     0                   0                  0
## 2                     0                   0                  0
## 3                     0                   0                  0
## 4                     0                   0                  0
## 5                     0                   0                  0
## 6                     0                   0                  0
##   contry_of_resNetherlands contry_of_resNew.Zealand contry_of_resNicaragua
## 1                        0                        0                      0
## 2                        0                        0                      0
## 3                        0                        0                      0
## 4                        0                        0                      0
## 5                        0                        0                      0
## 6                        0                        0                      0
##   contry_of_resNiger contry_of_resOman contry_of_resPakistan
## 1                  0                 0                     0
## 2                  0                 0                     0
## 3                  0                 0                     0
## 4                  0                 0                     0
## 5                  0                 0                     0
## 6                  0                 0                     0
##   contry_of_resPhilippines contry_of_resPortugal contry_of_resRomania
## 1                        0                     0                    0
## 2                        0                     0                    0
## 3                        0                     0                    0
## 4                        0                     0                    0
## 5                        0                     0                    0
## 6                        0                     0                    0
##   contry_of_resRussia contry_of_resSaudi.Arabia contry_of_resSerbia
## 1                   0                         0                   0
## 2                   0                         0                   0
## 3                   0                         0                   0
## 4                   0                         0                   0
## 5                   0                         0                   0
## 6                   0                         0                   0
##   contry_of_resSierra.Leone contry_of_resSouth.Africa contry_of_resSpain
## 1                         0                         0                  0
## 2                         0                         0                  0
## 3                         0                         0                  1
## 4                         0                         0                  0
## 5                         0                         0                  0
## 6                         0                         0                  0
##   contry_of_resSri.Lanka contry_of_resSweden contry_of_resTonga
## 1                      0                   0                  0
## 2                      0                   0                  0
## 3                      0                   0                  0
## 4                      0                   0                  0
## 5                      0                   0                  0
## 6                      0                   0                  0
##   contry_of_resTurkey contry_of_resUkraine contry_of_resUnited.Arab.Emirates
## 1                   0                    0                                 0
## 2                   0                    0                                 0
## 3                   0                    0                                 0
## 4                   0                    0                                 0
## 5                   0                    0                                 0
## 6                   0                    0                                 0
##   contry_of_resUnited.Kingdom contry_of_resUnited.States contry_of_resUruguay
## 1                           0                          1                    0
## 2                           0                           0                    0
## 3                           0                           0                    0
## 4                           0                           1                    0
## 5                           0                           0                    0
## 6                           0                           1                    0
##   contry_of_resViet.Nam used_app_before relationHealth.care.professional
## 1                     0               0                                0
## 2                     0               0                                0
## 3                     0               0                                0
## 4                     0               0                                0
## 5                     0               0                                0
## 6                     0               0                                0
##   relationOthers relationParent relationRelative relationSelf Class.ASD
## 1              0              0                0            1         0
## 2              0              0                0            1         0
## 3              0              1                0            0         1
## 4              0              0                0            1         0
## 5              1              0                0            0         0
## 6              0              0                0            1         1
```

```
# finding correaltion between variables

# using only numerical variables
num_cor_mat <- cor(cbind(num_aut_data, cat_aut_data$Class.ASD))
num_cor_mat
```

```
##                           A1_Score    A2_Score   A3_Score    A4_Score
## A1_Score               1.000000000  0.01235387 0.07497267 0.128815732
## A2_Score               0.012353866  1.00000000 0.22299723 0.157917566
## A3_Score               0.074972671  0.22299723 1.00000000 0.411962183
## A4_Score               0.128815732  0.15791757 0.41196218 1.000000000
## A5_Score               0.170417460  0.15272770 0.26397007 0.305830076
## A6_Score               0.110817604  0.18520946 0.26826005 0.294552941
## A7_Score               0.218457514 -0.04291160 0.07719548 0.150223778
## A8_Score               0.149078792  0.03371085 0.01602520 0.006711761
## A9_Score               0.146153846  0.20471525 0.31450483 0.327036547
## A10_Score              0.119585715  0.06748434 0.16719877 0.209678422
## age                    0.008118375  0.08257822 0.09855214 0.107580359
## result                 0.399616880  0.39143303 0.55160565 0.585248791
## cat_aut_data$Class.ASD 0.298322602  0.31086167 0.44066179 0.469541685
##                           A5_Score   A6_Score     A7_Score     A8_Score
## A1_Score               0.170417460 0.11081760  0.218457514  0.149078792
## A2_Score               0.152727704 0.18520946 -0.042911599  0.033710848
## A3_Score               0.263970068 0.26826005  0.077195477  0.016025203
## A4_Score               0.305830076 0.29455294  0.150223778  0.006711761
## A5_Score               1.000000000 0.39184869  0.237677618  0.100360028
## A6_Score               0.391848692 1.00000000  0.174868958  0.099062640
## A7_Score               0.237677618 0.17486896  1.000000000  0.083918040
## A8_Score               0.100360028 0.09906264  0.083918040  1.000000000
## A9_Score               0.396015052 0.47910011  0.188806572  0.100560301
## A10_Score              0.266358390 0.29375943  0.251077229  0.098761041
## age                    0.009127665 0.09221083 -0.001250767 -0.064874480
## result                 0.639051620 0.62987828  0.453988107  0.321994223
## cat_aut_data$Class.ASD 0.536664716 0.59186965  0.350969527  0.236361328
##                         A9_Score  A10_Score         age     result
## A1_Score               0.1461538 0.11958571  0.008118375 0.39961688
## A2_Score               0.2047153 0.06748434  0.082578217 0.39143303
## A3_Score               0.3145048 0.16719877  0.098552145 0.55160565
## A4_Score               0.3270365 0.20967842  0.107580359 0.58524879
## A5_Score               0.3960151 0.26635839  0.009127665 0.63905162
## A6_Score               0.4791001 0.29375943  0.092210833 0.62987828
## A7_Score               0.1888066 0.25107723 -0.001250767 0.45398811
## A8_Score               0.1005603 0.09876104 -0.064874480 0.32199422
## A9_Score               1.0000000 0.28256438  0.128297891 0.66103481
## A10_Score              0.2825644 1.00000000  0.046652646 0.53607543
## age                    0.1282979 0.04665265  1.000000000 0.09825975
## result                 0.6610348 0.53607543  0.098259748 1.00000000
## cat_aut_data$Class.ASD 0.6353617 0.38538689  0.132590609 0.82172939
##                        cat_aut_data$Class.ASD
## A1_Score                            0.2983226
## A2_Score                            0.3108617
## A3_Score                            0.4406618
## A4_Score                            0.4695417
## A5_Score                            0.5366647
## A6_Score                            0.5918696
## A7_Score                            0.3509695
## A8_Score                            0.2363613
## A9_Score                            0.6353617
## A10_Score                           0.3853869
## age                                 0.1325906
## result                              0.8217294
## cat_aut_data$Class.ASD              1.0000000
```

```r
# variable result is showing high correlation of 0.8217294 with the target variable
# Class.ASD
# none of the other variables show high correlation among themselves

# using only encoded categorical variables
cat_cor_mat <- cor(cat_aut_data)
# relationOthers has high correlation with ethinicityOthers of 0.8183654
# therefore we will drop relationOthers
cat_aut_data <- cat_aut_data[,-83]

# None of the machine learning algorithms that we are using make the assumptions of
# normality or in another words they don't assume the distribution to be normal

# We will apply min max normalisation on the dataset mainly for bringing age and
# result column on the same scale and since other columns are binary they wont get
# affected by min max scaling
# function to implement min max scaling
min_max_scaler <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}

# using min_max_scaler function to implement min max scaling
scaled_num_aut_data <- as.data.frame(lapply(num_aut_data, min_max_scaler))
```

```r
# the total number of features in our dataset excluding the target variable are 97
# there is no need to apply feature engineering or derived features as there are
# no variables that can be combined to form a new feature or which can be split to
# create two new features, and another reason for not applying any kind of
# transformation on our features is that our algorithms does make assumption
# of normality and will not be affected even if the data does not have normal
# distribution




# using PCA for selecting feautres
# preparing data for performing PCA
pca_data <- cbind(num_aut_data, cat_aut_data)
# removing the target column before performing PCA
pca_data <- pca_data[,-98]
colnames(pca_data)
```

```
##  [1] "A1_Score"                          "A2_Score"
##  [3] "A3_Score"                          "A4_Score"
##  [5] "A5_Score"                          "A6_Score"
##  [7] "A7_Score"                          "A8_Score"
##  [9] "A9_Score"                          "A10_Score"
## [11] "age"                               "result"
## [13] "gender"                            "ethnicityAsian"
## [15] "ethnicityBlack"                    "ethnicityHispanic"
## [17] "ethnicityLatino"                   "ethnicityMiddle.Eastern."
## [19] "ethnicityOthers"                   "ethnicityPasifika"
## [21] "ethnicitySouth.Asian"              "ethnicityTurkish"
## [23] "ethnicityWhite.European"           "jundice"
## [25] "austim"                            "contry_of_resAfghanistan"
## [27] "contry_of_resAmericanSamoa"        "contry_of_resAngola"
## [29] "contry_of_resArgentina"            "contry_of_resArmenia"
## [31] "contry_of_resAruba"                "contry_of_resAustralia"
## [33] "contry_of_resAustria"              "contry_of_resAzerbaijan"
## [35] "contry_of_resBahamas"              "contry_of_resBangladesh"
## [37] "contry_of_resBelgium"              "contry_of_resBolivia"
## [39] "contry_of_resBrazil"               "contry_of_resBurundi"
## [41] "contry_of_resCanada"               "contry_of_resChile"
## [43] "contry_of_resChina"                "contry_of_resCosta.Rica"
## [45] "contry_of_resCyprus"               "contry_of_resCzech.Republic"
## [47] "contry_of_resEcuador"              "contry_of_resEgypt"
## [49] "contry_of_resEthiopia"             "contry_of_resFinland"
## [51] "contry_of_resFrance"               "contry_of_resGermany"
## [53] "contry_of_resHong.Kong"            "contry_of_resIceland"
## [55] "contry_of_resIndia"                "contry_of_resIndonesia"
## [57] "contry_of_resIran"                 "contry_of_resIraq"
## [59] "contry_of_resIreland"              "contry_of_resItaly"
## [61] "contry_of_resJapan"                "contry_of_resJordan"
## [63] "contry_of_resKazakhstan"           "contry_of_resLebanon"
## [65] "contry_of_resMalaysia"             "contry_of_resMexico"
## [67] "contry_of_resNepal"                "contry_of_resNetherlands"
## [69] "contry_of_resNew.Zealand"          "contry_of_resNicaragua"
## [71] "contry_of_resNiger"                "contry_of_resOman"
## [73] "contry_of_resPakistan"             "contry_of_resPhilippines"
## [75] "contry_of_resPortugal"             "contry_of_resRomania"
## [77] "contry_of_resRussia"               "contry_of_resSaudi.Arabia"
## [79] "contry_of_resSerbia"               "contry_of_resSierra.Leone"
## [81] "contry_of_resSouth.Africa"         "contry_of_resSpain"
## [83] "contry_of_resSri.Lanka"            "contry_of_resSweden"
## [85] "contry_of_resTonga"                "contry_of_resTurkey"
## [87] "contry_of_resUkraine"              "contry_of_resUnited.Arab.Emirates"
## [89] "contry_of_resUnited.Kingdom"       "contry_of_resUnited.States"
## [91] "contry_of_resUruguay"              "contry_of_resViet.Nam"
## [93] "used_app_before"                   "relationHealth.care.professional"
## [95] "relationParent"                    "relationRelative"
## [97] "relationSelf"
```

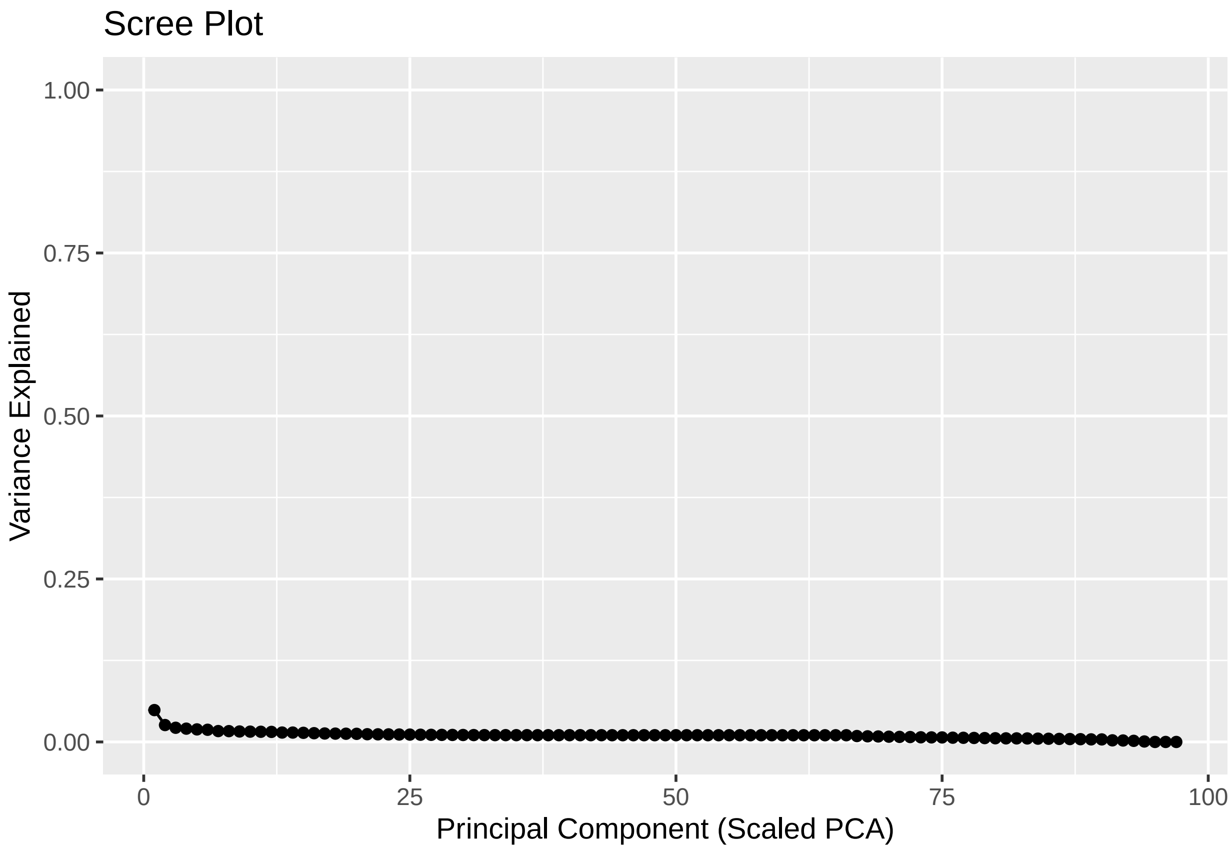
```
# performimg scaled PCA
pca_scaled <- prcomp(pca_data, scale. = TRUE, center = TRUE)
s_pca_scaled <- summary(pca_scaled)
s_pca_scaled$importance[2,]
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9    PC10
## 0.04889 0.02595 0.02177 0.02041 0.01926 0.01870 0.01667 0.01659 0.01609 0.01582
##    PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18    PC19    PC20
## 0.01564 0.01542 0.01445 0.01441 0.01401 0.01347 0.01299 0.01280 0.01264 0.01247
##    PC21    PC22    PC23    PC24    PC25    PC26    PC27    PC28    PC29    PC30
## 0.01188 0.01169 0.01161 0.01141 0.01128 0.01107 0.01092 0.01086 0.01074 0.01069
##    PC31    PC32    PC33    PC34    PC35    PC36    PC37    PC38    PC39    PC40
## 0.01055 0.01053 0.01047 0.01045 0.01043 0.01041 0.01040 0.01040 0.01038 0.01038
##    PC41    PC42    PC43    PC44    PC45    PC46    PC47    PC48    PC49    PC50
## 0.01038 0.01037 0.01037 0.01037 0.01036 0.01036 0.01035 0.01035 0.01035 0.01034
##    PC51    PC52    PC53    PC54    PC55    PC56    PC57    PC58    PC59    PC60
## 0.01034 0.01034 0.01034 0.01033 0.01033 0.01033 0.01033 0.01033 0.01033 0.01033
##    PC61    PC62    PC63    PC64    PC65    PC66    PC67    PC68    PC69    PC70
## 0.01032 0.01032 0.01032 0.01032 0.01032 0.01032 0.00909 0.00865 0.00844 0.00807
##    PC71    PC72    PC73    PC74    PC75    PC76    PC77    PC78    PC79    PC80
## 0.00779 0.00751 0.00716 0.00701 0.00697 0.00656 0.00640 0.00615 0.00592 0.00569
##    PC81    PC82    PC83    PC84    PC85    PC86    PC87    PC88    PC89    PC90
## 0.00551 0.00548 0.00541 0.00506 0.00491 0.00465 0.00446 0.00423 0.00396 0.00391
##    PC91    PC92    PC93    PC94    PC95    PC96    PC97
## 0.00249 0.00224 0.00171 0.00073 0.00000 0.00000 0.00000
```

```
var_explained_scaled <- pca_scaled$sdev^2 / sum(pca_scaled$sdev^2)
var_explained_scaled
```

```
##  [1] 4.889256e-02 2.594803e-02 2.176510e-02 2.040904e-02 1.926158e-02
##  [6] 1.869921e-02 1.666947e-02 1.658714e-02 1.608963e-02 1.581614e-02
## [11] 1.563715e-02 1.542423e-02 1.445229e-02 1.440710e-02 1.400860e-02
## [16] 1.347411e-02 1.298813e-02 1.279803e-02 1.263579e-02 1.247434e-02
## [21] 1.188109e-02 1.169255e-02 1.161020e-02 1.140667e-02 1.128418e-02
## [26] 1.106558e-02 1.091880e-02 1.085553e-02 1.073962e-02 1.068812e-02
## [31] 1.054898e-02 1.053097e-02 1.046834e-02 1.045115e-02 1.043235e-02
## [36] 1.041157e-02 1.040194e-02 1.039872e-02 1.038276e-02 1.037829e-02
## [41] 1.037630e-02 1.037073e-02 1.036893e-02 1.036726e-02 1.036068e-02
## [46] 1.035628e-02 1.035416e-02 1.035066e-02 1.034546e-02 1.034284e-02
## [51] 1.033965e-02 1.033812e-02 1.033652e-02 1.033415e-02 1.033211e-02
## [56] 1.033136e-02 1.032894e-02 1.032789e-02 1.032681e-02 1.032562e-02
## [61] 1.032492e-02 1.032468e-02 1.032440e-02 1.032399e-02 1.032396e-02
## [66] 1.032396e-02 9.092377e-03 8.649994e-03 8.440216e-03 8.066639e-03
## [71] 7.790055e-03 7.514609e-03 7.159540e-03 7.014764e-03 6.966526e-03
## [76] 6.560508e-03 6.399497e-03 6.151210e-03 5.923096e-03 5.688282e-03
## [81] 5.507526e-03 5.483477e-03 5.407426e-03 5.058265e-03 4.907720e-03
## [86] 4.649017e-03 4.458306e-03 4.233942e-03 3.961926e-03 3.905161e-03
## [91] 2.491824e-03 2.238729e-03 1.707774e-03 7.261360e-04 3.641344e-32
## [96] 1.865096e-32 1.001831e-32
```

```
# plotting scree plot for scaled PCA
qplot(c(1:97), var_explained_scaled) +
  geom_line() +
  xlab("Principal Component (Scaled PCA)") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```



```
# In scaled PCA, the first principal component explains 0.04889 or 4.9% of the
# variance and the second principal componenet explains 0.02595 or 2.6% of the
# variance
```

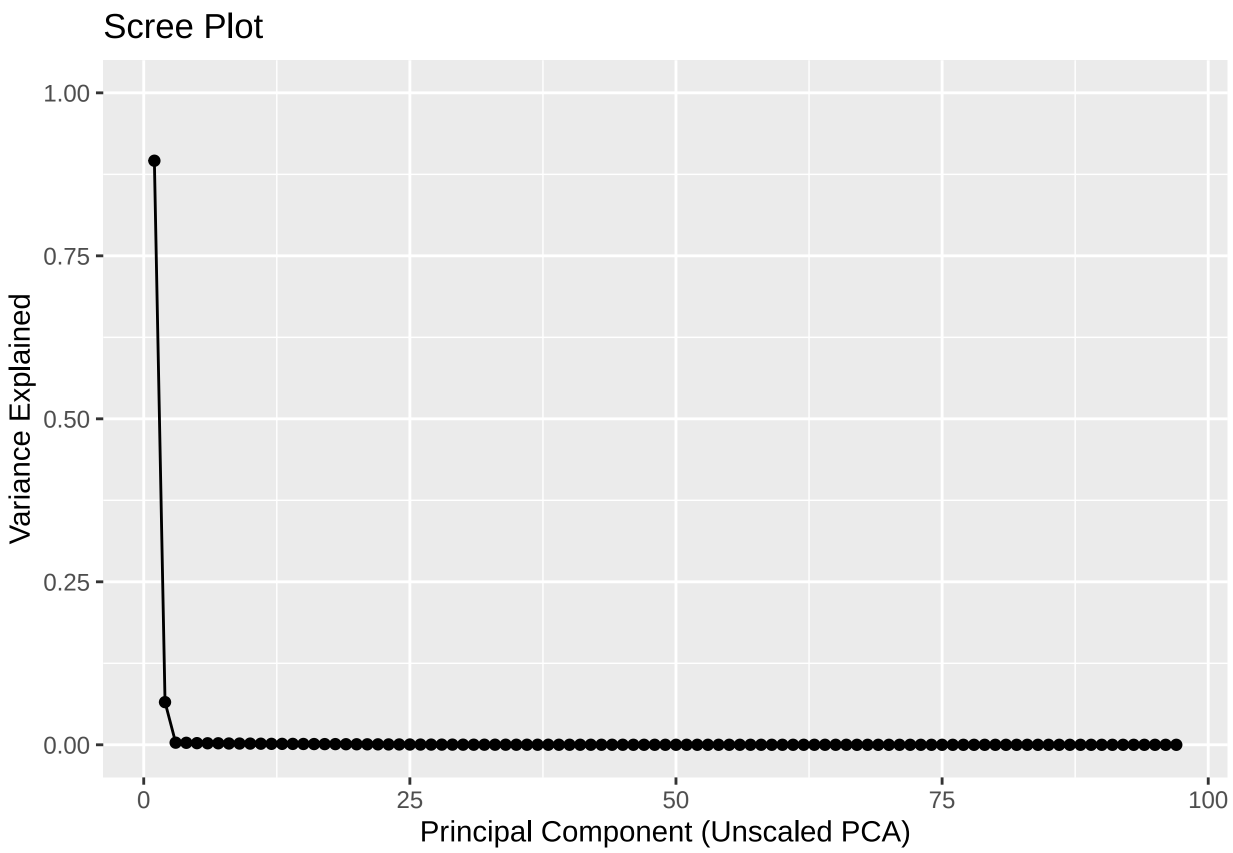
```r
# performing unscaled PCA
pca_unscaled <- prcomp(pca_data)
s_pca_unscaled <- summary(pca_unscaled)
s_pca_unscaled
```

```
## Importance of components:
##                           PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.7048 2.62280 0.58556 0.56836 0.51553 0.49585 0.49331
## Proportion of Variance 0.8959 0.06543 0.00326 0.00307 0.00253 0.00234 0.00231
## Cumulative Proportion  0.8959 0.96130 0.96457 0.96764 0.97017 0.97250 0.97482
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.45747 0.44993 0.43774 0.41800 0.39434 0.38862 0.38472
## Proportion of Variance 0.00199 0.00193 0.00182 0.00166 0.00148 0.00144 0.00141
## Cumulative Proportion  0.97681 0.97874 0.98056 0.98222 0.98370 0.98514 0.98654
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.36234 0.34819 0.33639 0.33164 0.30533 0.29257 0.27853
## Proportion of Variance 0.00125 0.00115 0.00108 0.00105 0.00089 0.00081 0.00074
## Cumulative Proportion  0.98779 0.98895 0.99002 0.99107 0.99195 0.99277 0.99351
##                           PC22    PC23    PC24    PC25    PC26    PC27   PC28
## Standard deviation     0.27380 0.24424 0.23817 0.21472 0.20061 0.18013 0.1776
## Proportion of Variance 0.00071 0.00057 0.00054 0.00044 0.00038 0.00031 0.0003
## Cumulative Proportion  0.99422 0.99479 0.99533 0.99577 0.99615 0.99646 0.9968
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.16740 0.14089 0.13616 0.13037 0.12906 0.12334 0.11707
## Proportion of Variance 0.00027 0.00019 0.00018 0.00016 0.00016 0.00014 0.00013
## Cumulative Proportion  0.99702 0.99721 0.99739 0.99755 0.99771 0.99785 0.99798
##                           PC36    PC37   PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.11102 0.10803 0.1043 0.09930 0.09768 0.09577 0.08868
## Proportion of Variance 0.00012 0.00011 0.0001 0.00009 0.00009 0.00009 0.00007
## Cumulative Proportion  0.99810 0.99821 0.9983 0.99841 0.99850 0.99859 0.99866
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.08394 0.08317 0.08144 0.07800 0.07566 0.07489 0.07388
## Proportion of Variance 0.00007 0.00007 0.00006 0.00006 0.00005 0.00005 0.00005
## Cumulative Proportion  0.99873 0.99880 0.99886 0.99892 0.99897 0.99902 0.99908
##                           PC50    PC51    PC52    PC53    PC54    PC55    PC56
## Standard deviation     0.07209 0.06962 0.06763 0.06474 0.06382 0.06322 0.06238
## Proportion of Variance 0.00005 0.00005 0.00004 0.00004 0.00004 0.00004 0.00004
## Cumulative Proportion  0.99913 0.99917 0.99921 0.99925 0.99929 0.99933 0.99937
##                           PC57    PC58    PC59    PC60    PC61    PC62    PC63
## Standard deviation     0.06084 0.05994 0.05613 0.05294 0.05232 0.05176 0.05111
## Proportion of Variance 0.00004 0.00003 0.00003 0.00003 0.00003 0.00003 0.00002
## Cumulative Proportion  0.99940 0.99944 0.99947 0.99949 0.99952 0.99955 0.99957
##                           PC64    PC65    PC66    PC67    PC68    PC69    PC70
## Standard deviation     0.05052 0.05019 0.04805 0.04405 0.04074 0.03774 0.03774
## Proportion of Variance 0.00002 0.00002 0.00002 0.00002 0.00002 0.00001 0.00001
## Cumulative Proportion  0.99960 0.99962 0.99964 0.99966 0.99968 0.99969 0.99970
##                           PC71    PC72    PC73    PC74    PC75    PC76    PC77
## Standard deviation     0.03774 0.03772 0.03770 0.03768 0.03761 0.03757 0.03750
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99972 0.99973 0.99974 0.99976 0.99977 0.99978 0.99980
##                           PC78    PC79    PC80    PC81    PC82    PC83    PC84
## Standard deviation     0.03742 0.03737 0.03733 0.03710 0.03686 0.03669 0.03659
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99981 0.99982 0.99984 0.99985 0.99986 0.99988 0.99989
##                           PC85    PC86    PC87    PC88    PC89    PC90    PC91
## Standard deviation     0.03640 0.03595 0.03584 0.03571 0.03521 0.03502 0.03341
## Proportion of Variance 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99990 0.99991 0.99993 0.99994 0.99995 0.99996 0.99997
##                           PC92    PC93    PC94      PC95      PC96     PC97
## Standard deviation     0.03329 0.03272 0.02874 4.343e-15 1.624e-15 2.24e-16
## Proportion of Variance 0.00001 0.00001 0.00001 0.000e+00 0.000e+00 0.00e+00
## Cumulative Proportion  0.99998 0.99999 1.00000 1.000e+00 1.000e+00 1.00e+00
```

```
var_explained_unscaled <- pca_unscaled$sdev^2 / sum(pca_unscaled$sdev^2)
var_explained_unscaled
```

```
##  [1] 8.958702e-01 6.543343e-02 3.261449e-03 3.072710e-03 2.527994e-03
##  [6] 2.338709e-03 2.314822e-03 1.990631e-03 1.925566e-03 1.822678e-03
## [11] 1.661943e-03 1.479140e-03 1.436535e-03 1.407821e-03 1.248821e-03
## [16] 1.153179e-03 1.076333e-03 1.046202e-03 8.867421e-04 8.141947e-04
## [21] 7.379317e-04 7.130993e-04 5.674283e-04 5.395552e-04 4.385650e-04
## [26] 3.828109e-04 3.086195e-04 2.999022e-04 2.665512e-04 1.888010e-04
## [31] 1.763498e-04 1.616571e-04 1.584417e-04 1.446959e-04 1.303671e-04
## [36] 1.172478e-04 1.110078e-04 1.035078e-04 9.380114e-05 9.076055e-05
## [41] 8.723607e-05 7.480831e-05 6.701534e-05 6.579001e-05 6.307995e-05
## [46] 5.786980e-05 5.444641e-05 5.334964e-05 5.192155e-05 4.943062e-05
## [51] 4.610092e-05 4.350214e-05 3.986910e-05 3.873665e-05 3.801895e-05
## [56] 3.700889e-05 3.521101e-05 3.417624e-05 2.997256e-05 2.665905e-05
## [61] 2.603710e-05 2.548463e-05 2.484677e-05 2.427758e-05 2.396233e-05
## [66] 2.195679e-05 1.845695e-05 1.578866e-05 1.354977e-05 1.354977e-05
## [71] 1.354848e-05 1.353519e-05 1.352157e-05 1.350200e-05 1.345758e-05
## [76] 1.342510e-05 1.337876e-05 1.331641e-05 1.328059e-05 1.325390e-05
## [81] 1.309374e-05 1.292344e-05 1.280331e-05 1.273584e-05 1.260238e-05
## [86] 1.229196e-05 1.222009e-05 1.213253e-05 1.179274e-05 1.166231e-05
## [91] 1.061660e-05 1.054368e-05 1.018294e-05 7.855719e-06 1.794417e-31
## [96] 2.508540e-32 4.772531e-34
```

```
# plotting scree plot for unscaled PCA
qplot(c(1:97), var_explained_unscaled) +
  geom_line() +
  xlab("Principal Component (Unscaled PCA)") +
  ylab("Variance Explained") +
  ggtitle("Scree Plot") +
  ylim(0, 1)
```



```
# In unscaled PCA, the first principal component explains 0.89587 or 89.6% of the
# variance and the second principal componenet explains 0.06543 or 6.5% of the
# variance, giving a cumulative explained variance of 0.96130 or 96.1%
# so we can use the first two principal components for visualizing our data in two
# dimensions in a scatter plot
# ans we can select first five principal components for using with Machine Learning
# algorithms
# First five principal components give us a cumulative proportion of 0.97017 or 97.02%

# After performing both scaled and unscaled PCA we can found out that unscaled PCA is
# performing better than scaled PCA in reducing the dimensions of the data

# choosing first five principal components as features for machine learning algorithms
# and adding the target column Class.ASD
Class.ASD <- cat_aut_data$Class.ASD
data <- cbind(as.data.frame(pca_unscaled$x[,1:5]), Class.ASD)

# splitting the data into training and testing set
# splitting is performed by random sampling of rown without replacement
# 20% of data is used as validation set and 80% as training set
# because training is the harder and the more complicated step of a machine learning
# algorithm and therefore training set should have a higher portion of data as
# compared to the testing or validation set
set.seed(10)
rows_test_set <- sample(rownames(data), 0.20 * nrow(data), replace = FALSE)
test_set <- data[rows_test_set,]
```

```r
train_set <- data[!row.names(data) %in% rows_test_set,]


# ALl the implemented ML algorthms will be evaluated using
# confusion matrix, AUC, precision, recall and F1-score

# 1.) Implementing SVM
# SVM is compatible with the features in the dataset
SVM <- svm(formula = Class.ASD ~ .,
           data = train_set,
           type = "C-classification",
           kernel = "radial")

summary(SVM)
```

```
##
## Call:
## svm(formula = Class.ASD ~ ., data = train_set, type = "C-classification",
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  115
##
##  ( 61 54 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```r
# using our SVM to make predictions on the validation set
svmpred <- predict(SVM , test_set)

# creating SVM confusion matrix
SVM_confusion_matrix = table(svmpred, test_set$Class.ASD)
SVM_confusion_matrix
```

```
##
## svmpred   0   1
##       0 102   0
##       1   0  38
```

```r
# SVM is able to correctly classify all 38 ASD patients and all 102 normal people

# Calculating SVM misclassification rate
SVM_miss_class_rate <- mean(svmpred != test_set$Class.ASD) * 100
SVM_miss_class_rate
```

```
## [1] 0
```

```r
# SVM has a misclassification rate of 0%

# Calculating SVM accuracy
SVM_acc <- sum(diag(SVM_confusion_matrix)) / sum(SVM_confusion_matrix) * 100
SVM_acc
```

```
## [1] 100
```

```r
# the accuracy of SVM is 100%

# finding true positive, true negative, false positive and false negative from
# SVM confusion matrix
true_pos_svm <- SVM_confusion_matrix[2,2]
true_neg_svm <- SVM_confusion_matrix[1,1]
false_pos_svm <- SVM_confusion_matrix[2,1]
false_neg_svm <- SVM_confusion_matrix[1,2]

# Calculating SVM precision
SVM_prec <- true_pos_svm/(true_pos_svm + false_pos_svm)
SVM_prec
```

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

```r
# the precision of SVM is 1

# Calculating SVM recall
SVM_rec <- true_pos_svm/(true_pos_svm + false_neg_svm)
SVM_rec
```

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

```r
# the recall of SVM is 1

# Calculating F1 score for SVM
SVM_F1 <- 2 * ((SVM_prec * SVM_rec)/(SVM_prec + SVM_rec))
SVM_F1
```

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

```r
# the F1 score for decision tree is 1

# We can use k-fold cross validation for SVM but we should not use it because the
# algorithm is already performing well and there is no point in splitting the dataset
# repeatedly and training/tesing the model on different portions of the dataset.



# 2.) Implementing Decision Tree
# Decision Tree is compatible with the features in the dataset
decision_tree <- rpart(Class.ASD ~., data = train_set, method = 'class')
summary(decision_tree)
```

```
## Call:
## rpart(formula = Class.ASD ~ ., data = train_set, method = "class")
##   n= 563
##
##          CP nsplit  rel error     xerror        xstd
## 1 0.986755      0 1.00000000 1.00000000 0.069615498
## 2 0.010000      1 0.01324503 0.01324503 0.009349003
##
## Variable importance
## PC2 PC1
##  98   2
##
## Node number 1: 563 observations,    complexity param=0.986755
##   predicted class=0  expected loss=0.268206  P(node) =1
##     class counts:   412   151
##    probabilities: 0.732 0.268
##   left son=2 (410 obs) right son=3 (153 obs)
##   Primary splits:
##       PC2 < -1.524075  to the right, improve=217.054100, (0 missing)
##       PC1 < -0.8545333 to the right, improve=  8.850475, (0 missing)
##       PC3 < -0.5720943 to the left,  improve=  7.738034, (0 missing)
##       PC4 < 0.7781198  to the right, improve=  4.446326, (0 missing)
##       PC5 < 0.7869657  to the right, improve=  3.325045, (0 missing)
##   Surrogate splits:
##       PC1 < -25.85237  to the right, agree=0.734, adj=0.02, (0 split)
##
## Node number 2: 410 observations
##   predicted class=0  expected loss=0  P(node) =0.7282416
##     class counts:   410     0
##    probabilities: 1.000 0.000
##
## Node number 3: 153 observations
##   predicted class=1  expected loss=0.0130719  P(node) =0.2717584
##     class counts:     2   151
##    probabilities: 0.013 0.987
```

```r
# using our decision tree to make predictions on the validation set
pred_dec_tree <- predict(decision_tree, test_set, type="class")

# creating decision tree confusion matrix
dec_tree_confusion_matrix = table(pred_dec_tree, test_set$Class.ASD)
dec_tree_confusion_matrix
```

```
##
## pred_dec_tree   0   1
##             0 101   1
##             1   1  37
```

```r
# Decision Tree is able to correctly classify 37 ASD patients and 101 normal people
# but it misclassifies 1 normal person as ASD patient (false positive) and
# misclassifies 1 ASD patient as normal person (false negative)

# Calculating decision tree misclassification rate
```

```
dec_tree_miss_class_rate <- mean(pred_dec_tree != test_set$Class.ASD) * 100
dec_tree_miss_class_rate
```

```
## [1] 1.428571
```

```
# Decision tree has a misclassification rate of 1.428571%

# Calculating decision tree accuracy
dec_tree_acc <- sum(diag(dec_tree_confusion_matrix)) / sum(dec_tree_confusion_matrix) * 100
dec_tree_acc
```

```
## [1] 98.57143
```

```
# the accuracy of decision tree is 98.57143%

# finding true positive, true negative, false positive and false negative from
# decision tree confusion matrix
true_pos_dec_tree <- dec_tree_confusion_matrix[2,2]
true_neg_dec_tree <- dec_tree_confusion_matrix[1,1]
false_pos_dec_tree <- dec_tree_confusion_matrix[2,1]
false_neg_dec_tree <- dec_tree_confusion_matrix[1,2]

# Calculating dec_tree precision
dec_tree_prec <- true_pos_dec_tree/(true_pos_dec_tree + false_pos_dec_tree)
dec_tree_prec
```

```
## [1] 0.9736842
```

```
# the precision of dec_tree is 0.9736842

# Calculating dec_tree recall
dec_tree_rec <- true_pos_dec_tree/(true_pos_dec_tree + false_neg_dec_tree)
dec_tree_rec
```

```
## [1] 0.9736842
```

```
# the recall of dec_tree is 0.9736842

# Calculating F1 score for dec_tree
dec_tree_F1 <- 2 * ((dec_tree_prec * dec_tree_rec)/(dec_tree_prec + dec_tree_rec))
dec_tree_F1
```

```
## [1] 0.9736842
```

```
# the F1 score for decision tree is 0.9736842

# implementing k fold cross validation for decision tree
# setting seed so that the results are reproducible
set.seed(10)
```

```r
# funstion trainControl generates parameters that control how models will be created
# here we are applying 10 fold cross validation
train_control <- trainControl(method = "cv", number = 10, savePredictions=TRUE)

# building the decision tree model with 10 fold cross validation
# we pass entire data inside train function because train and test splitting will
# be done by k fold cross validation
model <- train(factor(Class.ASD) ~., data = data,
               trControl = train_control,
               method = "rpart")


model
```

```
## CART
##
## 703 samples
##   5 predictor
##   2 classes: ’0’, ’1’
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 633, 633, 632, 632, 633, 633, ...
## Resampling results across tuning parameters:
##
##   cp        Accuracy   Kappa
##   0.000000  0.9914487  0.9780722
##   0.489418  0.9914487  0.9780722
##   0.978836  0.8036419  0.2780722
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.489418.
```

```r
# we are getting an accuracy of 0.9914475 at cp = 0.489418 using k-fold cross validation



# 3.) Implementing Logistic Regression
# Logistic Regression is compatible with the features in the dataset
log_reg_model <- glm(Class.ASD ~.,
                     data = train_set,
                     family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```r
summary(log_reg_model)
```

```
##
## Call:
## glm(formula = Class.ASD ~ ., family = "binomial", data = train_set)
```

```
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -5.138e-05  -2.100e-08  -2.100e-08   2.100e-08   6.283e-05
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -74.329  13368.736  -0.006    0.996
## PC1            -1.272    603.456  -0.002    0.998
## PC2           -43.353   7629.752  -0.006    0.995
## PC3            -2.106   6985.649   0.000    1.000
## PC4            -2.228   8204.870   0.000    1.000
## PC5             0.320   8733.300   0.000    1.000
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6.5473e+02  on 562  degrees of freedom
## Residual deviance: 5.2698e-08  on 557  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

```
# using our Logistic Regression model to make predictions on the validation set
pred_log_reg <- predict(log_reg_model, test_set, type="response")
pred_log_reg <- ifelse(pred_log_reg > 0.5, 1, 0)

# creating logistic regression confusion matrix
log_reg_confusion_matrix = table(pred_log_reg, test_set$Class.ASD)
log_reg_confusion_matrix
```

```
##
## pred_log_reg   0   1
##            0 102   0
##            1   0  38
```

```
# Logistic Regression is able to correctly classify all ASD patients and all normal
# people

# Calculating logistic regression misclassification rate
log_reg_miss_class_rate <- mean(pred_log_reg != test_set$Class.ASD) * 100
log_reg_miss_class_rate
```

```
## [1] 0
```

```
# Decision tree has a misclassification rate of 0%

# Calculating logistic regression accuracy
log_reg_acc <- sum(diag(log_reg_confusion_matrix)) / sum(log_reg_confusion_matrix) * 100
log_reg_acc
```

```
## [1] 100
```

```r
# the accuracy of decision tree is 100%

# finding true positive, true negative, false positive and false negative from
# logistic regression confusion matrix
true_pos_log_reg <- log_reg_confusion_matrix[2,2]
true_neg_log_reg <- log_reg_confusion_matrix[1,1]
false_pos_log_reg <- log_reg_confusion_matrix[2,1]
false_neg_log_reg <- log_reg_confusion_matrix[1,2]

# Calculating log_reg precision
log_reg_prec <- true_pos_log_reg/(true_pos_log_reg + false_pos_log_reg)
log_reg_prec
```

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

```r
# the precision of log_reg is 1

# Calculating log_reg recall
log_reg_rec <- true_pos_log_reg/(true_pos_log_reg + false_neg_log_reg)
log_reg_rec
```

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

```r
# the recall of log_reg is 1

# Calculating F1 score for log_reg
log_reg_F1 <- 2 * ((log_reg_prec * log_reg_rec)/(log_reg_prec + log_reg_rec))
log_reg_F1
```

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

```r
# the F1 score for logistic regression is 1

# We can use k-fold cross validation for logistic regression but we should not use it
# because the algorithm is already performing well and there is no point in splitting
# the dataset repeatedly and training/tesing the model on different portions of the
# dataset.



# 3.) Implementing Artificial Neural Network
# Logistic Regression is compatible with the features in the dataset

# fitting the neural network
set.seed(10)
ANN <- neuralnet(Class.ASD ~ .,
                 data = train_set,
                 hidden = c(4))

# number of neurons in the hidden layer taken as 1 less than the number of features
# making predictions using ANN
ANN_result <- compute(ANN, rep = 1, test_set[, -6])
ANN_predictions <- ANN_result$net.result
ANN_predictions <- ifelse(ANN_predictions > 0.5, 1, 0)

# creating ANN confusion matrix
ANN_confusion_matrix <- table(ANN_predictions, test_set$Class.ASD)
ANN_confusion_matrix
```

```
##
## ANN_predictions   0   1
##               0 102   0
##               1   0  38
```

```
# ANN is able to correctly classify all ASD patients and all normal people

# calulating ANN misclassification rate
ANN_misclass_rate <- mean(ANN_predictions != test_set$Class.ASD) * 100
ANN_misclass_rate
```

```
## [1] 0
```

```
# ANN misclassification rate is 0%

# calulating ANN accuracy
ANN_acc <- sum(diag(ANN_confusion_matrix)) / sum(ANN_confusion_matrix) * 100
ANN_acc
```

```
## [1] 100
```

```
# the accuracy from neural network is 100%

# calculating true positive, true negative, false positive and false negative
# from the ANN confusion matrix
true_pos_ANN <- ANN_confusion_matrix[2,2]
true_neg_ANN <- ANN_confusion_matrix[1,1]
false_pos_ANN <- ANN_confusion_matrix[2,1]
false_neg_ANN <- ANN_confusion_matrix[1,2]

# calculating ANN precision
ANN_prec <- true_pos_ANN/(true_pos_ANN + false_pos_ANN)
ANN_prec
```

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

```
# ANN precision is 1

# calculating ANN recall
ANN_recall <- true_pos_ANN/(true_pos_ANN + false_neg_ANN)
ANN_recall
```

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

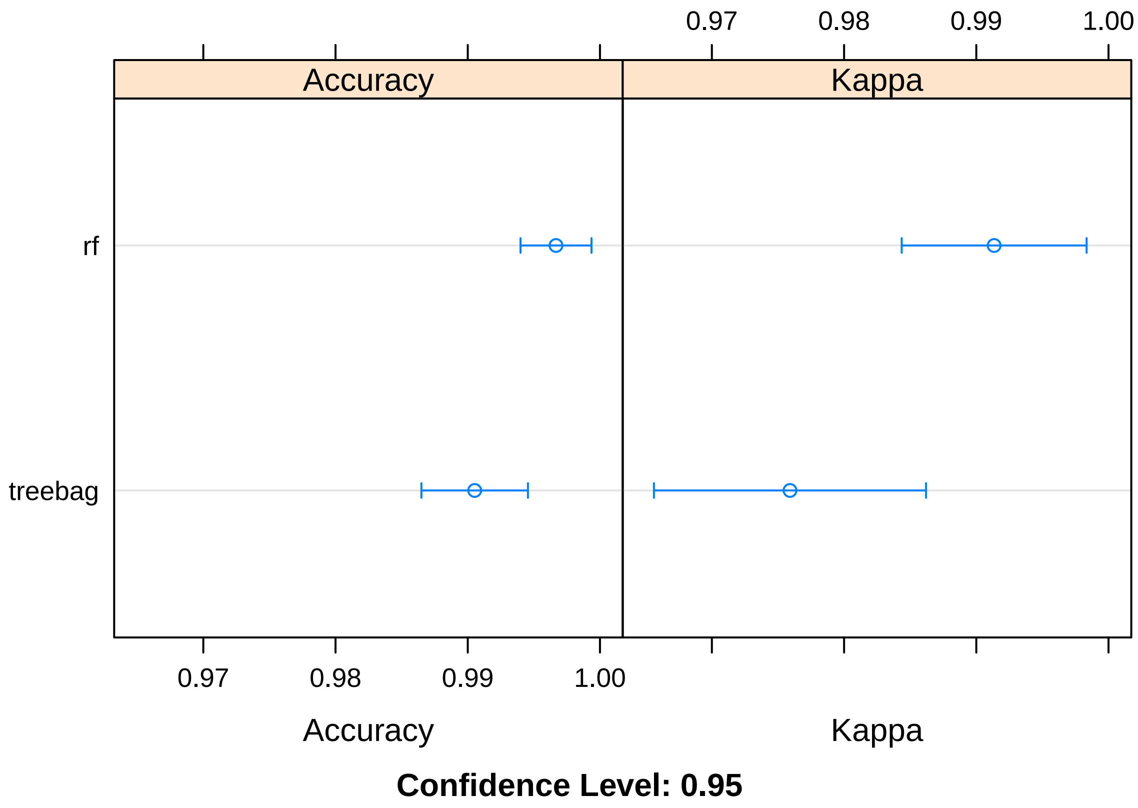
```
# ANN recalll is 1

# Calculating F1 score for log_reg
ANN_F1 <- 2 * ((ANN_prec * ANN_recall)/(ANN_prec + ANN_recall))
ANN_F1
```

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

```
# the F1 score for ANN is 1

# We can use k-fold cross validation for logistic regression but we should not use it
# because the algorithm is already performing well and there is no point in splitting
# the dataset repeatedly and training/tesing the model on different portions of the
# dataset.


# Applying two ensemble techniques bagging and boosting

# Applying two Bagging algorithms:
# 1.) Treebag
control <- trainControl(method="repeatedcv", number=10, repeats=3)
seed <- 7
metric <- "Accuracy"
# Bagged CART
set.seed(seed)
fit.treebag <- train(factor(Class.ASD)~., data=data, method="treebag", metric=metric, trControl=control

# 2.) Random Forest
set.seed(seed)
fit.rf <- train(factor(Class.ASD)~., data=data, method="rf", metric=metric, trControl=control)

# summarize results for both bagging algorithms
bagging_results <- resamples(list(treebag=fit.treebag, rf=fit.rf))
summary(bagging_results)
```

```
##
## Call:
## summary.resamples(object = bagging_results)
##
## Models: treebag, rf
## Number of resamples: 30
##
## Accuracy
##              Min.   1st Qu.    Median      Mean 3rd Qu. Max. NA's
## treebag 0.9571429 0.9857143 0.9859155 0.9905229       1    1    0
## rf      0.9714286 1.0000000 1.0000000 0.9966732       1    1    0
##
## Kappa
##              Min.   1st Qu.   Median      Mean 3rd Qu. Max. NA's
## treebag 0.8898216 0.9637107 0.964659 0.9759094       1    1    0
## rf      0.9252935 1.0000000 1.000000 0.9913496       1    1    0
```

```
dotplot(bagging_results)
```



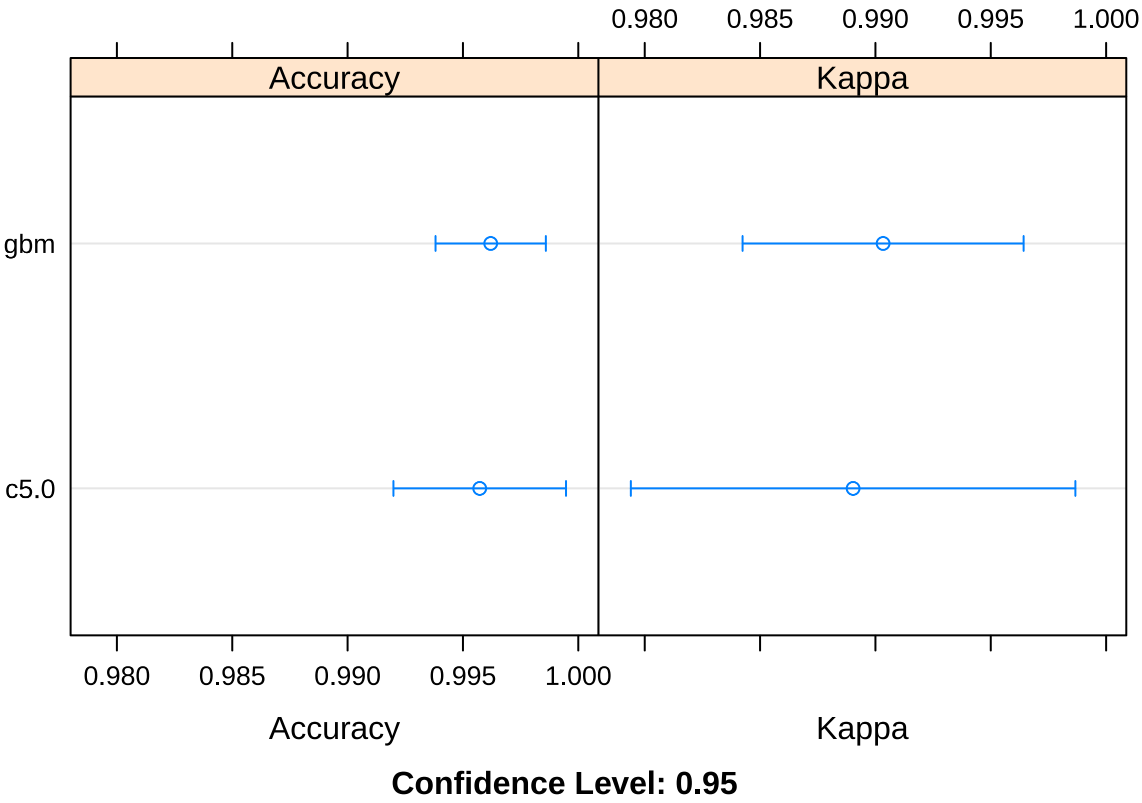
```
# treebag is giving a mean accuracy of 0.9856107 whereas random forest is giving
# a mean accuracy of 0.99926671

# Applying two boosting algorithms:
# C5.0
set.seed(seed)
fit.c50 <- train(factor(Class.ASD)~., data=data, method="C5.0", metric=metric, trControl=control)
```

```
## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 9 for this object. Predictions generated using 9
## trials

## Warning: 'trials' should be <= 9 for this object. Predictions generated using 9
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials
## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials
## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials
## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: ’trials’ should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: ’trials’ should be <= 1 for this object. Predictions generated using 1
## trials
## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials
## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 4 for this object. Predictions generated using 4
## trials

## Warning: 'trials' should be <= 4 for this object. Predictions generated using 4
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 8 for this object. Predictions generated using 8
## trials

## Warning: 'trials' should be <= 7 for this object. Predictions generated using 7
## trials

## Warning: 'trials' should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials

## Warning: 'trials' should be <= 1 for this object. Predictions generated using 1
## trials

## Warning: 'trials' should be <= 3 for this object. Predictions generated using 3
## trials
## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials

## Warning: 'trials' should be <= 6 for this object. Predictions generated using 6
## trials
```

```
# Stochastic Gradient Boosting
set.seed(seed)
fit.gbm <- train(factor(Class.ASD)~., data=data, method="gbm", metric=metric, trControl=control, verbos

# summarize results
boosting_results <- resamples(list(c5.0=fit.c50, gbm=fit.gbm))
summary(boosting_results)
```

```
##
## Call:
## summary.resamples(object = boosting_results)
##
## Models: c5.0, gbm
## Number of resamples: 30
##
## Accuracy
##           Min.   1st Qu. Median      Mean 3rd Qu. Max. NA's
## c5.0 0.9571429 1.0000000      1 0.9957277       1    1    0
## gbm  0.9855072 0.9894366      1 0.9962037       1    1    0
##
## Kappa
##           Min.   1st Qu. Median      Mean 3rd Qu. Max. NA's
## c5.0 0.8898216 1.0000000      1 0.9890325       1    1    0
## gbm  0.9617304 0.9734943      1 0.9903330       1    1    0
```

```
dotplot(boosting_results)
```



```
# Mean accuracy of C5.0 is 0.9971429
# Mean accuracy of gbm is also 0.9971429
# both the boosting algorithms are giving same accuracy

# Performing hyperparameter tuning for stochastic gradient boosting
hyperparameter_grid <- expand.grid(
  .n.trees = c(250, 500),
  .interaction.depth=c(2,3),
  .shrinkage=0.5,
  .n.minobsinnode=10
)

data_2 <- data[,-6]
target_class <- factor(ifelse(data$Class.ASD == 0, "No", "Yes"))
data_2 <- cbind(data_2, target_class)
fit_tuned <- train(target_class ~ . , data = data_2,
             method = "gbm",
             trControl = trainControl(method="cv", number = 5, verboseIter = TRUE, classProbs = TRUE),
             tuneGrid = hyperparameter_grid)
```

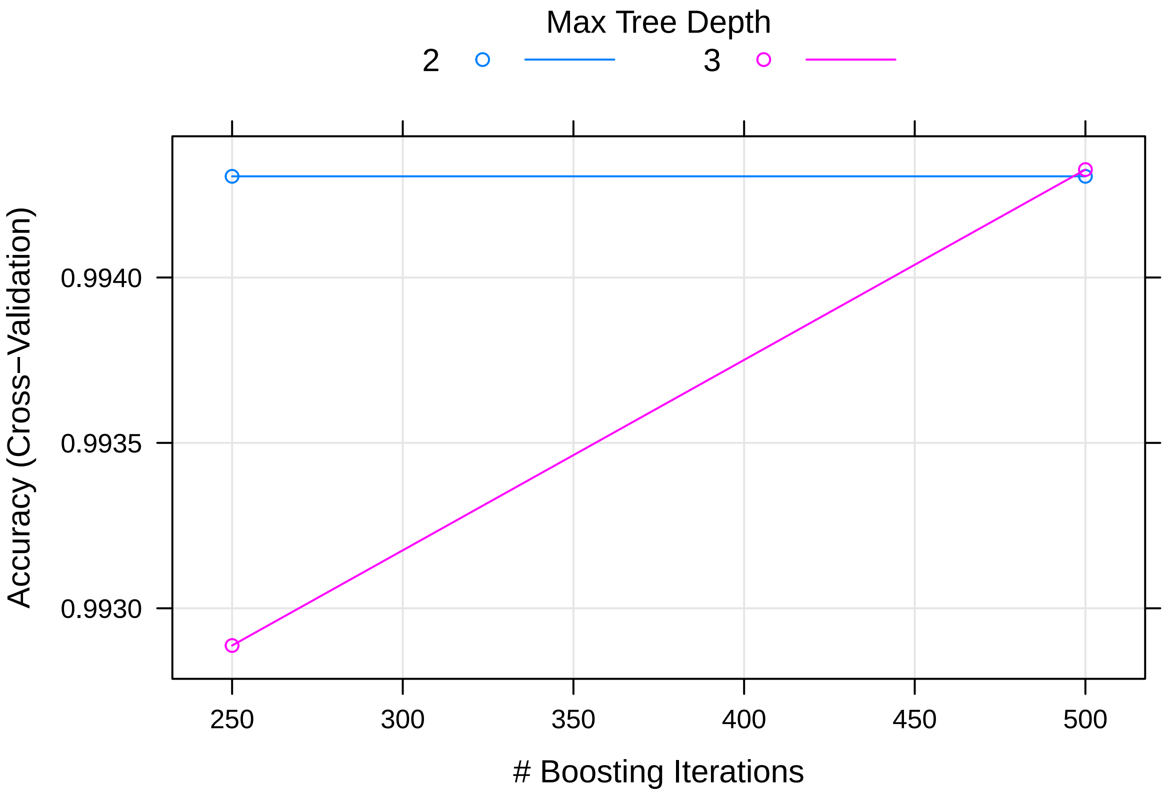
```
## + Fold1: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4599             nan     0.5000    0.3435
##      2        0.2674             nan     0.5000    0.0939
##      3        0.1683             nan     0.5000    0.0467
##      4        0.1118             nan     0.5000    0.0244
##      5        0.0811             nan     0.5000    0.0118
##      6        0.0524             nan     0.5000    0.0102
##      7        0.0416             nan     0.5000   -0.0002
##      8        0.0257             nan     0.5000    0.0046
##      9        0.0226             nan     0.5000   -0.0001
##     10        0.0220             nan     0.5000   -0.0009
##     20        0.0132             nan     0.5000   -0.0039
##     40        0.0054             nan     0.5000   -0.0008
##     60        0.0033             nan     0.5000   -0.0008
##     80        0.0004             nan     0.5000   -0.0001
##    100        0.0003             nan     0.5000   -0.0001
##    120        0.0003             nan     0.5000   -0.0001
##    140        0.0002             nan     0.5000   -0.0000
##    160        0.0000             nan     0.5000   -0.0000
##    180        0.0000             nan     0.5000   -0.0000
##    200        0.0000             nan     0.5000   -0.0000
##    220        0.0000             nan     0.5000   -0.0000
##    240        0.0000             nan     0.5000    0.0000
##    260        0.0001             nan     0.5000   -0.0001
##    280        0.0000             nan     0.5000   -0.0000
##    300        0.0000             nan     0.5000   -0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0002             nan     0.5000   -0.0001
##    360        0.0000             nan     0.5000   -0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0001             nan     0.5000   -0.0000
##    480        0.0001             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold1: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## + Fold1: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4595             nan     0.5000    0.3808
##      2        0.2714             nan     0.5000    0.0879
##      3        0.1679             nan     0.5000    0.0498
##      4        0.1078             nan     0.5000    0.0278
##      5        0.0616             nan     0.5000    0.0248
##      6        0.0382             nan     0.5000    0.0079
##      7        0.0265             nan     0.5000    0.0040
##      8        0.0205             nan     0.5000    0.0019
##      9        0.0164             nan     0.5000    0.0001
##     10        0.0134             nan     0.5000    0.0012
##     20        0.0065             nan     0.5000   -0.0007
##     40        0.0012             nan     0.5000   -0.0002
##     60        0.0012             nan     0.5000   -0.0003
##     80        0.0011             nan     0.5000   -0.0000
##    100        0.0009             nan     0.5000   -0.0000
##    120        0.0006             nan     0.5000   -0.0001
##    140        0.0008             nan     0.5000   -0.0002
##    160        0.0016             nan     0.5000   -0.0005
##    180        0.0005             nan     0.5000   -0.0001
##    200        0.0004             nan     0.5000    0.0000
##    220        0.0004             nan     0.5000   -0.0001
##    240        0.0004             nan     0.5000   -0.0001
##    260        0.0006             nan     0.5000   -0.0002
##    280        0.0003             nan     0.5000   -0.0000
##    300        0.0001             nan     0.5000   -0.0000
##    320        0.0006             nan     0.5000   -0.0002
##    340        0.0002             nan     0.5000   -0.0000
##    360        0.0000             nan     0.5000   -0.0000
##    380        0.0005             nan     0.5000   -0.0002
##    400        0.0001             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0003             nan     0.5000   -0.0001
##    480        0.0001             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold1: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## + Fold2: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4597             nan     0.5000    0.3460
##      2        0.2610             nan     0.5000    0.0956
##      3        0.1649             nan     0.5000    0.0466
##      4        0.1108             nan     0.5000    0.0229
##      5        0.0660             nan     0.5000    0.0160
##      6        0.0455             nan     0.5000    0.0089
##      7        0.0351             nan     0.5000    0.0039
##      8        0.0270             nan     0.5000    0.0014
##      9        0.0241             nan     0.5000    0.0007
##     10        0.0208             nan     0.5000   -0.0006
##     20        0.0027             nan     0.5000    0.0004
##     40        0.0007             nan     0.5000   -0.0001
##     60        0.0003             nan     0.5000    0.0000
##     80        0.0005             nan     0.5000    0.0001
##    100        0.0001             nan     0.5000    0.0000
##    120        0.0001             nan     0.5000   -0.0000
##    140        0.0002             nan     0.5000   -0.0001
##    160        0.0000             nan     0.5000   -0.0000
##    180        0.0000             nan     0.5000   -0.0000
##    200        0.0000             nan     0.5000   -0.0000
##    220        0.0000             nan     0.5000   -0.0000
##    240        0.0000             nan     0.5000   -0.0000
##    260        0.0000             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000   -0.0000
##    300        0.0000             nan     0.5000   -0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000   -0.0000
##    360        0.0000             nan     0.5000   -0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0000             nan     0.5000   -0.0000
##    480        0.0000             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold2: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## + Fold2: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4612             nan     0.5000    0.3445
##      2        0.2650             nan     0.5000    0.0968
##      3        0.1638             nan     0.5000    0.0510
##      4        0.1075             nan     0.5000    0.0256
##      5        0.0747             nan     0.5000    0.0120
##      6        0.0478             nan     0.5000    0.0118
##      7        0.0331             nan     0.5000    0.0051
##      8        0.0214             nan     0.5000    0.0030
##      9        0.0148             nan     0.5000    0.0031
##     10        0.0133             nan     0.5000   -0.0006
##     20        0.0049             nan     0.5000   -0.0002
##     40        0.0046             nan     0.5000   -0.0003
##     60        0.0003             nan     0.5000   -0.0001
##     80        0.0002             nan     0.5000   -0.0000
##    100        0.0000             nan     0.5000   -0.0000
##    120        0.0001             nan     0.5000   -0.0000
##    140        0.0000             nan     0.5000   -0.0000
##    160        0.0000             nan     0.5000   -0.0000
##    180        0.0000             nan     0.5000   -0.0000
##    200        0.0000             nan     0.5000    0.0000
##    220        0.0000             nan     0.5000   -0.0000
##    240        0.0000             nan     0.5000   -0.0000
##    260        0.0000             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000   -0.0000
##    300        0.0000             nan     0.5000   -0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000   -0.0000
##    360        0.0000             nan     0.5000    0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0000             nan     0.5000   -0.0000
##    480        0.0000             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold2: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## + Fold3: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4643             nan     0.5000    0.3402
##      2        0.2622             nan     0.5000    0.1039
##      3        0.1598             nan     0.5000    0.0501
##      4        0.1020             nan     0.5000    0.0281
##      5        0.0680             nan     0.5000    0.0150
##      6        0.0483             nan     0.5000    0.0077
##      7        0.0311             nan     0.5000    0.0057
##      8        0.0217             nan     0.5000    0.0027
##      9        0.0184             nan     0.5000    0.0005
##     10        0.0126             nan     0.5000    0.0013
##    20        0.0018             nan     0.5000    0.0001
##    40        0.0032             nan     0.5000   -0.0007
##    60        0.0015             nan     0.5000   -0.0004
##    80        0.0007             nan     0.5000   -0.0001
##   100        0.0005             nan     0.5000   -0.0001
##   120        0.0002             nan     0.5000   -0.0000
##   140        0.0001             nan     0.5000   -0.0000
##   160        0.0001             nan     0.5000   -0.0000
##   180        0.0001             nan     0.5000   -0.0000
##   200        0.0000             nan     0.5000   -0.0000
##   220        0.0000             nan     0.5000   -0.0000
##   240        0.0000             nan     0.5000    0.0000
##   260        0.0000             nan     0.5000    0.0000
##   280        0.0000             nan     0.5000    0.0000
##   300        0.0000             nan     0.5000   -0.0000
##   320        0.0000             nan     0.5000   -0.0000
##   340        0.0000             nan     0.5000   -0.0000
##   360        0.0000             nan     0.5000    0.0000
##   380        0.0000             nan     0.5000   -0.0000
##   400        0.0000             nan     0.5000    0.0000
##   420        0.0000             nan     0.5000   -0.0000
##   440        0.0000             nan     0.5000   -0.0000
##   460        0.0000             nan     0.5000   -0.0000
##   480        0.0000             nan     0.5000   -0.0000
##   500        0.0000             nan     0.5000   -0.0000
##
## - Fold3: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## + Fold3: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4548             nan     0.5000    0.3306
##      2        0.2636             nan     0.5000    0.0929
##      3        0.1634             nan     0.5000    0.0474
##      4        0.1085             nan     0.5000    0.0257
##      5        0.0762             nan     0.5000    0.0139
##      6        0.0536             nan     0.5000    0.0083
##      7        0.0316             nan     0.5000    0.0110
##      8        0.0264             nan     0.5000    0.0011
##      9        0.0172             nan     0.5000    0.0003
##     10        0.0144             nan     0.5000   -0.0012
##     20        0.0059             nan     0.5000   -0.0007
##     40        0.0032             nan     0.5000   -0.0007
##     60        0.0010             nan     0.5000   -0.0001
##     80        0.0035             nan     0.5000   -0.0008
##    100        0.0007             nan     0.5000   -0.0000
##    120        0.0004             nan     0.5000   -0.0000
##    140        0.0011             nan     0.5000    0.0006
##    160        0.0004             nan     0.5000   -0.0000
##    180        0.0003             nan     0.5000   -0.0000
##    200        0.0008             nan     0.5000   -0.0003
##    220        0.0001             nan     0.5000   -0.0000
##    240        0.0006             nan     0.5000   -0.0002
##    260        0.0001             nan     0.5000   -0.0000
##    280        0.0001             nan     0.5000   -0.0000
##    300        0.0001             nan     0.5000   -0.0000
##    320        0.0001             nan     0.5000   -0.0000
##    340        0.0001             nan     0.5000   -0.0000
##    360        0.0001             nan     0.5000   -0.0000
##    380        0.0001             nan     0.5000   -0.0000
##    400        0.0002             nan     0.5000   -0.0001
##    420        0.0003             nan     0.5000   -0.0001
##    440        0.0001             nan     0.5000   -0.0000
##    460        0.0002             nan     0.5000    0.0001
##    480        0.0001             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold3: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## + Fold4: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4479             nan     0.5000    0.3721
##      2        0.2580             nan     0.5000    0.0902
##      3        0.1564             nan     0.5000    0.0500
##      4        0.0967             nan     0.5000    0.0274
##      5        0.0634             nan     0.5000    0.0164
##      6        0.0416             nan     0.5000    0.0099
##      7        0.0297             nan     0.5000    0.0049
##      8        0.0213             nan     0.5000    0.0027
##      9        0.0116             nan     0.5000    0.0019
##     10        0.0102             nan     0.5000   -0.0002
##     20        0.0029             nan     0.5000   -0.0005
##     40        0.0006             nan     0.5000   -0.0001
##     60        0.0003             nan     0.5000   -0.0001
##     80        0.0002             nan     0.5000   -0.0000
##    100        0.0002             nan     0.5000   -0.0000
##    120        0.0000             nan     0.5000    0.0000
##    140        0.0000             nan     0.5000   -0.0000
##    160        0.0000             nan     0.5000    0.0000
##    180        0.0001             nan     0.5000   -0.0000
##    200        0.0001             nan     0.5000    0.0000
##    220        0.0000             nan     0.5000    0.0000
##    240        0.0000             nan     0.5000   -0.0000
##    260        0.0000             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000    0.0000
##    300        0.0000             nan     0.5000    0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000    0.0000
##    360        0.0000             nan     0.5000   -0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000    0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0000             nan     0.5000   -0.0000
##    480        0.0000             nan     0.5000    0.0000
##    500        0.0000             nan     0.5000    0.0000
##
## - Fold4: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## + Fold4: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4558             nan     0.5000    0.3571
##      2        0.2585             nan     0.5000    0.0960
##      3        0.1557             nan     0.5000    0.0504
##      4        0.0968             nan     0.5000    0.0287
##      5        0.0603             nan     0.5000    0.0174
##      6        0.0412             nan     0.5000    0.0078
##      7        0.0312             nan     0.5000    0.0034
##      8        0.0224             nan     0.5000    0.0023
##      9        0.0136             nan     0.5000    0.0012
##     10        0.0101             nan     0.5000    0.0010
##     20        0.0032             nan     0.5000   -0.0006
##     40        0.0009             nan     0.5000   -0.0001
##     60        0.0008             nan     0.5000    0.0000
##     80        0.0002             nan     0.5000   -0.0000
##    100        0.0001             nan     0.5000   -0.0000
##    120        0.0000             nan     0.5000    0.0000
##    140        0.0001             nan     0.5000   -0.0000
##    160        0.0000             nan     0.5000   -0.0000
##    180        0.0000             nan     0.5000   -0.0000
##    200        0.0000             nan     0.5000   -0.0000
##    220        0.0000             nan     0.5000    0.0000
##    240        0.0000             nan     0.5000    0.0000
##    260        0.0000             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000   -0.0000
##    300        0.0000             nan     0.5000   -0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000   -0.0000
##    360        0.0000             nan     0.5000    0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0000             nan     0.5000   -0.0000
##    480        0.0000             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold4: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## + Fold5: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4513             nan     0.5000    0.3659
##      2        0.2601             nan     0.5000    0.0941
##      3        0.1607             nan     0.5000    0.0458
##      4        0.1010             nan     0.5000    0.0279
##      5        0.0672             nan     0.5000    0.0143
##      6        0.0426             nan     0.5000    0.0101
##      7        0.0306             nan     0.5000    0.0053
##      8        0.0219             nan     0.5000    0.0012
##      9        0.0201             nan     0.5000   -0.0000
##     10        0.0163             nan     0.5000    0.0011
##     20        0.0044             nan     0.5000   -0.0003
##     40        0.0014             nan     0.5000   -0.0001
##     60        0.0017             nan     0.5000   -0.0002
##     80        0.0022             nan     0.5000   -0.0006
##    100        0.0009             nan     0.5000   -0.0000
##    120        0.0007             nan     0.5000   -0.0001
##    140        0.0008             nan     0.5000   -0.0001
##    160        0.0002             nan     0.5000    0.0000
##    180        0.0001             nan     0.5000   -0.0000
##    200        0.0001             nan     0.5000   -0.0000
##    220        0.0001             nan     0.5000   -0.0000
##    240        0.0000             nan     0.5000   -0.0000
##    260        0.0000             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000   -0.0000
##    300        0.0001             nan     0.5000   -0.0000
##    320        0.0000             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000   -0.0000
##    360        0.0000             nan     0.5000   -0.0000
##    380        0.0000             nan     0.5000   -0.0000
##    400        0.0000             nan     0.5000   -0.0000
##    420        0.0001             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0001             nan     0.5000    0.0000
##    480        0.0000             nan     0.5000   -0.0000
##    500        0.0001             nan     0.5000   -0.0000
##
## - Fold5: shrinkage=0.5, interaction.depth=2, n.minobsinnode=10, n.trees=500
## + Fold5: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4505             nan     0.5000    0.3436
##      2        0.2590             nan     0.5000    0.0943
##      3        0.1580             nan     0.5000    0.0482
##      4        0.1004             nan     0.5000    0.0262
##      5        0.0690             nan     0.5000    0.0137
##      6        0.0384             nan     0.5000    0.0128
##      7        0.0273             nan     0.5000    0.0047
##      8        0.0185             nan     0.5000    0.0010
##      9        0.0130             nan     0.5000    0.0019
##     10        0.0090             nan     0.5000    0.0023
##     20        0.0031             nan     0.5000   -0.0003
##     40        0.0034             nan     0.5000   -0.0007
##     60        0.0029             nan     0.5000   -0.0006
##     80        0.0105             nan     0.5000    0.0017
##    100        0.0016             nan     0.5000   -0.0005
##    120        0.0003             nan     0.5000   -0.0000
##    140        0.0002             nan     0.5000   -0.0000
##    160        0.0001             nan     0.5000   -0.0000
##    180        0.0004             nan     0.5000    0.0002
##    200        0.0002             nan     0.5000   -0.0001
##    220        0.0004             nan     0.5000   -0.0001
##    240        0.0001             nan     0.5000   -0.0000
##    260        0.0001             nan     0.5000   -0.0000
##    280        0.0000             nan     0.5000    0.0000
##    300        0.0000             nan     0.5000   -0.0000
##    320        0.0001             nan     0.5000   -0.0000
##    340        0.0000             nan     0.5000   -0.0000
##    360        0.0001             nan     0.5000   -0.0000
##    380        0.0002             nan     0.5000   -0.0001
##    400        0.0001             nan     0.5000   -0.0000
##    420        0.0000             nan     0.5000   -0.0000
##    440        0.0000             nan     0.5000   -0.0000
##    460        0.0002             nan     0.5000   -0.0000
##    480        0.0000             nan     0.5000   -0.0000
##    500        0.0000             nan     0.5000   -0.0000
##
## - Fold5: shrinkage=0.5, interaction.depth=3, n.minobsinnode=10, n.trees=500
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 500, interaction.depth = 3, shrinkage = 0.5, n.minobsinnode = 10 on full training s
## Iter   TrainDeviance   ValidDeviance   StepSize   Improve
##      1        0.4581             nan     0.5000    0.3488
##      2        0.2632             nan     0.5000    0.0920
##      3        0.1602             nan     0.5000    0.0515
##      4        0.1030             nan     0.5000    0.0275
##      5        0.0685             nan     0.5000    0.0151
##      6        0.0479             nan     0.5000    0.0089
##      7        0.0368             nan     0.5000    0.0040
##      8        0.0281             nan     0.5000    0.0030
##      9        0.0202             nan     0.5000    0.0015
##     10        0.0172             nan     0.5000    0.0003
##     20        0.0104             nan     0.5000   -0.0007
##     40        0.0103             nan     0.5000   -0.0011
##     60        0.0030             nan     0.5000   -0.0007
##     80        0.0016             nan     0.5000   -0.0003
##    100        0.0019             nan     0.5000   -0.0005
##    120        0.0003             nan     0.5000    0.0000
##    140        0.0005             nan     0.5000   -0.0002
##    160        0.0005             nan     0.5000   -0.0002
##    180        0.0003             nan     0.5000   -0.0000
##    200        0.0003             nan     0.5000   -0.0000
##    220        0.0004             nan     0.5000   -0.0001
##    240        0.0003             nan     0.5000   -0.0000
##    260        0.0002             nan     0.5000   -0.0001
##    280        0.0002             nan     0.5000   -0.0000
##    300        0.0003             nan     0.5000   -0.0001
##    320        0.0007             nan     0.5000   -0.0002
##    340        0.0002             nan     0.5000   -0.0000
##    360        0.0001             nan     0.5000   -0.0000
##    380        0.0015             nan     0.5000    0.0007
##    400        0.0004             nan     0.5000   -0.0000
##    420        0.0001             nan     0.5000   -0.0000
##    440        0.0003             nan     0.5000   -0.0000
##    460        0.0003             nan     0.5000   -0.0000
##    480        0.0005             nan     0.5000   -0.0002
##    500        0.0004             nan     0.5000   -0.0001
```

```
print(fit_tuned)
```

```
## Stochastic Gradient Boosting
##
## 703 samples
##   5 predictor
##   2 classes: 'No', 'Yes'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 564, 562, 562, 562, 562
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   2                  250      0.9943058  0.9855108
##   2                  500      0.9943058  0.9855108
##   3                  250      0.9928874  0.9818179
##   3                  500      0.9943262  0.9854695
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.5
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 500, interaction.depth =
##  3, shrinkage = 0.5 and n.minobsinnode = 10.
```

```
plot(fit_tuned)
```

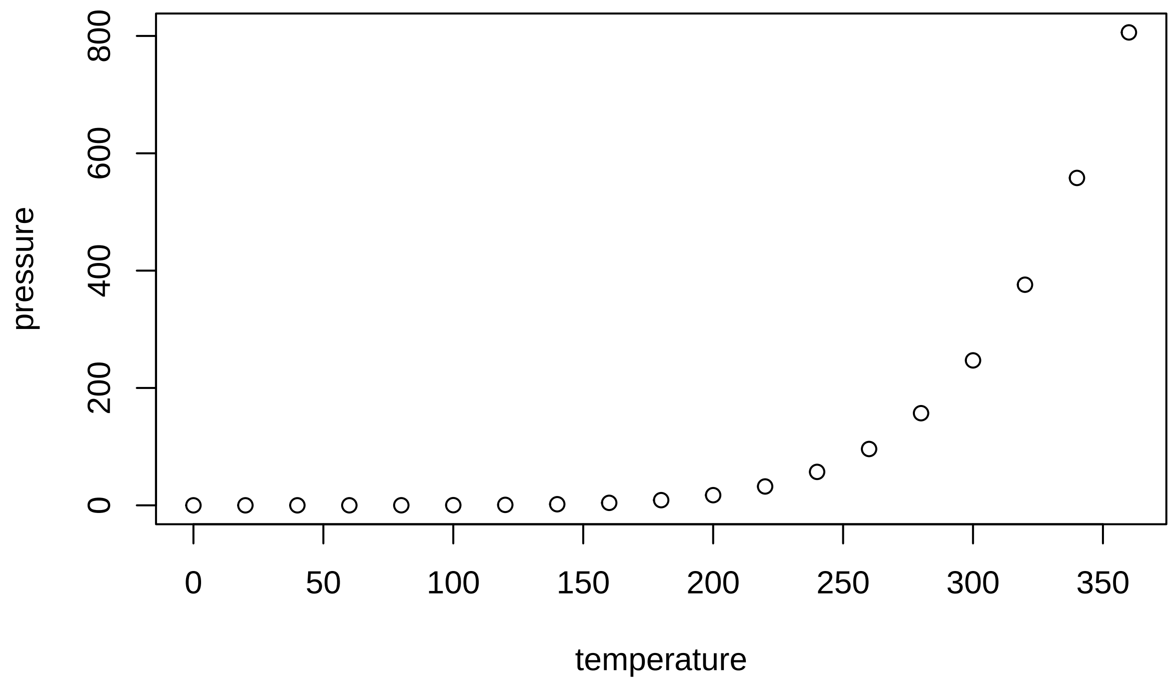


```
# The following accuracies were obtained corresponding to the hyperparameters
# interaction.depth  n.trees  Accuracy   Kappa
# 2                  250      0.9971631  0.9927946
# 2                  500      0.9957447  0.9892223
# 3                  250      0.9971631  0.9926734
# 3                  500      0.9957345  0.9891526

# We can see that the after hyperparameter tuning the accuracy of gbm slightly
# increased from 0.9971429 to 0.9971631 at interaction depth 2 and number of trees
# equal to 250


# To conclude, all the implemented algorithms performed well on our dataset
# in classifying the people into ASD patients and normal based on the 20 independent
# features. PCA was performed on these 20 features and first five principal components
# were selected as they covered more than 95% vraiation in the dataset and all the
# machine learning algorithms and ANN was implemented using these 5 principal
# components as independent features.
```

Including Plots

You can also embed plots, for example:

```
plot(pressure)
```



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.