FinalProj

Omar Hassoun

December 15, 2019

Contents


```
 bootr = function(B, train){
  mod <-svm(`Class/ASD`~.-result-country_of_residence, data=train, kernel="radial", gamma=1)

preds <- predict(mod)

#bootstrap:

#number of bootstrap resamples
bootpred <- matrix(ncol = length(preds), nrow = 100)
 #for reproducibility

#loop over n
for (i in 1:B) {
  ind = sample(nrow(train), replace = TRUE)
  bootdat <- train[ind,] #bootstrap resample of data
  bootmod <- svm(`Class/ASD`~.-result-country_of_residence, data = bootdat) #fit model to bootstrap res
  bootpred[i,] <- predict(bootmod, type="response") #calculate predictions from this model
}
bopred = colMeans(bootpred)
}
```

```r
library("readxl")
```

```
## Warning: package 'readxl' was built under R version 4.0.5
```

```r
autism = read_excel("AutismData.xlsx")
autism_original = autism
```

Data Cleaning

Type Conversion

```r
#We observe the variables to get an overview
dim(autism)
```

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

```r
summary(autism)
```

```
##     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   Length:704         Length:704        
##  1st Qu.:0.0000   1st Qu.:0.0000   Class :character   Class :character  
##  Median :0.0000   Median :1.0000   Mode  :character   Mode  :character  
##  Mean   :0.3239   Mean   :0.5739                                        
##  3rd Qu.:1.0000   3rd Qu.:1.0000                                        
##  Max.   :1.0000   Max.   :1.0000                                        
##   ethnicity           jundice             austim          country_of_residence
##  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_cat            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
##
##
##
```

```r
str(autism)
```

```
## tibble [704 x 21] (S3: tbl_df/tbl/data.frame)
##  $ A1_Score            : num [1:704] 1 1 1 1 1 1 0 1 1 1 ...
##  $ A2_Score            : num [1:704] 1 1 1 1 0 1 1 1 1 1 ...
##  $ A3_Score            : num [1:704] 1 0 0 0 0 1 0 1 0 1 ...
##  $ A4_Score            : num [1:704] 1 1 1 1 0 1 0 1 0 1 ...
##  $ A5_Score            : num [1:704] 0 0 1 0 0 1 0 0 1 0 ...
##  $ A6_Score            : num [1:704] 0 0 0 0 0 0 0 0 0 1 ...
##  $ A7_Score            : num [1:704] 1 0 1 1 0 1 0 0 0 1 ...
##  $ A8_Score            : num [1:704] 1 1 1 1 1 1 1 0 1 1 ...
##  $ A9_Score            : num [1:704] 0 0 1 0 0 1 0 1 1 1 ...
##  $ A10_Score           : num [1:704] 0 1 1 1 0 1 0 0 1 0 ...
##  $ age                 : chr [1:704] "26" "24" "27" "35" ...
##  $ gender              : chr [1:704] "f" "m" "m" "f" ...
##  $ ethnicity           : chr [1:704] "White-European" "Latino" "Latino" "White-European" ...
##  $ jundice             : chr [1:704] "no" "no" "yes" "no" ...
##  $ austim              : chr [1:704] "no" "yes" "yes" "yes" ...
##  $ country_of_residence: chr [1:704] "'United States'" "Brazil" "Spain" "'United States'" ...
##  $ used_app_before     : chr [1:704] "no" "no" "no" "no" ...
##  $ result              : num [1:704] 6 5 8 6 2 9 2 5 6 8 ...
##  $ age_cat             : chr [1:704] "'18 and more'" "'18 and more'" "'18 and more'" "'18 and more'"
##  $ relation            : chr [1:704] "Self" "Self" "Parent" "Self" ...
##  $ Class/ASD           : chr [1:704] "NO" "NO" "YES" "NO" ...
```

```r
# Type conversion
#get questions 1 to 10 and convert to binary
bin_vars = grep("A[0-9]+_Score", colnames(autism))
#these are the columns that are going to be changed
colnames(autism[bin_vars])
```

```
##  [1] "A1_Score"  "A2_Score"  "A3_Score"  "A4_Score"  "A5_Score"  "A6_Score"
##  [7] "A7_Score"  "A8_Score"  "A9_Score"  "A10_Score"
```

```r
summary(autism[bin_vars])
```

```
##     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
##  Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :1.0000
##  Mean   :0.3239   Mean   :0.5739
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

```r
str(autism[bin_vars])
```

```
## tibble [704 x 10] (S3: tbl_df/tbl/data.frame)
##  $ A1_Score : num [1:704] 1 1 1 1 1 1 0 1 1 1 ...
##  $ A2_Score : num [1:704] 1 1 1 1 0 1 1 1 1 1 ...
##  $ A3_Score : num [1:704] 1 0 0 0 0 1 0 1 0 1 ...
##  $ A4_Score : num [1:704] 1 1 1 1 0 1 0 1 0 1 ...
##  $ A5_Score : num [1:704] 0 0 1 0 0 1 0 0 1 0 ...
##  $ A6_Score : num [1:704] 0 0 0 0 0 0 0 0 0 1 ...
##  $ A7_Score : num [1:704] 1 0 1 1 0 1 0 0 0 1 ...
##  $ A8_Score : num [1:704] 1 1 1 1 1 1 1 0 1 1 ...
##  $ A9_Score : num [1:704] 0 0 1 0 0 1 0 1 1 1 ...
##  $ A10_Score: num [1:704] 0 1 1 1 0 1 0 0 1 0 ...
```

```r
#there are no missing values among these columns
autism[bin_vars] <- lapply(autism[bin_vars] , factor)
#some values that were characters were converted to numeric
#there are two missing observations in age that are represented by "?"
autism$age[autism$age == "?"] #they were 2 handled when converted into numeric
```

```
## [1] "?" "?"
```

```r
autism$age = as.numeric(autism$age)
```

```
## Warning: NAs introduced by coercion
```

```r
summary(autism$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    17.0    21.0    27.0    29.7    35.0   383.0       2
```

```r
#NAs will be handled after convertinng and vizualizng other variables
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## 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
```

```r
count(autism, ethnicity) #There are 2 variables called "others" and 95 "?"
```

```
## # A tibble: 12 x 2
##    ethnicity             n
```

```
##    <chr>              <int>
##  1 'Middle Eastern '    92
##  2 'South Asian'        36
##  3 ?                    95
##  4 Asian               123
##  5 Black                43
##  6 Hispanic             13
##  7 Latino               20
##  8 others                1
##  9 Others               30
## 10 Pasifika             12
## 11 Turkish               6
## 12 White-European      233
```

```
autism['others' == autism$ethnicity,'ethnicity'] =  "Others"#the obs name was fixed
autism['?' == autism$ethnicity,'ethnicity'] = NA
count(autism, gender)
```

```
## # A tibble: 2 x 2
##   gender     n
##   <chr>  <int>
## 1 f        337
## 2 m        367
```

```
autism$gender = as.factor(autism$gender)
#Jaundice
count(autism, jundice)
```

```
## # A tibble: 2 x 2
##   jundice     n
##   <chr>   <int>
## 1 no        635
## 2 yes        69
```

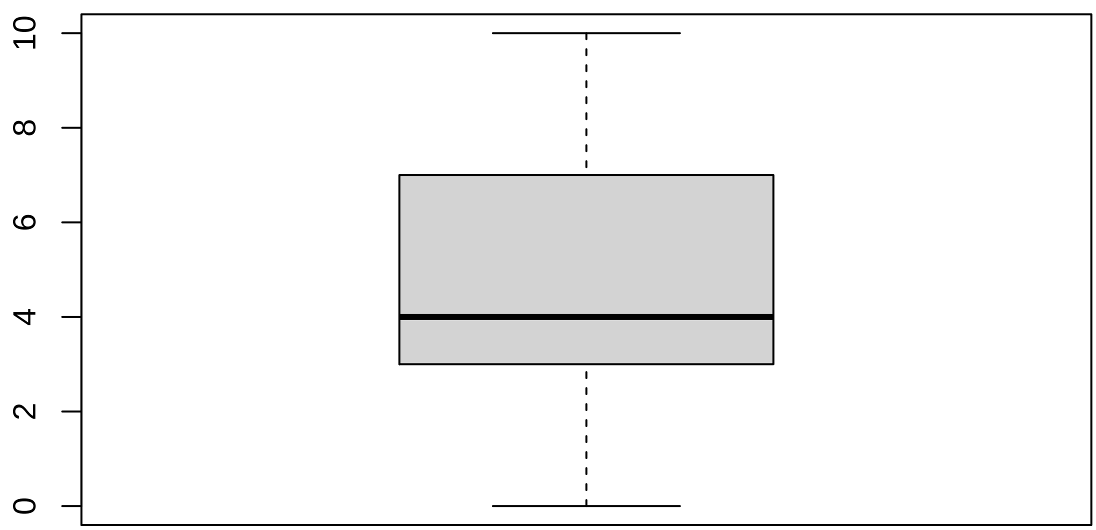
```
autism$jundice = as.factor(autism$jundice)
count(autism, country_of_residence)
```

```
## # A tibble: 67 x 2
##    country_of_residence       n
##    <chr>                  <int>
##  1 'Costa Rica'               1
##  2 'Czech Republic'           1
##  3 'Hong Kong'                1
##  4 'New Zealand'             81
##  5 'Saudi Arabia'             4
##  6 'Sierra Leone'             1
##  7 'South Africa'             2
##  8 'Sri Lanka'               14
##  9 'United Arab Emirates'    82
## 10 'United Kingdom'          77
## # ... with 57 more rows
```

```
#Converted into factors to use as part of predicttions, since
#diseases and conditions are affected by the enviroment
autism$country_of_residence = as.factor(autism$country_of_residence)
#converted to factor
count(autism, used_app_before) #count function used to check for NAs or NAs like ("?")
```

```
## # A tibble: 2 x 2
##   used_app_before     n
##   <chr>           <int>
## 1 no                692
## 2 yes                12
```

```r
autism$used_app_before = as.factor(autism$used_app_before)

boxplot(autism$result)
```




```r
count(autism, age_cat)
```

```
## # A tibble: 1 x 2
##   age_cat            n
##   <chr>          <int>
## 1 '18 and more'    704
```

```r
#factor again
#this is the class
autism$`Class/ASD` = as.factor(autism$`Class/ASD`)

count(autism, relation)
```

```
## # A tibble: 6 x 2
##   relation                        n
##   <chr>                       <int>
## 1 'Health care professional'      4
```

```
## 2 ?                              95
## 3 Others                          5
## 4 Parent                         50
## 5 Relative                       28
## 6 Self                          522
```

```r
autism$relation[autism$relation == "?"] = NA
autism$relation = as.factor(autism$relation)

count(autism, austim) #this is family PDD
```

```
## # A tibble: 2 x 2
##   austim     n
##   <chr>  <int>
## 1 no       613
## 2 yes       91
```

```r
autism$austim = as.factor(autism$austim)


autism$ethnicity = as.factor(autism$ethnicity)
```

NA Handling

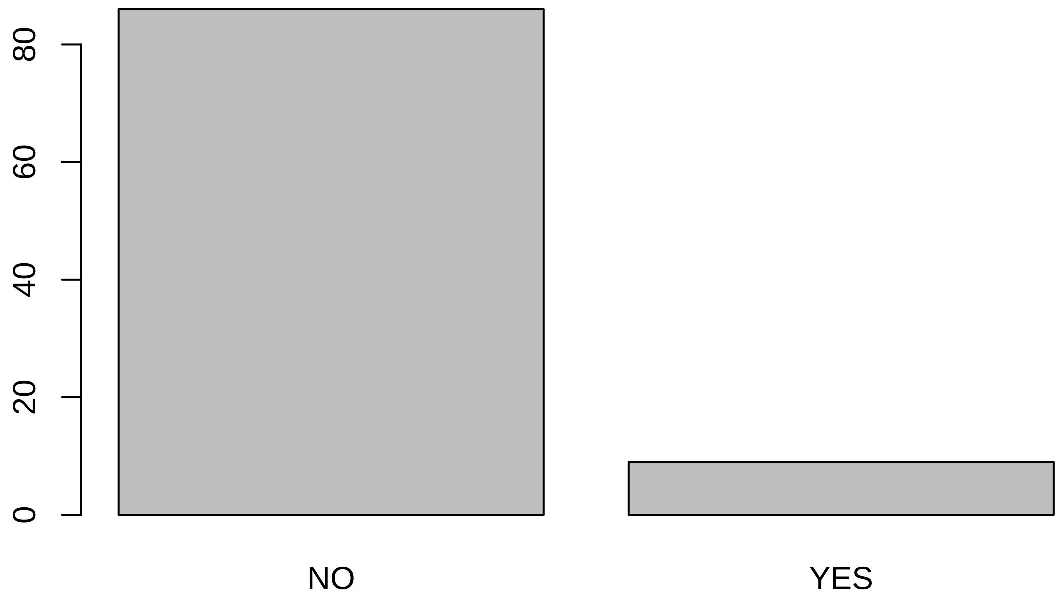
```r
#count the NAs
sum(is.na(autism))
```

```
## [1] 192
```

```r
#returns columns that has missing values
sapply(autism, function(x){
  sum(is.na(x))
})
```

```
##             A1_Score             A2_Score             A3_Score
##                    0                    0                    0
##             A4_Score             A5_Score             A6_Score
##                    0                    0                    0
##             A7_Score             A8_Score             A9_Score
##                    0                    0                    0
##            A10_Score                  age               gender
##                    0                    2                    0
##            ethnicity              jundice               austim
##                   95                    0                    0
## country_of_residence      used_app_before               result
##                    0                    0                    0
##              age_cat             relation            Class/ASD
##                    0                   95                    0
```

```r
#Most of the missing values are in the relation column
#because they are a lot of values, almost a 7th and could highly

plot(autism[is.na(autism$relation),'Class/ASD'])
```




```r
#we observe that missing values of individuals that have autism are higher proportion
#we decide to put "Unknown" in places where we have NAs in relation column and check if it affects pred
# Get levels and add "Unknown"
levels <- levels(autism$relation)
levels[length(levels) + 1] <- "Unknown"
autism$relation <- factor(autism$relation, levels = levels)
autism$relation[is.na(autism$relation)] <- "Unknown"
count(autism, relation)
```

```
## # A tibble: 6 x 2
##   relation                      n
##   <fct>                     <int>
## 1 'Health care professional'    4
## 2 Others                        5
## 3 Parent                       50
## 4 Relative                     28
## 5 Self                        522
## 6 Unknown                      95
```

```r
#There were missing values at age:
sum(is.na(autism$age))
```

```
## [1] 2
```

```r
autism[is.na(autism$age),]
```

```
## # A tibble: 2 x 21
##   A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score
```

```
##   <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>
## 1 0        0        0        0        0        0        0        0
## 2 0        1        0        0        1        0        1        0
## # ... with 13 more variables: A9_Score <fct>, A10_Score <fct>, age <dbl>,
## #   gender <fct>, ethnicity <fct>, jundice <fct>, austim <fct>,
## #   country_of_residence <fct>, used_app_before <fct>, result <dbl>,
## #   age_cat <chr>, relation <fct>, Class/ASD <fct>
```

```
# will replace it with the median of the ages, since the screeing method type was the same for all vari
#and there is an outlier that is increasing the mean a lot
autism$age[is.na(autism$age)] = median(autism$age)
sapply(autism, function(x){
  sum(is.na(x))
})
```

```
##            A1_Score            A2_Score            A3_Score
##                   0                   0                   0
##            A4_Score            A5_Score            A6_Score
##                   0                   0                   0
##            A7_Score            A8_Score            A9_Score
##                   0                   0                   0
##           A10_Score                 age              gender
##                   0                   2                   0
##           ethnicity             jundice              austim
##                  95                   0                   0
## country_of_residence     used_app_before              result
##                   0                   0                   0
##             age_cat            relation           Class/ASD
##                   0                   0                   0
```

```
#lastly we will handle the ethnicity column
autism[is.na(autism$ethnicity),]
```

```
## # A tibble: 95 x 21
##    A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score
##    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>
##  1 1        0        0        0        0        0        0        1
##  2 0        1        1        1        1        1        0        0
##  3 1        0        0        0        0        0        1        1
##  4 1        0        0        0        0        0        1        1
##  5 0        0        0        0        0        0        1        1
##  6 0        1        1        1        0        0        0        0
##  7 1        1        1        1        0        0        0        1
##  8 0        1        1        0        0        0        0        1
##  9 0        0        0        0        0        0        0        0
## 10 1        1        0        0        0        0        0        0
## # ... with 85 more rows, and 13 more variables: A9_Score <fct>,
## #   A10_Score <fct>, age <dbl>, gender <fct>, ethnicity <fct>, jundice <fct>,
## #   austim <fct>, country_of_residence <fct>, used_app_before <fct>,
## #   result <dbl>, age_cat <chr>, relation <fct>, Class/ASD <fct>
```

```
#apparently columns that have missing values in ethnicity also have missing values in relation
#often the ethnicity can be determined from the country of residence e.g: White people from some europe
#i've decided to change missing values for just "Unknown"

ethno_res = autism %>% group_by(country_of_residence) %>% count(ethnicity)
```

```r
#get indices of misisng values

levels <- levels(autism$ethnicity)
levels[length(levels) + 1] <- "Unknown"
autism$ethnicity <- factor(autism$ethnicity, levels = levels)
autism$ethnicity[is.na(autism$ethnicity)] <- "Unknown"
count(autism, ethnicity)
```

```
## # A tibble: 11 x 2
##    ethnicity            n
##    <fct>            <int>
##  1 'Middle Eastern '    92
##  2 'South Asian'        36
##  3 Asian               123
##  4 Black                43
##  5 Hispanic             13
##  6 Latino               20
##  7 Others               31
##  8 Pasifika             12
##  9 Turkish               6
## 10 White-European      233
## 11 Unknown              95
```

```r
sum(is.na(autism))
```

```
## [1] 2
```

```r
#finally, we will handle the missing values in age
missing_ages = which(is.na(autism$age))
autism[missing_ages,]
```

```
## # A tibble: 2 x 21
##   A1_Score A2_Score A3_Score A4_Score A5_Score A6_Score A7_Score A8_Score
##   <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>    <fct>
## 1 0        0        0        0        0        0        0        0
## 2 0        1        0        0        1        0        1        0
## # ... with 13 more variables: A9_Score <fct>, A10_Score <fct>, age <dbl>,
## #   gender <fct>, ethnicity <fct>, jundice <fct>, austim <fct>,
## #   country_of_residence <fct>, used_app_before <fct>, result <dbl>,
## #   age_cat <chr>, relation <fct>, Class/ASD <fct>
```

```r
#I've decided to replace the missing ages with some value
#but first I would like to know if age is a significant predictor for the class
#we previously observed an outlier, so wr are excluding it
age.glm = glm(`Class/ASD`~age, data = autism[-max(autism$age, na.rm = TRUE),], family = binomial)
summary(age.glm)
```

```
##
## Call:
## glm(formula = `Class/ASD` ~ age, family = binomial, data = autism[-max(autism$age,
##     na.rm = TRUE), ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9623  -0.7904  -0.7693   1.5477   1.6681
##
```

```
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.238690   0.195373  -6.340  2.3e-10 ***
## age          0.007850   0.005884   1.334    0.182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 815.19  on 700  degrees of freedom
## Residual deviance: 812.87  on 699  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 816.87
##
## Number of Fisher Scoring iterations: 4
```

```
#the p value for the age is not significant, so I will replace age with with the median
age_noNA = autism$age
age_noNA[missing_ages] = median(age_noNA, na.rm = TRUE)
sum(is.na(age_noNA))
```

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

```
autism$age = age_noNA
#all mising values were handled
sum(is.na(autism))
```

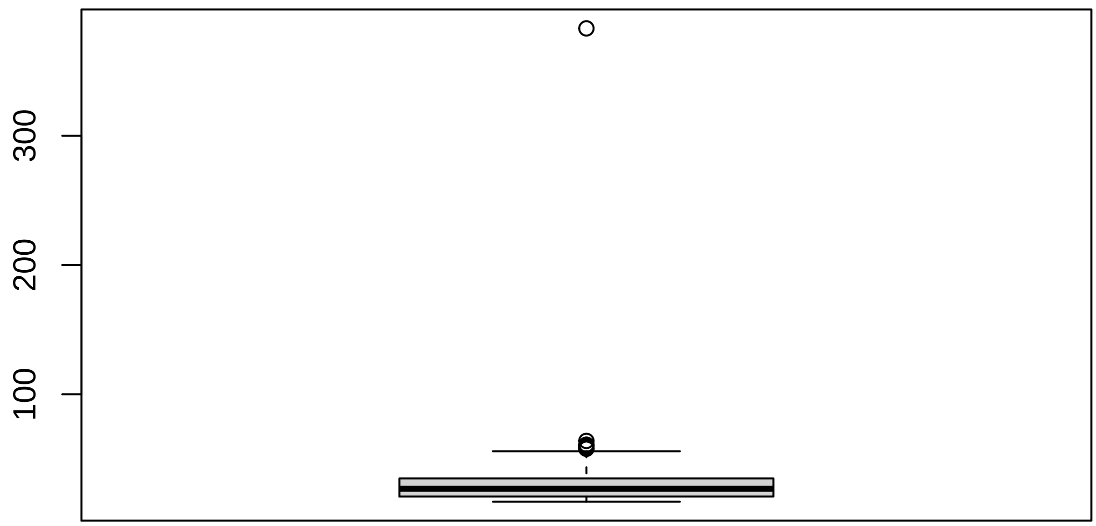
```
## [1] 0
```

Outliers

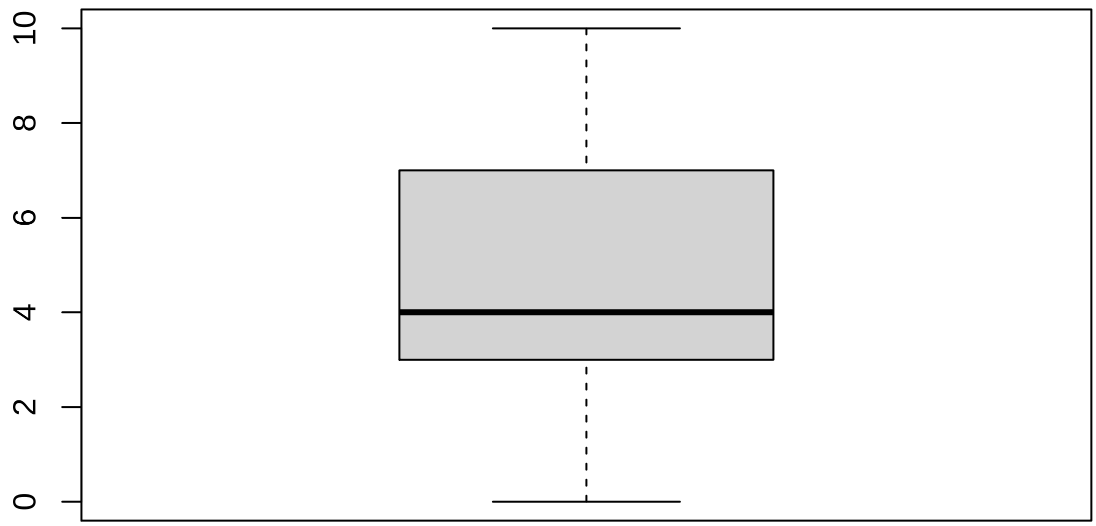
```
#get all numerical variables
numeric_cols = sapply(autism, is.numeric)
autism[,numeric_cols]
```

```
## # A tibble: 704 x 2
##      age result
##    <dbl>  <dbl>
##  1    26      6
##  2    24      5
##  3    27      8
##  4    35      6
##  5    40      2
##  6    36      9
##  7    17      2
##  8    64      5
##  9    29      6
## 10    17      8
## # ... with 694 more rows
```

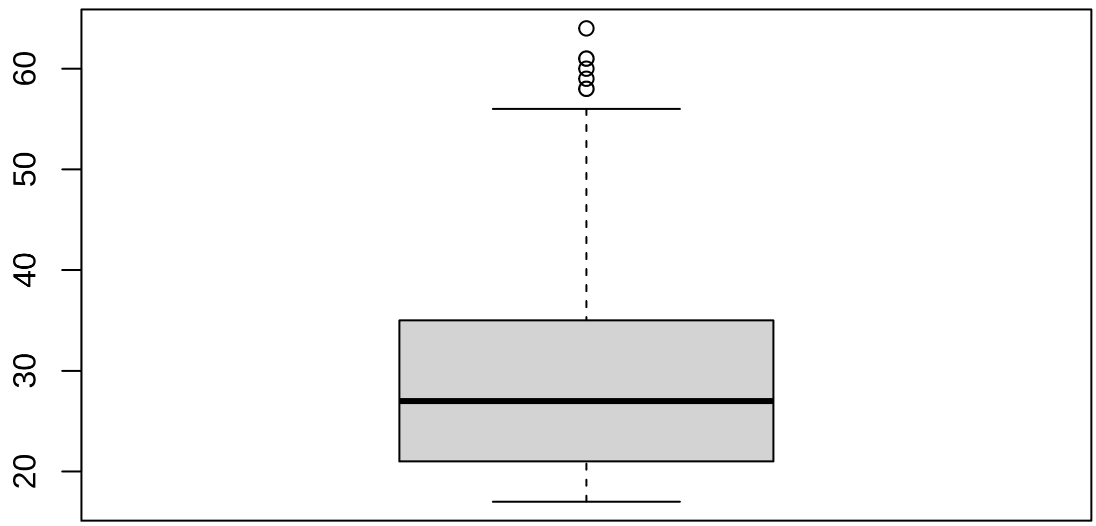
```
#we just have 2 numeric variables
boxplot(autism$age)
```




```r
boxplot(autism$result)
```




```
#age clearly has an outlier, for an age that does not make sense
#so we will replace that age for the median
autism$age[which(max(autism$age) == autism$age)] = median(autism$age)
boxplot(autism$age)$out #after removing the variable, we observe other variables outside
```




```
## [1] 64 58 60 58 61 59 61
```

```
#will not remove them from the data since they are not too far from the mean of the data + 3 standard d
```

Removal

```
#there is a column (age_cat) that contains a single value "18 and more", that will be removed,
#since it's not useful for predictions because there is no variability and they probably used the test
#adults in the those ages below 18
sum(autism$age < 18)
```

```
## [1] 18
```

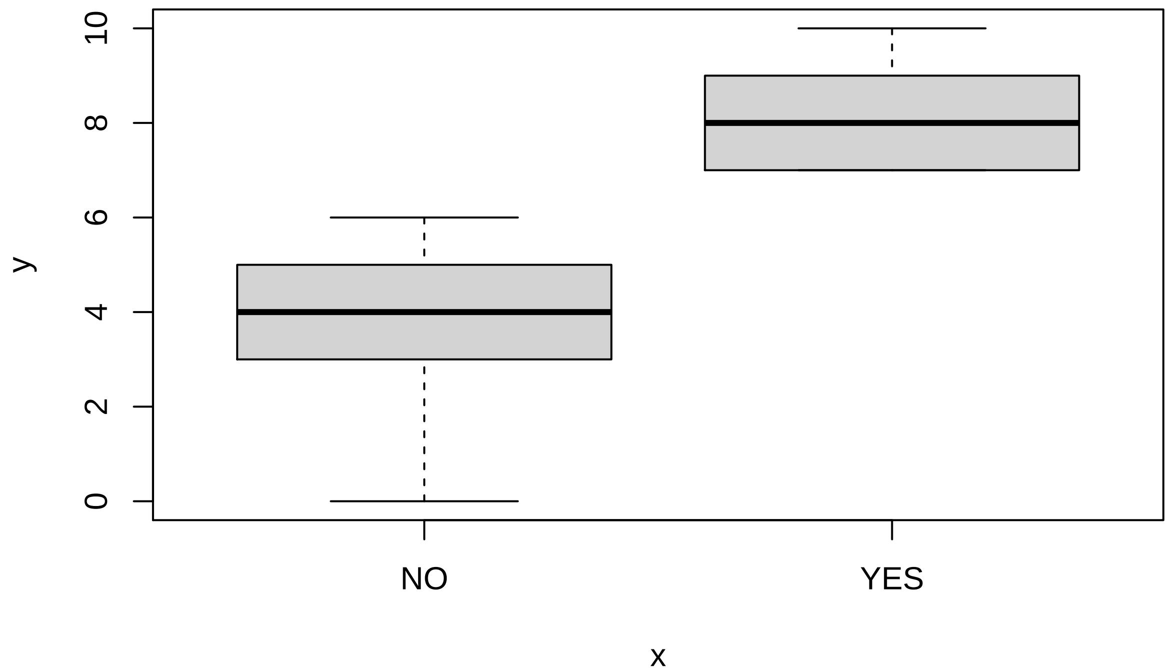
```
autism$age[autism$age < 18]
```

```
##  [1] 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17 17
```

```
autism = autism[,-which(colnames(autism) == "age_cat")]
```

Useful Plots

```
#to check results against the Class variable
plot(autism$`Class/ASD`, autism$result)
```




```
#we observe that all variables above 6 correspond to ASD and below, do not have ASD
autism %>% group_by(result, `Class/ASD`) %>% count(`Class/ASD`)
```

```
## # A tibble: 11 x 3
## # Groups:   result, Class/ASD [11]
##    result `Class/ASD`     n
##     <dbl> <fct>       <int>
##  1      0 NO             14
##  2      1 NO             33
##  3      2 NO             74
##  4      3 NO            110
##  5      4 NO            131
##  6      5 NO             83
##  7      6 NO             70
##  8      7 YES            57
##  9      8 YES            55
## 10      9 YES            47
## 11     10 YES            30
```

```
result.vs.class = glm(result~`Class/ASD`, data = autism)
summary(result.vs.class)
```

```
##
## Call:
## glm(formula = result ~ `Class/ASD`, data = autism)
##
## Deviance Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -3.6311  -1.2646   0.3689   1.3689   2.3689
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.63107    0.06291   57.72   <2e-16 ***
## `Class/ASD`YES  4.63348    0.12141   38.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.037999)
##
##     Null deviance: 4399.0  on 703  degrees of freedom
## Residual deviance: 1430.7  on 702  degrees of freedom
## AIC: 2503.1
##
## Number of Fisher Scoring iterations: 2
```

Model Building

Trees

Naive Training Tree

```
library(tree)
```

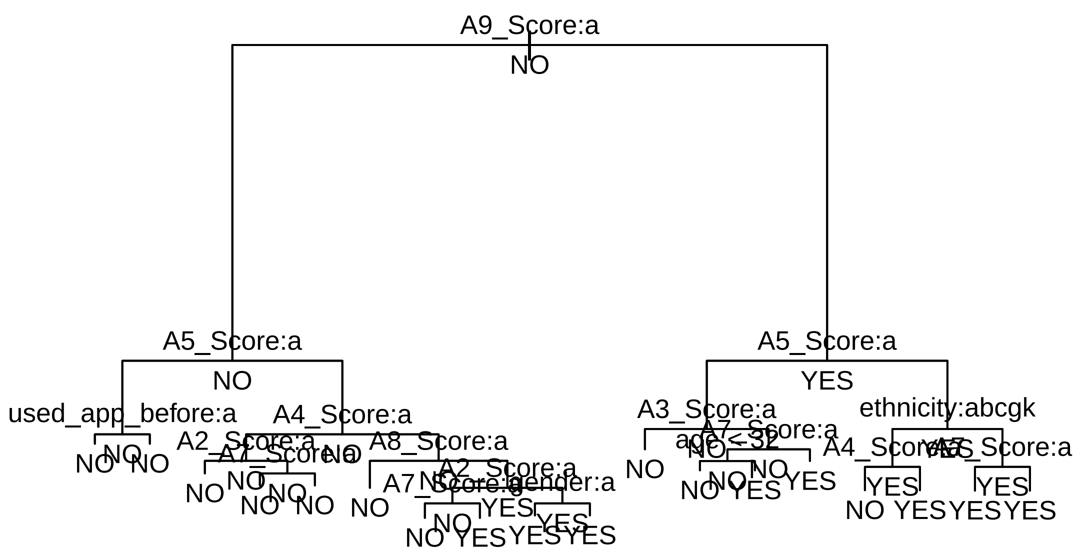
```
## Warning: package 'tree' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
#Tree function does not allow to use factor with more than 32 levels, so I decided to convert each of
#the countries into integers and fit it into the model, if I find out that it's a useful predictor, I w
#keep it
#Training Tree was built with the purpose of collecting the training error
#of a tree using  anaive approach
country_id = as.numeric(autism$country_of_residence)
autism_update = autism %>% mutate(country_id = country_id)
#result should not be used in the model, since Class/ASD variable is obtained through results
tree.attempt = tree(`Class/ASD`~.-country_of_residence-result, data = autism_update)
summary(tree.attempt)
```

```
##
## Classification tree:
## tree(formula = `Class/ASD` ~ . - country_of_residence - result,
##     data = autism_update)
## Variables actually used in tree construction:
##  [1] "A9_Score"        "A5_Score"        "used_app_before" "A4_Score"
##  [5] "A2_Score"        "A7_Score"        "A8_Score"        "gender"
##  [9] "A3_Score"        "age"             "ethnicity"
## Number of terminal nodes:  18
## Residual mean deviance:  0.2662 = 182.6 / 686
## Misclassification error rate: 0.0625 = 44 / 704
```

```r
library(rpart)
plot(tree.attempt,
   main="Classification Tree for ASD")
text(tree.attempt, all=TRUE, cex=.8)
```




```r
tree.attempt
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 704 819.100 NO ( 0.731534 0.268466 )
##    2) A9_Score: 0 476 250.100 NO ( 0.926471 0.073529 )
##      4) A5_Score: 0 304  13.430 NO ( 0.996711 0.003289 )
##        8) used_app_before: no 299   0.000 NO ( 1.000000 0.000000 ) *
##        9) used_app_before: yes 5   5.004 NO ( 0.800000 0.200000 ) *
##      5) A5_Score: 1 172 171.000 NO ( 0.802326 0.197674 )
##       10) A4_Score: 0 87  38.270 NO ( 0.942529 0.057471 )
##         20) A2_Score: 0 57   0.000 NO ( 1.000000 0.000000 ) *
##         21) A2_Score: 1 30  27.030 NO ( 0.833333 0.166667 )
##           42) A7_Score: 0 18   0.000 NO ( 1.000000 0.000000 ) *
##           43) A7_Score: 1 12  16.300 NO ( 0.583333 0.416667 ) *
##       11) A4_Score: 1 85 109.100 NO ( 0.658824 0.341176 )
##         22) A8_Score: 0 30   8.769 NO ( 0.966667 0.033333 ) *
##         23) A8_Score: 1 55  76.230 YES ( 0.490909 0.509091 )
##           46) A2_Score: 0 27  30.900 NO ( 0.740741 0.259259 )
##             92) A7_Score: 0 14   0.000 NO ( 1.000000 0.000000 ) *
```

```
##             93) A7_Score: 1 13  17.940 YES ( 0.461538 0.538462 ) *
##           47) A2_Score: 1 28  31.490 YES ( 0.250000 0.750000 )
##             94) gender: f 13   0.000 YES ( 0.000000 1.000000 ) *
##             95) gender: m 15  20.730 YES ( 0.466667 0.533333 ) *
##    3) A9_Score: 1 228 287.400 YES ( 0.324561 0.675439 )
##      6) A5_Score: 0 49  49.590 NO ( 0.795918 0.204082 )
##       12) A3_Score: 0 26   0.000 NO ( 1.000000 0.000000 ) *
##       13) A3_Score: 1 23  31.490 NO ( 0.565217 0.434783 )
##         26) A7_Score: 0 15  15.010 NO ( 0.800000 0.200000 )
##           52) age < 32 10   0.000 NO ( 1.000000 0.000000 ) *
##           53) age > 32 5   6.730 YES ( 0.400000 0.600000 ) *
##         27) A7_Score: 1 8   6.028 YES ( 0.125000 0.875000 ) *
##      7) A5_Score: 1 179 176.900 YES ( 0.195531 0.804469 )
##       14) ethnicity: 'Middle Eastern ','South Asian',Asian,Others,Unknown 53  73.300 YES ( 0.471698 
##         28) A4_Score: 0 18  12.560 NO ( 0.888889 0.111111 ) *
##         29) A4_Score: 1 35  39.900 YES ( 0.257143 0.742857 ) *
##       15) ethnicity: Black,Hispanic,Latino,Pasifika,Turkish,White-European 126  69.860 YES ( 0.07936
##         30) A7_Score: 0 47  48.650 YES ( 0.212766 0.787234 ) *
##         31) A7_Score: 1 79   0.000 YES ( 0.000000 1.000000 ) *
```

```r
#Na
```

Validation Set Tree

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 4.0.5
## Loading required package: lattice
```

```r
#Separate Data into Test and Train set
set.seed(1) #0.8 was not chosen with any criteria
shuffle =sample(nrow(autism_update))
autism_update = autism_update[shuffle,]
split = round(nrow(autism_update) * .8)
train = autism_update[1:split,]
test = autism_update[(split+1):nrow(autism_update),]
tree.validation = tree(`Class/ASD`~.-country_of_residence-result, data = train)
tree.pred=predict(tree.validation ,test,type="class")
confusionMatrix(tree.pred, test$`Class/ASD`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO  91   6
##        YES  4  40
##
##                Accuracy : 0.9291
##                  95% CI : (0.8734, 0.9655)
##     No Information Rate : 0.6738
##     P-Value [Acc > NIR] : 3.408e-13
##
```

```
##                   Kappa : 0.8368
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.9579
##             Specificity : 0.8696
##          Pos Pred Value : 0.9381
##          Neg Pred Value : 0.9091
##              Prevalence : 0.6738
##          Detection Rate : 0.6454
##    Detection Prevalence : 0.6879
##       Balanced Accuracy : 0.9137
##
##        'Positive' Class : NO
##
```

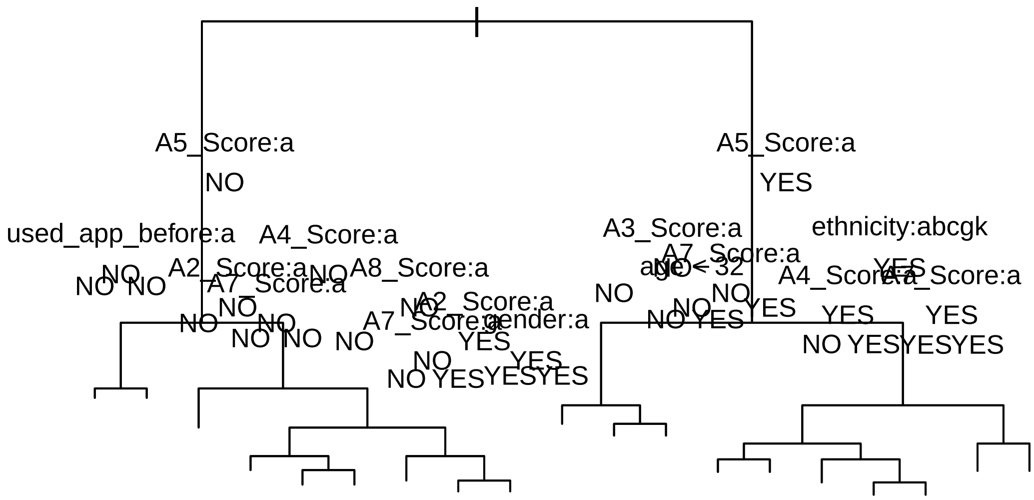
```r
mean(tree.pred == test$`Class/ASD`)
```

```
## [1] 0.929078
```

```r
summary(tree.validation)
```

```
##
## Classification tree:
## tree(formula = `Class/ASD` ~ . - country_of_residence - result,
##     data = train)
## Variables actually used in tree construction:
##  [1] "A9_Score"   "A5_Score"   "ethnicity"  "A10_Score"  "A2_Score"
##  [6] "A4_Score"   "A7_Score"   "A8_Score"   "country_id" "A3_Score"
## [11] "A6_Score"   "A1_Score"
## Number of terminal nodes:  19
## Residual mean deviance:  0.1895 = 103.1 / 544
## Misclassification error rate: 0.04796 = 27 / 563
```

```r
plot(tree.validation,
   main="Classification Tree for ASD")
text(tree.attempt, all=TRUE, cex=.8)
```



Cost Complexity Prunning

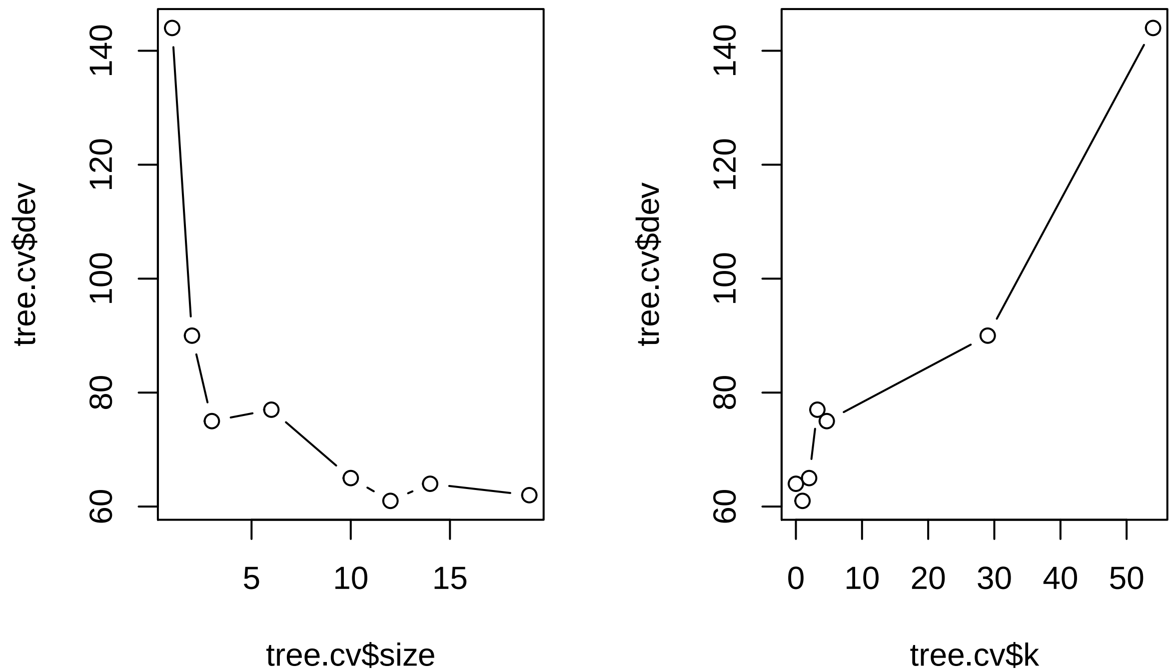
```r
set.seed(2)
tree.cv =cv.tree(tree.validation ,FUN=prune.misclass)
names(tree.cv)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
tree.cv
```

```
## $size
## [1] 19 14 12 10  6  3  2  1
##
## $dev
## [1]  62  64  61  65  77  75  90 144
##
## $k
## [1]      -Inf  0.000000  1.000000  2.000000  3.250000  4.666667 29.000000
## [8] 54.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```r
#Tree with the lowest error rate is the one with 15 terminal nodes
par(mfrow=c(1,2))
plot(tree.cv$size ,tree.cv$dev ,type="b")
plot(tree.cv$k ,tree.cv$dev ,type="b")
```



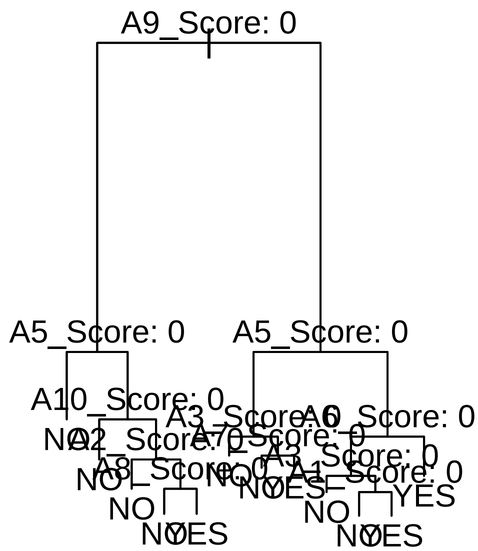

```r
#We observe in the plot that tree with less terminal nodes and lower alpha, tend to perform better

#now that we have the parameters, it's time to prune

tree.prune =prune.misclass(tree.validation ,best=12)
plot(tree.prune)
text(tree.prune ,pretty =0)
summary(tree.prune)
```

```
##
## Classification tree:
## snip.tree(tree = tree.validation, nodes = c(4L, 15L, 22L, 59L,
## 28L, 47L))
## Variables actually used in tree construction:
## [1] "A9_Score"   "A5_Score"   "A10_Score" "A2_Score"   "A8_Score"   "A3_Score"
## [7] "A7_Score"   "A6_Score"   "A1_Score"
## Number of terminal nodes:  12
## Residual mean deviance:  0.3177 = 175.1 / 551
## Misclassification error rate: 0.05151 = 29 / 563
```

```r
tree.prune.pred = predict(tree.prune, test, type = "class")
confusionMatrix(tree.prune.pred, test$`Class/ASD`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO  90   9
##        YES  5  37
##
##                Accuracy : 0.9007
##                  95% CI : (0.839, 0.9446)
##     No Information Rate : 0.6738
##     P-Value [Acc > NIR] : 2.383e-10
##
##                   Kappa : 0.769
##
##  Mcnemar's Test P-Value : 0.4227
##
##             Sensitivity : 0.9474
##             Specificity : 0.8043
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.8810
##              Prevalence : 0.6738
##          Detection Rate : 0.6383
##    Detection Prevalence : 0.7021
##       Balanced Accuracy : 0.8759
##
##        'Positive' Class : NO
##
```

```
#Tree accuracy apparently decreased after pruning, it could be explained
#by the fact that less variables were used
```



Random Forests

Bagging

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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

```
tree.bag = randomForest(`Class/ASD`~.-result-country_of_residence, data=train, mtry=ncol(autism_update)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

```
tree.bag
```

```
##
```

```
## Call:
##  randomForest(formula = `Class/ASD` ~ . - result - country_of_residence,      data = train, mtry = n
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 18
##
##         OOB estimate of  error rate: 5.68%
## Confusion matrix:
##      NO YES class.error
## NO  406  14  0.03333333
## YES  18 125  0.12587413
```

```
yhat.bag = predict(tree.bag, newdata = test, type="Class")
confusionMatrix(yhat.bag, test$`Class/ASD`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO  91   8
##        YES  4  38
##
##                Accuracy : 0.9149
##                  95% CI : (0.8561, 0.9552)
##     No Information Rate : 0.6738
##     P-Value [Acc > NIR] : 1.073e-11
##
##                   Kappa : 0.802
##
##  Mcnemar's Test P-Value : 0.3865
##
##             Sensitivity : 0.9579
##             Specificity : 0.8261
##          Pos Pred Value : 0.9192
##          Neg Pred Value : 0.9048
##              Prevalence : 0.6738
##          Detection Rate : 0.6454
##    Detection Prevalence : 0.7021
##       Balanced Accuracy : 0.8920
##
##        'Positive' Class : NO
##
```

```
#Prediction accuracy does not seem to improve after bagging
```

Normal Random Forests

```
tree.rf = randomForest(`Class/ASD`~.-country_of_residence-result, data=train, mtry=sqrt(20), importance
yhat.rf = predict(tree.rf, newdata=test, type="Class")
summary(tree.rf)
```

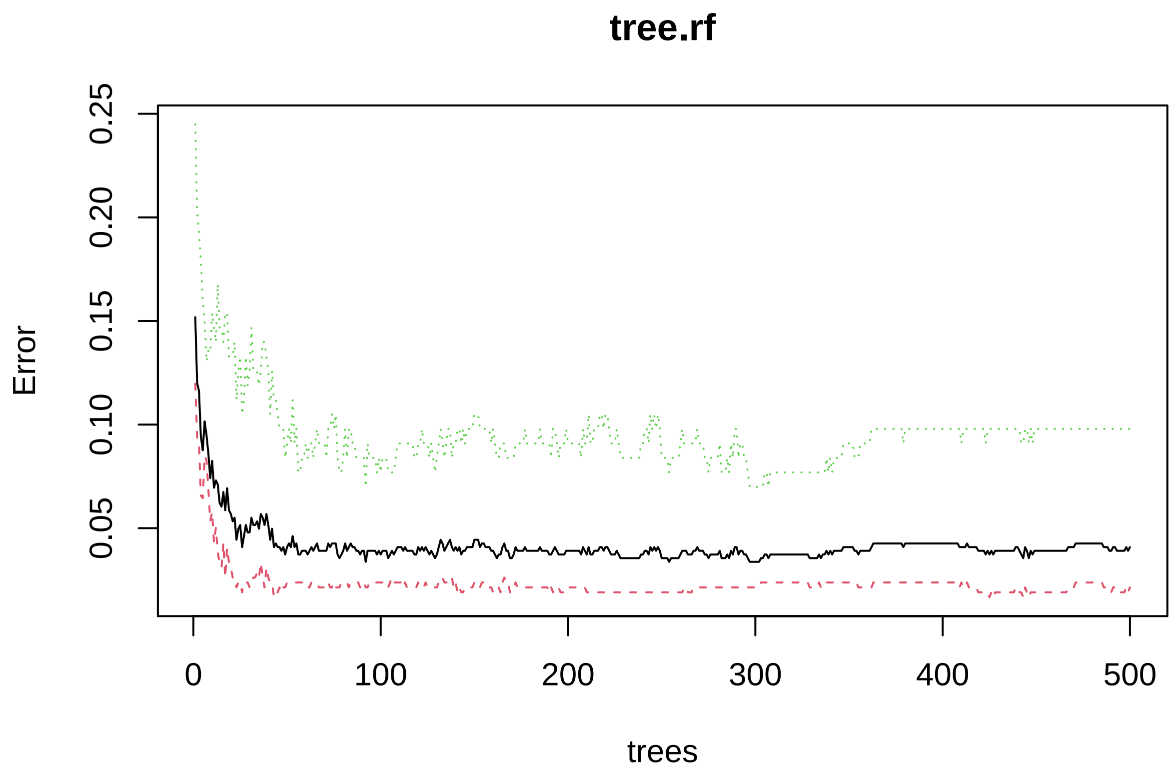
```
##                 Length Class  Mode
## call              5    -none- call
## type              1    -none- character
## predicted       563    factor numeric
```

```
## err.rate        1500   -none- numeric
## confusion          6   -none- numeric
## votes           1126   matrix numeric
## oob.times        563   -none- numeric
## classes            2   -none- character
## importance        72   -none- numeric
## importanceSD      54   -none- numeric
## localImportance    0   -none- NULL
## proximity          0   -none- NULL
## ntree              1   -none- numeric
## mtry               1   -none- numeric
## forest            14   -none- list
## y                563   factor numeric
## test               0   -none- NULL
## inbag              0   -none- NULL
## terms              3   terms  call
```

```
confusionMatrix(yhat.rf, test$`Class/ASD`)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction NO YES
##        NO  93   7
##        YES  2  39
##
##                Accuracy : 0.9362
##                  95% CI : (0.8823, 0.9704)
##     No Information Rate : 0.6738
##     P-Value [Acc > NIR] : 5.231e-14
##
##                   Kappa : 0.8506
##
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9789
##             Specificity : 0.8478
##          Pos Pred Value : 0.9300
##          Neg Pred Value : 0.9512
##              Prevalence : 0.6738
##          Detection Rate : 0.6596
##    Detection Prevalence : 0.7092
##       Balanced Accuracy : 0.9134
##
##        'Positive' Class : NO
##
```

```
plot(tree.rf)
```




```
oob.err = double(ncol(autism_update))
test.err = double(ncol(autism_update))
for(mtry in 1:ncol(autism_update)){
   tree.rf = randomForest(`Class/ASD`~.-country_of_residence-result, data=train, mtry=mtry, importance='
  oob.err[mtry] =  mean(tree.rf$err.rate[,1])
  pred = predict(tree.rf, test, type="Class")
  test.err[mtry] = mean(pred != test$`Class/ASD`)
}
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid
## range
```

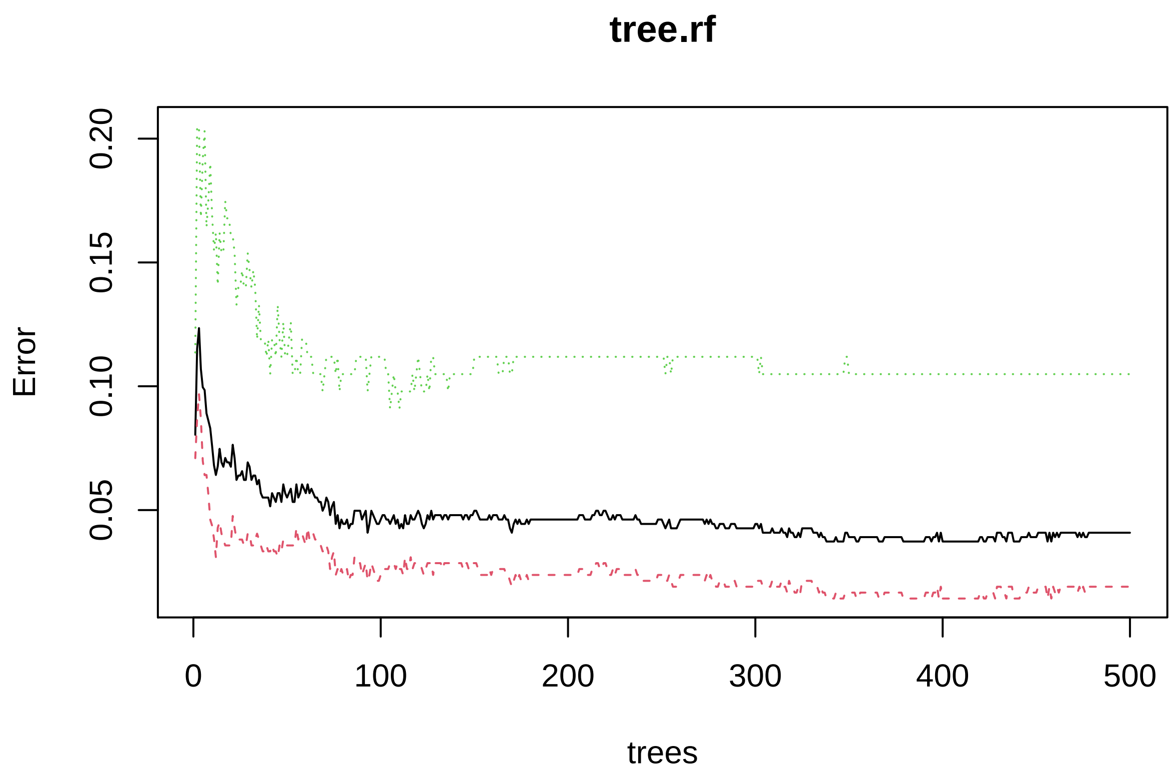
```
which(min(test.err) == test.err)
```

```
## [1] 4 5 6 7
```

```
which(min(oob.err) == oob.err)
```

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

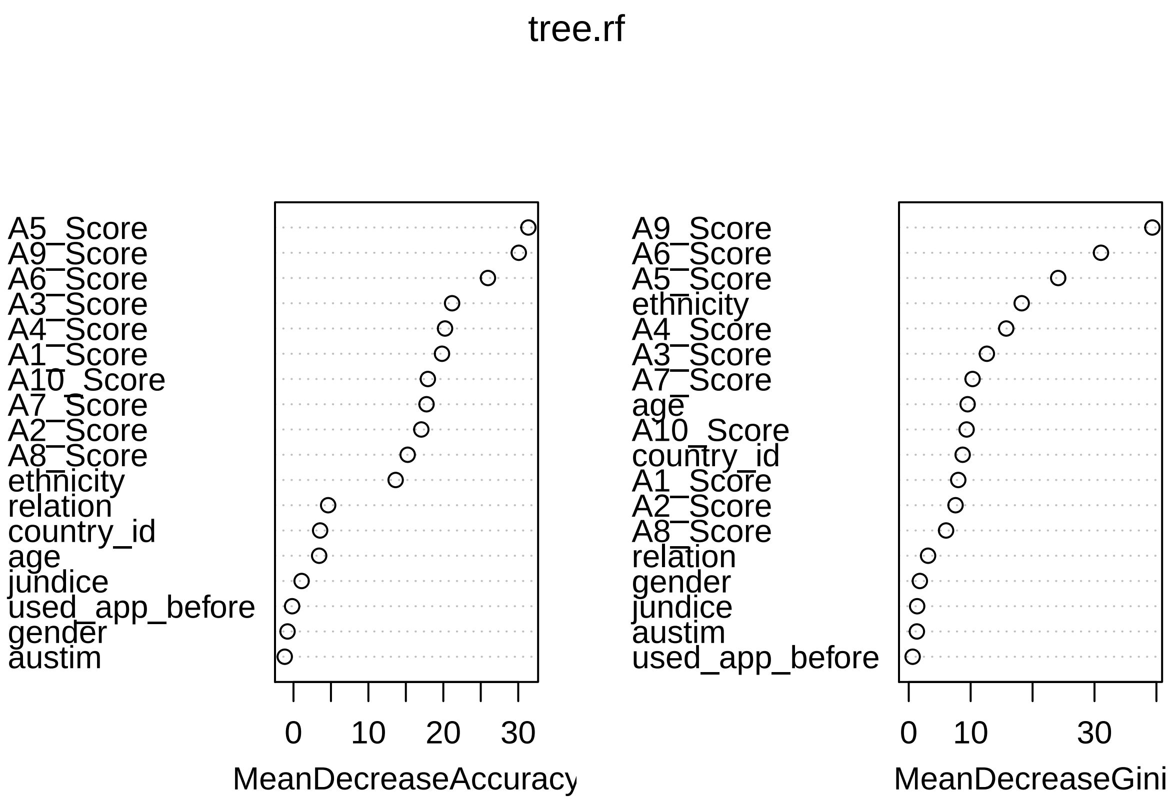
```
#will choose the 4 as the mtry
tree.rf = randomForest(`Class/ASD`~.-country_of_residence-result, data=train, mtry=4, importance=TRUE)
plot(tree.rf)
```




```r
importance(tree.rf)
```

```
##                          NO        YES MeanDecreaseAccuracy MeanDecreaseGini
## A1_Score        10.29885675 18.4335027           19.8236927        7.9959851
## A2_Score        11.84475410 14.3498806           17.0605239        7.5586080
## A3_Score        11.51604643 20.5584608           21.1719692       12.6161125
## A4_Score        10.07324720 18.9726673           20.2289707       15.7460699
## A5_Score        13.76866821 30.2336809           31.3511665       24.1383324
## A6_Score        17.33698804 23.7760556           25.9558504       31.0467675
## A7_Score        12.08857629 15.6902348           17.7596655       10.3110459
## A8_Score         7.66210895 15.3149490           15.2427637        6.0274368
## A9_Score        19.03916762 27.6660359           30.0749820       39.3443037
## A10_Score        4.67558115 19.6621937           17.9356015        9.3475967
## age              3.67064494  0.7856685            3.4278474        9.5090292
## gender          -0.81740265 -0.3328592           -0.7949848        1.7974259
## ethnicity        5.95980258 13.1576104           13.6365658       18.2450849
## jundice          2.77071203 -1.8090255            1.0867384        1.3596412
## austim          -0.07768648 -1.3994212           -1.1675617        1.3135796
## used_app_before  0.19668618 -1.0202377           -0.1811075        0.6297376
## relation         2.80535351  4.1598144            4.6285694        3.1209015
## country_id       2.25894563  2.7744406            3.5551376        8.7079381
```

```r
varImpPlot(tree.rf)
```



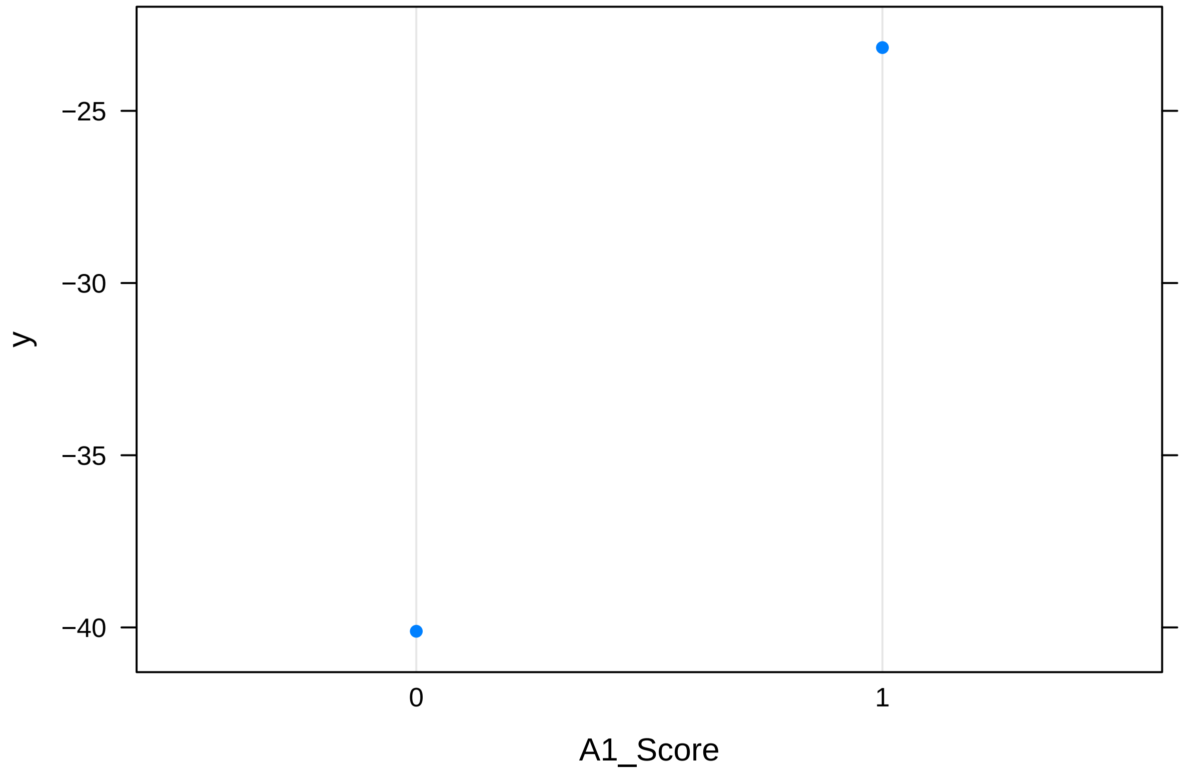
Boosting

```
#make a column for boosting
library("gbm")
```

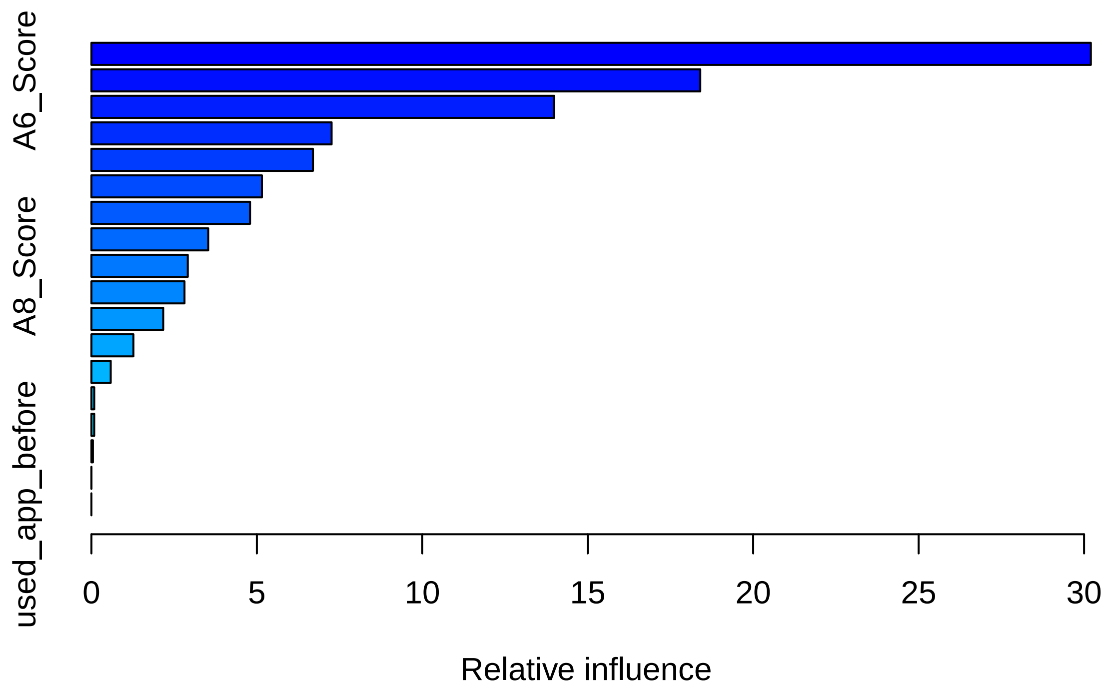
```
## Warning: package 'gbm' was built under R version 4.0.5
```

```
## Loaded gbm 2.1.8
```

```
class_numeric = ifelse(as.character(train$`Class/ASD`) == "YES", 1, 0)
train_boost = cbind(train, class_numeric)
class_numeric = ifelse(as.character(test$`Class/ASD`) == "YES", 1, 0)
test_boost = cbind(test, class_numeric)
boost = gbm(class_numeric~.-country_of_residence-result-`Class/ASD`, data = train_boost, distribution =
plot(boost)
```




```
summary(boost)
```



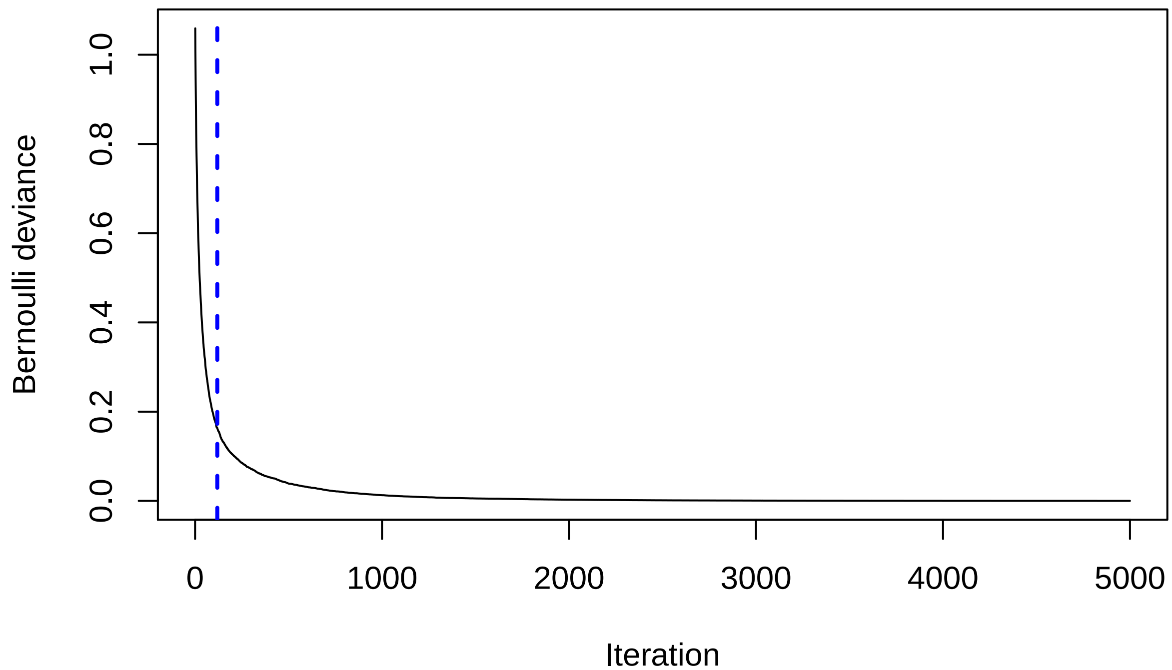

```
##                             var      rel.inf
## A9_Score               A9_Score 3.020570e+01
## A6_Score               A6_Score 1.840155e+01
## A5_Score               A5_Score 1.398851e+01
## A4_Score               A4_Score 7.259362e+00
## A3_Score               A3_Score 6.695213e+00
## ethnicity             ethnicity 5.152673e+00
## A7_Score               A7_Score 4.794799e+00
## A1_Score               A1_Score 3.530535e+00
## A8_Score               A8_Score 2.913157e+00
## A2_Score               A2_Score 2.813868e+00
## A10_Score             A10_Score 2.171086e+00
## country_id           country_id 1.270485e+00
## age                         age 5.839666e-01
## relation               relation 8.646913e-02
## gender                   gender 8.569774e-02
## jundice                 jundice 4.596676e-02
## austim                   austim 9.528958e-04
## used_app_before used_app_before 0.000000e+00
```

```
boost.pred = predict(boost, test_boost, n.trees = 5000)
mean((boost.pred-test_boost$class_numeric)^2)
```

```
## [1] 2336.384
```

```
oob.naive = gbm.perf(boost, method="OOB")
```

```
## OOB generally underestimates the optimal number of iterations although predictive performance is reas
```




```
oob.naive
```

```
## [1] 119
## attr(,"smoother")
## Call:
## loess(formula = object$oobag.improve ~ x, enp.target = min(max(4,
##     length(x)/10), 50))
##
## Number of Observations: 5000
## Equivalent Number of Parameters: 39.99
## Residual Standard Error: 0.000658
```

```
#Number of predictors for this model was similar as other models
#Now we will try tuning
```

Support Vector Machines

Linear

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
#We scale them, because not all variables use the same units
#We are setting the default cost for learning purposes
svm.linear=svm(`Class/ASD`~.-result-country_of_residence, data=train, kernel ="linear", type = "C-class
svm.linear
```

```
##
## Call:
## svm(formula = `Class/ASD` ~ . - result - country_of_residence, data = train,
##     kernel = "linear", type = "C-classification", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  72
```

```
svm.linear$index
```

```
##  [1]   8  22  38  43 114 122 128 146 150 158 198 209 215 219 231 238 252 260 292
## [20] 293 330 331 344 351 356 387 422 431 446 555   6  73  82  96 110 111 144 154
## [39] 159 163 167 175 179 207 223 239 265 279 280 290 299 321 380 392 406 416 433
## [58] 436 438 445 458 467 469 493 500 510 534 539 550 554 561 563
```

```
svm.linear.pred = predict(svm.linear, test)
mean(svm.linear.pred == test$`Class/ASD`)
```

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

```
#The vectors are linear
summary(svm.linear)
```

```
##
## Call:
## svm(formula = `Class/ASD` ~ . - result - country_of_residence, data = train,
##     kernel = "linear", type = "C-classification", scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  72
##
##  ( 30 42 )
##
##
## Number of Classes:  2
##
## Levels:
##  NO YES
```

```
#implementing it with smaller cost

 svmfit=svm(`Class/ASD`~.-result-country_of_residence, data=train , kernel ="linear", cost=0.1, scale=T
 svmfit.pred = predict(svmfit, test)
 mean(svmfit.pred == test$`Class/ASD`) #accuracy decreased
```

```
## [1] 0.9574468
```

```r
summary(svmfit) #the number of support vector increased as we decreased the margin from 1 to 0.1
```

```
##
## Call:
## svm(formula = `Class/ASD` ~ . - result - country_of_residence, data = train,
##     kernel = "linear", cost = 0.1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##
## Number of Support Vectors:  111
##
##  ( 55 56 )
##
##
## Number of Classes:  2
##
## Levels:
##  NO YES
```

```r
#makes sense since we increased the margin
svmfit$index
```

```
##   [1]   1   8  21  22  38  43  49  56  74  89 114 122 128 139 146 150 158 162
##  [19] 168 183 192 196 198 209 215 219 226 231 238 241 252 260 264 292 293 330
##  [37] 331 344 347 351 356 378 379 387 401 422 426 427 431 446 452 460 485 517
##  [55] 555   6  14  27  35  53  57  73  77  82  96 110 111 144 147 154 159 163
##  [73] 167 175 187 207 213 223 239 265 277 279 280 290 299 321 370 380 381 388
##  [91] 392 406 411 413 416 433 435 436 438 445 458 467 469 493 500 510 534 549
## [109] 554 561 563
```

```r
tune.out =tune(svm, `Class/ASD`~.-result-country_of_residence, data=train, kernel="linear", ranges=lis
svmfit.best = tune.out$best.model
summary(svmfit.best)
```

```
##
## Call:
## best.tune(method = svm, train.x = `Class/ASD` ~ . - result - country_of_residence,
##     data = train, ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5,
##         10, 100)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  72
##
##  ( 30 42 )
##
##
```

```
## Number of Classes:  2
##
## Levels:
##  NO YES
```

Radial

```r
svm.radial = svm(`Class/ASD`~.-result-country_of_residence, data=train, kernel="radial", gamma=1)
summary(svm.radial)
```

```
##
## Call:
## svm(formula = `Class/ASD` ~ . - result - country_of_residence, data = train,
##     kernel = "radial", gamma = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  545
##
##  ( 139 406 )
##
##
## Number of Classes:  2
##
## Levels:
##  NO YES
```

```r
tune.out.radial = tune(svm, `Class/ASD`~.-result-country_of_residence, data=train, kernel="radial", rang
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.25401003 0.06140489
## 2 1e-02 0.03724937 0.03077004
## 3 1e-01 0.02474937 0.02663706
## 4 1e+00 0.00000000 0.00000000
## 5 5e+00 0.00000000 0.00000000
## 6 1e+01 0.00000000 0.00000000
## 7 1e+02 0.00000000 0.00000000
```

```r
radial.bestmodel = tune.out.radial$best.model
summary(radial.bestmodel)
```

```
##
## Call:
## best.tune(method = svm, train.x = `Class/ASD` ~ . - result - country_of_residence,
##     data = train, ranges = list(0.1, 1, 10, 100, 1000), kernel = "radial",
##     gamma = c(0.5, 1, 2, 3, 4))
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  382
##
##  ( 104 278 )
##
##
## Number of Classes:  2
##
## Levels:
##  NO YES
```

```r
#Number of support vectors increased significantly after using a radial kernel,
radial.pred = predict(radial.bestmodel, test)
mean(radial.pred == test$`Class/ASD`)
```

```
## [1] 0.8510638
```

```r
#Radial Kernel performs worse than the linear, most likely because a linear boundary is more appropiate
```

Conclusion

Most accurate model among Trees was the bagging model, with an accuracy of .93. The most accurate SVM model was the linear, with an accuracy of 1. Trees already produce their own bootstrap when bagging.