Machine Learning Analysis on Cancer Data

Pesantez, Alejandro

Sun, 3/27/2022

Objective

In this project, we'll be analyzing a set of data using three machine learning methods.

Load libraries:

```
library(tidyverse)
library(ggplot2)
library(rpart)
library(partykit)
library(randomForest)
library(class)
library(caret)
library(ROCR)
library(GGally)
```

The Data

The physicians have identified a data set that consists of over 500 measurements from Fine Needle Aspiration (FNA) of breast tissue masses. In an FNA, a small needle is used to extract a sample of cells from a tissue mass. The cells are then photographed under a microscope. The resulting photographs are entered into graphical imaging software. A trained technician uses a mouse pointer to draw the boundary of the nuclei. The software then calculated each of ten characteristics for the nuclei.

The data consists of measurements of the cell nuclei for the following characteristics:

1. radius,
2. texture,
3. perimeter,
4. area,
5. smoothness (local variation in radius lengths),
6. compactness (perimeterˆ2/ area - 1.0),
7. concavity (severity of concave portions of the contour),
8. concave points (number of concave portions of the counter),
9. symmetry, and
10. fractal dimension ("coastline approximation" -1).

Measurements of these ten characteristics are summarized for all cells in the sample. The data set consists of the mean, standard error of the mean, and maximum of the 10 characteristics, for a total of 30 observations for each. Additionally, the data includes an identification number and a variable that indicates if the tissue mass is malignant (M) or benign (B).

Load data

```r
fna <- read.csv("FNA_cancer.csv")
glimpse(fna)
```

```
## Rows: 569
## Columns: 33
## $ id                      <int> 842302, 842517, 84300903, 84348301, 84358402, ~
## $ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
## $ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450~
## $ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.9~
## $ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, ~
## $ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, ~
## $ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0~
## $ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0~
## $ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0~
## $ concave.points_mean     <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0~
## $ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087~
## $ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0~
## $ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345~
## $ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902~
## $ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.18~
## $ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.~
## $ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.0114~
## $ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.0246~
## $ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0~
## $ concave.points_se       <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.0188~
## $ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0~
## $ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.0051~
## $ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.8~
## $ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.6~
## $ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40,~
## $ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, ~
## $ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791~
## $ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249~
## $ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0~
## $ concave.points_worst    <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0~
## $ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985~
## $ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0~
## $ X                       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
```

The Task

We've been asked by the physicians to conduct an analysis of the data using three of the classification methods we've seen in this class, and provide a video presentation that describes those results.

For our analysis we'll be:

- performing basic exploratory data analysis,
- splitting the data into test and training data,
- build a classification algorithm using decision trees (prune your tree appropriately),
- build a classification algorithm using random forest/ bagging (adjust the parameters of the forest appropriately), and
- build a classification algorithm using Kth Nearest Neighbors (tune the value of K appropriately).

EDA

Convert variable(s) to approprate variable types

```
fna_tidy <- fna %>% mutate( id = as.character(id), diagnosis = as.factor(diagnosis) )
# Checks each variables data type
sapply(fna_tidy, "class")
```

```
##                       id                diagnosis              radius_mean
##              "character"                 "factor"                "numeric"
##             texture_mean           perimeter_mean                area_mean
##                "numeric"                "numeric"                "numeric"
##          smoothness_mean          compactness_mean           concavity_mean
##                "numeric"                "numeric"                "numeric"
##      concave.points_mean            symmetry_mean  fractal_dimension_mean
##                "numeric"                "numeric"                "numeric"
##                radius_se               texture_se             perimeter_se
##                "numeric"                "numeric"                "numeric"
##                  area_se            smoothness_se          compactness_se
##                "numeric"                "numeric"                "numeric"
##             concavity_se        concave.points_se              symmetry_se
##                "numeric"                "numeric"                "numeric"
##     fractal_dimension_se             radius_worst            texture_worst
##                "numeric"                "numeric"                "numeric"
##          perimeter_worst               area_worst          smoothness_worst
##                "numeric"                "numeric"                "numeric"
##        compactness_worst          concavity_worst     concave.points_worst
##                "numeric"                "numeric"                "numeric"
##           symmetry_worst fractal_dimension_worst                        X
##                "numeric"                "numeric"                "logical"
```

Descriptive Statistic for each variable

```
summary(fna_tidy)
```

```
##       id            diagnosis  radius_mean      texture_mean   perimeter_mean
##  Length:569         B:357     Min.   : 6.981   Min.   : 9.71   Min.   : 43.79
##  Class :character   M:212     1st Qu.:11.700   1st Qu.:16.17   1st Qu.: 75.17
##  Mode  :character             Median :13.370   Median :18.84   Median : 86.24
##                               Mean   :14.127   Mean   :19.29   Mean   : 91.97
##                               3rd Qu.:15.780   3rd Qu.:21.80   3rd Qu.:104.10
##                               Max.   :28.110   Max.   :39.28   Max.   :188.50
##    area_mean      smoothness_mean   compactness_mean  concavity_mean
##  Min.   : 143.5   Min.   :0.05263   Min.   :0.01938   Min.   :0.00000
##  1st Qu.: 420.3   1st Qu.:0.08637   1st Qu.:0.06492   1st Qu.:0.02956
##  Median : 551.1   Median :0.09587   Median :0.09263   Median :0.06154
##  Mean   : 654.9   Mean   :0.09636   Mean   :0.10434   Mean   :0.08880
##  3rd Qu.: 782.7   3rd Qu.:0.10530   3rd Qu.:0.13040   3rd Qu.:0.13070
##  Max.   :2501.0   Max.   :0.16340   Max.   :0.34540   Max.   :0.42680
##  concave.points_mean symmetry_mean    fractal_dimension_mean   radius_se
##  Min.   :0.00000     Min.   :0.1060   Min.   :0.04996        Min.   :0.1115
##  1st Qu.:0.02031     1st Qu.:0.1619   1st Qu.:0.05770        1st Qu.:0.2324
##  Median :0.03350     Median :0.1792   Median :0.06154        Median :0.3242
##  Mean   :0.04892     Mean   :0.1812   Mean   :0.06280        Mean   :0.4052
##  3rd Qu.:0.07400     3rd Qu.:0.1957   3rd Qu.:0.06612        3rd Qu.:0.4789
##  Max.   :0.20120     Max.   :0.3040   Max.   :0.09744        Max.   :2.8730
```

```
##    texture_se      perimeter_se       area_se        smoothness_se
##  Min.   :0.3602   Min.   : 0.757   Min.   :  6.802   Min.   :0.001713
##  1st Qu.:0.8339   1st Qu.: 1.606   1st Qu.: 17.850   1st Qu.:0.005169
##  Median :1.1080   Median : 2.287   Median : 24.530   Median :0.006380
##  Mean   :1.2169   Mean   : 2.866   Mean   : 40.337   Mean   :0.007041
##  3rd Qu.:1.4740   3rd Qu.: 3.357   3rd Qu.: 45.190   3rd Qu.:0.008146
##  Max.   :4.8850   Max.   :21.980   Max.   :542.200   Max.   :0.031130
##  compactness_se      concavity_se     concave.points_se   symmetry_se
##  Min.   :0.002252   Min.   :0.00000   Min.   :0.000000   Min.   :0.007882
##  1st Qu.:0.013080   1st Qu.:0.01509   1st Qu.:0.007638   1st Qu.:0.015160
##  Median :0.020450   Median :0.02589   Median :0.010930   Median :0.018730
##  Mean   :0.025478   Mean   :0.03189   Mean   :0.011796   Mean   :0.020542
##  3rd Qu.:0.032450   3rd Qu.:0.04205   3rd Qu.:0.014710   3rd Qu.:0.023480
##  Max.   :0.135400   Max.   :0.39600   Max.   :0.052790   Max.   :0.078950
##  fractal_dimension_se  radius_worst   texture_worst   perimeter_worst
##  Min.   :0.0008948    Min.   : 7.93   Min.   :12.02   Min.   : 50.41
##  1st Qu.:0.0022480    1st Qu.:13.01   1st Qu.:21.08   1st Qu.: 84.11
##  Median :0.0031870    Median :14.97   Median :25.41   Median : 97.66
##  Mean   :0.0037949    Mean   :16.27   Mean   :25.68   Mean   :107.26
##  3rd Qu.:0.0045580    3rd Qu.:18.79   3rd Qu.:29.72   3rd Qu.:125.40
##  Max.   :0.0298400    Max.   :36.04   Max.   :49.54   Max.   :251.20
##    area_worst     smoothness_worst  compactness_worst concavity_worst
##  Min.   : 185.2   Min.   :0.07117   Min.   :0.02729   Min.   :0.0000
##  1st Qu.: 515.3   1st Qu.:0.11660   1st Qu.:0.14720   1st Qu.:0.1145
##  Median : 686.5   Median :0.13130   Median :0.21190   Median :0.2267
##  Mean   : 880.6   Mean   :0.13237   Mean   :0.25427   Mean   :0.2722
##  3rd Qu.:1084.0   3rd Qu.:0.14600   3rd Qu.:0.33910   3rd Qu.:0.3829
##  Max.   :4254.0   Max.   :0.22260   Max.   :1.05800   Max.   :1.2520
##  concave.points_worst symmetry_worst   fractal_dimension_worst    X
##  Min.   :0.00000      Min.   :0.1565   Min.   :0.05504         Mode:logical
##  1st Qu.:0.06493      1st Qu.:0.2504   1st Qu.:0.07146         NA's:569
##  Median :0.09993      Median :0.2822   Median :0.08004
##  Mean   :0.11461      Mean   :0.2901   Mean   :0.08395
##  3rd Qu.:0.16140      3rd Qu.:0.3179   3rd Qu.:0.09208
##  Max.   :0.29100      Max.   :0.6638   Max.   :0.20750
```

Counts total number of NAs in each variable in the data set

```
na_count <-sapply(fna_tidy, function(y) sum(length(which(is.na(y)))))
data.frame(na_count)
```

```
##                          na_count
## id                              0
## diagnosis                       0
## radius_mean                     0
## texture_mean                    0
## perimeter_mean                  0
## area_mean                       0
## smoothness_mean                 0
## compactness_mean                0
## concavity_mean                  0
## concave.points_mean             0
## symmetry_mean                   0
## fractal_dimension_mean          0
## radius_se                       0
```

```
## texture_se                    0
## perimeter_se                  0
## area_se                       0
## smoothness_se                 0
## compactness_se                0
## concavity_se                  0
## concave.points_se             0
## symmetry_se                   0
## fractal_dimension_se          0
## radius_worst                  0
## texture_worst                 0
## perimeter_worst               0
## area_worst                    0
## smoothness_worst              0
## compactness_worst             0
## concavity_worst               0
## concave.points_worst          0
## symmetry_worst                0
## fractal_dimension_worst       0
## X                           569
```

Add binary variable for response

```
fna_tidy$diagnosis_binary <- as.factor(ifelse(fna_tidy$diagnosis == "M", 1, 0))
```

Select desirable variables

```
fna_tidy <- fna_tidy %>% dplyr::select(id, diagnosis, diagnosis_binary, radius_mean,
                                       texture_mean, perimeter_mean, area_mean,
                                       smoothness_mean, compactness_mean, concavity_mean,
                                       concave.points_mean, symmetry_mean,
                                       fractal_dimension_mean)

dim(fna_tidy)
```

```
## [1] 569  13
```

Correlation between variables

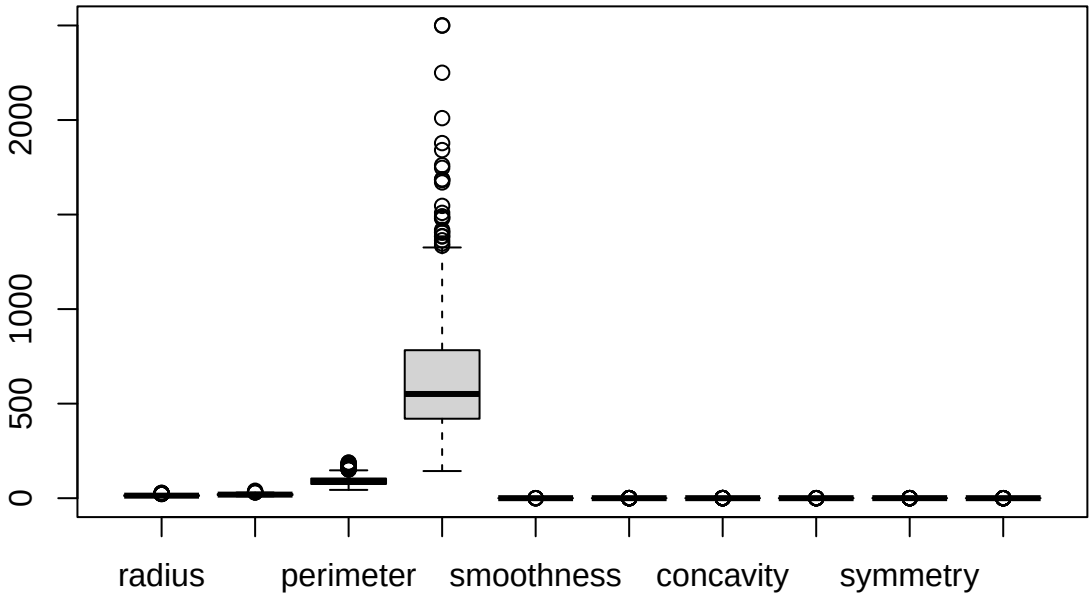
```
round(cor(fna_tidy[4:13]), 4)
```

```
##                         radius_mean texture_mean perimeter_mean area_mean
## radius_mean                  1.0000       0.3238         0.9979    0.9874
## texture_mean                 0.3238       1.0000         0.3295    0.3211
## perimeter_mean               0.9979       0.3295         1.0000    0.9865
## area_mean                    0.9874       0.3211         0.9865    1.0000
## smoothness_mean              0.1706      -0.0234         0.2073    0.1770
## compactness_mean             0.5061       0.2367         0.5569    0.4985
## concavity_mean               0.6768       0.3024         0.7161    0.6860
## concave.points_mean          0.8225       0.2935         0.8510    0.8233
## symmetry_mean                0.1477       0.0714         0.1830    0.1513
## fractal_dimension_mean      -0.3116      -0.0764        -0.2615   -0.2831
##                         smoothness_mean compactness_mean concavity_mean
## radius_mean                      0.1706           0.5061         0.6768
## texture_mean                    -0.0234           0.2367         0.3024
## perimeter_mean                   0.2073           0.5569         0.7161
## area_mean                        0.1770           0.4985         0.6860
```

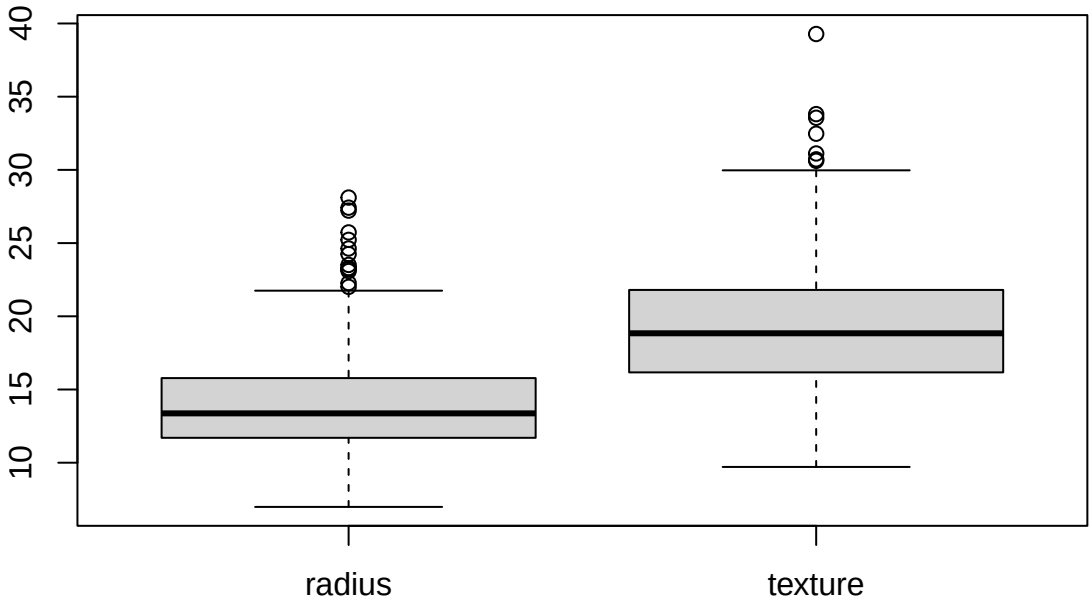
```
## smoothness_mean                 1.0000           0.6591          0.5220
## compactness_mean                0.6591           1.0000          0.8831
## concavity_mean                  0.5220           0.8831          1.0000
## concave.points_mean             0.5537           0.8311          0.9214
## symmetry_mean                   0.5578           0.6026          0.5007
## fractal_dimension_mean          0.5848           0.5654          0.3368
##                       concave.points_mean symmetry_mean fractal_dimension_mean
## radius_mean                        0.8225        0.1477                -0.3116
## texture_mean                       0.2935        0.0714                -0.0764
## perimeter_mean                     0.8510        0.1830                -0.2615
## area_mean                          0.8233        0.1513                -0.2831
## smoothness_mean                    0.5537        0.5578                 0.5848
## compactness_mean                   0.8311        0.6026                 0.5654
## concavity_mean                     0.9214        0.5007                 0.3368
## concave.points_mean                1.0000        0.4625                 0.1669
## symmetry_mean                      0.4625        1.0000                 0.4799
## fractal_dimension_mean             0.1669        0.4799                 1.0000
```

Checking for outliers

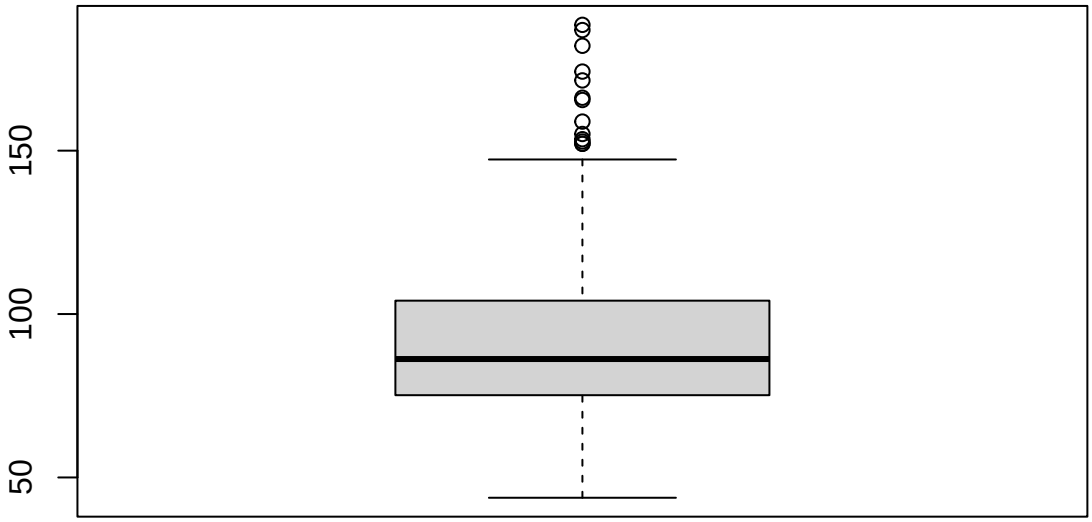
```
# creating boxplots to check for outliers on all variables
boxplot(fna_tidy[4:13], names = c("radius","texture","perimeter","area",
                                  "smoothness", "compactness", "concavity",
                                  "concave points", "symmetry",
                                  "Fractal dimension"))
```



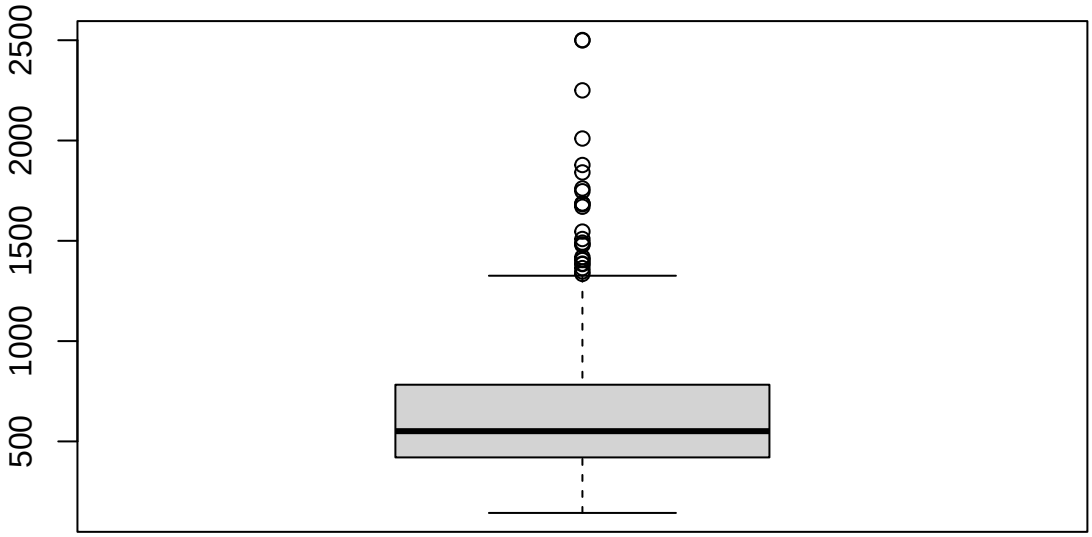

```
# examines the outliers on all variables by an individual basis
boxplot(fna_tidy[4:5], names = c("radius","texture"))
```



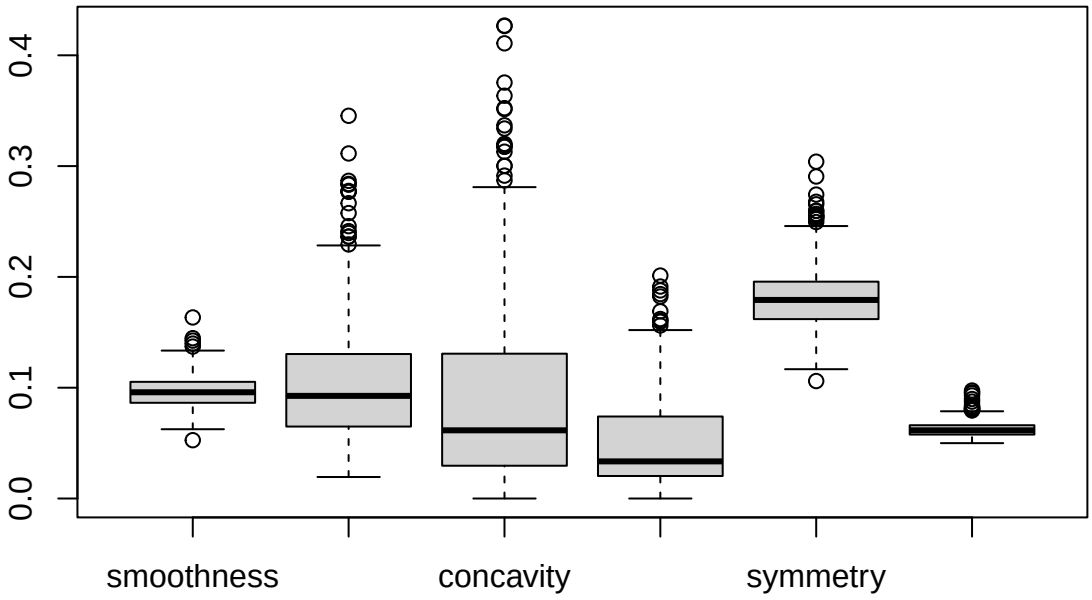

```
boxplot(fna_tidy[6], names = c("perimeter"))
```




```
boxplot(fna_tidy[7], names = c("area"))
```

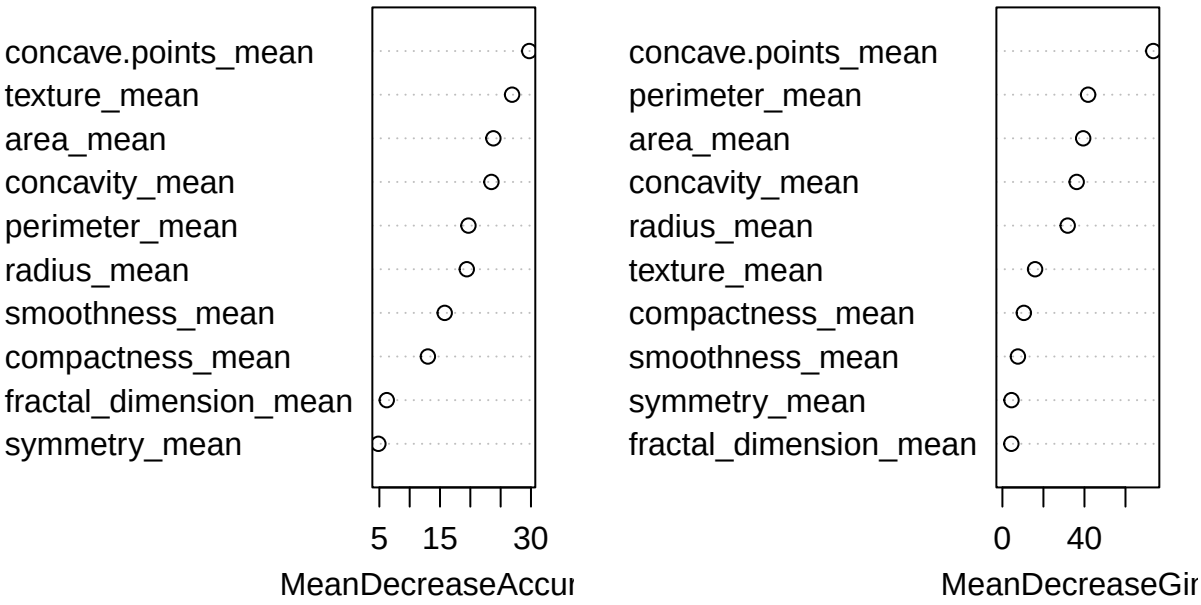



```
boxplot(fna_tidy[8:13], names = c("smoothness", "compactness", "concavity",
                                  "concave points", "symmetry",
                                  "Fractal dimension"))
```



Variable importance plot

```
regressor <- randomForest(as.factor(diagnosis_binary) ~ . , fna_tidy[3:13], importance=TRUE)
varImpPlot(regressor)
```



Re-scaling appropriate variables between 0 and 1 scale

```
# Function that re-scales the data
rescale_x <- function(x){(x-min(x))/(max(x)-min(x))}

# Creates new data.frame containing re-scaled variables
fna_tidy_rescale <- fna_tidy

# Re-scales the necessarily variables from the data set
fna_tidy_rescale$radius_mean <- rescale_x(fna_tidy_rescale$radius_mean)
fna_tidy_rescale$texture_mean <- rescale_x(fna_tidy_rescale$texture_mean)
fna_tidy_rescale$perimeter_mean <- rescale_x(fna_tidy_rescale$perimeter_mean)
fna_tidy_rescale$area_mean <- rescale_x(fna_tidy_rescale$area_mean)
fna_tidy_rescale$smoothness_mean <- rescale_x(fna_tidy_rescale$smoothness_mean)
fna_tidy_rescale$compactness_mean <- rescale_x(fna_tidy_rescale$compactness_mean)
fna_tidy_rescale$concavity_mean <- rescale_x(fna_tidy_rescale$concavity_mean)
fna_tidy_rescale$concave.points_mean <- rescale_x(fna_tidy_rescale$concave.points_mean)
fna_tidy_rescale$symmetry_mean <- rescale_x(fna_tidy_rescale$symmetry_mean)
fna_tidy_rescale$fractal_dimension_mean <- rescale_x(fna_tidy_rescale$fractal_dimension_mean)

glimpse(fna_tidy_rescale)
```

```
## Rows: 569
## Columns: 13
## $ id                    <chr> "842302", "842517", "84300903", "84348301", "84~
## $ diagnosis             <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, M,~
## $ diagnosis_binary      <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ radius_mean           <dbl> 0.5210374, 0.6431445, 0.6014956, 0.2100904, 0.6~
## $ texture_mean          <dbl> 0.0226581, 0.2725736, 0.3902604, 0.3608387, 0.1~
## $ perimeter_mean        <dbl> 0.5459885, 0.6157833, 0.5957432, 0.2335015, 0.6~
```

```
## $ area_mean              <dbl> 0.36373277, 0.50159067, 0.44941676, 0.10290562,~
## $ smoothness_mean        <dbl> 0.5937528, 0.2898799, 0.5143089, 0.8113208, 0.4~
## $ compactness_mean       <dbl> 0.7920373, 0.1817680, 0.4310165, 0.8113613, 0.3~
## $ concavity_mean         <dbl> 0.70313964, 0.20360825, 0.46251172, 0.56560450,~
## $ concave.points_mean    <dbl> 0.7311133, 0.3487575, 0.6356859, 0.5228628, 0.5~
## $ symmetry_mean          <dbl> 0.6863636, 0.3797980, 0.5095960, 0.7762626, 0.3~
## $ fractal_dimension_mean <dbl> 0.60551811, 0.14132266, 0.21124684, 1.00000000,~
```

Train and Test data

Split the data into train and test

```
set.seed(1997)
n <- nrow(fna)
test_indx <- sample.int(n, round(n*0.2))
train_data <- fna_tidy_rescale[-test_indx,]
test_data <- fna_tidy_rescale[test_indx,]

glimpse(train_data)
```

```
## Rows: 455
## Columns: 13
## $ id                     <chr> "842517", "84348301", "843786", "844359", "8445~
## $ diagnosis              <fct> M, M, M, M, M, M, M, M, M, M, M, M, M, M, M, B,~
## $ diagnosis_binary       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,~
## $ radius_mean            <dbl> 0.6431445, 0.2100904, 0.2588386, 0.5333428, 0.3~
## $ texture_mean           <dbl> 0.2725736, 0.3608387, 0.2025702, 0.3473115, 0.3~
## $ perimeter_mean         <dbl> 0.6157833, 0.2335015, 0.2679842, 0.5238753, 0.3~
## $ area_mean              <dbl> 0.50159067, 0.10290562, 0.14150583, 0.38027572,~
## $ smoothness_mean        <dbl> 0.2898799, 0.8113208, 0.6786133, 0.3791640, 0.5~
## $ compactness_mean       <dbl> 0.1817680, 0.8113613, 0.4619962, 0.2748911, 0.4~
## $ concavity_mean         <dbl> 0.20360825, 0.56560450, 0.36972821, 0.26405811,~
## $ concave.points_mean    <dbl> 0.3487575, 0.5228628, 0.4020378, 0.3677932, 0.2~
## $ symmetry_mean          <dbl> 0.3797980, 0.7762626, 0.5186869, 0.3707071, 0.5~
## $ fractal_dimension_mean <dbl> 0.14132266, 1.00000000, 0.55117944, 0.15711879,~
```

```
glimpse(test_data)
```

```
## Rows: 114
## Columns: 13
## $ id                     <chr> "871641", "90769601", "893988", "917897", "8846~
## $ diagnosis              <fct> B, B, B, B, B, B, M, B, B, B, B, B, B, M, B, M,~
## $ diagnosis_binary       <fct> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,~
## $ radius_mean            <dbl> 0.1939988, 0.1963652, 0.2157698, 0.1356430, 0.1~
## $ texture_mean           <dbl> 0.16909029, 0.23368279, 0.03415624, 0.20189381,~
## $ perimeter_mean         <dbl> 0.1825720, 0.1843687, 0.2068966, 0.1327483, 0.1~
## $ area_mean              <dbl> 0.09722163, 0.10078473, 0.11266172, 0.06349947,~
## $ smoothness_mean        <dbl> 0.4330595, 0.2607204, 0.3009840, 0.3817821, 0.3~
## $ compactness_mean       <dbl> 0.11671063, 0.05815594, 0.12364272, 0.19879149,~
## $ concavity_mean         <dbl> 0.055365511, 0.032075914, 0.032029053, 0.054592~
## $ concave.points_mean    <dbl> 0.12837972, 0.06809145, 0.04426938, 0.12007952,~
## $ symmetry_mean          <dbl> 0.2555556, 0.2277778, 0.3904040, 0.1651515, 0.4~
## $ fractal_dimension_mean <dbl> 0.35235889, 0.24262848, 0.23251896, 0.39911542,~
```

Classification Algorithm using Decision Trees method

Defining diagnosis formula

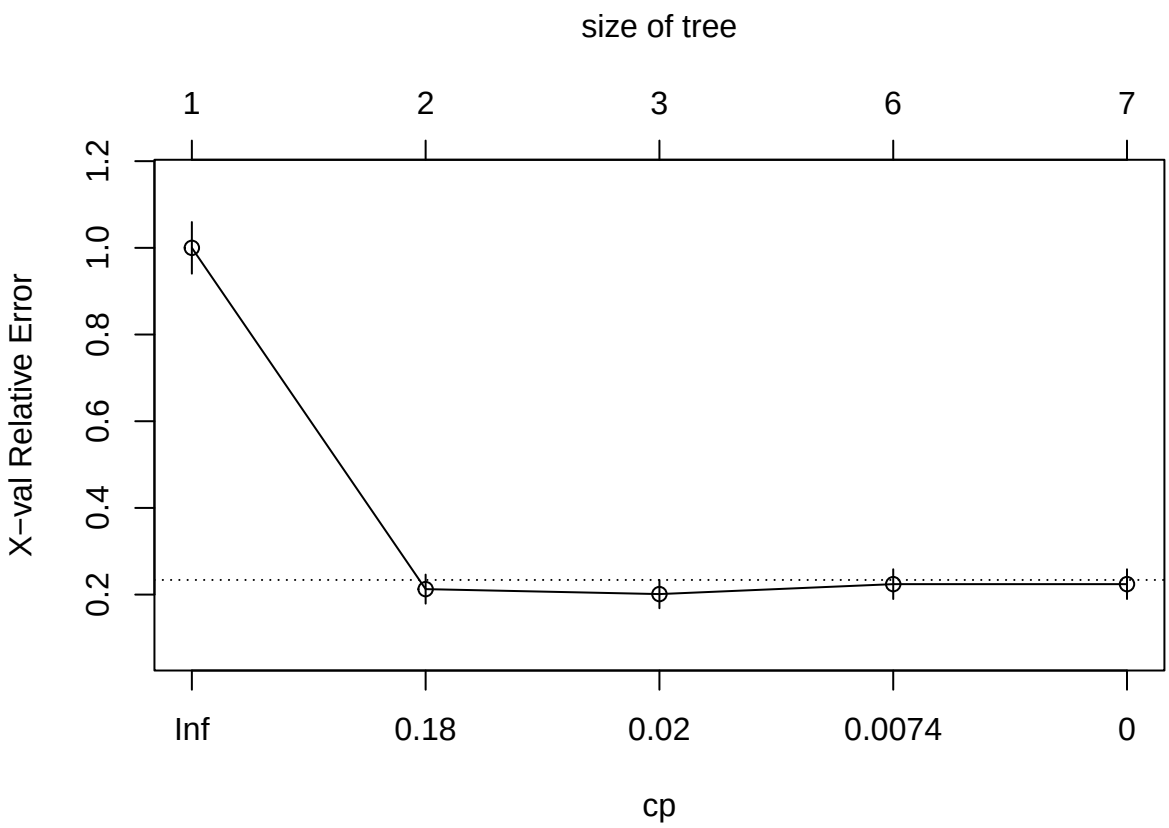
```
# Creates formula for diagnosis
diagnosis_form <- as.formula(diagnosis ~ radius_mean + texture_mean + perimeter_mean
                             + area_mean + smoothness_mean + compactness_mean +
                               concavity_mean + concave.points_mean + symmetry_mean
                             + fractal_dimension_mean)
```

Decision Tree Method (when cp=0)

```
set.seed(1997)

# Creates decision tree, where pruning is used to avoid overfitting
diagnosis_tree_full <- rpart(diagnosis_form, train_data, cp = 0)

# Creates cp of model
printcp(diagnosis_tree_full)
```

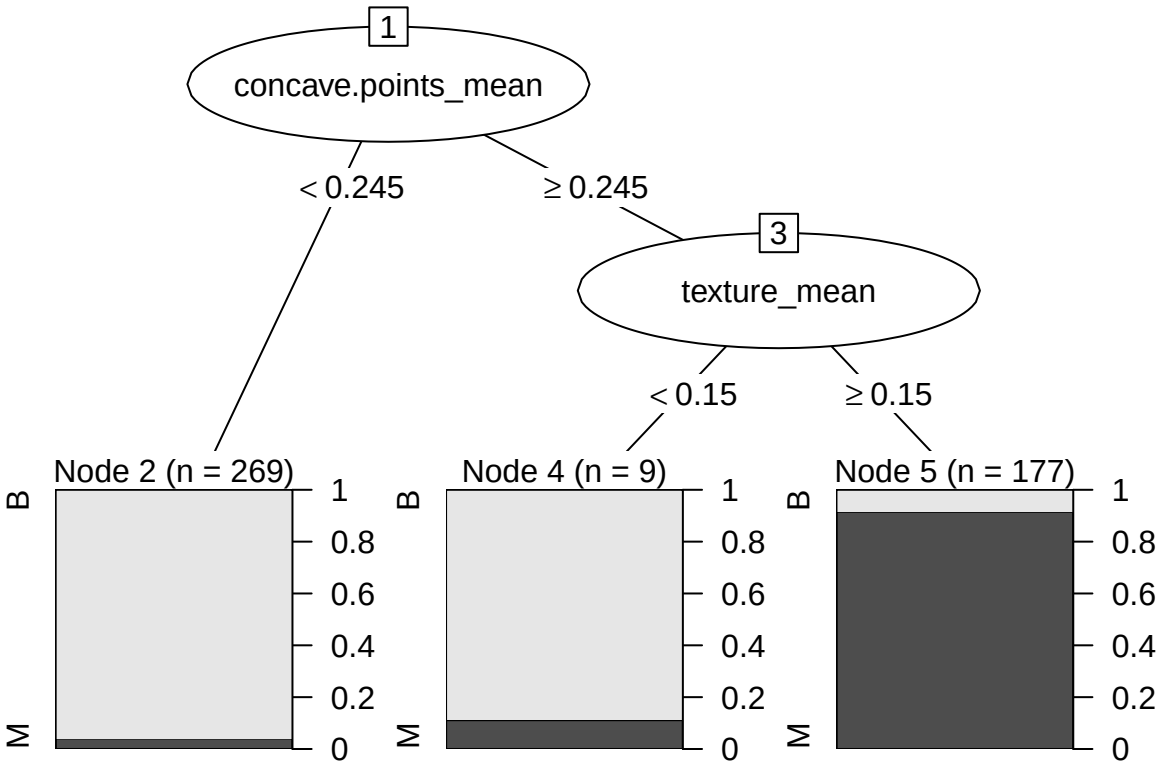
```
##
## Classification tree:
## rpart(formula = diagnosis_form, data = train_data, cp = 0)
##
## Variables actually used in tree construction:
## [1] area_mean           concave.points_mean perimeter_mean
## [4] texture_mean
##
## Root node error: 174/455 = 0.38242
##
## n= 455
##
##          CP nsplit rel error  xerror     xstd
## 1 0.8045977      0   1.00000 1.00000 0.059576
## 2 0.0402299      1   0.19540 0.21264 0.033507
## 3 0.0095785      2   0.15517 0.20115 0.032667
## 4 0.0057471      5   0.12644 0.22414 0.034318
## 5 0.0000000      6   0.12069 0.22414 0.034318
```

```
plotcp(diagnosis_tree_full)
```



Decision Tree Method after pruning

```
set.seed(1997)
# Creates decision tree
diagnosis_tree <- rpart(diagnosis_form, train_data, cp = .02)
plot(as.party(diagnosis_tree))
```




```
# Creates predictions for diagnosis using the decision tree with the removal of ID, diagnosis, and diag
diagnosis_tree_preds <- predict(diagnosis_tree, newdata = test_data[c(-1,-2,-3)], type = "class")
# Creates Confusion Matrix
confusionMatrix(diagnosis_tree_preds, test_data$diagnosis, positive ="M")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 71  8
##          M  5 30
##
##                Accuracy : 0.886
##                  95% CI : (0.8129, 0.9379)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 5.927e-08
##
##                   Kappa : 0.7383
##
##  Mcnemar's Test P-Value : 0.5791
##
##             Sensitivity : 0.7895
##             Specificity : 0.9342
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.8987
##              Prevalence : 0.3333
##          Detection Rate : 0.2632
##    Detection Prevalence : 0.3070
##       Balanced Accuracy : 0.8618
##
```

```
##        'Positive' Class : M
##
```

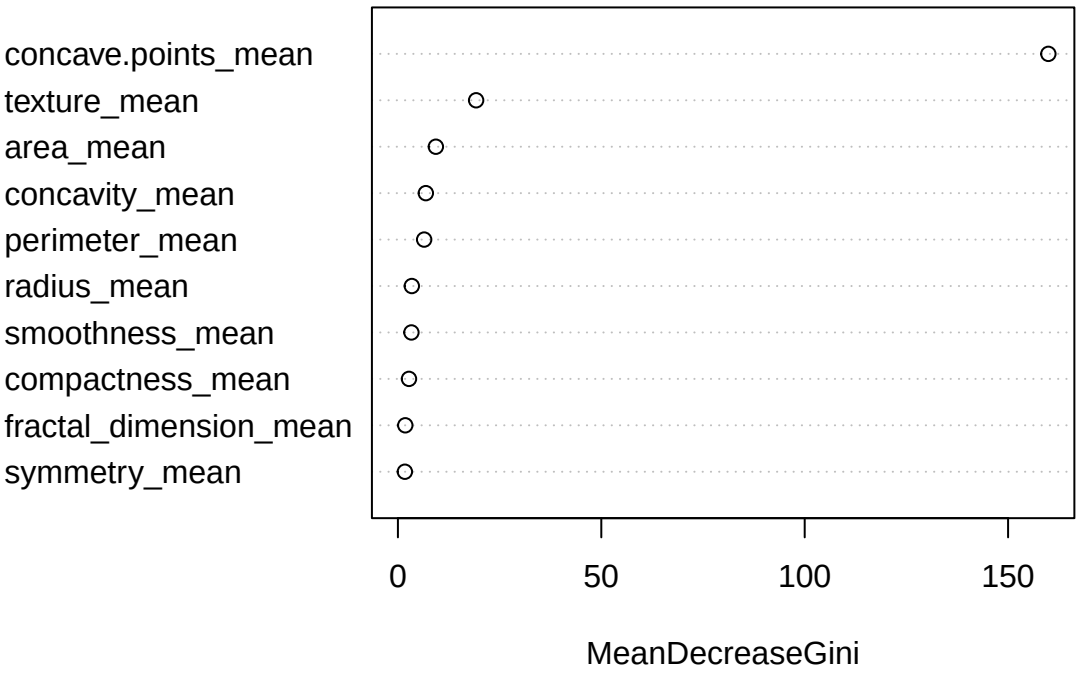
Classification Algorithm using random forest/ bagging method

Bagging Method Analysis:

```
set.seed(1997)

# Creates bagging
diagnosis_bag <- randomForest(diagnosis_form, data = train_data, mtry = 10, ntree = 201)

# Creates predictions for diagnosis
diagnosis_bag_preds <- predict(diagnosis_bag, newdata = test_data[c(-1,-2,-3)])

# Creates Confusion Matrix
confusionMatrix(diagnosis_bag_preds, test_data$diagnosis, positive = 'M')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 73  4
##          M  3 34
##
##                Accuracy : 0.9386
##                  95% CI : (0.8776, 0.975)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 3.102e-12
##
##                   Kappa : 0.8609
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8947
##             Specificity : 0.9605
##          Pos Pred Value : 0.9189
##          Neg Pred Value : 0.9481
##              Prevalence : 0.3333
##          Detection Rate : 0.2982
##    Detection Prevalence : 0.3246
##       Balanced Accuracy : 0.9276
##
##        'Positive' Class : M
##
```

```
#looking at variable importance
varImpPlot(diagnosis_bag)
```



Random Forest Method Anaylsis:

```
set.seed(1997)

# Creates various random forest models
randomForest(diagnosis_form, data = train_data, mtry = 1, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 1,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 6.37%
## Confusion matrix:
##     B   M class.error
## B 269  12  0.04270463
## M  17 157  0.09770115
```

```
randomForest(diagnosis_form, data = train_data, mtry = 2, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 2,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 6.81%
## Confusion matrix:
```

```
##     B   M class.error
## B 267  14  0.04982206
## M  17 157  0.09770115
```

```
randomForest(diagnosis_form, data = train_data, mtry = 3, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 3,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 6.59%
## Confusion matrix:
##     B   M class.error
## B 267  14  0.04982206
## M  16 158  0.09195402
```

```
randomForest(diagnosis_form, data = train_data, mtry = 4, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 4,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 6.37%
## Confusion matrix:
##     B   M class.error
## B 267  14  0.04982206
## M  15 159  0.08620690
```

```
randomForest(diagnosis_form, data = train_data, mtry = 5, ntree = 201) #best model
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 5,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 5.49%
## Confusion matrix:
##     B   M class.error
## B 269  12  0.04270463
## M  13 161  0.07471264
```

```
randomForest(diagnosis_form, data = train_data, mtry = 6, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 6,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 6
```

```
##
##         OOB estimate of  error rate: 5.93%
## Confusion matrix:
##     B   M class.error
## B 267  14  0.04982206
## M  13 161  0.07471264
```

```
randomForest(diagnosis_form, data = train_data, mtry = 7, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 7,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 6.59%
## Confusion matrix:
##     B   M class.error
## B 266  15  0.05338078
## M  15 159  0.08620690
```

```
randomForest(diagnosis_form, data = train_data, mtry = 8, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 8,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 8
##
##         OOB estimate of  error rate: 6.15%
## Confusion matrix:
##     B   M class.error
## B 267  14  0.04982206
## M  14 160  0.08045977
```

```
randomForest(diagnosis_form, data = train_data, mtry = 9, ntree = 201)
```

```
##
## Call:
##  randomForest(formula = diagnosis_form, data = train_data, mtry = 9,      ntree = 201)
##                Type of random forest: classification
##                      Number of trees: 201
## No. of variables tried at each split: 9
##
##         OOB estimate of  error rate: 6.37%
## Confusion matrix:
##     B   M class.error
## B 267  14  0.04982206
## M  15 159  0.08620690
```

Best model for Random Forest Method

```
set.seed(1997)

# Choose the best random forest model, # note: m = 5
```

```r
diagnosis_forest <- randomForest(diagnosis_form, data = train_data, mtry = 5, ntree = 201)
# Creates predictions for diagnosis using random forest
diagnosis_forest_preds <- predict(diagnosis_forest, newdata = test_data[c(-1,-2,-3)])
# Creates Confusion Matrix
confusionMatrix(diagnosis_forest_preds, test_data$diagnosis, positive = 'M')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 72  4
##          M  4 34
##
##                Accuracy : 0.9298
##                  95% CI : (0.8664, 0.9692)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 2.121e-11
##
##                   Kappa : 0.8421
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8947
##             Specificity : 0.9474
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.9474
##              Prevalence : 0.3333
##          Detection Rate : 0.2982
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.9211
##
##        'Positive' Class : M
##
```

Classification Algorithm using Kth Nearest Neighbors (KNN) method

```r
set.seed(1997)
diagnosis_knn3 <- as.factor(knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnos
table(diagnosis_knn3, test_data$diagnosis)
```

```
##
## diagnosis_knn3  B  M
##              B 72  4
##              M  4 34
```

```r
set.seed(1997)
diagnosis_knn5 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 5)
table(diagnosis_knn5, test_data$diagnosis)
```

```
##
## diagnosis_knn5  B  M
##              B 74  4
##              M  2 34
```

```r
set.seed(1997)
diagnosis_knn7 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 7)
table(diagnosis_knn7, test_data$diagnosis)
```

```
##
## diagnosis_knn7  B  M
##              B 74  4
##              M  2 34
```

```r
set.seed(1997)
diagnosis_knn9 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 9)
table(diagnosis_knn9, test_data$diagnosis)
```

```
##
## diagnosis_knn9  B  M
##              B 74  4
##              M  2 34
```

```r
set.seed(1997)
diagnosis_knn11 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 
table(diagnosis_knn11, test_data$diagnosis)
```

```
##
## diagnosis_knn11  B  M
##               B 75  4
##               M  1 34
```

```r
set.seed(1997)
diagnosis_knn13 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 
table(diagnosis_knn13, test_data$diagnosis)
```

```
##
## diagnosis_knn13  B  M
##               B 75  4
##               M  1 34
```

```r
set.seed(1997)
diagnosis_knn17 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 
table(diagnosis_knn17, test_data$diagnosis)
```

```
##
## diagnosis_knn17  B  M
##               B 75  5
##               M  1 33
```

```r
#square root of n model
set.seed(1997)
diagnosis_knn24 <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis, k = 
table(diagnosis_knn24, test_data$diagnosis)
```

```
##
## diagnosis_knn24  B  M
##               B 76  6
##               M  0 32
```

As a result, the best fitted model was when k = 11.

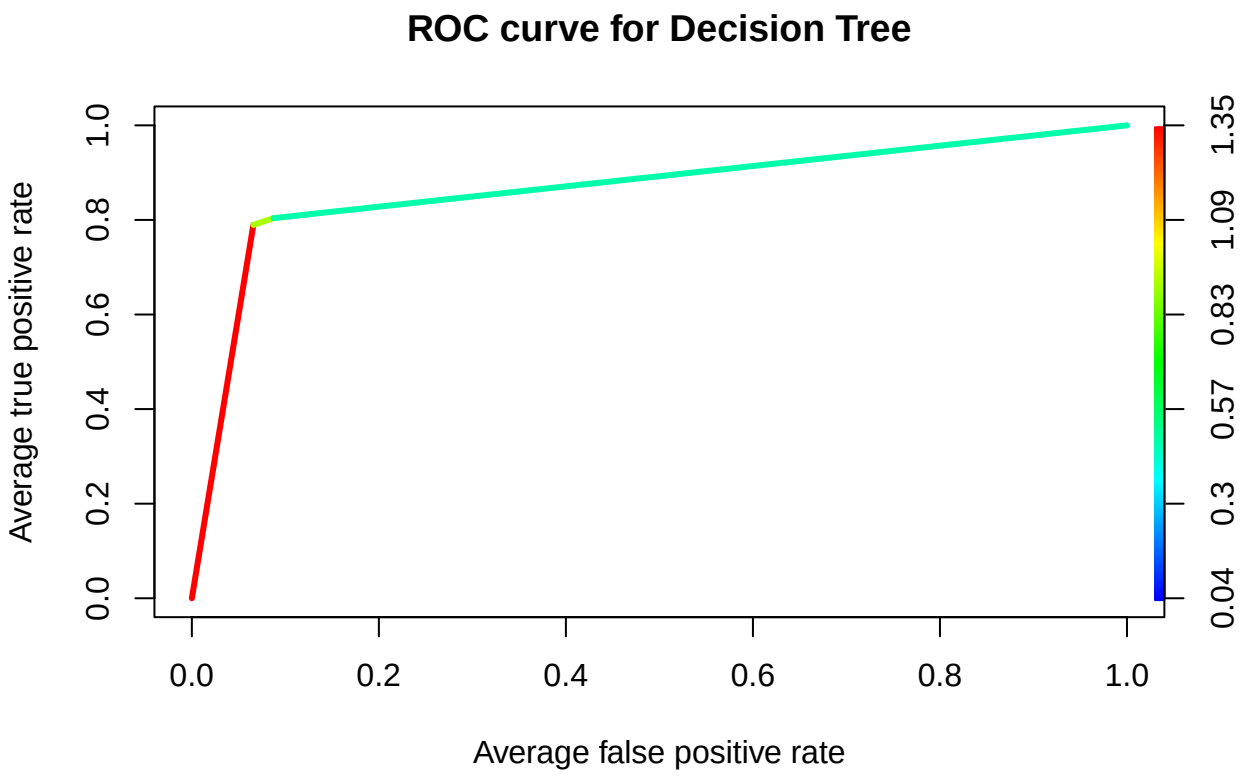
```r
confusionMatrix(diagnosis_knn11, test_data$diagnosis, positive = 'M')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 75  4
##          M  1 34
##
##                Accuracy : 0.9561
##                  95% CI : (0.9006, 0.9856)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 4.243e-14
##
##                   Kappa : 0.8993
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.8947
##             Specificity : 0.9868
##          Pos Pred Value : 0.9714
##          Neg Pred Value : 0.9494
##              Prevalence : 0.3333
##          Detection Rate : 0.2982
##    Detection Prevalence : 0.3070
##       Balanced Accuracy : 0.9408
##
##        'Positive' Class : M
##
```

ROC Curves for All Methods

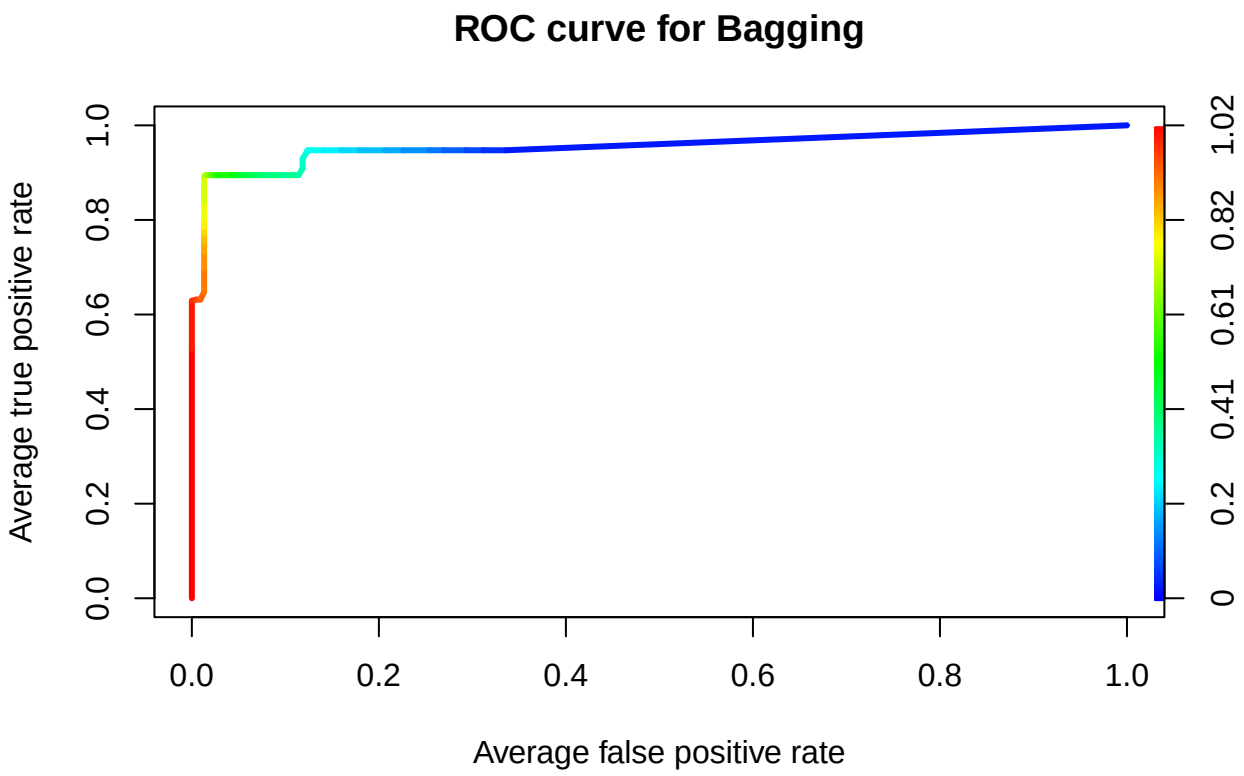
Decision Tree Method

```r
set.seed(1997)

diagnosis_tree_rocpreds <- predict(diagnosis_tree, newdata = test_data[c(-1,-2,-3)], type = "prob")
roc_tree_preds <- prediction(diagnosis_tree_rocpreds[,2], test_data$diagnosis)
roc_tree_perf <- performance(roc_tree_preds, "tpr", "fpr")
plot(roc_tree_perf, avg= "threshold", colorize=T, lwd=3, main="ROC curve for Decision Tree")
```



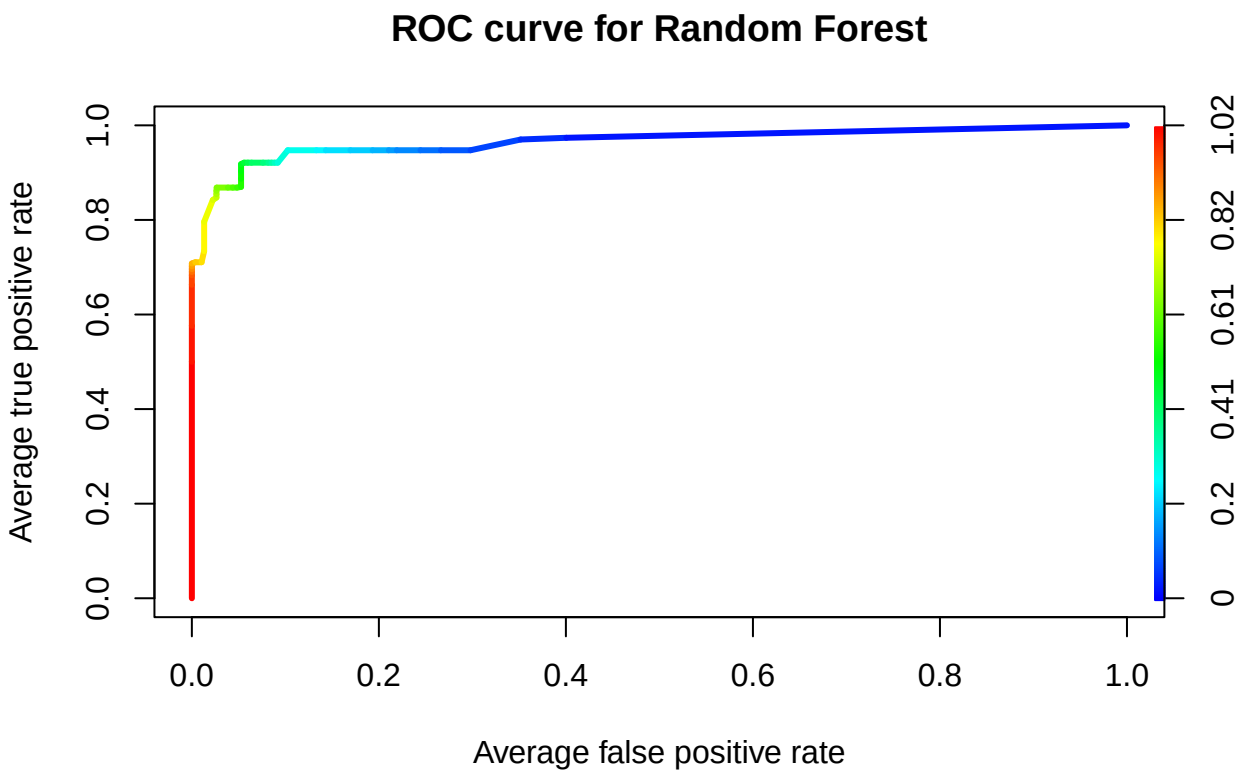
Bagging Method

```
set.seed(1997)

diagnosis_bag_rocpreds <- predict(diagnosis_bag, newdata = test_data[c(-1,-2,-3)], type = "prob")
roc_bag_preds <- prediction(diagnosis_bag_rocpreds[,2], test_data$diagnosis)
roc_bag_perf <- performance(roc_bag_preds, "tpr", "fpr")
plot(roc_bag_perf, avg= "threshold", colorize=T, lwd=3, main="ROC curve for Bagging")
```



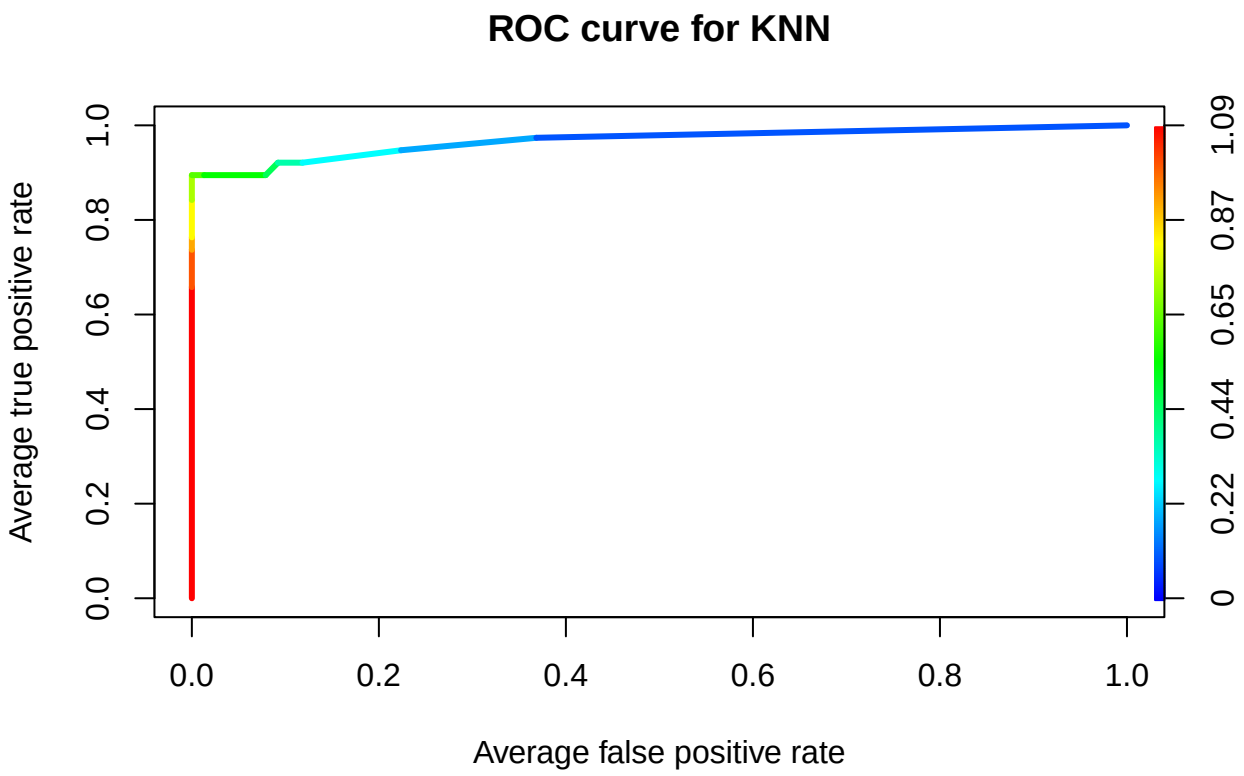
Random Forest Method

```
set.seed(1997)

diagnosis_forest_rocpreds <- predict(diagnosis_forest, newdata = test_data[c(-1,-2,-3)], type = "prob")
roc_forest_preds <- prediction(diagnosis_forest_rocpreds[,2], test_data$diagnosis)
roc_forest_perf <- performance(roc_forest_preds, "tpr", "fpr")
plot(roc_forest_perf, avg= "threshold", colorize=T, lwd=3, main="ROC curve for Random Forest")
```



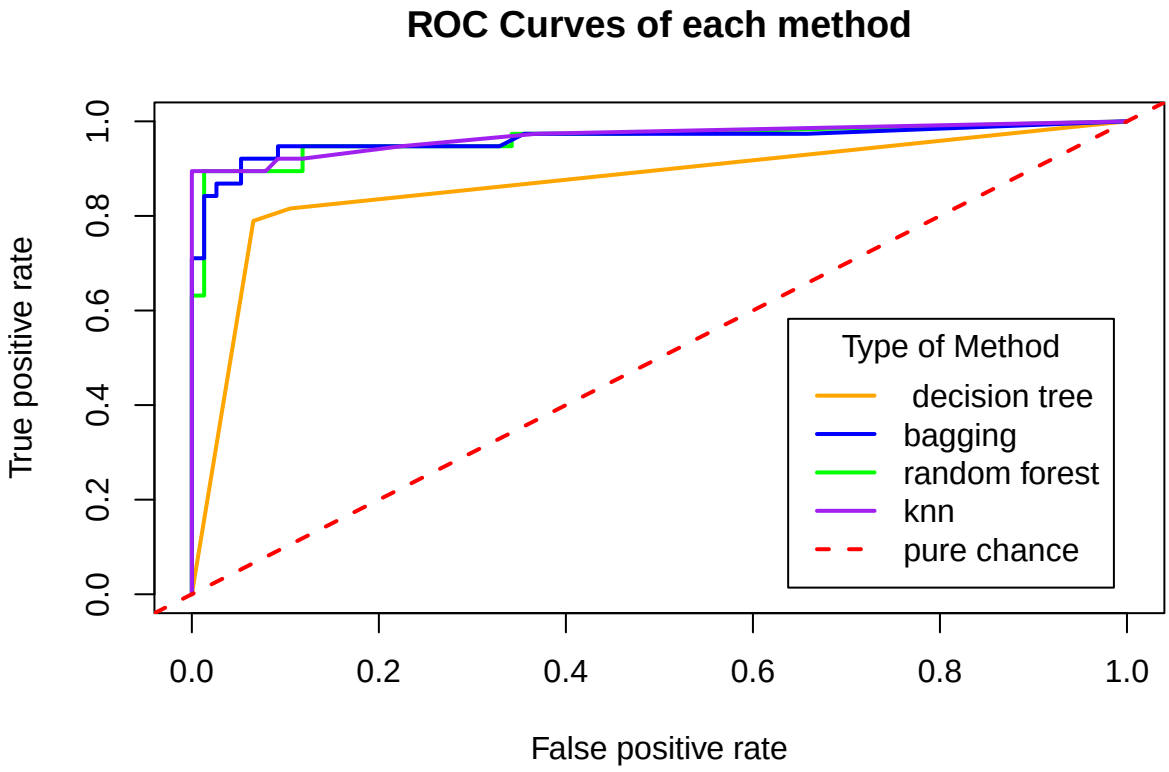
KNN Method

```
set.seed(1997)

diagnosis_knn_prob <- knn(train_data[c(-1,-2,-3)], test_data[c(-1,-2,-3)], cl = train_data$diagnosis_bi
# extracts the probabilities from the KNN method using the attribute function
prob <- attr(diagnosis_knn_prob, "prob")
# Since it takes majority voting, we must manually account for benign causes, which are defined as "0"
diagnosis_knn_rocpreds <- ifelse(diagnosis_knn_prob == "0", 1-prob, prob)
# plots ROC curve
roc_knn_preds <- prediction(diagnosis_knn_rocpreds, test_data$diagnosis)
roc_knn_perf <- performance(roc_knn_preds, "tpr", "fpr")
plot(roc_knn_perf, avg= "threshold", colorize=T, lwd=3, main="ROC curve for KNN")
```



Combining plot results of all four methods

```
set.seed(1997)

# plot curves on same graph
plot(roc_tree_perf, col = "orange", main = "ROC Curves of each method", lwd =2) # decision tree, 4
plot(roc_bag_perf, add=T, col = "green", lwd =2) # bagging, 2
plot(roc_forest_perf, add=T, col = "blue", lwd =2) # randomForest, 3
plot(roc_knn_perf, add=T, avg= "threshold", col = "purple", lwd =2) # knn, 1
abline(a = 0, b = 1, col = "red", lwd =2, lty=2)
legend(x = "bottomright",
       inset = 0.05,
       legend = c(" decision tree","bagging","random forest","knn","pure chance"),
        col = c("orange","blue","green","purple", "red"),
       lty = c(1,1,1,1,2),
       lwd = 2,
       title = "Type of Method"
       )
```


ROC Curves of each method
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
Type of Method
decision tree
bagging
random forest
knn
pure chance