Final project in Data Science at Harvard University

# Parkinsons Disease with ensemble methods

Jan Thomsen

2021

# Contents

# Chapter 1

# Abstract

This is the final assignment for the Harvard Data Science Professional certificate Program with Professor of Biostatistics Rafael Irizarry from Harvard University.

In this capstone project, we had to choose your own dataset and we have to analyze it and show our machine learning knowledge.

My motivation for diving into the area of Parkinson Disease is that the last 2-3 years i have lived a process helping my father who has been diagnosed with Alzheimer Disease, which is somewhat related. The process from showing symptoms to actually being diagnosed and then degenerate into an unconscious state, has been a big challenge as the son.

My conditions to do the medical analysis is on on at third party basis and the emphasis has been to show my knowledge that i have accomplished during these courses.

This is also the final assignment for the Harvard Data Science Professional certificate Program with Professor of Biostatistics Rafael Irizarry from Harvard University.

It is the 9th and last course in the Data Science series offered by Harvard University:

- **1. R basics**
- **2. Visualization**
- **3. Probability**
- **4. Inference and modeling**
- **5. Productivity tools**
- **6. Wrangling**
- **7. Linear regression**
- **8. Machine learning**
- **9. Capstone**

In this capstone project, we given the dataset and instructions we have to clean, analyze and modeling it and show our Data Science knowledge."

The target group for this report is obligated from Harvard to be an unknowing audience to the course materials.

**Nomenclature**

Here is the definitions of the focal concepts and variables in the project.

- **PD** Parkinson Disease
- **Subject** - Integer that uniquely identifies each subject
- **Age** - Subject age
- **Sex** - Subject gender '0' - male, '1' - female
- **Test_time** - Time since recruitment into the trial. The integer part is the number of days since recruitment
- **Motor_UPDRS** - Clinician's motor UPDRS score, linearly interpolated
- **Total_UPDRS** - Clinician's total UPDRS score, linearly interpolated
- **Jitter (%)** - Jitter(Abs), Jitter. RAP, Jitter. PPQ5, Jitter. DDP: Several measures of variation in fundamental frequency (Frequency parameters)
- **Shimmer** - Shimmer (dB), Shimmer. APQ3, Shimmer. APQ5, Shimmer. APQ11
- **Shimmer.DDA** - Several measures of variation in amplitude (Amplitude parameters)
- **NHR - HNR** - Two measures of ratio of noise to tonal components in the voice
- **RPDE** - A nonlinear dynamical complexity measure
- **DFA** - Signal fractal scaling exponent
- **PPE** - A nonlinear measure of fundamental frequency variation

# Chapter 2

# Executive Summary

Firstly the situation with the Parkinson Disease '(PD)' has become increasingly worrying especially when you experience it entering your personal life - my father is diagnosed with PD. The main motivation for working with this dataset is also to find significant variables in identifying patients suffering from Parkinson's disease with the means of the knowledge i have gained on my courses.

During the research presented in the dataset, I plan to focus on the following:

1. Explore variables of PD through visualization of variable distribution and anticipated correlations
2. Which variables are most important
3. Perform tests on the same dataset through appropriate methods for regression and classification
4. Conclude on the performances of the analyzed models

The **regression** analysis of severity (total_UPDRS) concluded that the bagged model because it had the lowest RMSE of the 5 models that was used to analyze the dataset. The age of the patient was the most important factor of the 10 predictors. The research shows that with 10 predictor the severity can be predicted with 2,15 RMSE.

The **classification** issue predicting the gender of patient giving the 10 predictors in the investigation. I have used various model to optimize the prediction of the sex, ending with $> 99\%$ accuracy. here it was the tuned boosted model, that had the highest accuracy.

The conclusions in this report suggest, that It should be considered to reduce the measurements of Parkinson Disease, which subsequently will impact the use resources to reach the same conclusions with accuracy.

The focal point should not be Big Data, but Good data.

# Chapter 3

# Introduction

Parkinson's disease is a neurodegenerative disorder of central nervous system that causes limited or complete loss of motor reflexes, speech, behavior, mental processing, and other vital functions. It is normally detected in elderly people and causes disorders in speech and motor abilities (writing, balance, etc.) of 90% of the patients. Occurring Alzheimer, PD is the second common neurological health problem in elder ages and it is estimated that nearly 10 million people all around the world and approximately 11k people in Denmark are suffering from this disease. Denmark, my country, has the 15. large proportion of elderly people $> 65 = 19.5\%$ out of a population of 5.8 mio.

Particularly, PD is generally seen in one out of every hundred people aged over 65. At present, there is no known cure for the disease. Although, there is significant amount of drug therapies to decrease complications caused by the disorder,

PD is usually diagnosed and treated using invasive methods. Therefore, this complicates the process of diagnosis and treatment of patients who are grieving from the disease.

The main reason behind the popularity of PD diagnosis from speech impairments is that tele-diagnosis and tele-monitoring systems based on voice signals are low in cost and easy to self-use . Such systems lower the inconvenience and cost of physical visits of PD patients to the medical clinic, enable the early diagnosis of the disease, and also lessen the workload of medical personnel.

People with Parkinsonism (PWP) suffer from speech impairments like dysphonia (defective use of the voice), hypophonia (reduced volume), monotone (reduced pitch range), and dysarthria (difficulty with articulation of sounds or syllables). Even though there are many studies aiming at diagnosing and monitoring PD using these impairments, the origin of these studies leans to diagnose basic voice disorders.

```
# Install all needed libraries if it is not present
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(kableExtra)) install.packages("kableExtra")
if(!require(tidyr)) install.packages("tidyr")
if(!require(stringr)) install.packages("stringr")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(gbm)) install.packages("gbm")
if(!require(dplyr)) install.packages("dplyr")
if(!require(caret)) install.packages("caret")
if(!require(randomForest)) install.packages("randomForest")
```

```r
if(!require(reshape2)) install.packages("reshape2")
if(!require(pROC)) install.packages("pROC")
if(!require(mvnormtest)) install.packages("mvnormtest")
if(!require(tibble)) install.packages("tibble")
if(!require(corrplot)) install.packages("corrplot")
if(!require(matrixStats)) install.packages("matrixStats")
if(!require(e1071)) install.packages("e1071")
if(!require(class)) install.packages("class")
if(!require(rpart)) install.packages("rpart")
if(!require(rpart.plot)) install.packages("rpart.plot")
if(!require(lattice)) install.packages("lattice")
if(!require(funModeling)) install.packages("funModeling")
if(!require(stringr)) install.packages("stringr")
if(!require(caTools)) install.packages("caTools")
if(!require(alookr)) install.packages("alookr")
if(!require(AppliedPredictiveModeling)) install.packages("AppliedPredictiveModeling")
if(!require(readr)) install.packages("readr")
if(!require(gt)) install.packages("gt")
if(!require(RSQLite)) install.packages("RSQLite")
if(!require(knitr)) install.packages("knitr")
if(!require(RhpcBLASctl)) install.packages("RhpcBLASctl")
if(!require(formattable)) install.packages("formattable")
if(!require(ggpubr)) install.packages("ggpubr")
if(!require(RColorBrewer)) install.packages("RColorBrewer")
if(!require(ModelMetrics)) install.packages("ModelMetrics")
if(!require(mmtable2)) install.packages("mmtable2")
if(!require(GGally)) install.packages("GGally")
if(!require(mlbench)) install.packages("mlbench")
if(!require(plyr)) install.packages("plyr")

#Loading the packages
library(plyr)
library(mlbench)
library(GGally)
library(mmtable2)
library(ModelMetrics)
library(RColorBrewer)
library(formattable)
library(ggpubr)
library(RhpcBLASctl)
library(knitr)
library(gt)
library(RSQLite)
library(AppliedPredictiveModeling)
library(readr)
library(tidyverse)
library(kableExtra)
library(tidyr)
```

```
library(ggplot2)
library(caret)
library(dplyr)
library(gbm)
library(tibble)
library(reshape2)
library(randomForest)
library(corrplot)
library(matrixStats)
library(mvnormtest)
library(pROC)
library(e1071)
library(class)
library(rpart)
library(rpart.plot)
library(lattice)
library(funModeling)
library(stringr)
library(caTools)
library(alookr)
```

# Chapter 4

# Exploratory Data Analysis

## 4.1 The Dataset

The dataset was created by Athanasios Tsanas and Max Little of the University of Oxford, in collaboration with 10 medical centers in the US and Intel Corporation who developed the tele-monitoring device to record the speech signals. The original study used a range of linear and nonlinear regression methods to predict the clinician's Parkinson's disease symptom score on the UPDRS scale.

This dataset is composed of a range of biomedical voice measurements from **42 people** with early-stage Parkinson's disease recruited to a six-month trial of a tele-monitoring device for remote symptom progression monitoring. The recordings were automatically captured in the patient's homes.

Columns in the dataset contain subject number, subject age, subject gender, time interval from baseline recruitment date, motor UPDRS, total UPDRS, and 16 biomedical voice measures. Each row corresponds to one of 5,875 voice recording from these individuals.

The main aim of the data is to predict the motor and total UPDRS scores ('motor_UPDRS' and 'total_UPDRS') from the 16 voice measures.

The data is in ASCII CSV format. The rows of the CSV file contain an instance corresponding to one voice recording. There are around 200 recordings per patient, the subject number of the patient is identified in the first column.

```
#Read the data file
parkinsons <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/
#parkinsons <- read.csv("~/Desktop/telemonitoring_parkinsons_updrs.csv", header = TRUE)
glimpse(parkinsons, width = getOption("width"))
```

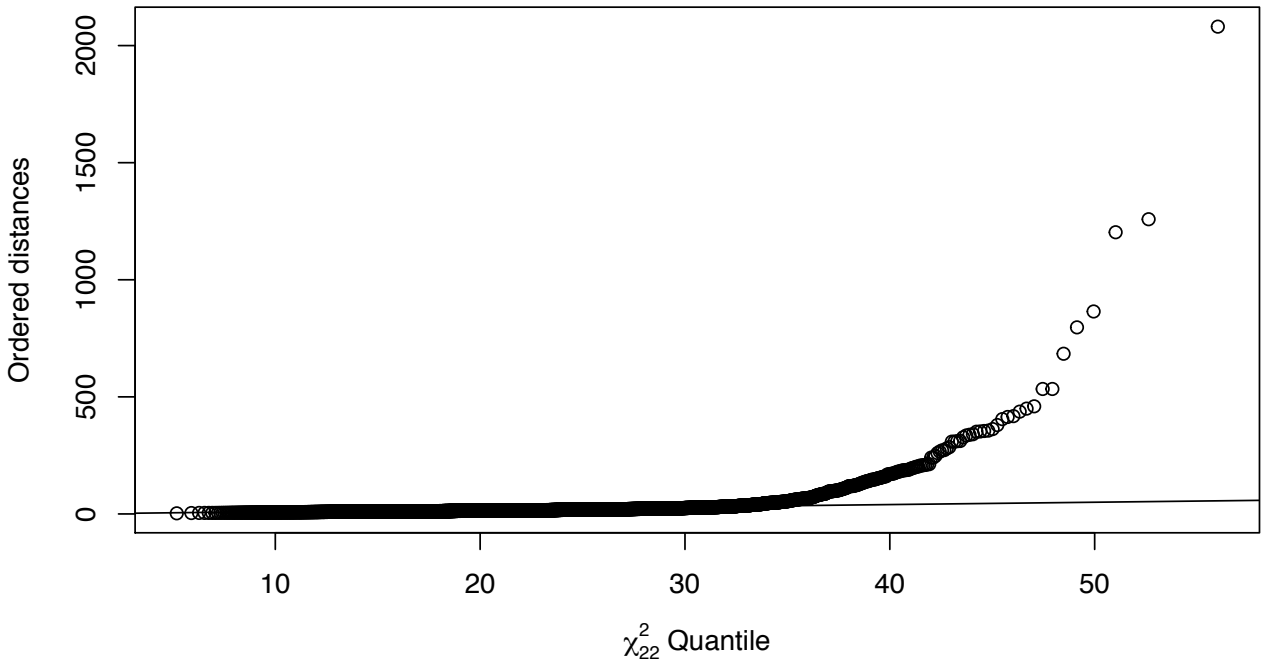
```
## Rows: 5,875
## Columns: 22
## $ subject.      <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ age           <int> 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, 72, ~
## $ sex           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ test_time     <dbl> 5.6431, 12.6660, 19.6810, 25.6470, 33.6420, 40.6520, 47.~
## $ motor_UPDRS   <dbl> 28.199, 28.447, 28.695, 28.905, 29.187, 29.435, 29.682, ~
## $ total_UPDRS   <dbl> 34.398, 34.894, 35.389, 35.810, 36.375, 36.870, 37.363, ~
## $ Jitter...     <dbl> 0.00662, 0.00300, 0.00481, 0.00528, 0.00335, 0.00353, 0.~
```

```
## $ Jitter.Abs.   <dbl> 3.380e-05, 1.680e-05, 2.462e-05, 2.657e-05, 2.014e-05, 2~
## $ Jitter.RAP    <dbl> 0.00401, 0.00132, 0.00205, 0.00191, 0.00093, 0.00119, 0.~
## $ Jitter.PPQ5   <dbl> 0.00317, 0.00150, 0.00208, 0.00264, 0.00130, 0.00159, 0.~
## $ Jitter.DDP    <dbl> 0.01204, 0.00395, 0.00616, 0.00573, 0.00278, 0.00357, 0.~
## $ Shimmer       <dbl> 0.02565, 0.02024, 0.01675, 0.02309, 0.01703, 0.02227, 0.~
## $ Shimmer.dB.   <dbl> 0.230, 0.179, 0.181, 0.327, 0.176, 0.214, 0.445, 0.212, ~
## $ Shimmer.APQ3  <dbl> 0.01438, 0.00994, 0.00734, 0.01106, 0.00679, 0.01006, 0.~
## $ Shimmer.APQ5  <dbl> 0.01309, 0.01072, 0.00844, 0.01265, 0.00929, 0.01337, 0.~
## $ Shimmer.APQ11 <dbl> 0.01662, 0.01689, 0.01458, 0.01963, 0.01819, 0.02263, 0.~
## $ Shimmer.DDA   <dbl> 0.04314, 0.02982, 0.02202, 0.03317, 0.02036, 0.03019, 0.~
## $ NHR           <dbl> 0.014290, 0.011112, 0.020220, 0.027837, 0.011625, 0.0094~
## $ HNR           <dbl> 21.640, 27.183, 23.047, 24.445, 26.126, 22.946, 22.506, ~
## $ RPDE          <dbl> 0.41888, 0.43493, 0.46222, 0.48730, 0.47188, 0.53949, 0.~
## $ DFA           <dbl> 0.54842, 0.56477, 0.54405, 0.57794, 0.56122, 0.57243, 0.~
## $ PPE           <dbl> 0.16006, 0.10810, 0.21014, 0.33277, 0.19361, 0.19500, 0.~
```

## 4.2 Checking Multivariate normality

To see if there are probable outliers in the dataset i perform a multivariate normal analysis.

```
#designing the multivariate normality analysis
x <- parkinsons
cm <- colMeans(x)
S <- cov(x)
d <- apply(x, 1, function(x) t(x - cm) %*% solve(S) %*% (x - cm))
# Chi-Square plot:
plot(qchisq((1:nrow(x) - 1/2) / nrow(x), df = ncol(x)),
     sort(d),
     xlab = expression(paste(chi[22]^2,
                             " Quantile")),
     ylab = "Ordered distances")
abline(a = 0, b = 1)
```




From the above figure, it can observed that our multivariate data is not completely normally distributed. There may be some outliers in our data set.

# Chapter 5

# Data Cleaning and Outlier Removal

The first step is going through the dataset and determine any missing value or outlier to take necessary measures. This step is fundamental to prepare the data for a beneficial analysis.

## 5.1 Check null values

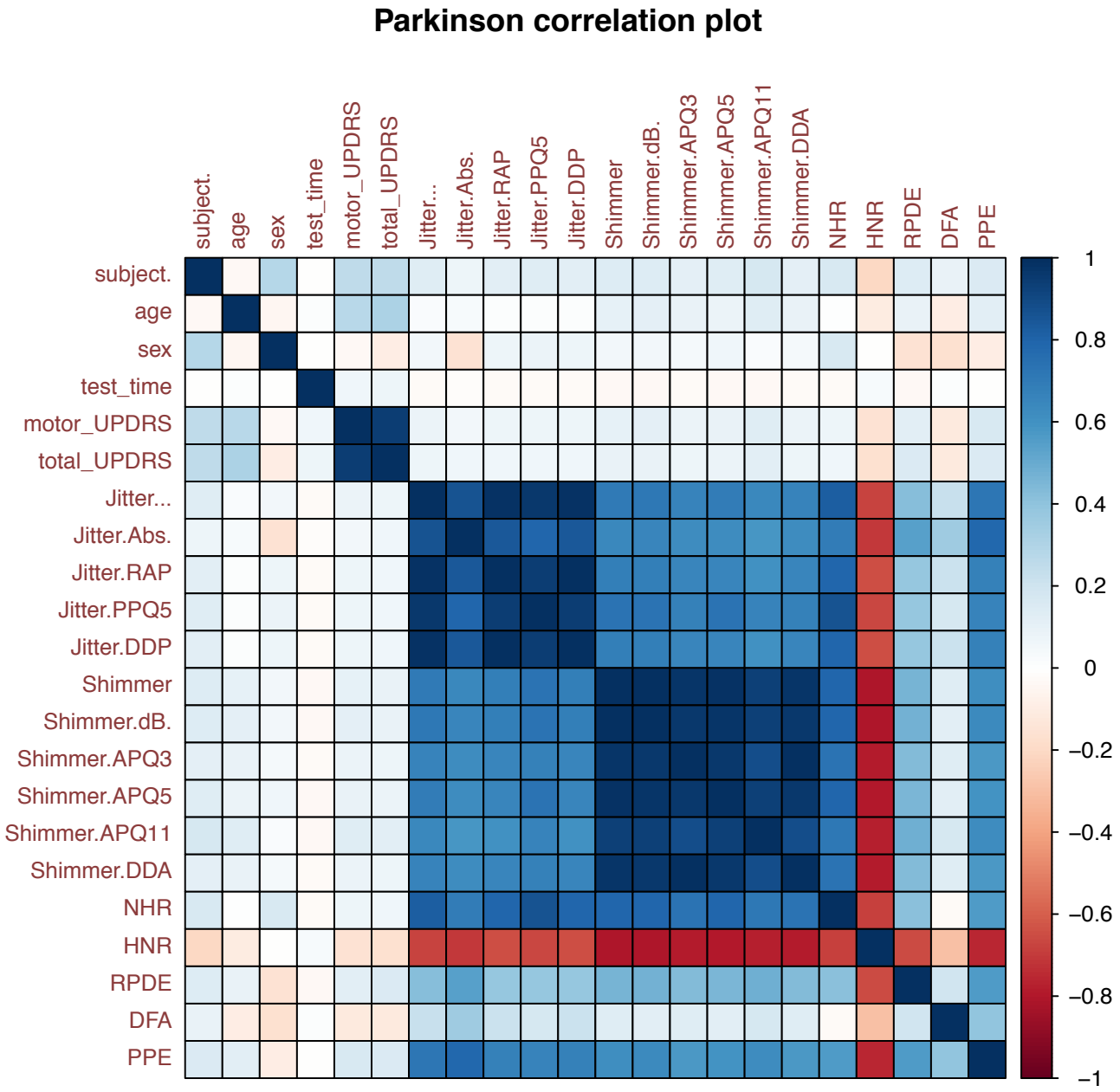
```
#checking if any data is missing
missing <- apply(parkinsons, 2, function(x)
  round(100 * (length(which(is.na(x))))/length(x) , digits = 1))
knitr::kable(missing)
```

| | x |
|---|---|
| subject. | 0 |
| age | 0 |
| sex | 0 |
| test_time | 0 |
| motor_UPDRS | 0 |
| total_UPDRS | 0 |
| Jitter... | 0 |
| Jitter.Abs. | 0 |
| Jitter.RAP | 0 |
| Jitter.PPQ5 | 0 |
| Jitter.DDP | 0 |
| Shimmer | 0 |
| Shimmer.dB. | 0 |
| Shimmer.APQ3 | 0 |
| Shimmer.APQ5 | 0 |
| Shimmer.APQ11 | 0 |
| Shimmer.DDA | 0 |
| NHR | 0 |
| HNR | 0 |
| RPDE | 0 |
| DFA | 0 |
| PPE | 0 |

There are no missing values in our dataset.

## 5.2 Check correlations between the variables

```
#checking correlation
corrplot(cor(parkinsons), type="full", method ="color", title = "Parkinson correlation plot"
```




It should be noticed that all the jitter variables highly correlates with Shimmer variables.

## 5.3 Outlier detection

```
# overview of data
summary(parkinsons[,-3])
```

```
##     subject.          age         test_time        motor_UPDRS
##  Min.   : 1.00   Min.   :36.0   Min.   : -4.263   Min.   : 5.038
##  1st Qu.:10.00   1st Qu.:58.0   1st Qu.: 46.847   1st Qu.:15.000
##  Median :22.00   Median :65.0   Median : 91.523   Median :20.871
##  Mean   :21.49   Mean   :64.8   Mean   : 92.864   Mean   :21.296
##  3rd Qu.:33.00   3rd Qu.:72.0   3rd Qu.:138.445   3rd Qu.:27.596
##  Max.   :42.00   Max.   :85.0   Max.   :215.490   Max.   :39.511
##   total_UPDRS      Jitter...          Jitter.Abs.          Jitter.RAP
##  Min.   : 7.00   Min.   :0.000830   Min.   :2.250e-06   Min.   :0.000330
##  1st Qu.:21.37   1st Qu.:0.003580   1st Qu.:2.244e-05   1st Qu.:0.001580
##  Median :27.58   Median :0.004900   Median :3.453e-05   Median :0.002250
##  Mean   :29.02   Mean   :0.006154   Mean   :4.403e-05   Mean   :0.002987
##  3rd Qu.:36.40   3rd Qu.:0.006800   3rd Qu.:5.333e-05   3rd Qu.:0.003290
##  Max.   :54.99   Max.   :0.099990   Max.   :4.456e-04   Max.   :0.057540
##   Jitter.PPQ5         Jitter.DDP          Shimmer          Shimmer.dB.
##  Min.   :0.000430   Min.   :0.000980   Min.   :0.00306   Min.   :0.026
##  1st Qu.:0.001820   1st Qu.:0.004730   1st Qu.:0.01912   1st Qu.:0.175
##  Median :0.002490   Median :0.006750   Median :0.02751   Median :0.253
##  Mean   :0.003277   Mean   :0.008962   Mean   :0.03404   Mean   :0.311
##  3rd Qu.:0.003460   3rd Qu.:0.009870   3rd Qu.:0.03975   3rd Qu.:0.365
##  Max.   :0.069560   Max.   :0.172630   Max.   :0.26863   Max.   :2.107
##   Shimmer.APQ3       Shimmer.APQ5      Shimmer.APQ11       Shimmer.DDA
##  Min.   :0.00161   Min.   :0.00194   Min.   :0.00249   Min.   :0.00484
##  1st Qu.:0.00928   1st Qu.:0.01079   1st Qu.:0.01566   1st Qu.:0.02783
##  Median :0.01370   Median :0.01594   Median :0.02271   Median :0.04111
##  Mean   :0.01716   Mean   :0.02014   Mean   :0.02748   Mean   :0.05147
##  3rd Qu.:0.02057   3rd Qu.:0.02375   3rd Qu.:0.03272   3rd Qu.:0.06173
##  Max.   :0.16267   Max.   :0.16702   Max.   :0.27546   Max.   :0.48802
##       NHR                HNR              RPDE              DFA
##  Min.   :0.000286   Min.   : 1.659   Min.   :0.1510   Min.   :0.5140
##  1st Qu.:0.010955   1st Qu.:19.406   1st Qu.:0.4698   1st Qu.:0.5962
##  Median :0.018448   Median :21.920   Median :0.5423   Median :0.6436
##  Mean   :0.032120   Mean   :21.680   Mean   :0.5415   Mean   :0.6532
##  3rd Qu.:0.031463   3rd Qu.:24.444   3rd Qu.:0.6140   3rd Qu.:0.7113
##  Max.   :0.748260   Max.   :37.875   Max.   :0.9661   Max.   :0.8656
##       PPE
##  Min.   :0.02198
##  1st Qu.:0.15634
##  Median :0.20550
##  Mean   :0.21959
##  3rd Qu.:0.26449
##  Max.   :0.73173
```

The total_UPDRS (Unified Parkinson's Disease Ratings Score) is the main variable of interest, which

determines the clinical impression of Parkinson's disease (PD) severity.

Furthermore the subject number is a ascending number which as an independent variable do not explain anything about the outcome other than making the individuals unique.

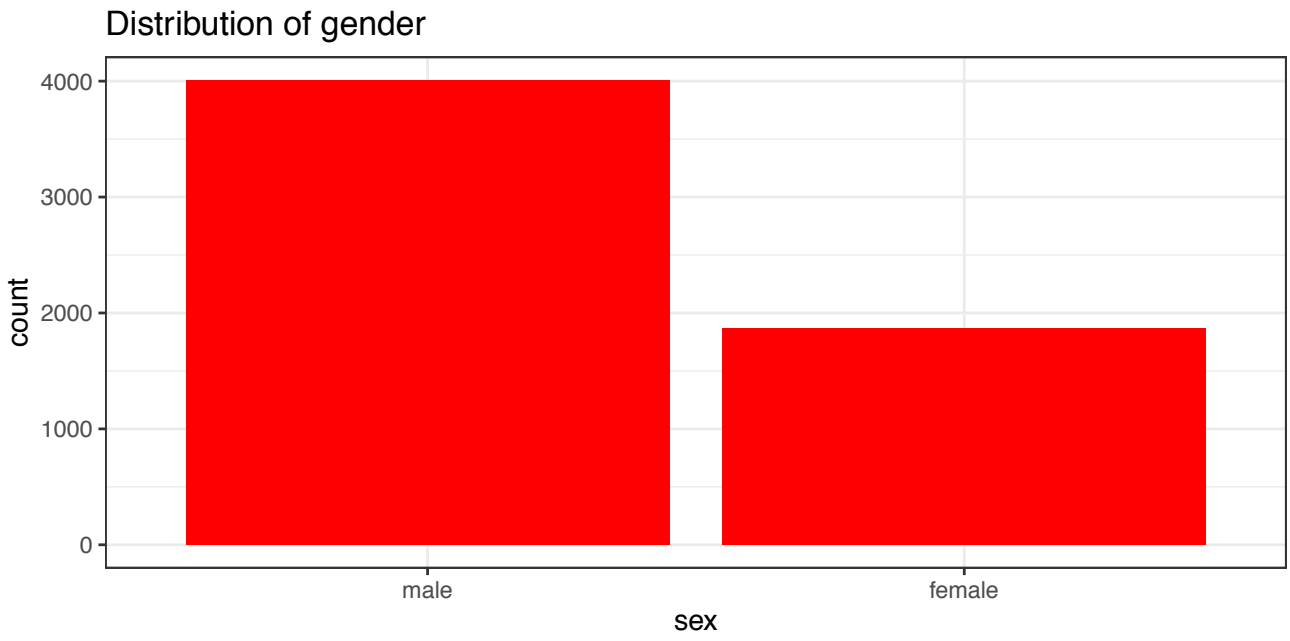
These outliers will be removed in section "ensemble methods" section.

# Chapter 6

# Visualization

For visualization purposes i will focus on severity and gender.

```
#proportions of measurements
# Distribution of the sex column
df <- as.factor(parkinsons$sex)
parkinsons$sex <- factor(parkinsons$sex,
                  levels=c(0,1),
                  labels=c("male","female"))

options(repr.plot.width=4, repr.plot.height=4)
ggplot(parkinsons, aes(x=sex))+geom_bar(fill="red",alpha=1)+theme_bw()+labs(title="Distribut
```




It seems that males are overly represented in the dataset.

### Parkinson and gender

```
#average severity and testtime distributed on gender
```

```r
parkinsons$sex <- as.factor(parkinsons$sex)
data_wrangled <- parkinsons %>%
  group_by(parkinsons$sex) %>%
  summarise(avg = mean(total_UPDRS), sd = sd(total_UPDRS))

print(data_wrangled)
```

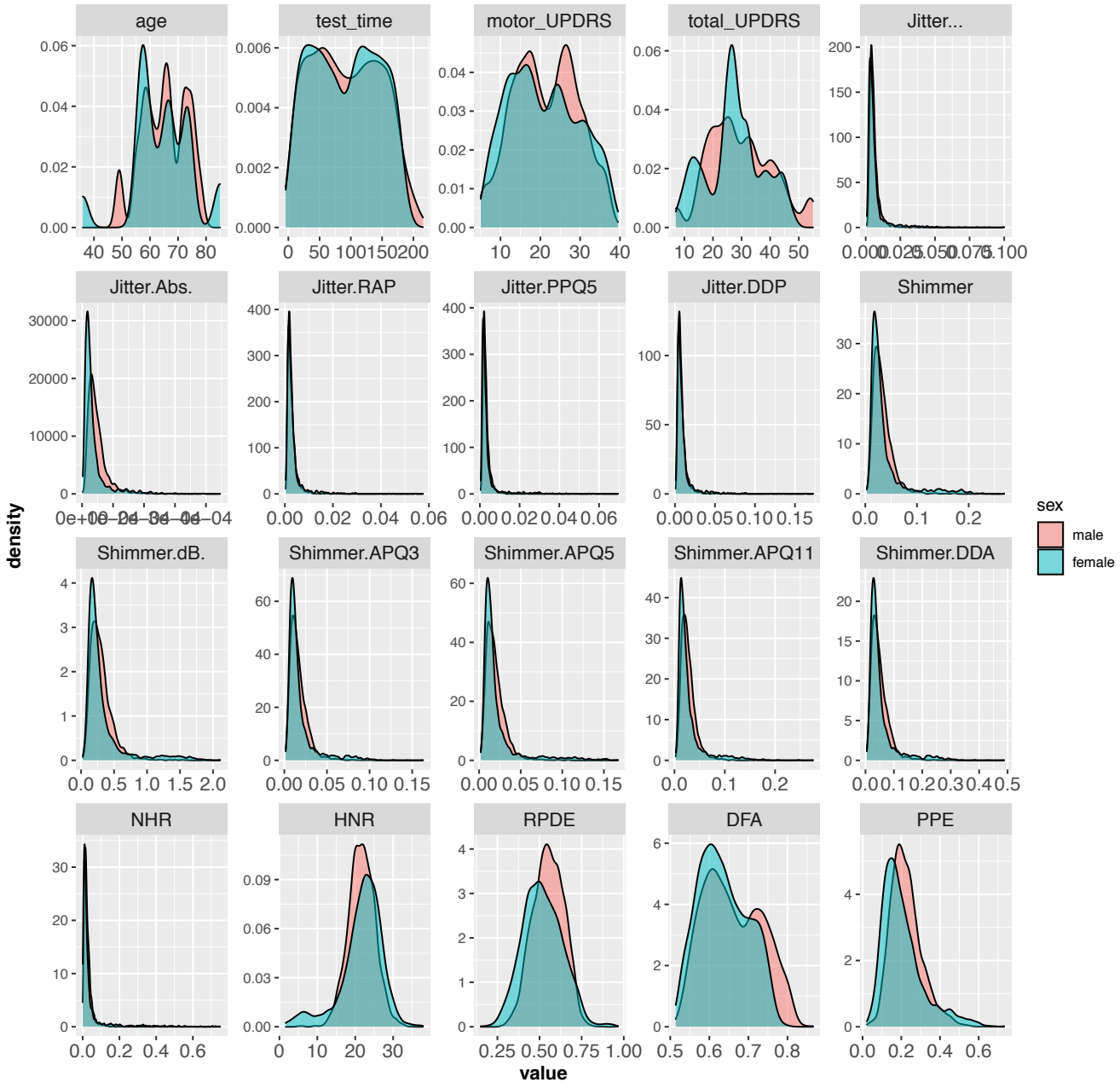
```
##        avg       sd
## 1 29.01894 10.70028
```

The average male severity is 29.72 total_UPDRS with a standard deviation of 11.00. The average female severity is 27.51 with a 9.85 standard deviation.

Additional research show that PD comes approx. 2 years later for women than men.

```
# Severity and sex plot
parkinsons$sex <- as.factor(parkinsons$sex)
mu <- ddply(parkinsons, "sex", summarise, grp.mean=mean(total_UPDRS))
head(mu)

p<-ggplot(parkinsons, aes(x=total_UPDRS, fill=sex)) +
  geom_density(alpha=0.5)+
  geom_vline(data=mu, aes(xintercept=grp.mean, color=sex),
             linetype="dashed", size=1) + ggtitle("Densityplot of total_UPDRS and sex") + th
axis.title.x = element_text(color="black", size=16, face="bold"),
axis.title.y = element_text(color="black", size=16, face="bold")
)

#facet density plot  of the genders
attach(parkinsons)
parkinsons$sex=as.factor(parkinsons$sex)
park.m=melt(parkinsons[,-1], id.vars = "sex")

p <- ggplot(data = park.m, aes(x=value)) +  geom_density(aes(fill=sex), alpha = 0.5)
p <- p + facet_wrap( ~ variable, scales = "free") + ggtitle("Densityplots on dataset predict
plot.title = element_text(color="blue", size=20, face="bold.italic"),
           axis.text.x = element_text( size = 12 ),
           axis.title = element_text( size = 12, face = "bold" ),
           legend.position="right",
           strip.text = element_text(size = 12))
p
```

## *Densityplots on dataset predictors*

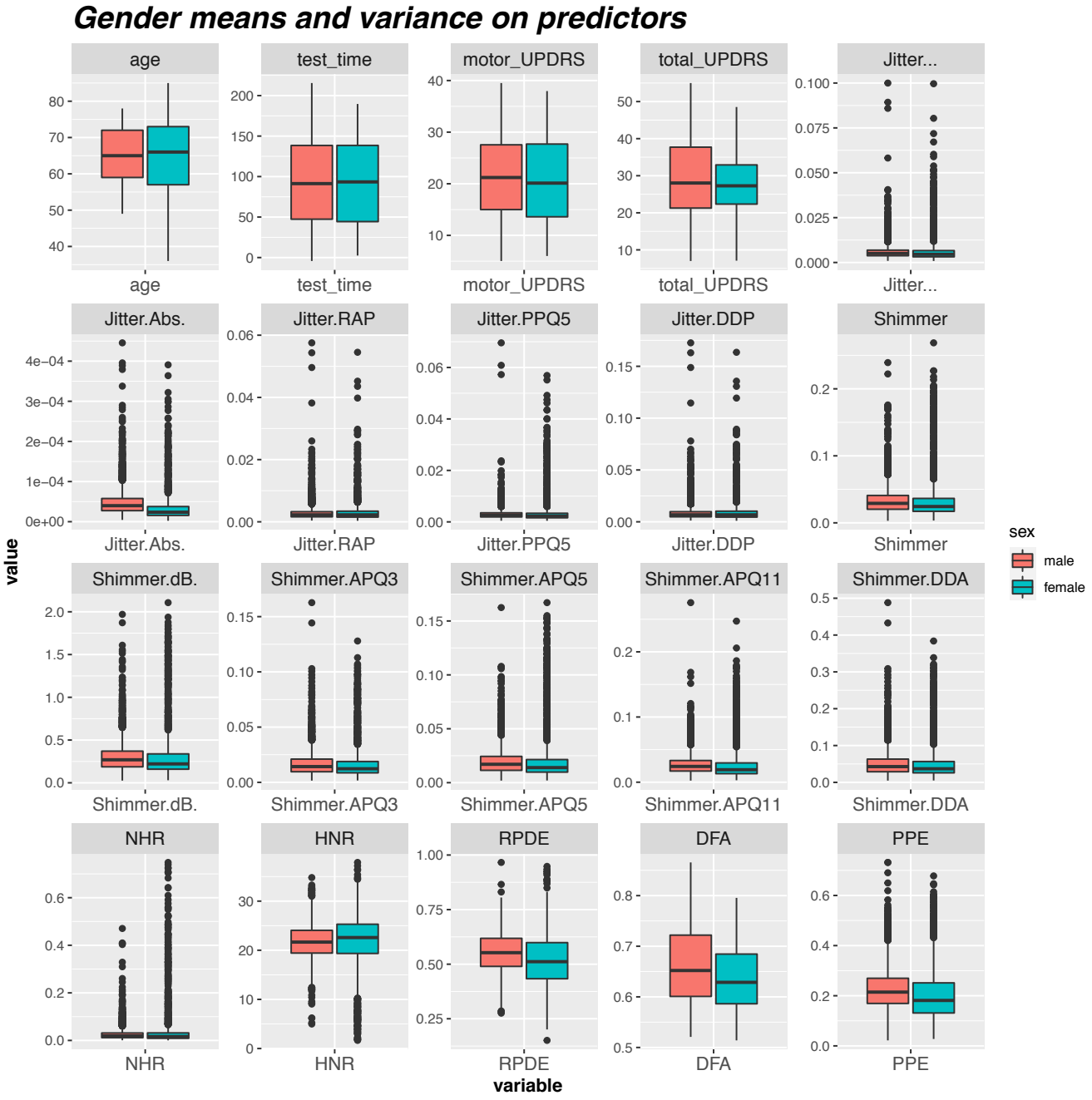



```
#Average and variance plot on gender
attach(parkinsons)
```

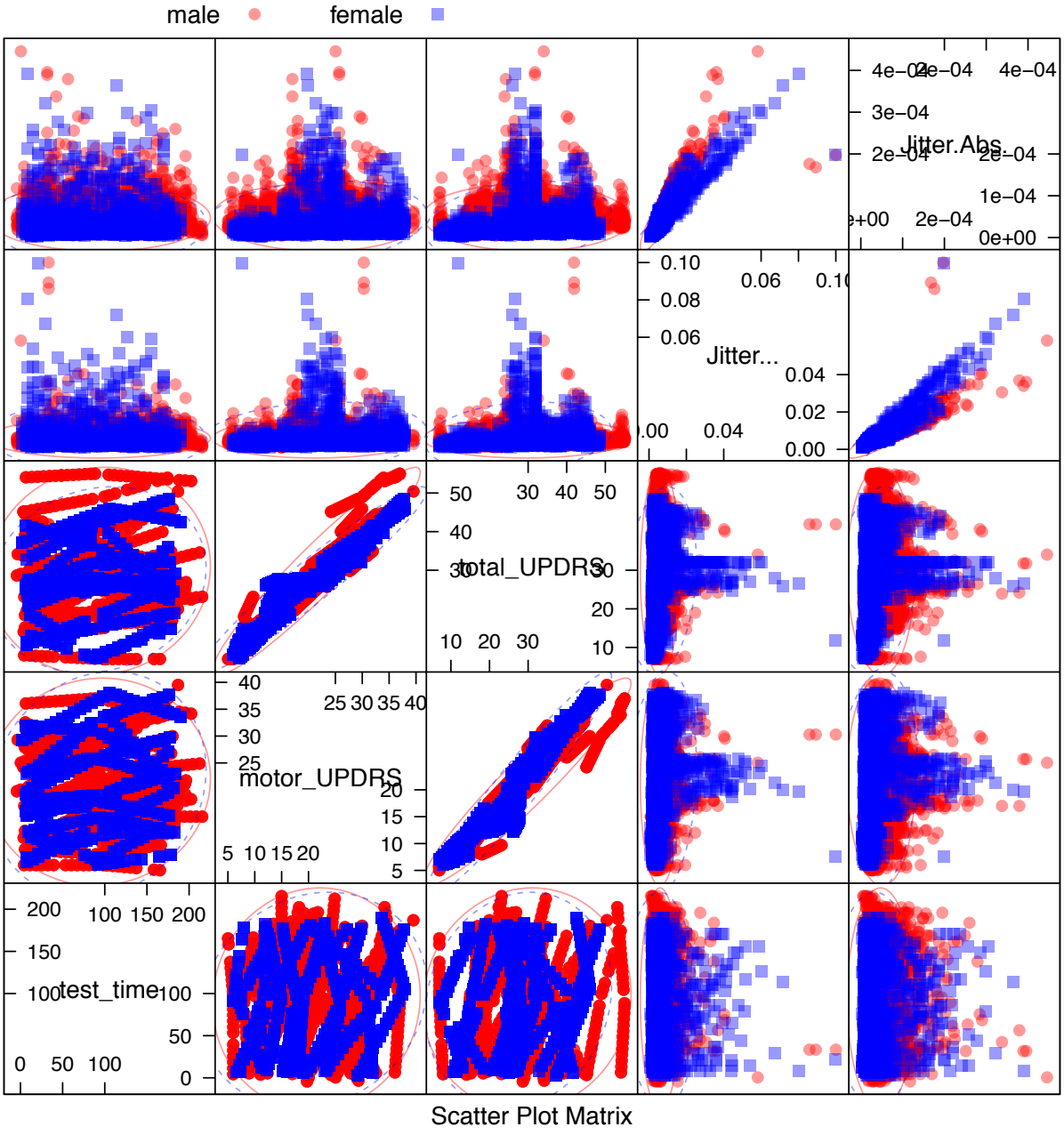
```
## The following objects are masked from parkinsons (pos = 3):
##
##     age, DFA, HNR, Jitter..., Jitter.Abs., Jitter.DDP, Jitter.PPQ5,
##     Jitter.RAP, motor_UPDRS, NHR, PPE, RPDE, sex, Shimmer,
##     Shimmer.APQ11, Shimmer.APQ3, Shimmer.APQ5, Shimmer.dB.,
##     Shimmer.DDA, subject., test_time, total_UPDRS
```

```
parkinsons$sex=as.factor(parkinsons$sex)
park.m=melt(parkinsons[,-1], id.vars = "sex")
```

```
p <- ggplot(data = park.m, aes(x=variable, y=value)) +
  geom_boxplot(aes(fill=sex))
p + facet_wrap( ~ variable, scales="free") + ggtitle("Gender means and variance on predictor
plot.title = element_text(color="black", size=20, face="bold.italic"),
          axis.text.x = element_text( size = 12 ),
          axis.title = element_text( size = 12, face = "bold" ),
          legend.position="right",
          strip.text = element_text(size = 12))
```



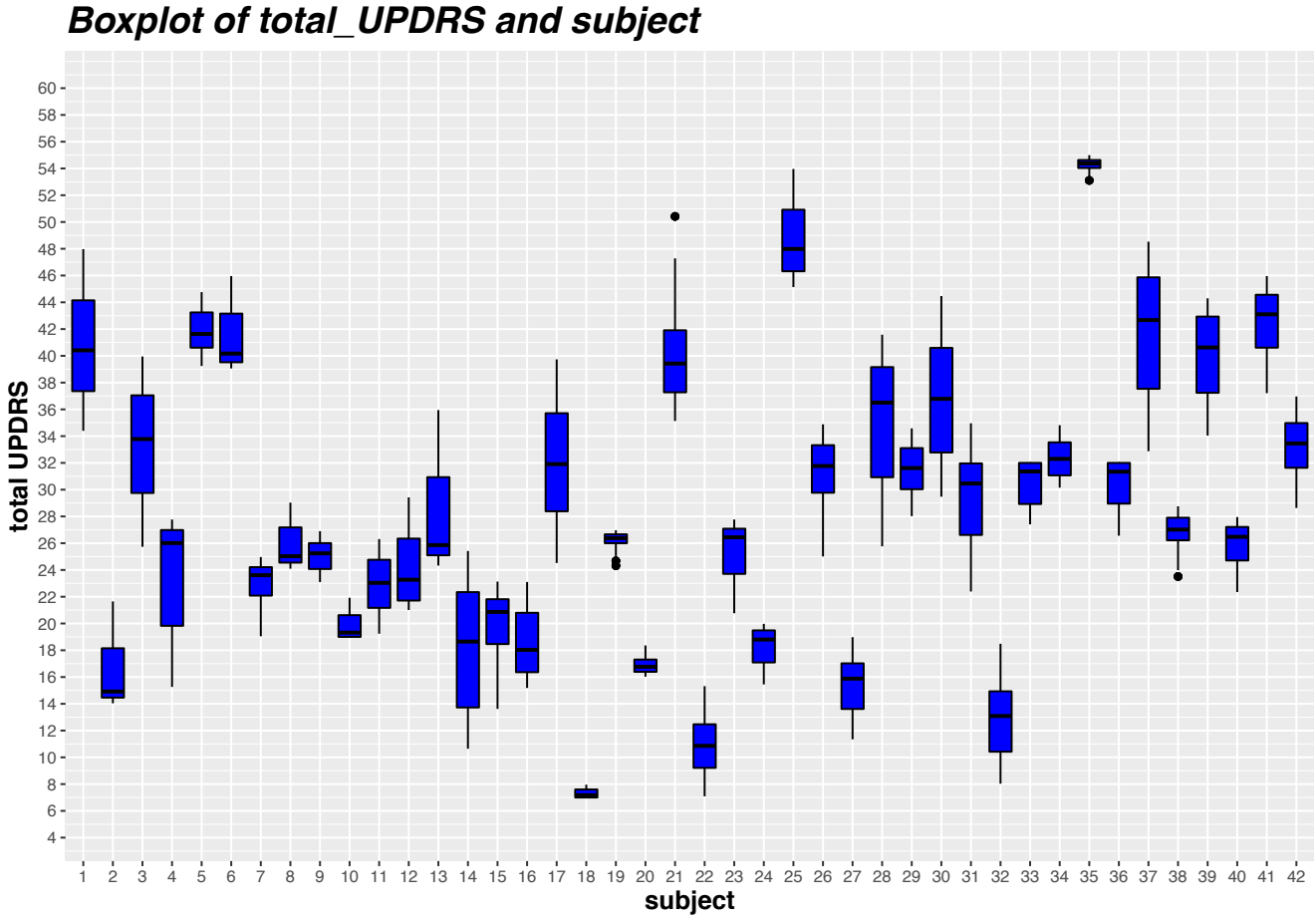
```
#Featureplot on variables on gender
transparentTheme(trans = .4)
caret::featurePlot(x = parkinsons[, 4:8],
            y = parkinsons$sex,
            plot = "ellipse", jitter = TRUE,
            auto.key = list(columns = 5))
```




```
#Title: Gender Scatterplot on predictors
```

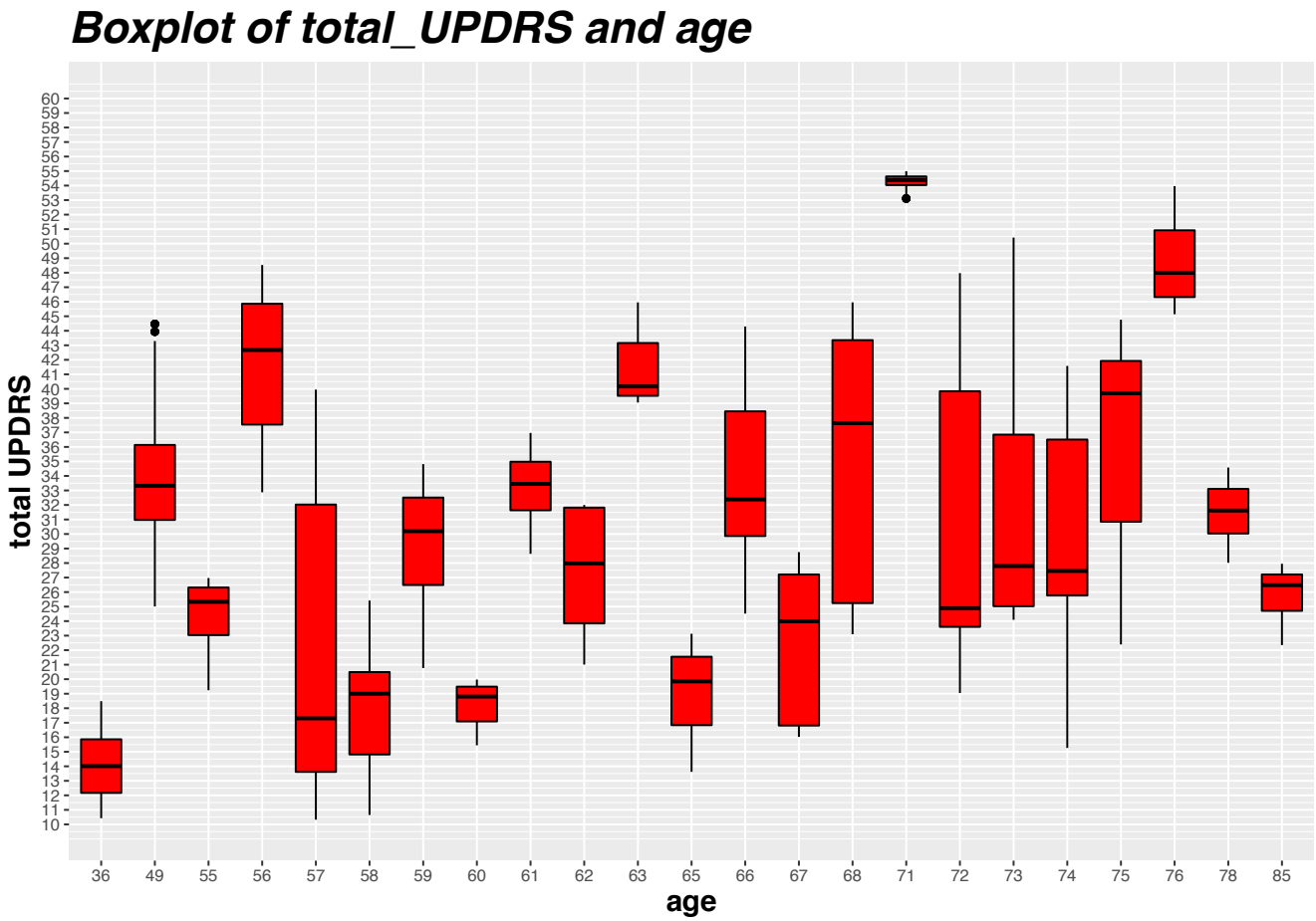
### Boxplots for PD severity of different patients

```
# boxplot for total UPDRS by different subjects
fill <- "blue"
line <- "black"
ggplot(parkinsons, aes(x =as.factor(parkinsons$subject.), y =parkinsons$total_UPDRS)) +
        geom_boxplot(fill = fill, colour = line) +
        scale_y_continuous(name = "total UPDRS",
                           breaks = seq(0, 60, 2),
                           limits=c(5, 60)) +
        scale_x_discrete(name = "subject") +
        ggtitle("Boxplot of total_UPDRS and subject")+ theme(
plot.title = element_text(color="black", size=20, face="bold.italic"),
axis.title.x = element_text(color="black", size=14, face="bold"),
axis.title.y = element_text(color="black", size=14, face="bold")
)
```



### Boxplots of PD severity and age

```
#Severity and age plot
fill <- "red"
line <- "black"
ggplot(parkinsons, aes(x =as.factor(parkinsons$age), y =parkinsons$total_UPDRS)) +
        geom_boxplot(fill = fill, colour = line) +
        scale_y_continuous(name = "total UPDRS",
                           breaks = seq(10, 60, 1),
                           limits=c(10, 60)) +
        scale_x_discrete(name = "age") +
        ggtitle("Boxplot of total_UPDRS and age") +theme(
plot.title = element_text(color="black", size=24, face="bold.italic"),
axis.title.x = element_text(color="black", size=16, face="bold"),
axis.title.y = element_text(color="black", size=16, face="bold")
)
```




Not surprisingly PD comes with age as a important contributing factor.

# Chapter 7

# Ensemble methods

Before diving into the regression and classification calculations I would be beneficial to define what ensemble methods actually are.

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers (experts) and to allow them to vote.

Before the regression and the classification analysis it is feasible to reduce the complexity and the dataset and reduce the number of variables.

#### 7.0.0.1 Reduction of variables

To ensure the focus on the focal point in the analysis i choose to reduce the variable that are highly correlated (Pearsons x>0.9).

```
# Remove correlated variables x>0.9
parkinsons$sex <- as.factor(parkinsons$sex)
parkinsons$age <- as.numeric(parkinsons$age)
parkinsons <- treatment_corr(parkinsons) #, corr_thres = 0.9, treat = FALSE)
```

```
## * remove variables whose strong correlation (pearson >= 0.8)
```

```
##  - remove motor_UPDRS   : with total_UPDRS (0.9472)
##  - remove Jitter...     : with Jitter.Abs. (0.8656)
##  - remove Jitter...     : with Jitter.RAP (0.9842)
##  - remove Jitter...     : with Jitter.PPQ5 (0.9682)
##  - remove Jitter...     : with Jitter.DDP (0.9842)
##  - remove Jitter...     : with NHR (0.8253)
##  - remove Jitter.Abs.   : with Jitter.RAP (0.8446)
##  - remove Jitter.Abs.   : with Jitter.DDP (0.8446)
##  - remove Jitter.RAP    : with Jitter.PPQ5 (0.9472)
##  - remove Jitter.RAP    : with Jitter.DDP (1)
##  - remove Jitter.PPQ5   : with Jitter.DDP (0.9472)
##  - remove Jitter.PPQ5   : with NHR (0.8649)
##  - remove Shimmer       : with Shimmer.dB. (0.9923)
##  - remove Shimmer       : with Shimmer.APQ3 (0.9798)
```

```
##  - remove Shimmer       : with Shimmer.APQ5 (0.9849)
##  - remove Shimmer       : with Shimmer.APQ11 (0.9355)
##  - remove Shimmer       : with Shimmer.DDA (0.9798)
##  - remove Shimmer       : with HNR (-0.8014)
##  - remove Shimmer.dB.   : with Shimmer.APQ3 (0.968)
##  - remove Shimmer.dB.   : with Shimmer.APQ5 (0.9764)
##  - remove Shimmer.dB.   : with Shimmer.APQ11 (0.9363)
##  - remove Shimmer.dB.   : with Shimmer.DDA (0.968)
##  - remove Shimmer.dB.   : with HNR (-0.8025)
##  - remove Shimmer.APQ3  : with Shimmer.APQ5 (0.9627)
##  - remove Shimmer.APQ3  : with Shimmer.APQ11 (0.8857)
##  - remove Shimmer.APQ3  : with Shimmer.DDA (1)
##  - remove Shimmer.APQ5  : with Shimmer.APQ11 (0.9389)
##  - remove Shimmer.APQ5  : with Shimmer.DDA (0.9627)
##  - remove Shimmer.APQ11 : with Shimmer.DDA (0.8857)
```

```
# Number of columns after removing correlated variables
ncol(parkinsons)
```

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

```
names(parkinsons)
```

```
##  [1] "subject."    "age"         "sex"         "test_time"   "total_UPDRS"
##  [6] "Jitter.DDP"  "Shimmer.DDA" "NHR"         "HNR"         "RPDE"
## [11] "DFA"         "PPE"
```

Variable subject Furthermore we need to exclude the subject variable because it is a number without information that contributes to qualify the total UPDRS.

```
#Excluding variable subject
parkinsons <- parkinsons[-1]
str(parkinsons)
```

```
## 'data.frame':    5875 obs. of  11 variables:
##  $ age        : num  72 72 72 72 72 72 72 72 72 72 ...
##  $ sex        : Factor w/ 2 levels "male","female": 1 1 1 1 1 1 1 1 1 1 ...
##  $ test_time  : num  5.64 12.67 19.68 25.65 33.64 ...
##  $ total_UPDRS: num  34.4 34.9 35.4 35.8 36.4 ...
##  $ Jitter.DDP : num  0.01204 0.00395 0.00616 0.00573 0.00278 ...
##  $ Shimmer.DDA: num  0.0431 0.0298 0.022 0.0332 0.0204 ...
##  $ NHR        : num  0.0143 0.0111 0.0202 0.0278 0.0116 ...
##  $ HNR        : num  21.6 27.2 23 24.4 26.1 ...
##  $ RPDE       : num  0.419 0.435 0.462 0.487 0.472 ...
##  $ DFA        : num  0.548 0.565 0.544 0.578 0.561 ...
##  $ PPE        : num  0.16 0.108 0.21 0.333 0.194 ...
```

We will have 10 variables to test and make predictions.

# Chapter 8

# Regression

## 8.1 Training set and test set

Data partition in training and test sets The dataset is relatively large, hence I choose make a datasplit 50/50.

```
#Data Partition
data2 <- parkinsons
set.seed(18)
pd_idx = sample(1:nrow(data2), nrow(data2) / 2)
pd_trn = data2[pd_idx,]
pd_tst = data2[-pd_idx,]
```

## 8.2 Regression model evaluation metrics

The dependent variable will be total_UPDRS, which is a approximated Gaussian distribution. The independent variables will be 10 variables (predictors) as i will exclude the subject number because it will bias the outcome unnecessarily.

There are three metrics which are generally used for evaluation of Regression problems (like Linear Regression, Decision Tree Regression, Random Forest Regression etc.).

**Mean Absolute Error (MAE)**: This measures the absolute average distance between the real data and the predicted data, but it fails to punish large errors in prediction. **Mean Square Error (MSE)**: This measures the squared average distance between the real data and the predicted data. Here, larger errors are well noted (better than MAE). But the disadvantage is that it also squares up the units of data as well. So, evaluation with different units is not at all justified. **Root Mean Squared Error (RMSE)**: This is actually the square root of MSE. Also, this metrics solves the problem of squaring the units. Hence, all the metrics above measures the average model prediction error ranging between 0 to infinity with negatively oriented scores which means lower the evaluation value better is your model.

In short, MSE evaluates with squared units which is clearly unjustified. Mathematically, taking up absolute value for evaluation is quite undesirable, making RMSE a distinct advantageous over other two.

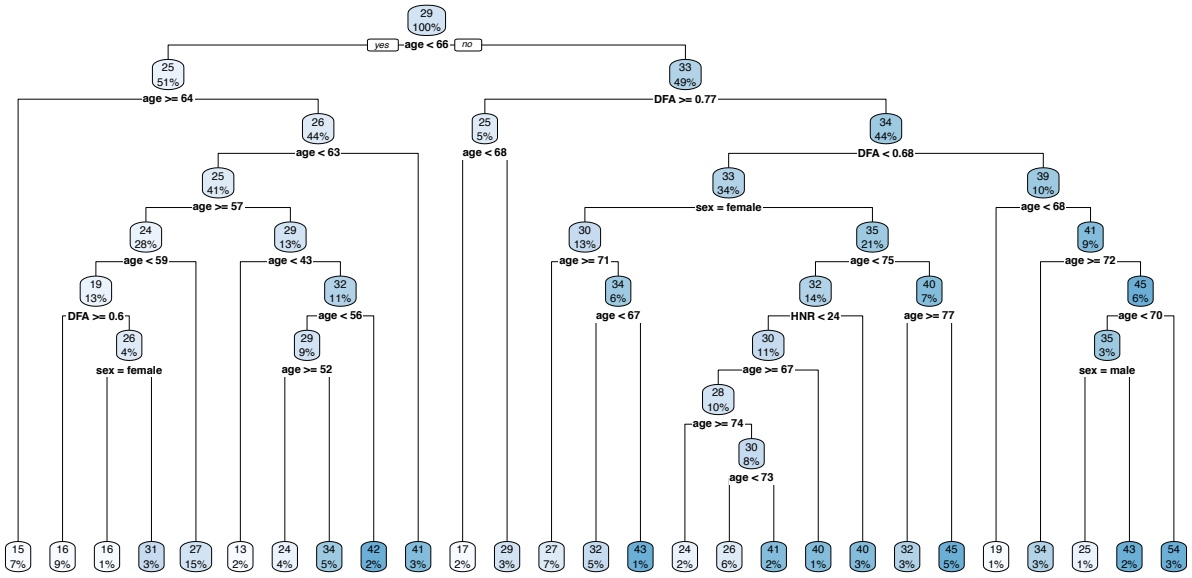
## 8.3 Decision trees (rpart)

The rpart algorithm works by splitting the dataset recursively, which means that the subsets that arise from a split are further split until a predetermined termination criterion is reached. At each step, the split is made based on the independent variable that results in the largest possible reduction in heterogeneity of the dependent (predicted) variable.

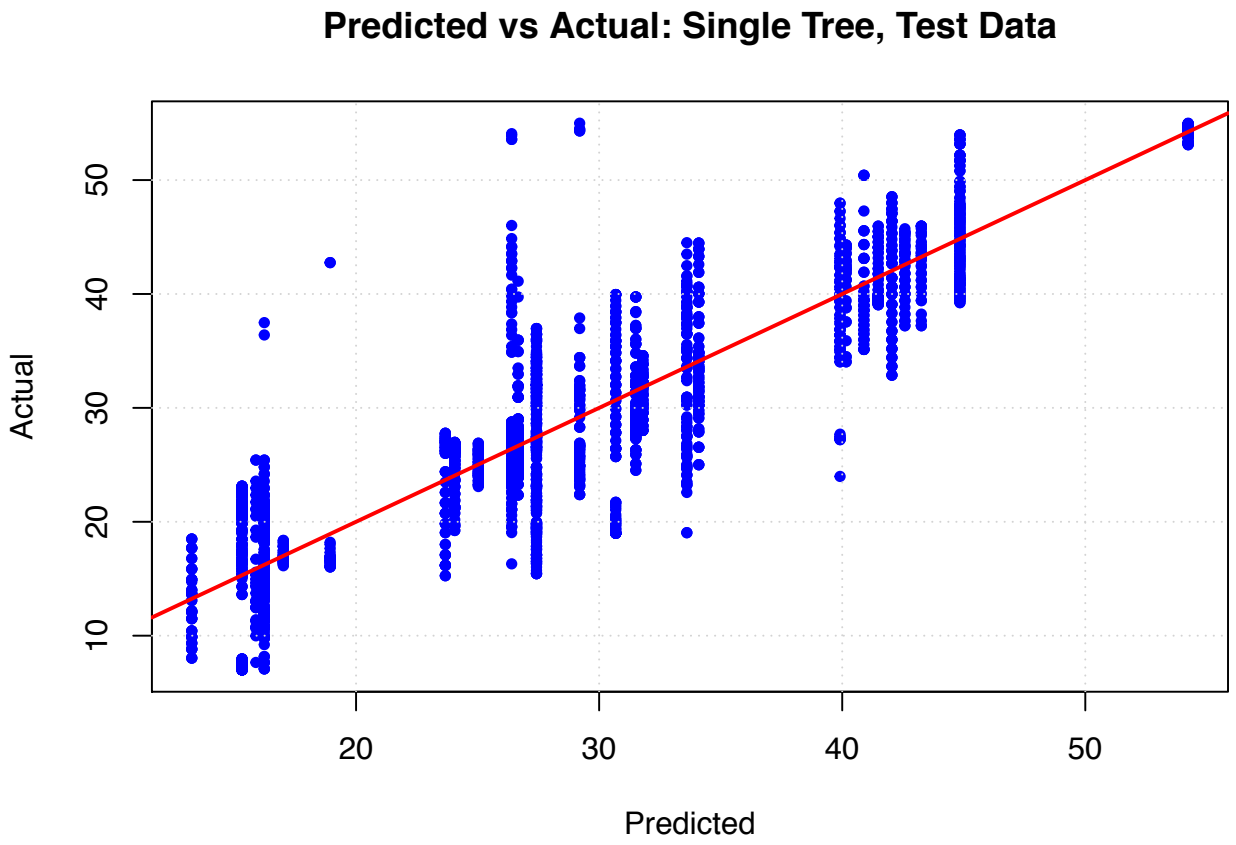
Splitting rules can be constructed in many different ways, all of which are based on the notion of impurity- a measure of the degree of heterogeneity of the leaf nodes. Put another way, a leaf node that contains a single class is homogeneous and has impurity=0. There are three popular impurity quantification methods: Entropy (aka information gain), Gini Index and Classification Error.

The rpart algorithm offers the entropy and Gini index methods as choices.

```
#Designing the tree
pd_tree = rpart(total_UPDRS ~ ., data = pd_trn)

#Predicting the tree
pd_tree_tst_pred = predict(pd_tree, newdata = pd_tst)


rpart.plot(pd_tree)
```




```
#plotting predicted and actual single tree
plot(pd_tree_tst_pred, pd_tst$total_UPDRS,
     xlab = "Predicted", ylab = "Actual",
     main = "Predicted vs Actual: Single Tree, Test Data",
     col = "blue", pch = 20)
grid()
abline(0, 1, col = "red", lwd = 2)
```

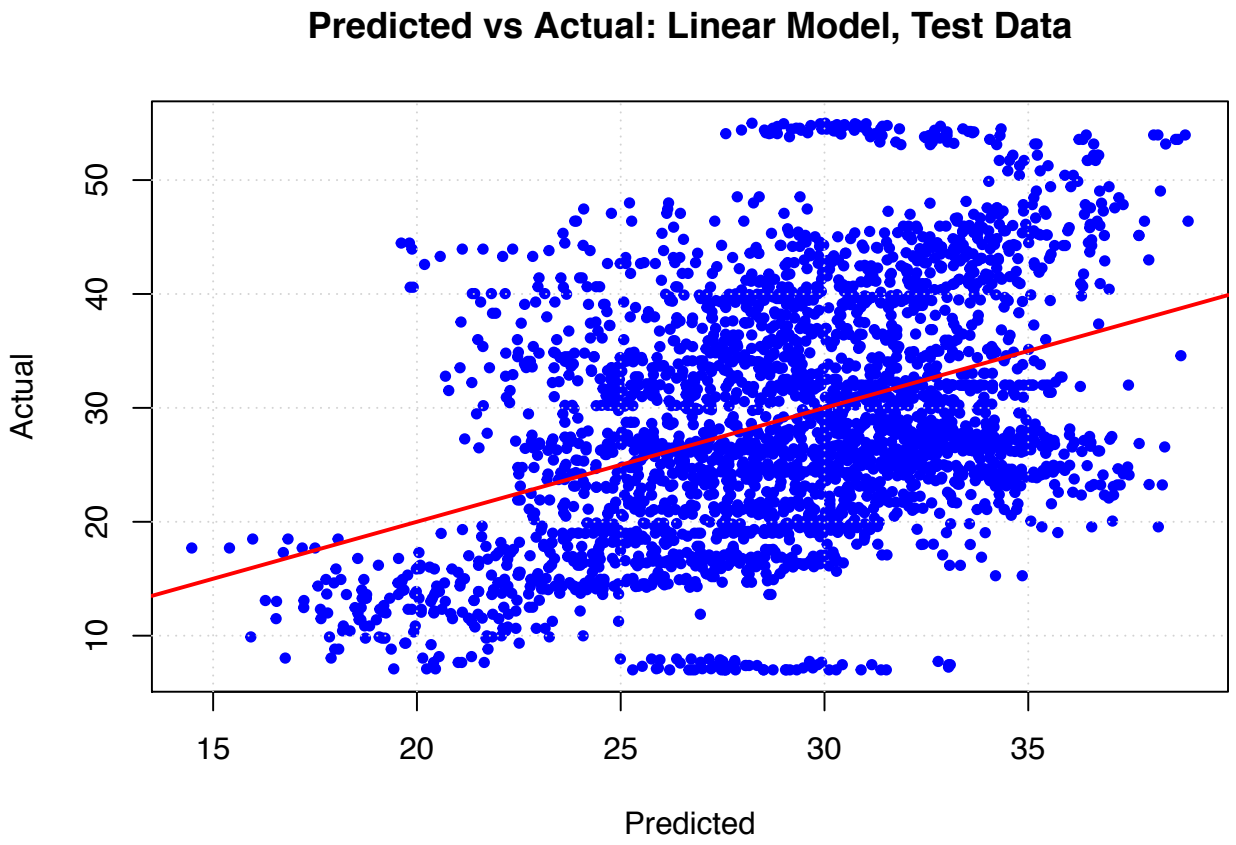



```
#calculating RMSE of rpart model
calc_rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
(tree_tst_rmse = calc_rmse(pd_tree_tst_pred, pd_tst$total_UPDRS))
```

```
## [1] 4.836149
```

## 8.4 Linear Model(lm)

```
#Designing the lm model
pd_lm = lm(total_UPDRS ~ ., data = pd_trn)

pd_lm_tst_pred = predict(pd_lm, newdata = pd_tst)

#plotting the lm model actual vs. predicted
plot(pd_lm_tst_pred, pd_tst$total_UPDRS,
     xlab = "Predicted", ylab = "Actual",
     main = "Predicted vs Actual: Linear Model, Test Data",
     col = "blue", pch = 20)
grid()
abline(0, 1, col = "red", lwd = 2)
```




```
(lm_tst_rmse = calc_rmse(pd_lm_tst_pred, pd_tst$total_UPDRS))
```

```
## [1] 9.622532
```

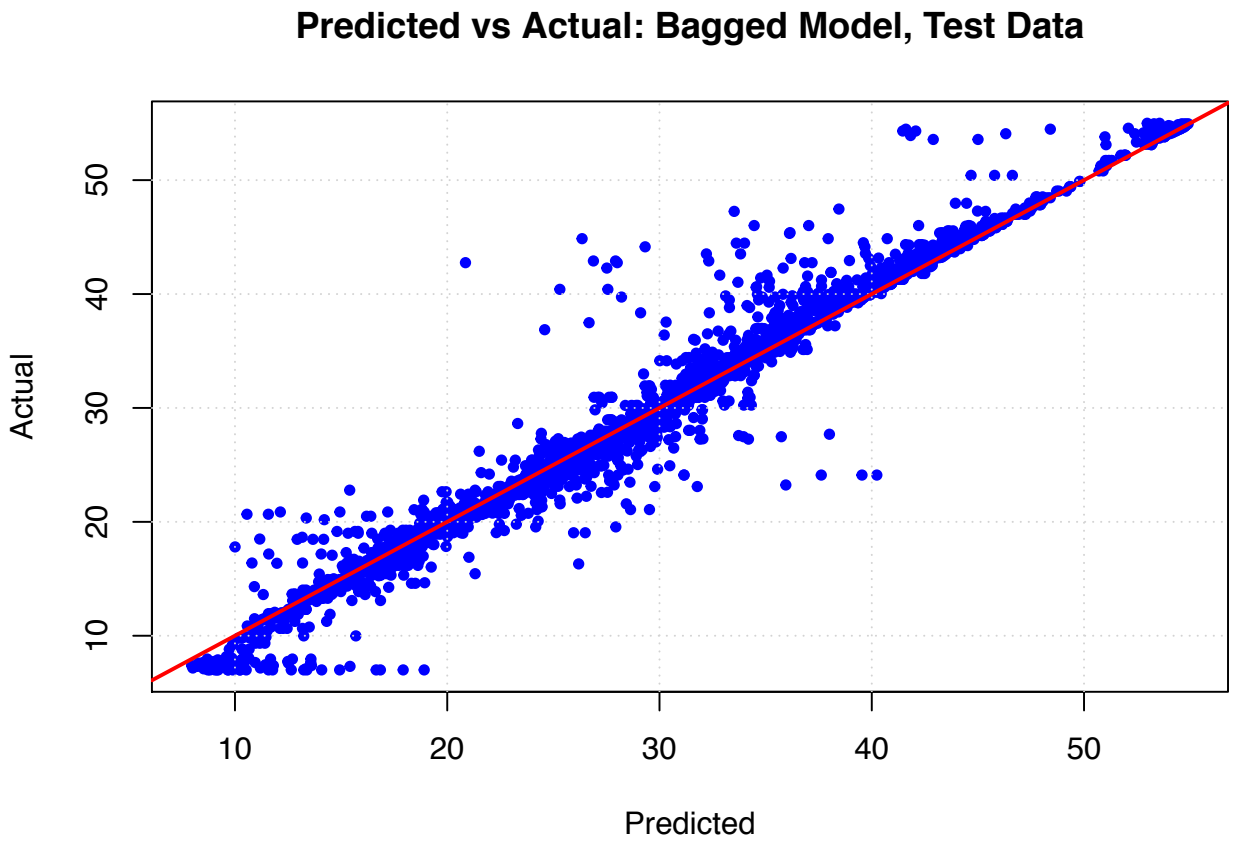
## 8.5 Bagging (rf)

We now fit a bagged model, using the randomForest package. Bagging is actually a special case of a random forest where mtry is equal to p, the number of predictors.

```
#Bagging the RF algorithm
pd_bag = randomForest(total_UPDRS ~ ., data = pd_trn, mtry = 10,
                          importance = TRUE, ntrees = 500)
pd_bag
```

```
##
## Call:
##  randomForest(formula = total_UPDRS ~ ., data = pd_trn, mtry = 10,      importance = TRUE
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 4.903923
##                     % Var explained: 95.78
```

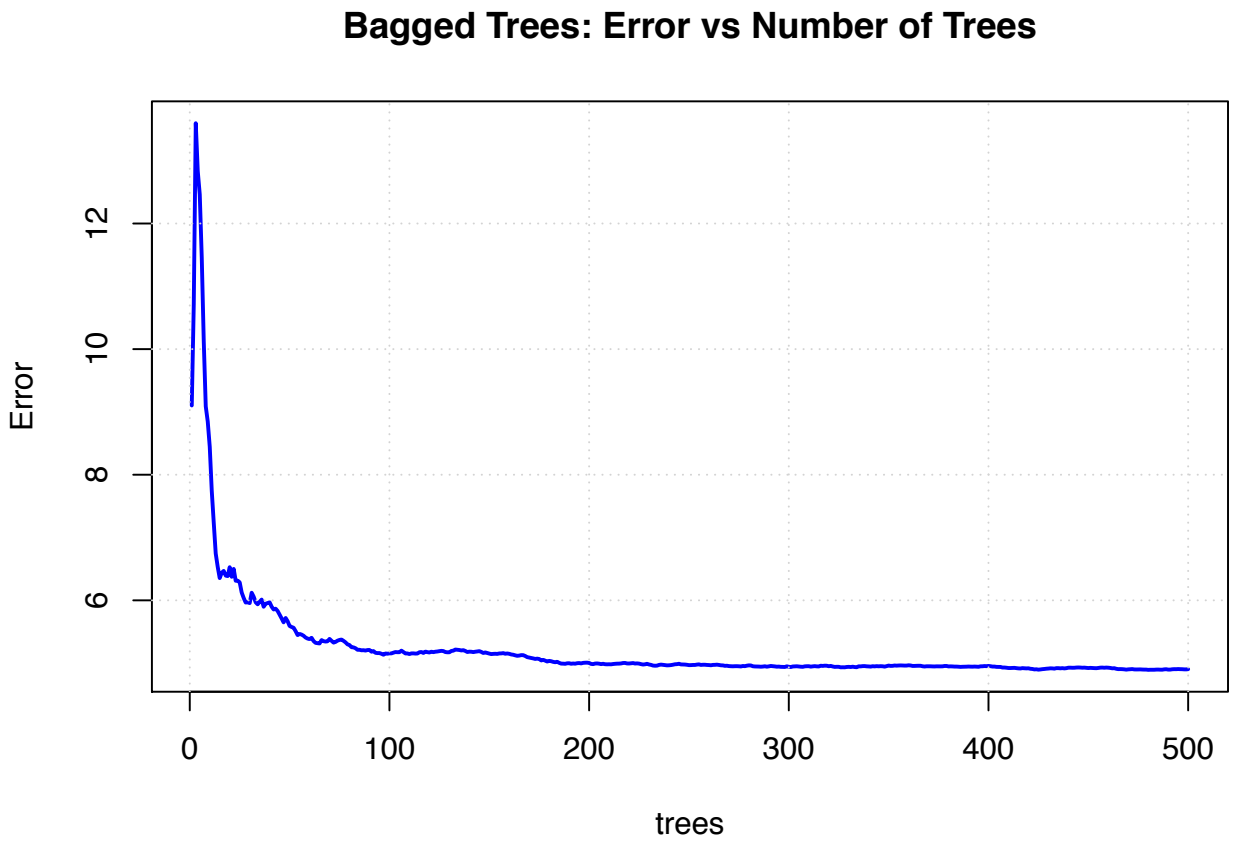
```r
#plotting the bagged model actual vs. predicted
pd_bag_tst_pred = predict(pd_bag, newdata = pd_tst)
plot(pd_bag_tst_pred,pd_tst$total_UPDRS,
     xlab = "Predicted", ylab = "Actual",
     main = "Predicted vs Actual: Bagged Model, Test Data",
     col = "blue", pch = 20)
grid()
abline(0, 1, col = "red", lwd = 2)
```




```r
(bag_tst_rmse = calc_rmse(pd_bag_tst_pred, pd_tst$total_UPDRS))
```

```
## [1] 2.145503
```

```r
#plotting the error of the trees
plot(pd_bag, col = "blue", lwd = 2, main = "Bagged Trees: Error vs Number of Trees")
grid()
```




Here we see two interesting results. First, the predicted versus actual plot no longer has a small number of predicted values. Second, our test error has dropped dramatically. Also note that the "Mean of squared residuals" which is output by randomForest is the Out of Bag estimate of the error.

## 8.6 Random Forest

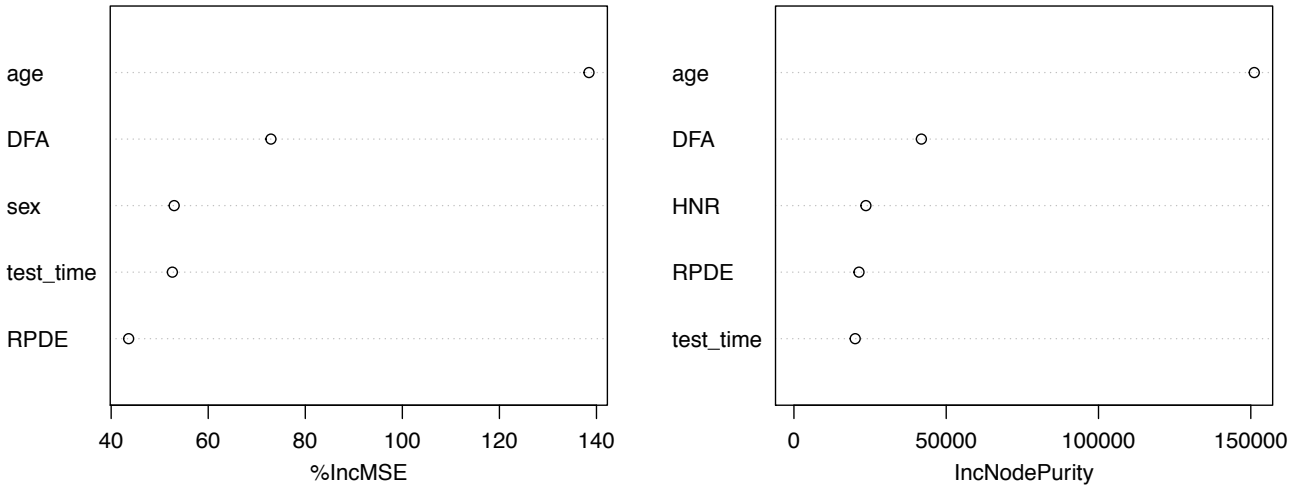
```
#Designing the Random Forest model
pd_forest = randomForest(total_UPDRS ~ ., data = pd_trn, mtry = 4,
                         importance = TRUE, ntrees = 500)
pd_forest
```

```
##
## Call:
##  randomForest(formula = total_UPDRS ~ ., data = pd_trn, mtry = 4,      importance = TRUE
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 14.59607
##                     % Var explained: 87.44
```

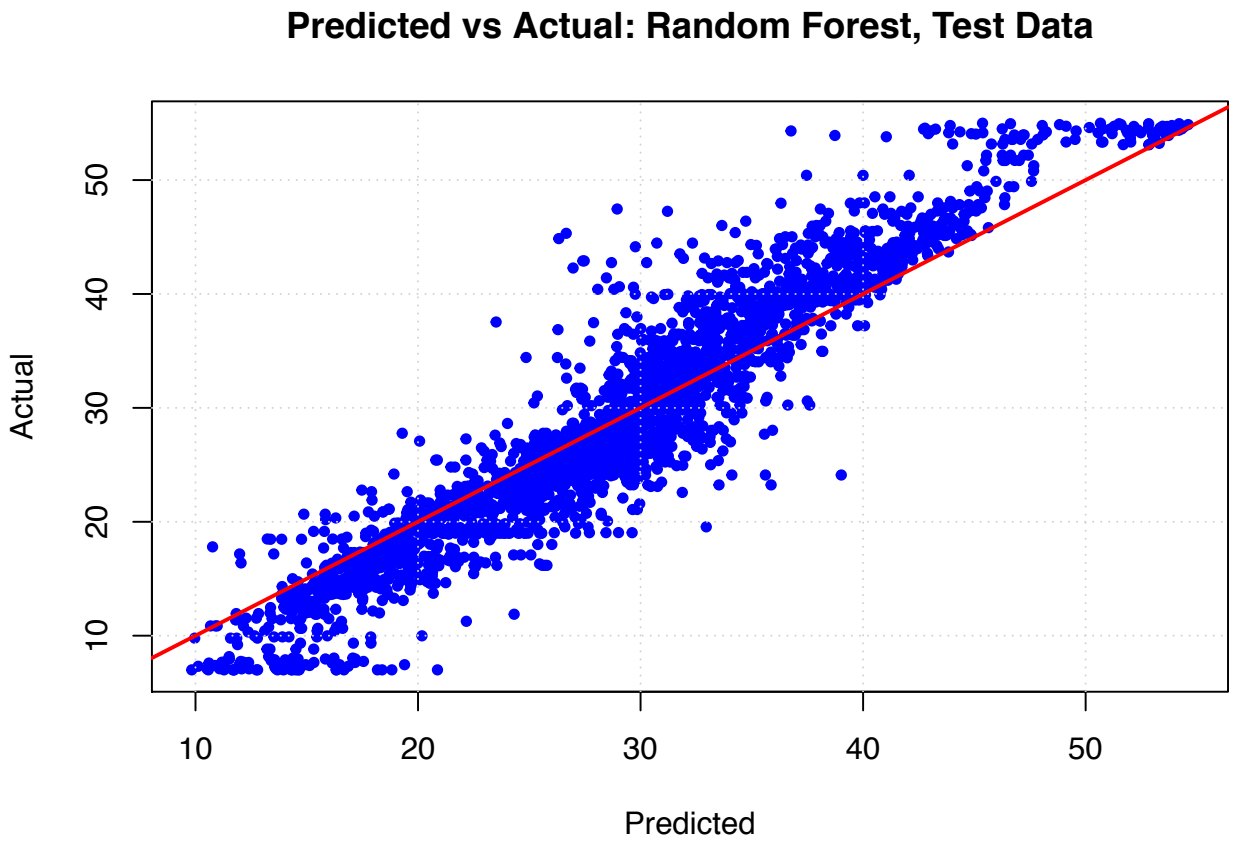
```
#Checking the most important variables
varImpPlot(pd_forest,
```

```
      sort = T,
      n.var = 5,
      main = "RF Top 5 - Variable Importance")
```

RF Top 5 – Variable Importance




```
#Random forest plot actual versus predicted values
pd_forest_tst_pred = predict(pd_forest, newdata = pd_tst)
plot(pd_forest_tst_pred, pd_tst$total_UPDRS,
     xlab = "Predicted", ylab = "Actual",
     main = "Predicted vs Actual: Random Forest, Test Data",
     col = "blue", pch = 20)
grid()
abline(0, 1, col = "red", lwd = 2)
```




```
#Calculating the RMSE of RF
(forest_tst_rmse = calc_rmse(pd_forest_tst_pred, pd_tst$total_UPDRS))
```

```
## [1] 3.735826
```

Here we note three RMSE's. The training RMSE (which is optimistic), the OOB RMSE (which is a reasonable estimate of the test error) and the test RMSE. Also note that variables importance was calculated.

```
pd_forest_trn_pred = predict(pd_forest, newdata = pd_trn)
forest_trn_rmse = calc_rmse(pd_forest_trn_pred, pd_trn$total_UPDRS)
forest_oob_rmse = calc_rmse(pd_forest$predicted, pd_trn$total_UPDRS)
```

## 8.7 Boosting with gbm

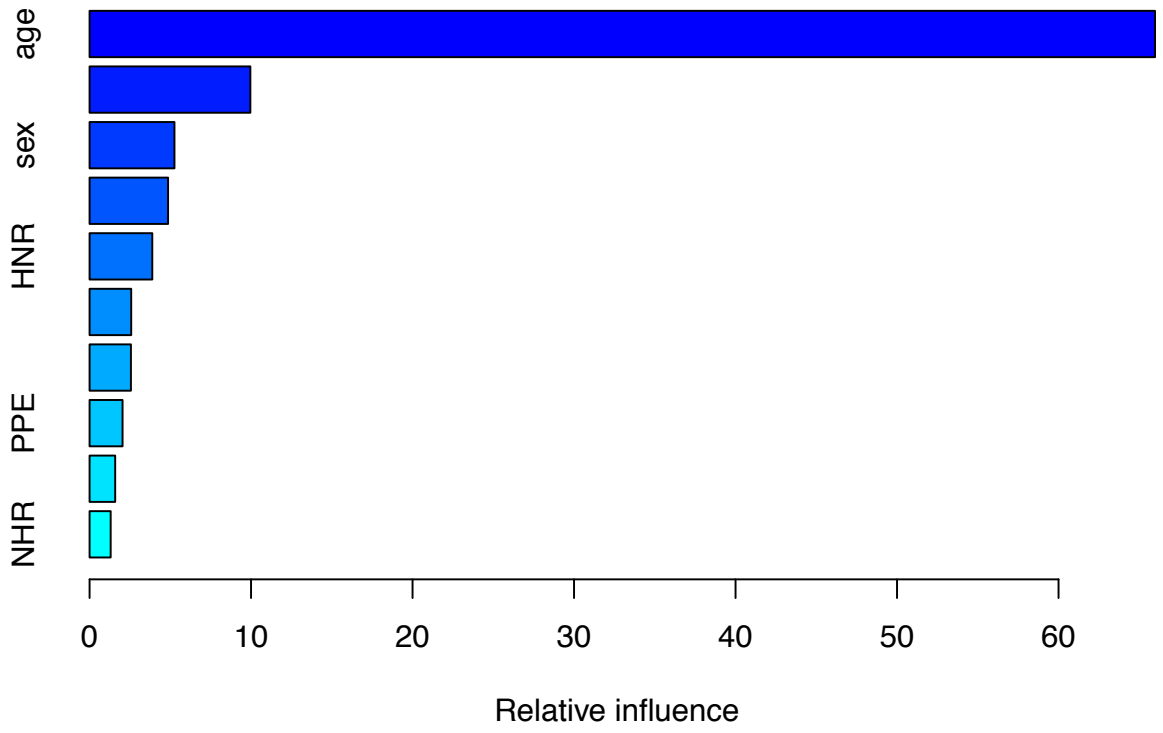
Lastly, we try a boosted model, which by default will produce a nice variable importance plot as well as plots of the marginal effects of the predictors. We use the gbm package.

```
pd_boost = gbm(total_UPDRS ~ ., data = pd_trn, distribution = "gaussian",
                    n.trees = 5000, interaction.depth = 4, shrinkage = 0.01)
pd_boost
```

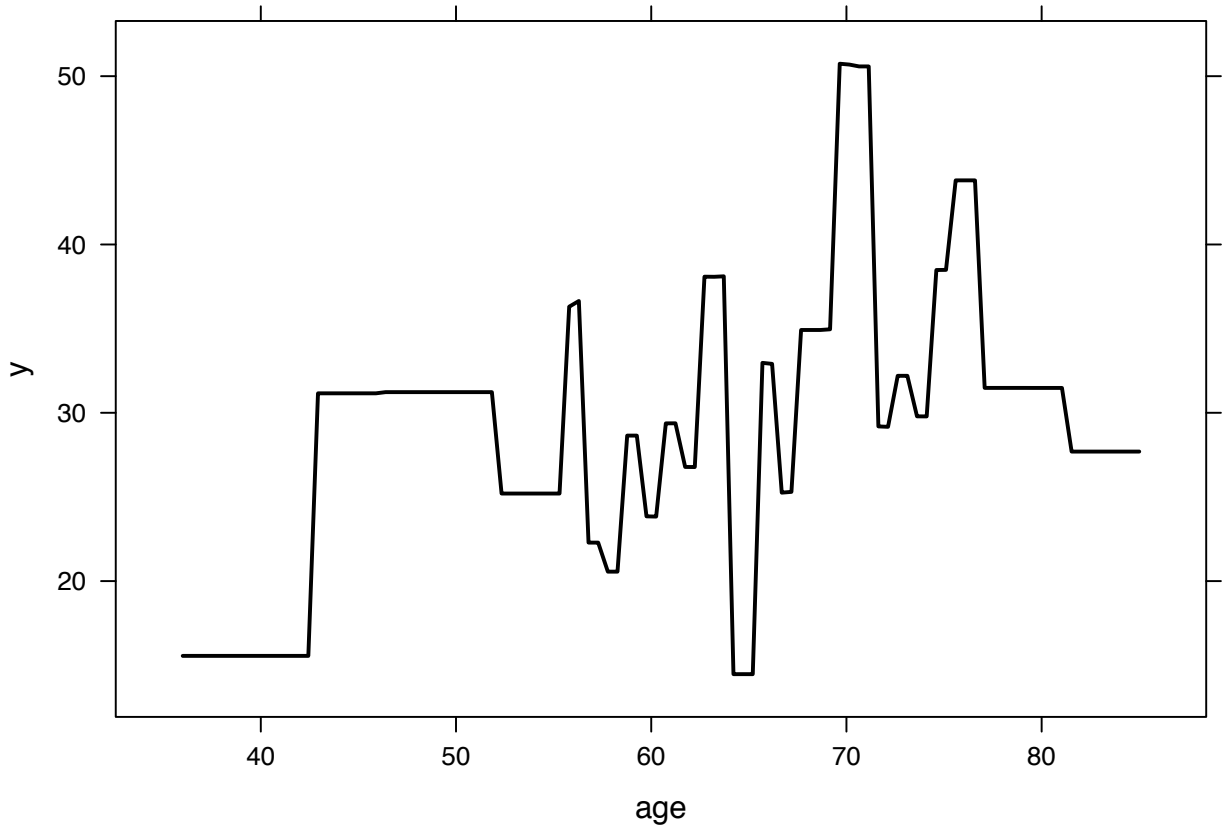
```
## gbm(formula = total_UPDRS ~ ., distribution = "gaussian", data = pd_trn,
##     n.trees = 5000, interaction.depth = 4, shrinkage = 0.01)
```

```
## A gradient boosted model with gaussian loss function.
## 5000 iterations were performed.
## There were 10 predictors of which 10 had non-zero influence.
```

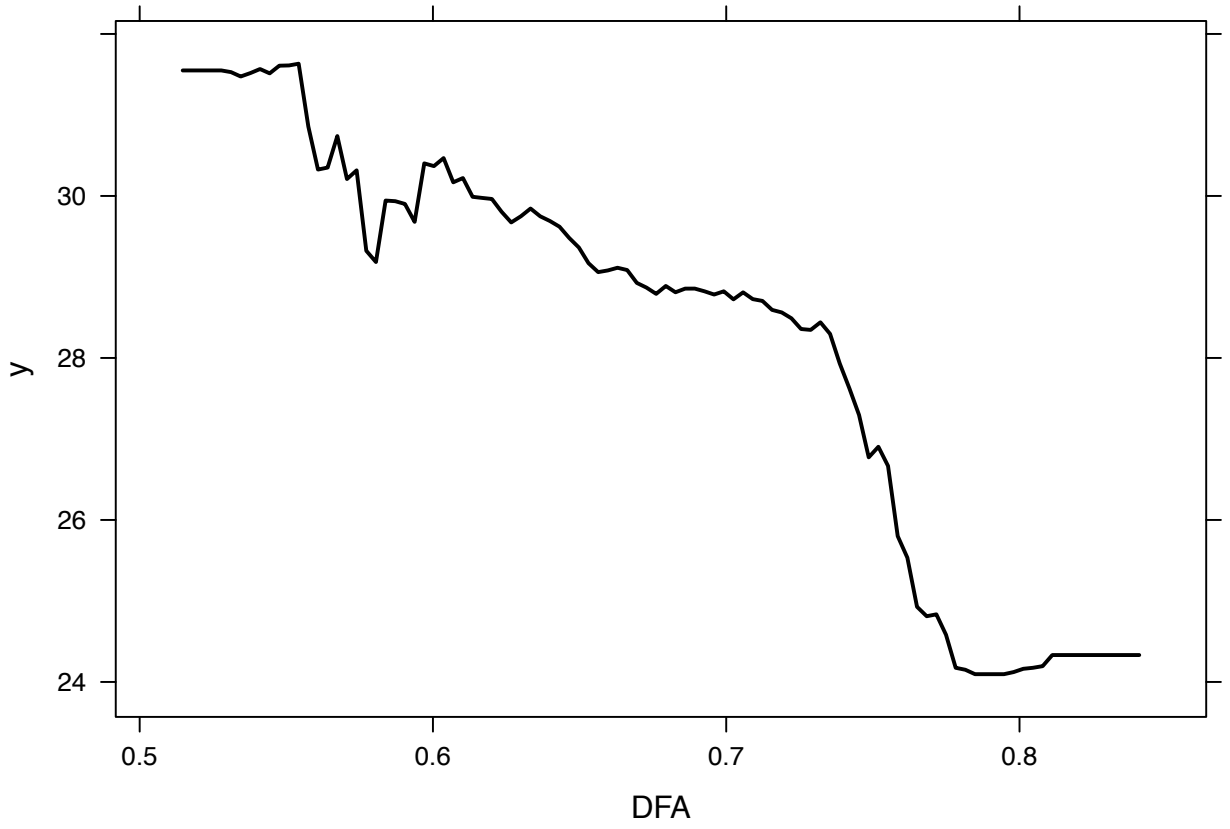
```
tibble::as_tibble(summary(pd_boost))
```



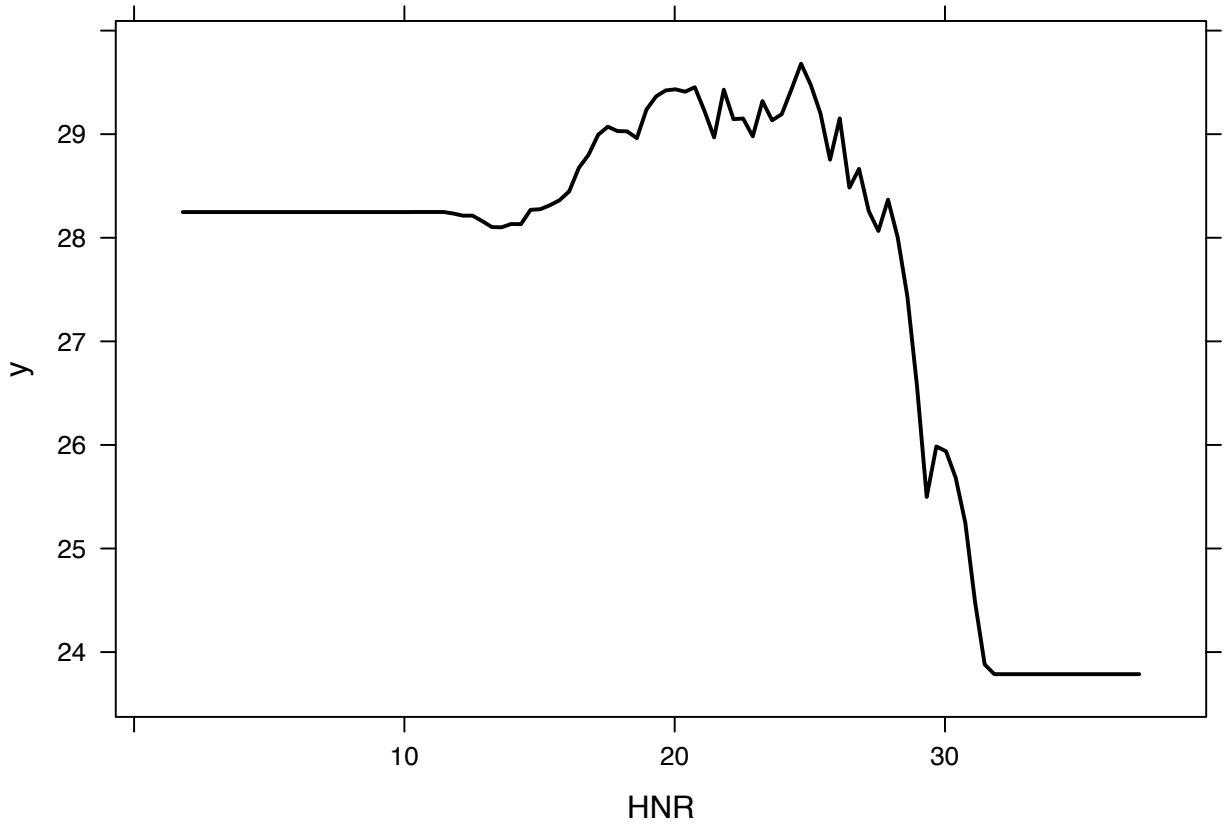

```
par(mfrow = c(1, 3))
plot(pd_boost, i = "age", col = "blue", lwd = 2)
```




```
plot(pd_boost, i = "DFA", col = "blue", lwd = 2)
```




```
plot(pd_boost, i = "HNR", col = "blue", lwd = 2)
```

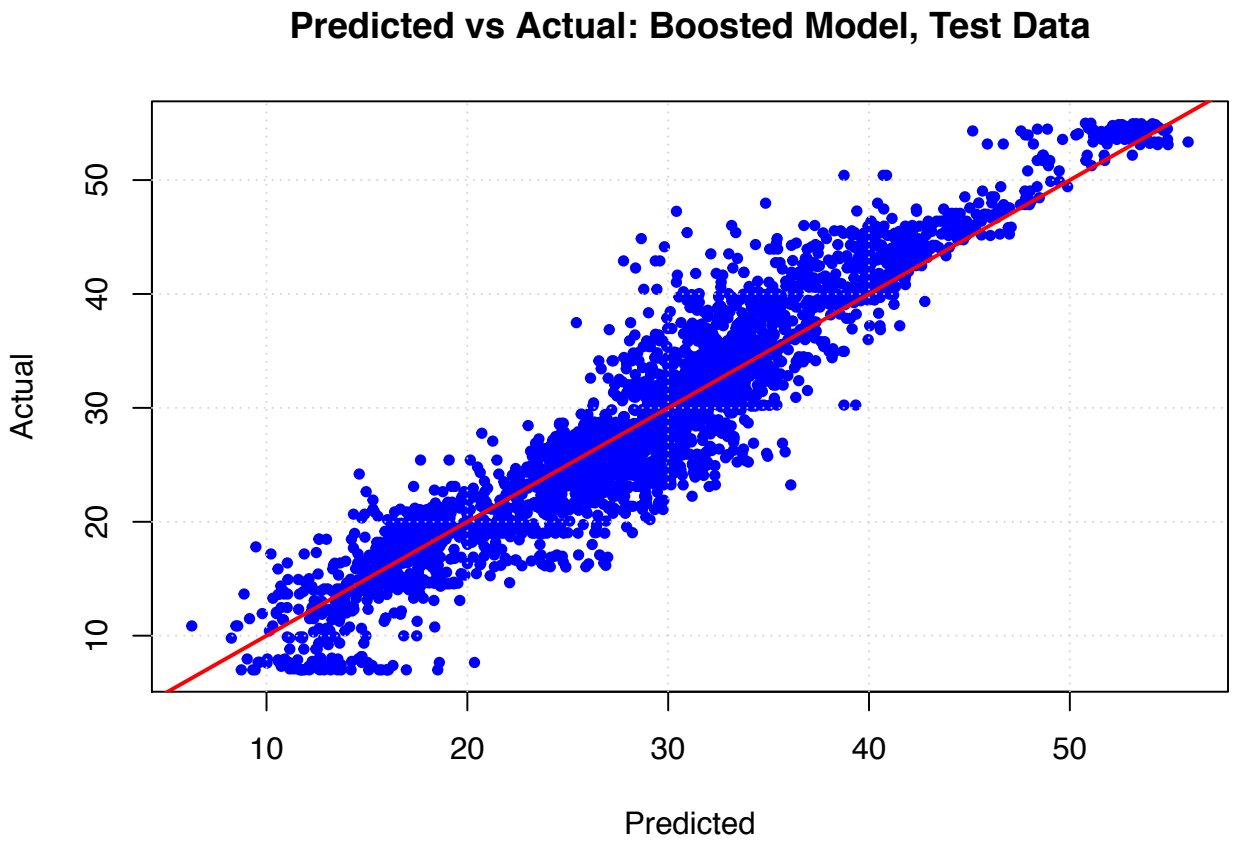



```
pd_boost_tst_pred = predict(pd_boost, newdata = pd_tst, n.trees = 5000)
```

```
(boost_tst_rmse = calc_rmse(pd_boost_tst_pred, pd_tst$total_UPDRS))
```

```
## [1] 3.501507
```

```
plot(pd_boost_tst_pred, pd_tst$total_UPDRS,
     xlab = "Predicted", ylab = "Actual",
     main = "Predicted vs Actual: Boosted Model, Test Data",
     col = "blue", pch = 20)
grid()
abline(0, 1, col = "red", lwd = 2)
```




## 8.8 Summary regression

```
#Resume of the testerrors RMSE
(
  pd_rmse = data.frame(
  Model = c("Single Tree", "Linear Model", "Bagging",  "Random Forest",  "Boosting"),
  TestError = c(tree_tst_rmse, lm_tst_rmse, bag_tst_rmse, forest_tst_rmse, boost_tst_rmse)
)
)

print(pd_rmse)
```

```
##           Model TestError
## 1   Single Tree  4.836149
## 2  Linear Model  9.622532
## 3       Bagging  2.145503
## 4 Random Forest  3.735826
## 5      Boosting  3.501507
```

As we can see the boosted model does no better than the bagged model which has a RMSE of 2.14.

The charts of actual and predicted results shows some clear patterns in the different algorithms and their behavior or performance. The linear model for example in that chart it is relatively easy to see

that there is large distance from our model line than it is on the bagged model.

In this case each of the ensemble methods perform much better than the Linear model!

# Chapter 9

# Classification

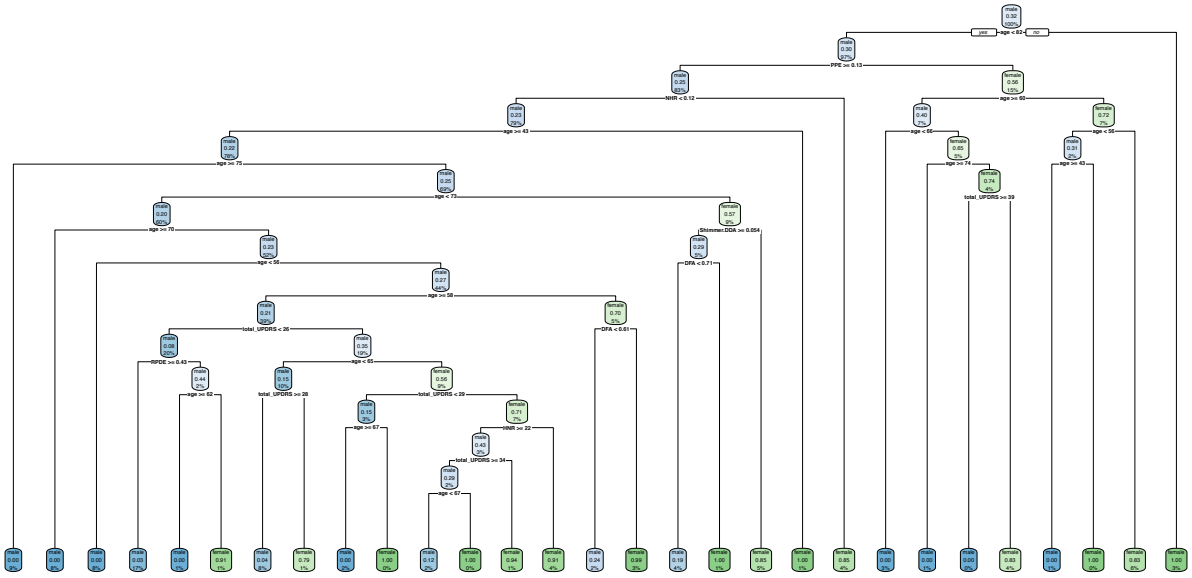
In this classification challenge the predictor will show to outcomes - 0=male, 1= female.

For the evaluation i will use the accuracy as metric.

Let us start with the tree model from the rpart package.

## 9.1 Tree model (rpart)

```
#setting the accumulation function
calc_acc = function(actual, predicted) {
  mean(actual == predicted)
}

#designing the rpart tree
pdc_tree = rpart(sex ~ ., data = pd_trn)

#plotting the tree
rpart.plot(pdc_tree)
```



```
#Predicting the tree
pdc_tree_tst_pred = predict(pdc_tree, pd_tst, type = "class")
table(predicted = pdc_tree_tst_pred, actual = pd_tst$sex)
```

```
##           actual
## predicted male female
##    male   1885     85
##    female  124    844
```

```
#Calculating the accuracy
(tree_tst_acc = calc_acc(predicted = pdc_tree_tst_pred, actual = pd_tst$sex))
```

```
## [1] 0.9288632
```

## 9.2 Logistic regression (glm)

```
#Designing the glm
pdc_glm = glm(sex ~ ., data = pd_trn, family = "binomial")
pdc_glm
```

```
##
## Call:  glm(formula = sex ~ ., family = "binomial", data = pd_trn)
##
## Coefficients:
## (Intercept)          age    test_time  total_UPDRS   Jitter.DDP  Shimmer.DDA
##    8.111815     0.014529     0.001217    -0.020886    16.419778   -12.755383
##         NHR          HNR         RPDE          DFA          PPE
##   23.660742    -0.144128    -6.469868    -2.042826    -7.650955
##
## Degrees of Freedom: 2936 Total (i.e. Null);  2926 Residual
## Null Deviance:       3679
```

```
## Residual Deviance: 3223  AIC: 3245
```

```
pdc_glm_tst_pred = ifelse(predict(pdc_glm, pd_tst, "response") > 0.5,
                          "female", "male")
table(predicted = pdc_glm_tst_pred, actual = pd_tst$sex)
```

```
##           actual
## predicted male female
##    female  103    266
##    male   1906    663
```

```
(glm_tst_acc = calc_acc(predicted = pdc_glm_tst_pred, actual = pd_tst$sex))
```

```
## [1] 0.7392784
```

## 9.3 Bagging (rf)

```
# Bagging calculation
pdc_bag = randomForest(sex ~ ., data = pd_trn, mtry = 10,
                       importance = TRUE, ntrees = 500)
pdc_bag
```

```
##
## Call:
##  randomForest(formula = sex ~ ., data = pd_trn, mtry = 10, importance = TRUE,      ntrees
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 1.7%
## Confusion matrix:
##        male female class.error
## male   1976     23  0.01150575
## female   27    911  0.02878465
```

```
#Predicting the bag
pdc_bag_tst_pred = predict(pdc_bag, newdata = pd_tst)
table(predicted = pdc_bag_tst_pred, actual = pd_tst$sex)
```

```
##           actual
## predicted male female
##    male   1981     40
##    female   28    889
```

```
(bag_tst_acc = calc_acc(predicted = pdc_bag_tst_pred, actual = pd_tst$sex))
```

```
## [1] 0.976855
```

## 9.4 Random Forest

For classification, the suggested mtry for a random forest is
$\sqrt{p}$.

```
# Designing the RF
pdc_forest = randomForest(sex ~ ., data = pd_trn, mtry = 3, importance = TRUE, ntrees = 500)
pdc_forest
```

```
##
## Call:
##  randomForest(formula = sex ~ ., data = pd_trn, mtry = 3, importance = TRUE,      ntrees
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 3.17%
## Confusion matrix:
##        male female class.error
## male   1979     20  0.01000500
## female   73    865  0.07782516
```

```
#Predicting the forest
pdc_forest_tst_perd = predict(pdc_forest, newdata = pd_tst)
table(predicted = pdc_forest_tst_perd, actual = pd_tst$sex)
```

```
##          actual
## predicted male female
##    male   1991     96
##    female   18    833
```

```
(forest_tst_acc = calc_acc(predicted = pdc_forest_tst_perd, actual = pd_tst$sex))
```

```
## [1] 0.9611981
```

Here we see each of the ensemble methods performing better than a single tree, however, they still fall behind logistic regression. Sometimes a simple linear model will beat more complicated models! This is why you should always try a logistic regression for classification.

## 9.5 Boosting with gbm

To perform boosting, we modify the response to be 0 and 1 to work with gbm.

```
# A boosted gbm model
pdc_trn_mod = pd_trn
pdc_trn_mod$sex = as.numeric(ifelse(pdc_trn_mod$sex == "female", "0", "1"))

pdc_boost = gbm(sex ~ ., data = pdc_trn_mod, distribution = "bernoulli",
                n.trees = 5000, interaction.depth = 4, shrinkage = 0.01)
pdc_boost
```

```
## gbm(formula = sex ~ ., distribution = "bernoulli", data = pdc_trn_mod,
```

```
##     n.trees = 5000, interaction.depth = 4, shrinkage = 0.01)
## A gradient boosted model with bernoulli loss function.
## 5000 iterations were performed.
## There were 10 predictors of which 10 had non-zero influence.
```

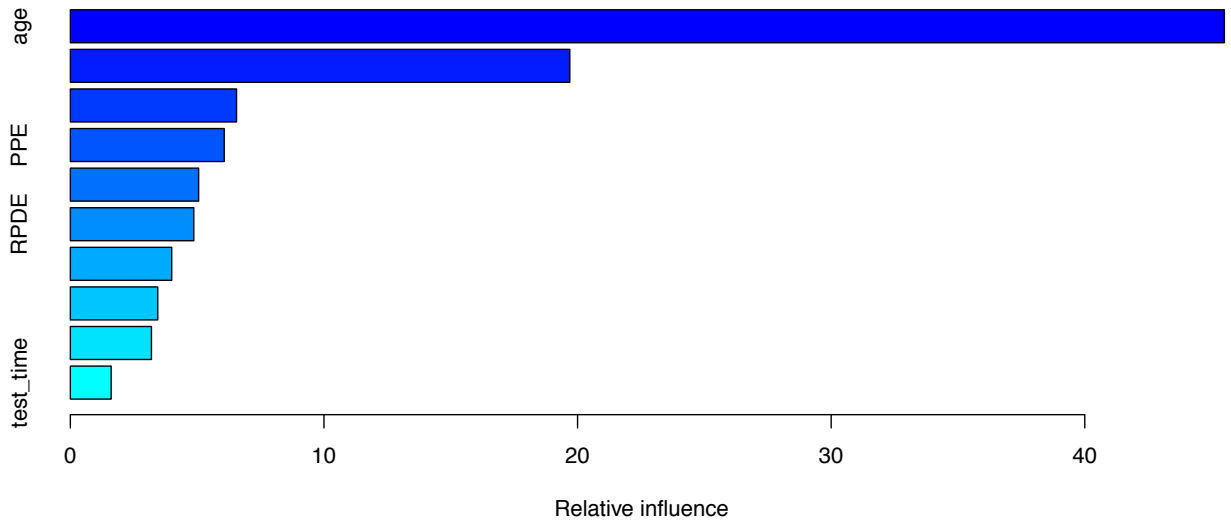
```
#Predicting the boosted gbm
pdc_boost_tst_pred = ifelse(predict(pdc_boost, pd_tst, n.trees = 5000, "response") > 0.5,
                            "male", "female")
table(predicted = pdc_boost_tst_pred, actual = pd_tst$sex)
```

```
##           actual
## predicted male female
##    female    5    912
##    male   2004     17
```

```
(boost_tst_acc = calc_acc(predicted = pdc_boost_tst_pred, actual = pd_tst$sex))
```

```
## [1] 0.9925119
```

```
tibble::as_tibble(summary(pdc_boost))
```




```
# Collecting the accuracies
(pdc_acc = data.frame(
  Model = c("Single Tree", "Logistic Regression", "Bagging",  "Random Forest",  "Boosting"),
  TestAccuracy = c(tree_tst_acc, glm_tst_acc, bag_tst_acc, forest_tst_acc, boost_tst_acc)
)
)
```

```
print(pdc_acc)
```

```
##                 Model TestAccuracy
## 1         Single Tree    0.9288632
## 2 Logistic Regression    0.7392784
```

```
## 3             Bagging    0.9768550
## 4       Random Forest    0.9611981
## 5            Boosting    0.9925119
```

## 9.6 Tuning - Cross Validation

So far we fit bagging, boosting and random forest models, but I did not tune any of them, I simply used certain or the default, somewhat arbitrary, parameters. Now we will see how to modify the tuning parameters to make these models better.

Bagging: Actually just a subset of Random Forest with mtry =
p . Random Forest: mtry Boosting: n.trees, interaction.depth, shrinkage,n.minobsinnode

Also note that with these tree-based ensemble methods there are two resampling solutions for tuning the model:

Out of Bag Cross-Validation Using Out of Bag samples is advantageous with these methods as compared to Cross-Validation since it removes the need to refit the model and is thus much more computationally efficient. Unfortunately OOB methods cannot be used with gbm models.

### 9.6.1 Random Forest and bagging

In this section i will perform for both OOB and cross-validation methods.

oob = trainControl(method = "oob") cv_5 = trainControl(method = "cv", number = 5)

To tune a Random Forest in caret we will use method = "rf" which uses the randomForest function in the background. Here we elect to use the OOB training control that we created. We could also use cross-validation, however it will likely select a similar model, but require much more time.

We setup a grid of mtry values which include all possible values since there are 10 predictors in the dataset. An mtry of 10 is bagging.

```
#Tuning
# tuning with caret
oob = trainControl(method = "oob")
cv_5 = trainControl(method = "cv", number = 5)
```

```
dim(pd_trn)
```

```
## [1] 2937   11
```

```
rf_grid =  expand.grid(mtry = 1:10)
library(caret)
set.seed(825)
pdc_rf_tune = train(sex ~ ., data = pd_trn,
                    method = "rf",
                    trControl = oob,
                    verbose = FALSE,
                    tuneGrid = rf_grid)
pdc_rf_tune
```

```
## Random Forest
##
## 2937 samples
##   10 predictor
##    2 classes: 'male', 'female'
##
## No pre-processing
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.9267961  0.8239989
##    2    0.9543752  0.8924999
##    3    0.9662921  0.9211949
##    4    0.9761662  0.9446769
##    5    0.9829758  0.9606185
##    6    0.9812734  0.9567664
##    7    0.9836568  0.9623010
##    8    0.9833163  0.9615483
##    9    0.9839973  0.9632012
##   10    0.9836568  0.9623651
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 9.
```

```
best_tun_mod <- calc_acc(predict(pdc_rf_tune, pd_tst), pd_tst$sex)
(tuned_tst_acc = calc_acc(predict(pdc_rf_tune, pd_tst), pd_tst$sex))
```

```
## [1] 0.9785568
```

```
pdc_rf_tune$bestTune
```

Based on these results, we would select the random forest model with an mtry of 9. Note that based on the OOB estimates.

### 9.6.2 Boosting

We now tune a boosted tree model. We will use the cross-validation tune control setup above. We will fit the model using gbm with caret.

Setting up the tuning grid:

- **interaction.depth**: How many splits to use with each tree.
- **n.trees**: The number of trees to use.
- **shrinkage**: The shrinkage parameters, which controls how fast the method learns.
- **n.minobsinnode**: The minimum number of observations in a node of the tree.

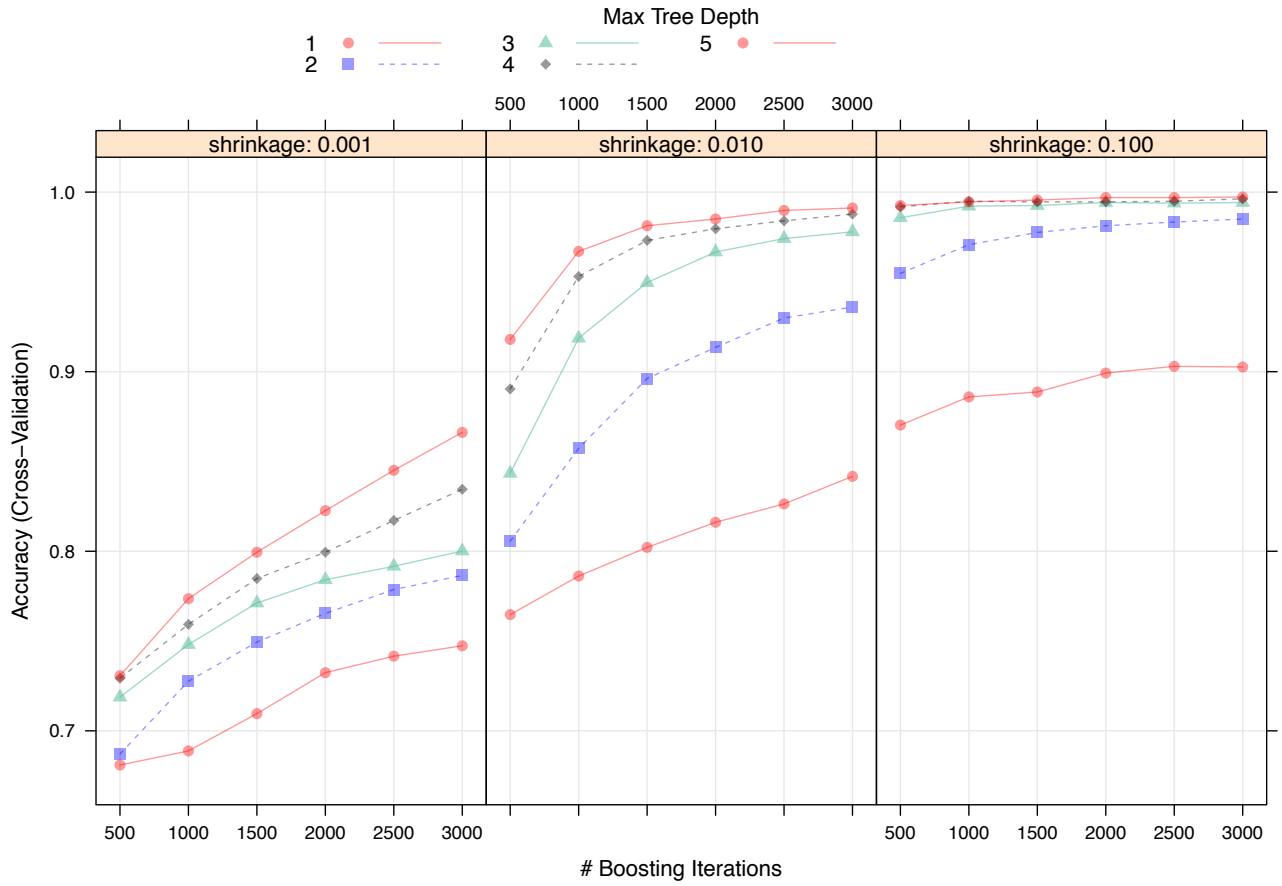
```
## [1] 0.9965963
```

```
## [1] 0.9965963
```

```
#calculating the accuracy
(boost_tst_acc = calc_acc(predicted = pdc_boost_tst_pred, actual = pd_tst$sex))
```

```
## [1] 0.9925119
```

```
#plotting the tuned tree
plot(pdc_gbm_tune)
```




Notice that the tuned model does more accurately on the test set than the arbitrary boosted model we had fit above, with the marginally different parameters. We could perhaps try a larger tuning grid, but at this point it seems unlikely that we could find a much more fitting model.

```
print(pdc_gbm_tune$bestTune)
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 90    3000                 5       0.1             10
```

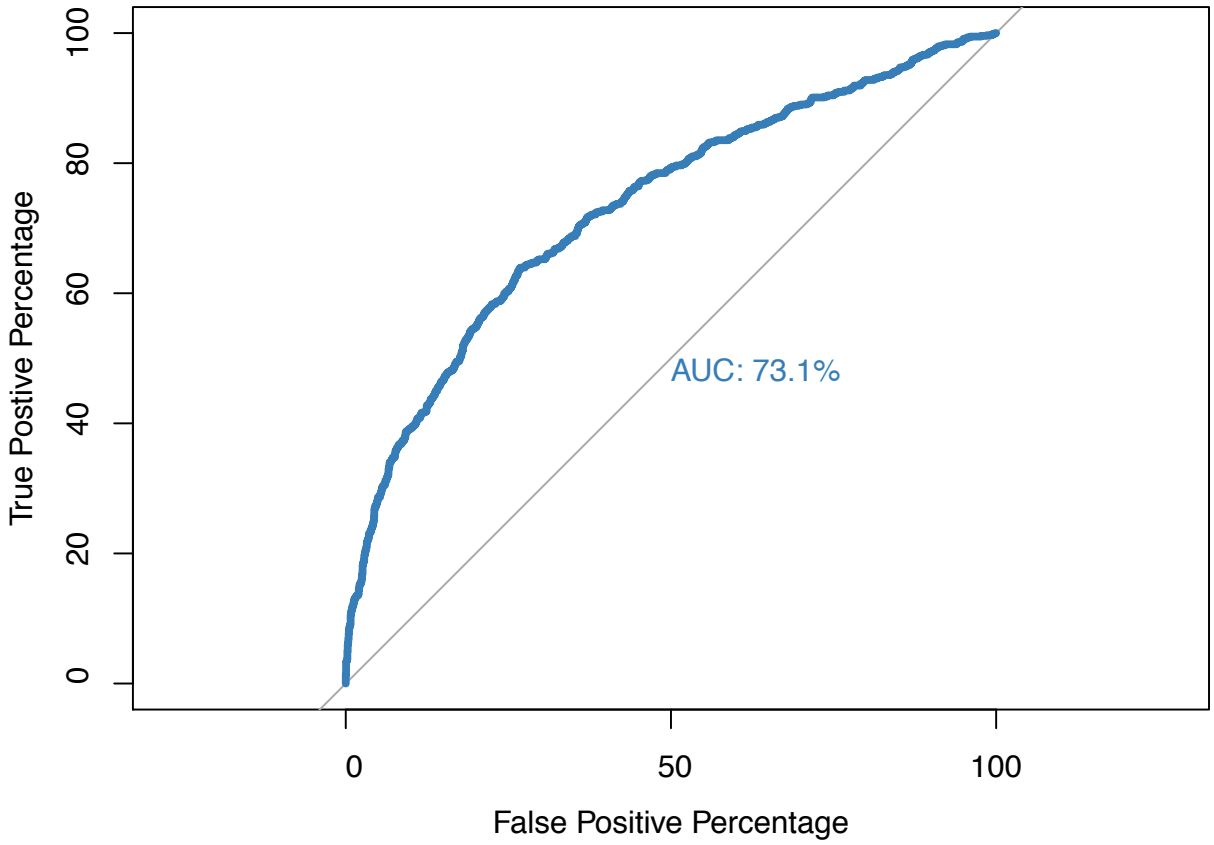
## 9.7 ROC curve

Receiver operating characteristics (ROC) visual representations are useful for organizing classifiers and visualizing their performance. ROC graphs are frequently used in medical decision making, and in recent years have been used progressively in machine learning and data mining research. Although ROC graphs are apparently straightforward, there are some common misinterpretations and difficulties when using them in practice.

```
# Calculating the ROC boosted tuned model
test_prob = predict(pdc_glm, newdata = pd_tst, type = "response")
test_roc = roc(pd_tst$sex ~ test_prob, plot = TRUE, print.auc = TRUE, legacy.axes=TRUE, perc
```

```
## Setting levels: control = male, case = female
```

```
## Setting direction: controls < cases
```




Instead of manually checking cutoffs, we can create an ROC curve, which will sweep through all possible cutoffs, and plot the sensitivity and specificity.

A good model will have a high AUC, that is as often as possible a high sensitivity and specificity. The confusion matrix's are placed under the different sections related to the algorithms.

Some types of errors in the classification process are very important to understand and I will define than below.

**Type I and Type II errors**

*Type I error*, also known as a "false positive": the error of rejecting a null hypothesis when it is actually true. In other words, this is the error of accepting an alternative hypothesis (the real hypothesis of interest) when the results can be attributed to chance. Plainly speaking, it occurs when we are observing a difference when in truth there is none (or more specifically - no statistically significant difference).

*Type II error*, also known as a "false negative": the error of not rejecting a null hypothesis when the alternative hypothesis is the true state of nature. In other words, this is the error of failing to accept an alternative hypothesis when you don't have adequate power. Plainly speaking, it occurs when we are failing to observe a difference when in truth there is one.

## 9.8 Summary classification

```
#Resume of the results
(pdc_acc = data.frame(
  Model = c("Single Tree", "Logistic Regression", "Bagging",  "Random Forest",  "Boosting",
  TestAccuracy = c(tree_tst_acc, glm_tst_acc, bag_tst_acc, forest_tst_acc, boost_tst_acc, tu
)
)
```

```
print(pdc_acc)
```

```
##                  Model TestAccuracy
## 1          Single Tree    0.9288632
## 2 Logistic Regression    0.7392784
## 3              Bagging    0.9768550
## 4        Random Forest    0.9611981
## 5             Boosting    0.9925119
## 6          Tuned model    0.9785568
## 7 Boosted tuned Model    0.9965963
```

In the algorithm design phase it is vital to reduce these errors to improve accuracy and if we take a look on the analysis performed it is clear that all algorithms do well except the logistic regression. All models reached above 90% accuracy and will predict well, but in this case the tuned boosted model was the winner with 99.65% accuracy.

In my appendix I show an visual example what the boosted algorithm actually does compared to a single tree. It expand and widens the "horizon" and make the decision boundaries optimal to improve the predictions of the outcomes, in this case male or female.

In the process i also tried to use the superlearner and ranger packages, but i rejected it for this project, because the results might seem like a black box to understand and subsequently explain - therefore chose these methods as the probability of learning and understanding connections between stages in the analysis is far better for me.

```
knitr::purl("merging.Rmd", documentation = 0)
```

# Chapter 10

# Conclusion/results

The research i have made on this project analyzing 5800 instances with 23 variables. After cleaning the data and made a 0.9 cutoff in the correlation, i reached the following results and conclusions:

- The Regression reached the lowest RMSE was the bagged model.
- The tuned boosted gbm reached the highest accuracy

The **regression** analysis of severity (total_UPDRS) concluded that the bagged model was the final model because it had the lowest RMSE of the 5 models that was used to analyze the dataset. The age of the patient was the most important factor of the 10 predictors.

The **classification** issue predicting the gender of patient giving the 10 predictors the investigation could be reduced because one will get a good performance with reduced knowledge and the public could reduce the measurements i.e. use minor resources to reach the same conclusions.

**My take-out from the course** The knowledge that i have gained before this course - was started in 1990 with statistics with pen and paper, so I have refreshed and updated my statistics, but most importantly - R - statistics calculations, R Markdown and the power of the new technology, i can certainly relate to now.

It has been a very interesting journey from my perspective. The core for me is that it is very important to improve the communication of knowledge, the visual part. That's a science. Its far more important than the quantity of charts that you may have in one report. With the present analysis it could be argued that this conclusion is robust.

**Future work** To further improve upon the performance of a regression and classification models for this area, one could include other models and investigate their performances. In addition, an ensemble of multiple models could be tried to see if any meaningful improvements could be gained.

Finally I would like to mention that the dataset is large and complex and to improve the output for further improving the detection of PD the public should improved the quality of the dataset.

It think using resources to improve the quality in dialog with machine learning artist will reach far more benefits than making dataset larger or optimizing algorithms.

The focal point should not be Big Data, but Good Data.

Jan Thomsen

Denmark, 2021

# Chapter 11

# Appendix

## 11.1 Tree versus boosted boundaries

MLBench: Distributed Machine Learning Benchmark

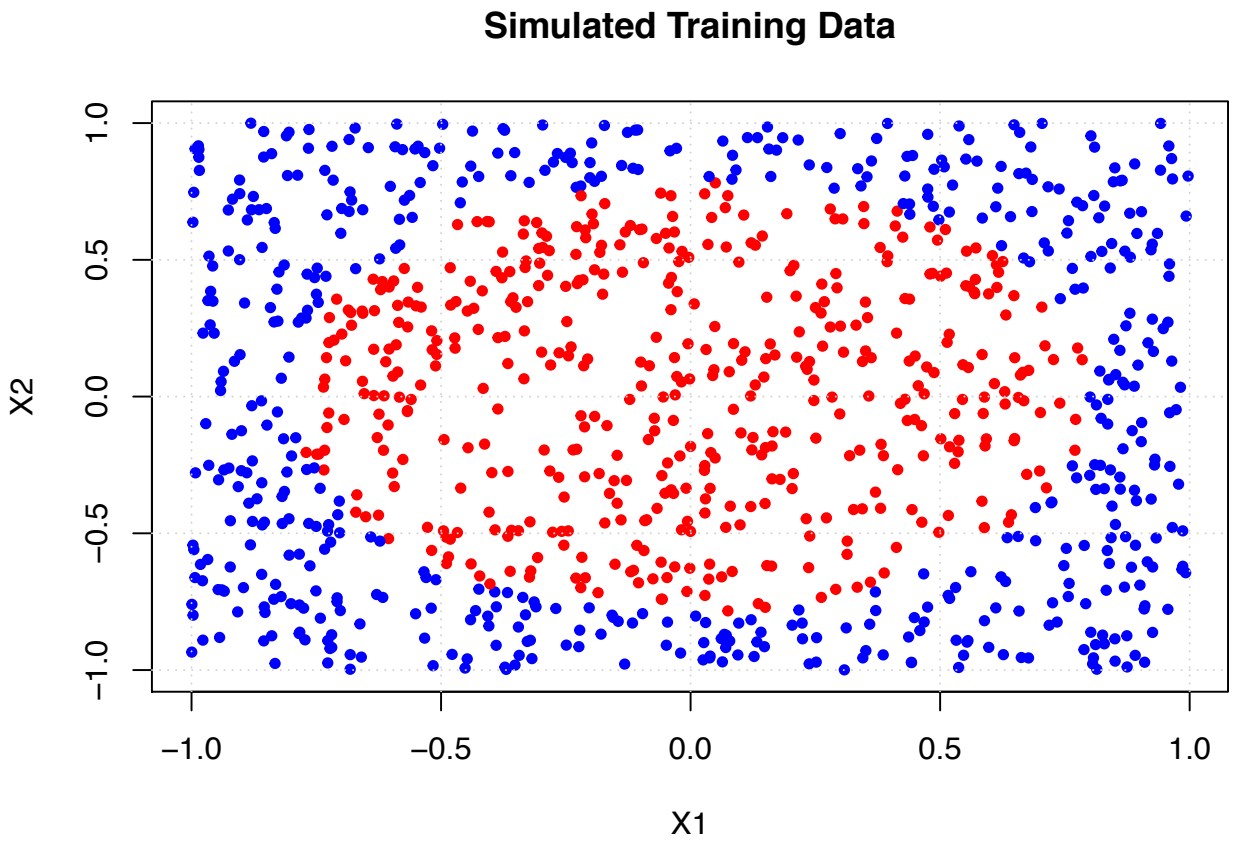
MLBench is a framework for distributed machine learning. Its purpose is to improve transparency, reproducibility, robustness, and to provide fair performance measures as well as reference implementations, helping adoption of distributed machine learning methods both in industry and in the academic community.

MLBench is public, open source and vendor independent, and has two main goals:

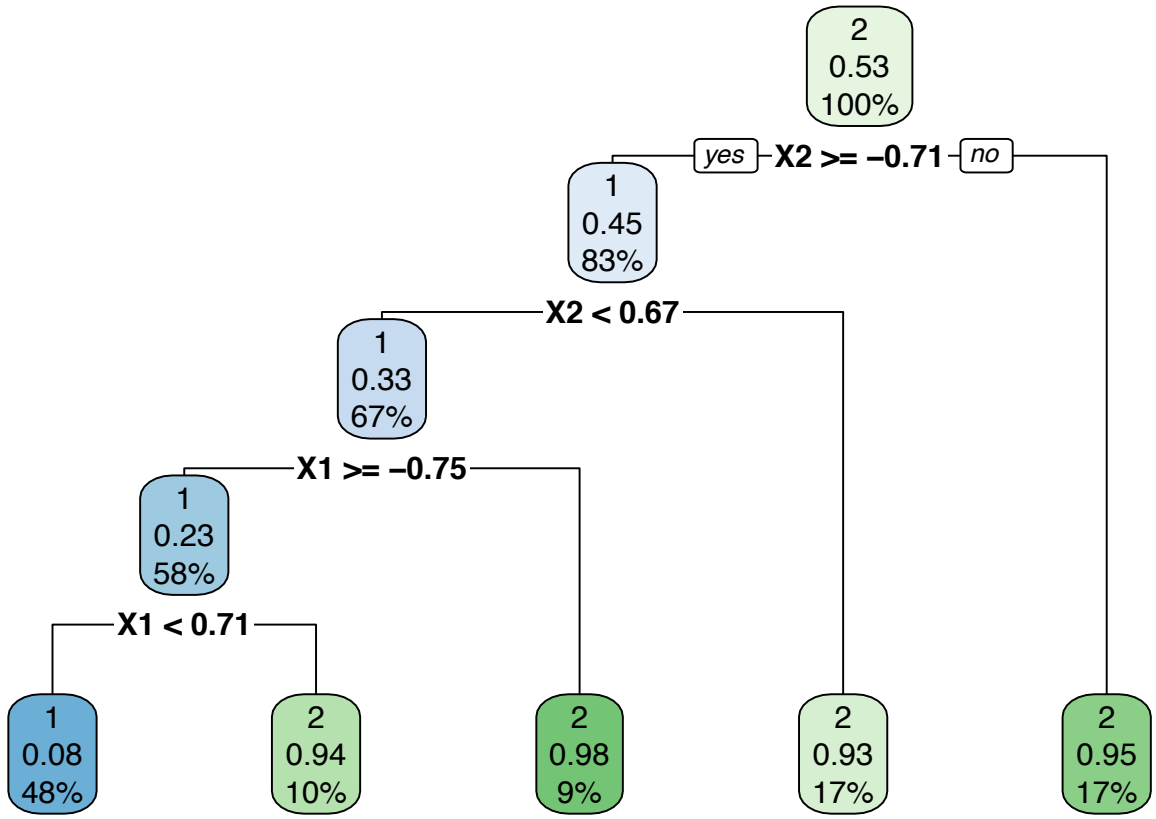
to be an easy-to-use and fair benchmarking suite for algorithms as well as for systems (software frameworks and hardware). to provide re-usable and reliable reference implementations of distributed ML algorithms. For more details on the benchmarking tasks, see Benchmark Tasks and Benchmark Results

Sponsors EPFL PwC Google Facebook

```
#setting training and test data
set.seed(42)
sim_trn = mlbench.circle(n = 1000, d = 2)
sim_trn = data.frame(sim_trn$x, class = as.factor(sim_trn$classes))
sim_tst = mlbench.circle(n = 1000, d = 2)
sim_tst = data.frame(sim_tst$x, class = as.factor(sim_tst$classes))

#Defining true positives and plotting
sim_trn_col = ifelse(sim_trn$class == 1, "red", "blue")
plot(sim_trn$X1, sim_trn$X2, col = sim_trn_col,
     xlab = "X1", ylab = "X2", main = "Simulated Training Data", pch = 20)
grid()
```

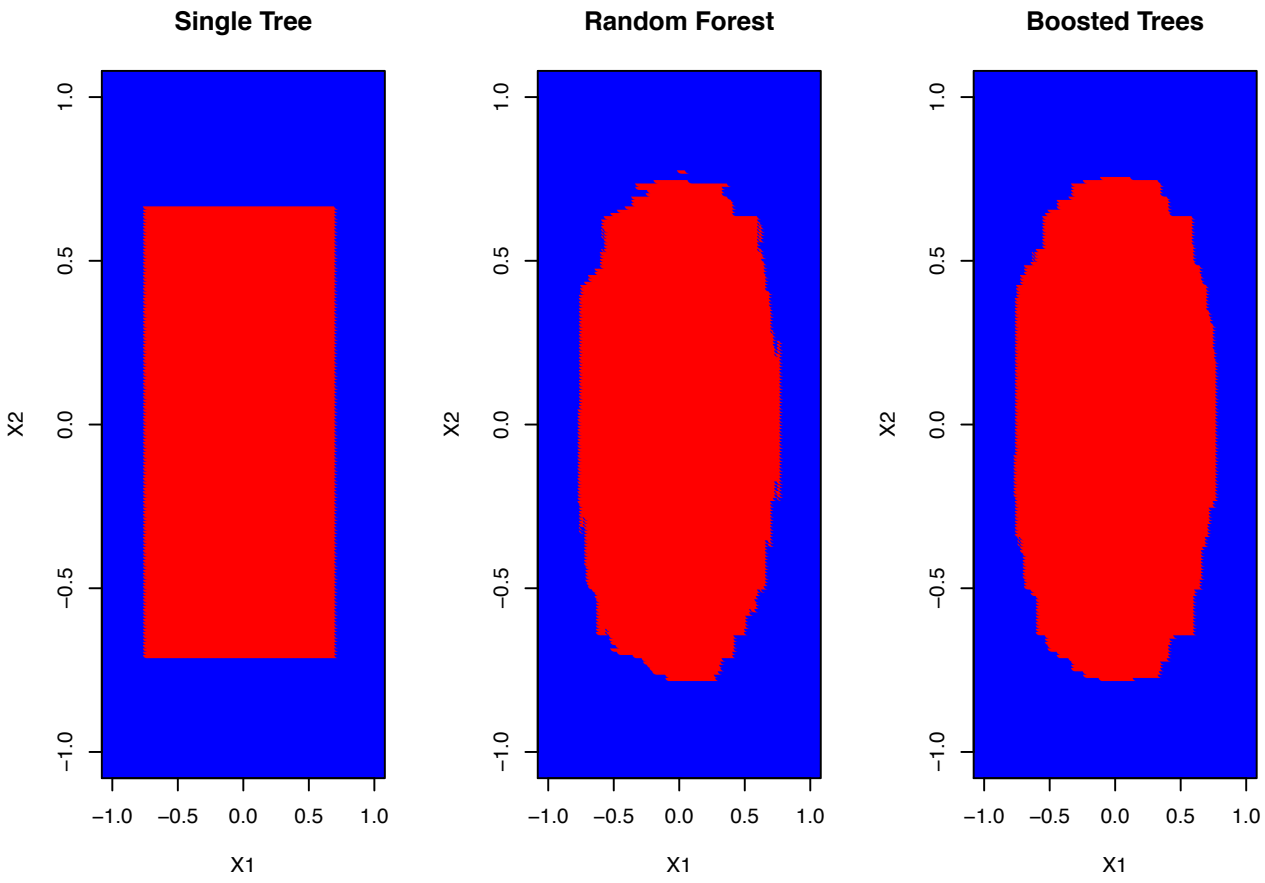



```
cv_5 = trainControl(method = "cv", number = 5)
oob  = trainControl(method = "oob")

sim_tree_cv = train(class ~ .,
                    data = sim_trn,
                    trControl = cv_5,
                    method = "rpart")

#Print tree
rpart.plot(sim_tree_cv$finalModel)
```




```
rf_grid = expand.grid(mtry = c(1, 2))
sim_rf_oob = train(class ~ .,
                   data = sim_trn,
                   trControl = oob,
                   tuneGrid = rf_grid)

gbm_grid =  expand.grid(interaction.depth = 1:5,
                        n.trees = (1:6) * 500,
                        shrinkage = c(0.001, 0.01, 0.1),
                        n.minobsinnode = 10)

sim_gbm_cv = train(class ~ .,
                   data = sim_trn,
                   method = "gbm",
                   trControl = cv_5,
                   verbose = FALSE,
                   tuneGrid = gbm_grid)
#plotting the grid
plot_grid = expand.grid(
  X1 = seq(min(sim_tst$X1) - 1, max(sim_tst$X1) + 1, by = 0.01),
  X2 = seq(min(sim_tst$X2) - 1, max(sim_tst$X2) + 1, by = 0.01)
)
```

```
tree_pred = predict(sim_tree_cv, plot_grid)
rf_pred   = predict(sim_rf_oob, plot_grid)
gbm_pred  = predict(sim_gbm_cv, plot_grid)

tree_col = ifelse(tree_pred == 1, "red", "blue")
rf_col   = ifelse(rf_pred == 1, "red", "blue")
gbm_col  = ifelse(gbm_pred == 1, "red", "blue")


par(mfrow = c(1, 3))
plot(plot_grid$X1, plot_grid$X2, col = tree_col,
     xlab = "X1", ylab = "X2", pch = 20, main = "Single Tree",
     xlim = c(-1, 1), ylim = c(-1, 1))
plot(plot_grid$X1, plot_grid$X2, col = rf_col,
     xlab = "X1", ylab = "X2", pch = 20, main = "Random Forest",
     xlim = c(-1, 1), ylim = c(-1, 1))
plot(plot_grid$X1, plot_grid$X2, col = gbm_col,
     xlab = "X1", ylab = "X2", pch = 20, main = "Boosted Trees",
     xlim = c(-1, 1), ylim = c(-1, 1))
```



# Chapter 12

# Bibliography


1. Professor Rafael Irizarry - Data Science course 1- 9 at HarvardX
2. David Dalpiaz - R for Statistical Learning
3. Sami M. Halawani and Amir Ahmad - Ensemble Methods for Prediction of Parkinson Disease -Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh, Saudi Arabia, 2019
4. Jianxin Zhang et al. - Exploring risk factors and predicting UPDRS score based on Parkinson's speech signals, 2017
5. Houtao Deng -Interpreting Tree Ensembles with inTrees, 2019
6. Max Kuhn - Building Predictive Models in R Using the caret Package 2008
7. Iqra Nissar et al. - Voice-Based Detection of Parkinson's Disease through Ensemble Machine Learning Approach: A Performance Study, 2018
8. Tom Fawcett, An introduction to ROC analysis, Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA, 2005
9. PBR, https://www.prb.org/resources/countries-with-the-oldest-populations-in-the-world/