Statistical Learning Project

Statistical Learning Final Project

Group 2: Sarah, Pavel, Rose, Catherine, Shravya East

```
#load the necessary packages
library(plyr)
library(tidyverse)
library(reshape2)
library(readxl)
library(caret)
library(rpart)
library(partykit)
library(randomForest)
library(class)
library (rminer)
library(e1071)
library(mlbench)
library(plyr)
library(DMwR)

#Read in the data
dat <- read_excel("Absenteeism_at_work.xls")

#View the data
glimpse(dat)

## Observations: 740
## Variables: 21
## $ ID                                <dbl> 11, 36, 3, 7, 11, 3, 10, 20,...
## $ `Reason for absence`              <dbl> 26, 0, 23, 7, 23, 23, 22, 23...
## $ `Month of absence`                <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7...
## $ `Day of the week`                 <dbl> 3, 3, 4, 5, 5, 6, 6, 6, 2, 2...
## $ Seasons                           <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ `Transportation expense`          <dbl> 289, 118, 179, 279, 289, 179...
## $ `Distance from Residence to Work` <dbl> 36, 13, 51, 5, 36, 51, 52, 5...
## $ `Service time`                    <dbl> 13, 18, 18, 14, 13, 18, 3, 1...
## $ Age                               <dbl> 33, 50, 38, 39, 33, 38, 28, ...
## $ `Work load Average/day`           <dbl> 239554, 239554, 239554, 2395...
## $ `Hit target`                      <dbl> 97, 97, 97, 97, 97, 97, 97, ...
## $ `Disciplinary failure`            <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0...
## $ Education                         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 3...
## $ Son                               <dbl> 2, 1, 0, 2, 2, 0, 1, 4, 2, 1...
## $ `Social drinker`                  <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0...
## $ `Social smoker`                   <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0...
## $ Pet                               <dbl> 1, 0, 0, 0, 1, 0, 4, 0, 0, 1...
```

```
## $ Weight                               <dbl> 90, 98, 89, 68, 90, 89, 80, ...
## $ Height                               <dbl> 172, 178, 170, 168, 172, 170...
## $ `Body mass index`                    <dbl> 30, 31, 31, 24, 30, 31, 27, ...
## $ `Absenteeism time in hours`          <dbl> 4, 0, 2, 4, 2, 2, 8, 4, 40, ...
```

Pre-Processing Data

```
#Set factored variables as factors
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
#set all categorical variables as ordered factors
dat[col] <- lapply(dat[col], as.factor)
dat[col] <- lapply(dat[col], ordered)

#Rename the columns for easier use
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")

#View the data
glimpse(dat)
```

```
## Observations: 740
## Variables: 21
## $ ID                   <ord> 11, 36, 3, 7, 11, 3, 10, 20, 14, 1, 20,...
## $ Reason               <ord> 26, 0, 23, 7, 23, 23, 22, 23, 19, 22, 1...
## $ Month                <ord> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, ...
## $ Day                  <ord> 3, 3, 4, 5, 5, 6, 6, 6, 2, 2, 2, 3, 4, ...
## $ Seasons              <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ Transportation_expense <dbl> 289, 118, 179, 279, 289, 179, 361, 260,...
## $ Distance             <dbl> 36, 13, 51, 5, 36, 51, 52, 50, 12, 11, ...
## $ Service_time         <dbl> 13, 18, 18, 14, 13, 18, 3, 11, 14, 14, ...
## $ Age                  <dbl> 33, 50, 38, 39, 33, 38, 28, 36, 34, 37,...
## $ Work_load            <dbl> 239554, 239554, 239554, 239554, 239554,...
## $ Hit_target           <dbl> 97, 97, 97, 97, 97, 97, 97, 97, 97, 97,...
## $ Disciplinary_failure <ord> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ Education            <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, ...
## $ Children             <dbl> 2, 1, 0, 2, 2, 0, 1, 4, 2, 1, 4, 4, 4, ...
## $ Social_drinker       <ord> 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, ...
## $ Social_smoker        <ord> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ Pet                  <dbl> 1, 0, 0, 0, 1, 0, 4, 0, 0, 1, 0, 0, 0, ...
## $ Weight               <dbl> 90, 98, 89, 68, 90, 89, 80, 65, 95, 88,...
## $ Height               <dbl> 172, 178, 170, 168, 172, 170, 172, 168,...
## $ BMI                  <dbl> 30, 31, 31, 24, 30, 31, 27, 23, 25, 29,...
## $ Absent_time          <dbl> 4, 0, 2, 4, 2, 2, 8, 4, 40, 8, 8, 8, 8,...
```

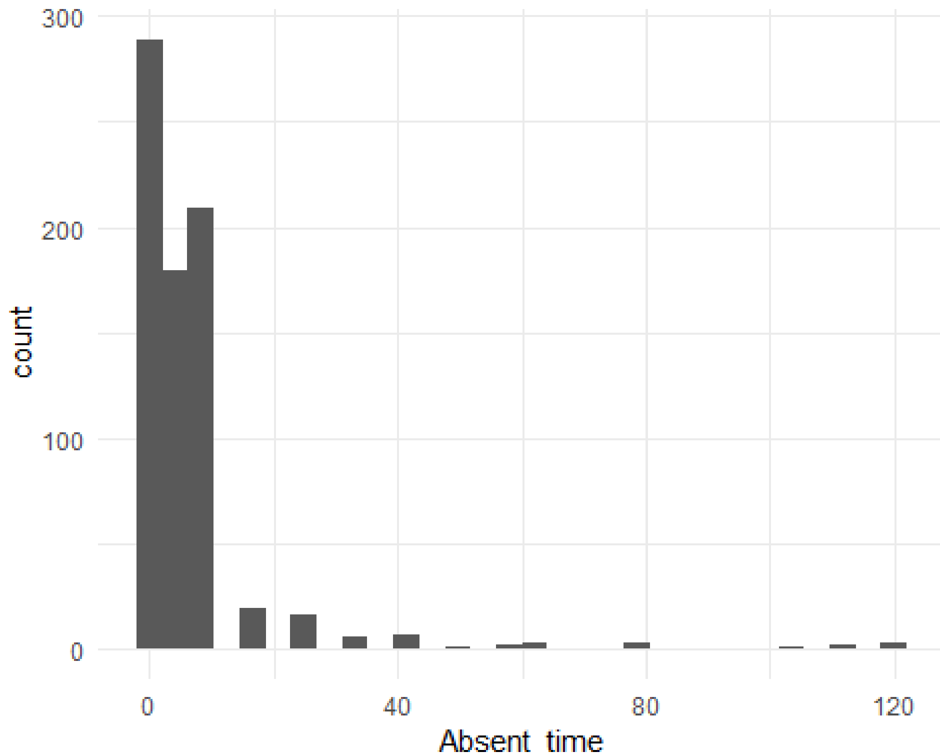
```
#create a list of the numeric variables in the data set
nums <- unlist(lapply(dat, is.numeric))
```

```r
#create a smaller data set of just numeric variables
dat.num <- dat[ , nums]
```

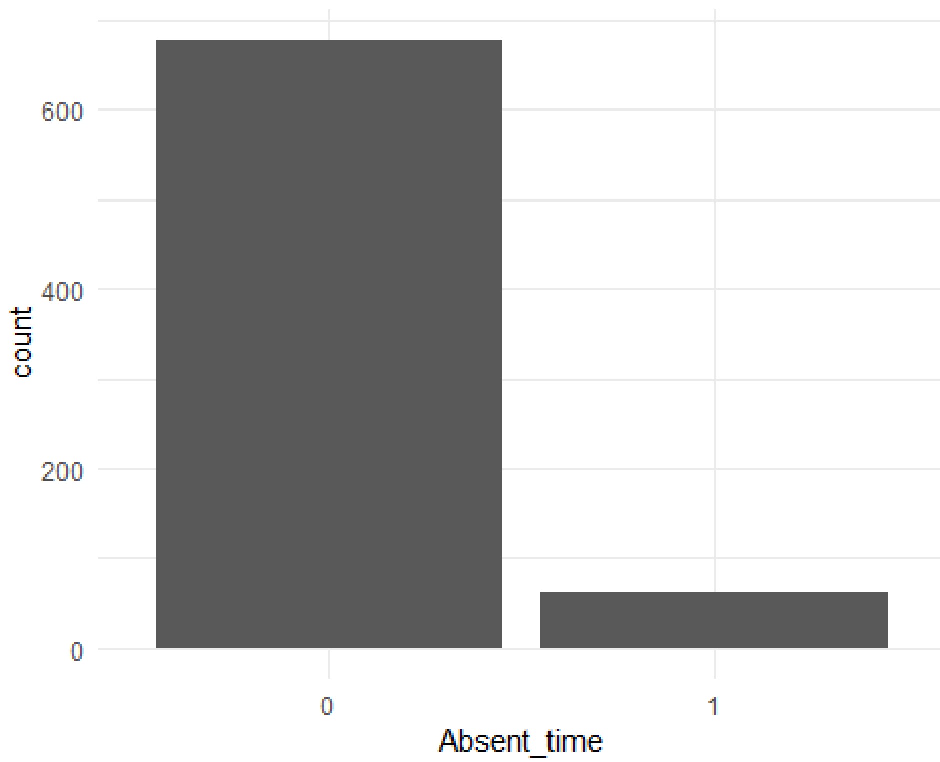
EDA Response Variable

Absent_time

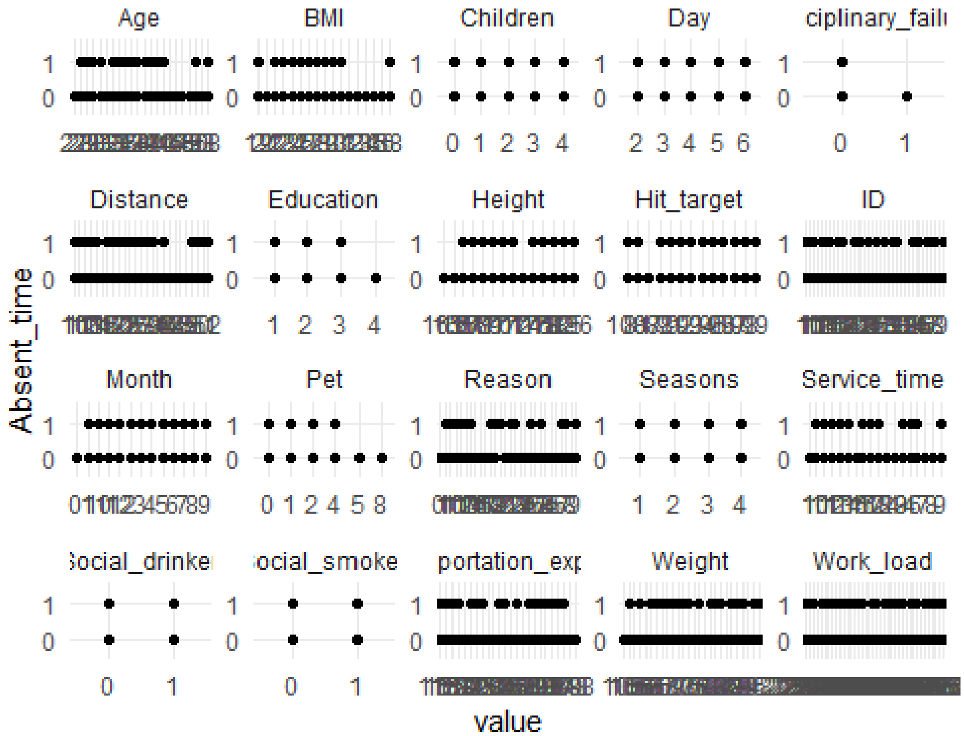
```r
summary(dat$Absent_time)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   2.000   3.000   6.924   8.000 120.000

dat %>%
  count(Absent_time)

## # A tibble: 19 x 2
##    Absent_time     n
##          <dbl> <int>
##  1           0    44
##  2           1    88
##  3           2   157
##  4           3   112
##  5           4    60
##  6           5     7
##  7           7     1
##  8           8   208
##  9          16    19
## 10          24    16
## 11          32     6
## 12          40     7
## 13          48     1
## 14          56     2
## 15          64     3
## 16          80     3
## 17         104     1
## 18         112     2
## 19         120     3

#plot the Absent_time
ggplot(data = dat,
       aes(x = Absent_time)) +
  geom_histogram() +
  theme_minimal()

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
#change variable represent missed time one day or greater
dat <- dat %>%
  mutate(Absent_time = ifelse(dat$Absent_time <= 8,0,1))

#save Absent_time as a factor in the data set
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

#plot the Absent_time
ggplot(data = dat,
       aes(x = Absent_time)) +
  geom_bar() +
  theme_minimal()
```




```r
#plot all variables vs. Absent_time
dat %>%
  gather(-Absent_time, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Absent_time)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

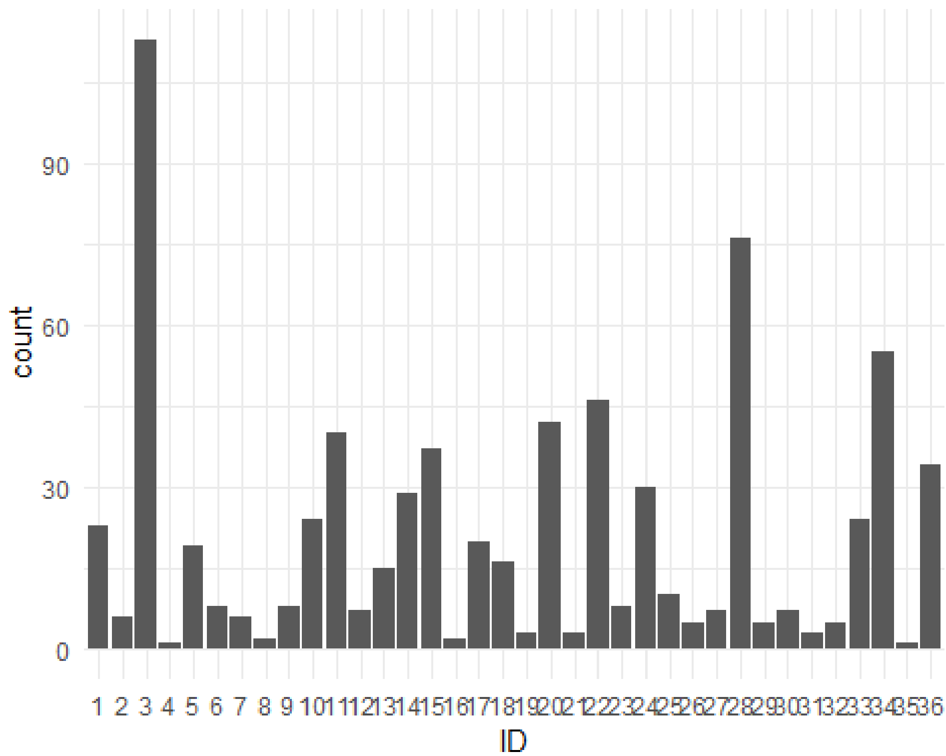


EDA Predictors

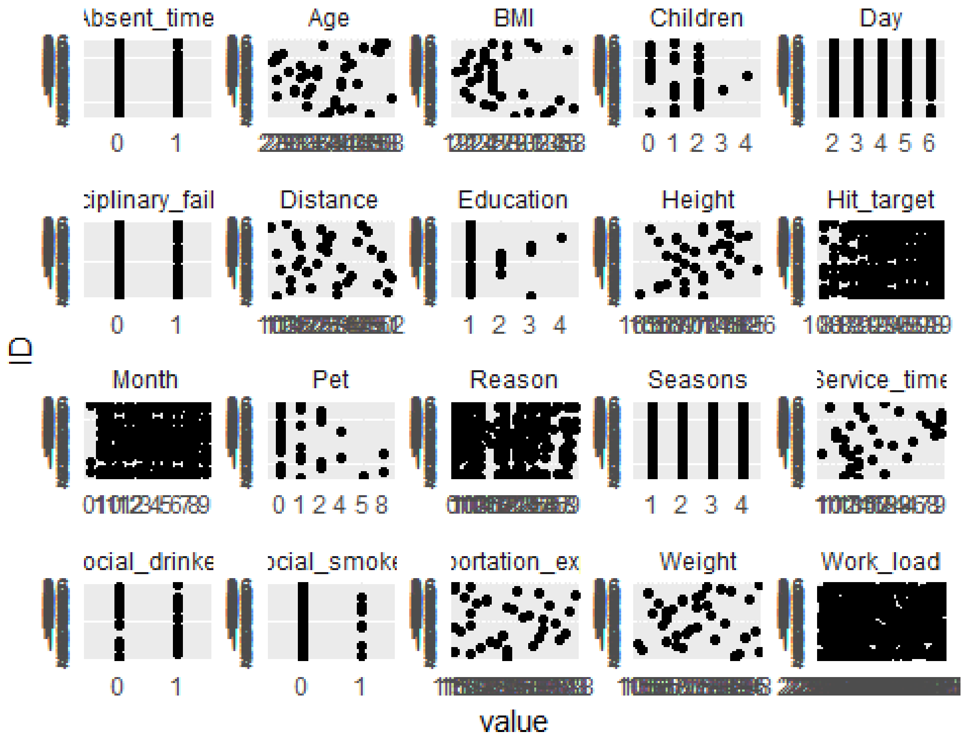
ID

```
#frequency table by ID
dat %>%
  count(ID)

## # A tibble: 36 x 2
##    ID        n
##    <ord> <int>
##  1 1        23
##  2 2         6
##  3 3       113
##  4 4         1
##  5 5        19
##  6 6         8
##  7 7         6
##  8 8         2
##  9 9         8
## 10 10       24
## # ... with 26 more rows

#bar chart
dat %>%
  ggplot(aes(x = ID)) +
```

```r
  geom_bar() +
  theme_minimal()
```




```r
#ID
dat %>%
  gather(-ID, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = ID)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

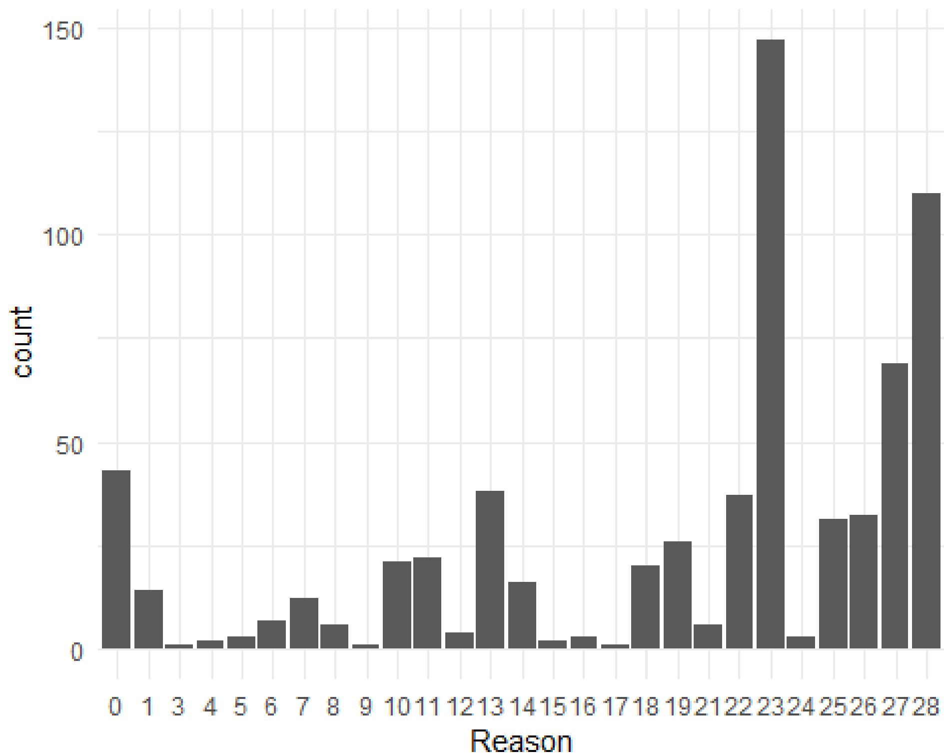


Reason

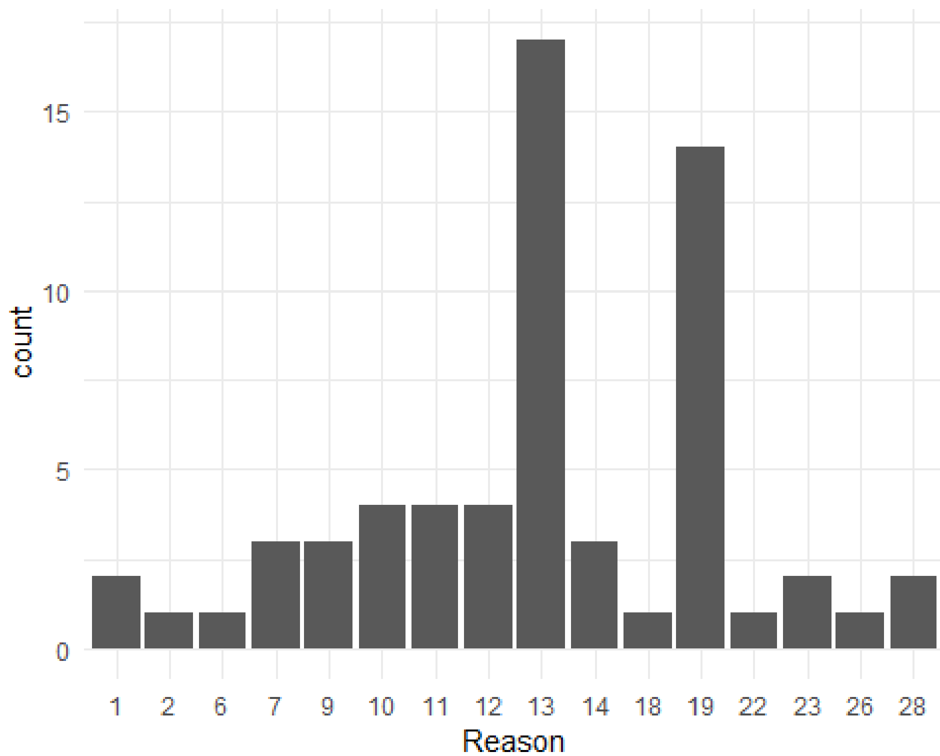
```
#frequency table by Reason for Absence
dat %>%
  count(Reason)
```

```
## # A tibble: 28 x 2
##    Reason      n
##    <ord>   <int>
##  1 0          43
##  2 1          16
##  3 2           1
##  4 3           1
##  5 4           2
##  6 5           3
##  7 6           8
##  8 7          15
##  9 8           6
## 10 9           4
## # ... with 18 more rows
```

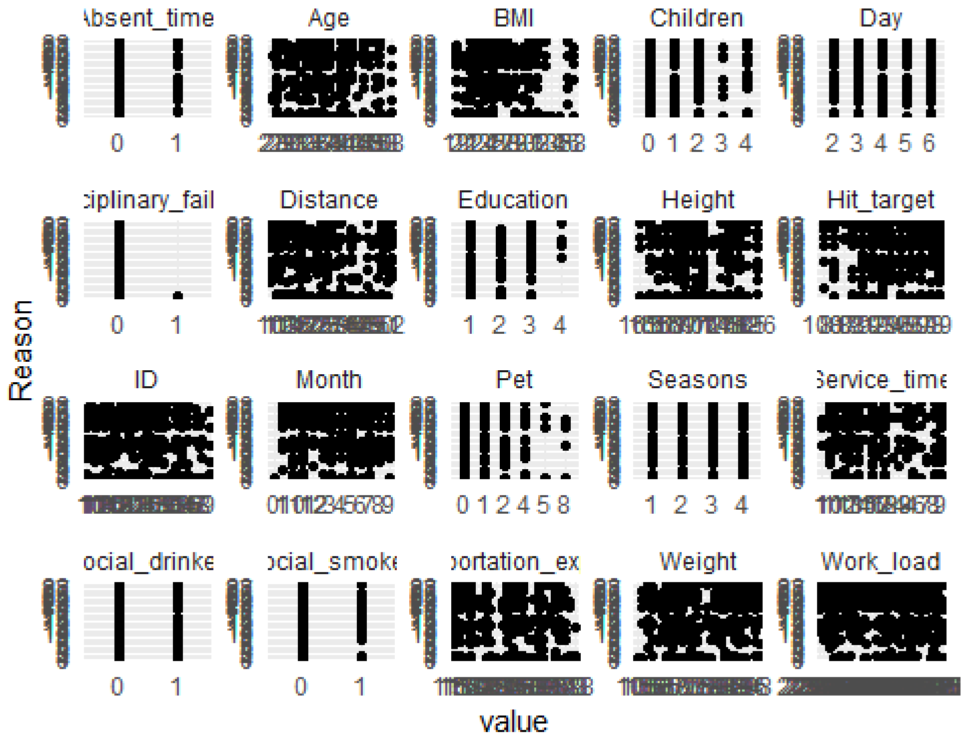
```
#bar chart
dat %>%
  filter(Absent_time==0) %>%
  ggplot(aes(x=Reason)) +
  geom_bar() +
  theme_minimal()
```




```
dat %>%
  filter(Absent_time==1) %>%
  ggplot(aes(x=Reason)) +
  geom_bar() +
  theme_minimal()
```

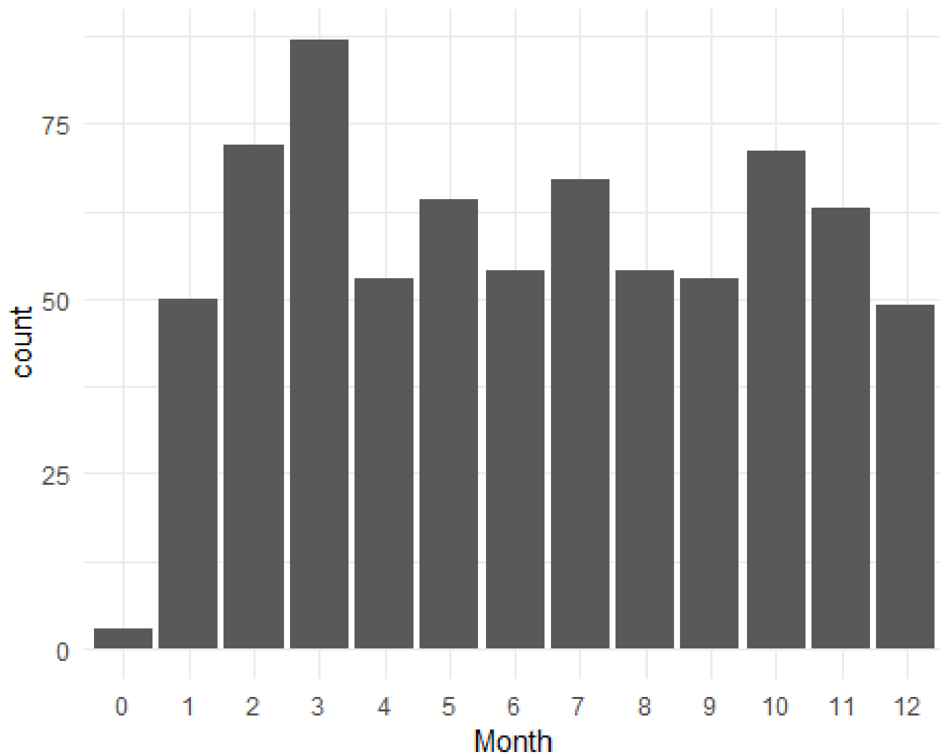



```
#Reason for absence
table(dat %>%
  filter(Reason==0) %>%
  select(Absent_time))

##
##  0  1
## 43  0

dat %>%
  gather(-Reason, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Reason)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

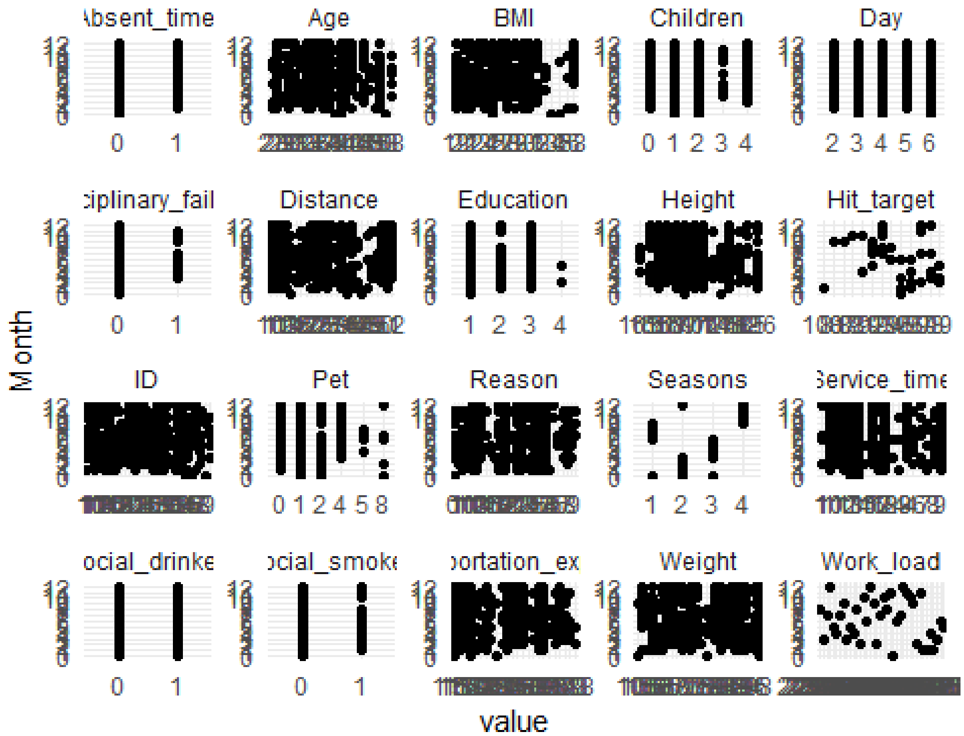


Month

```
#frequency table by Month of Absence
dat %>%
  count(Month)

## # A tibble: 13 x 2
##    Month     n
##    <ord> <int>
##  1 0         3
##  2 1        50
##  3 2        72
##  4 3        87
##  5 4        53
##  6 5        64
##  7 6        54
##  8 7        67
##  9 8        54
## 10 9        53
## 11 10       71
## 12 11       63
## 13 12       49

#bar chart
dat %>%
  ggplot(aes(x=Month)) +
  geom_bar() +
  theme_minimal()
```

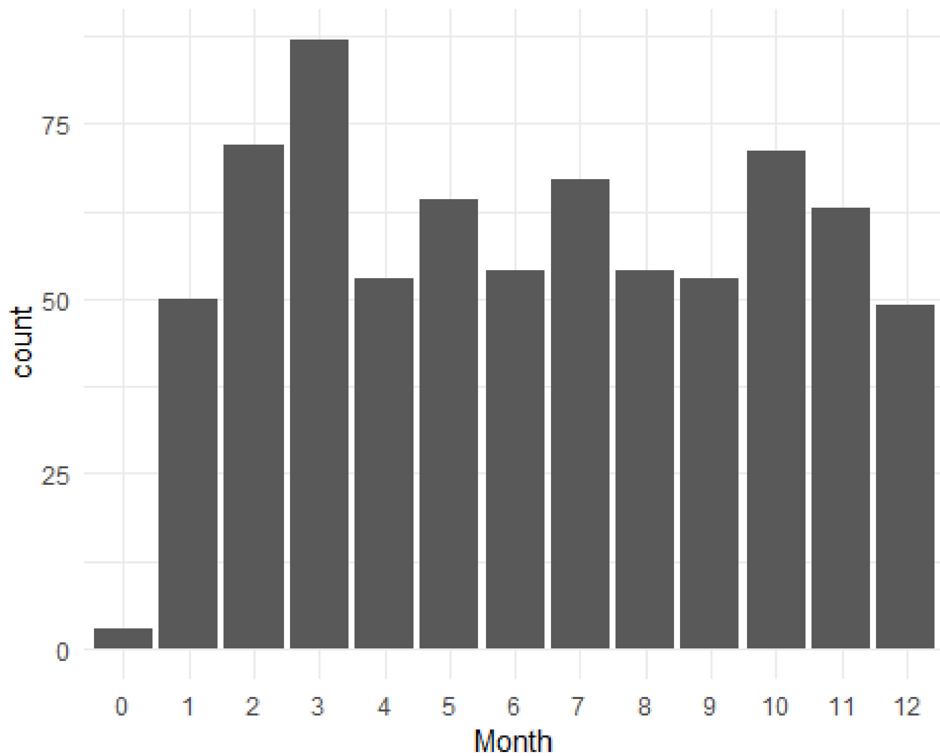



```
dat %>%
  gather(-Month, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Month)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

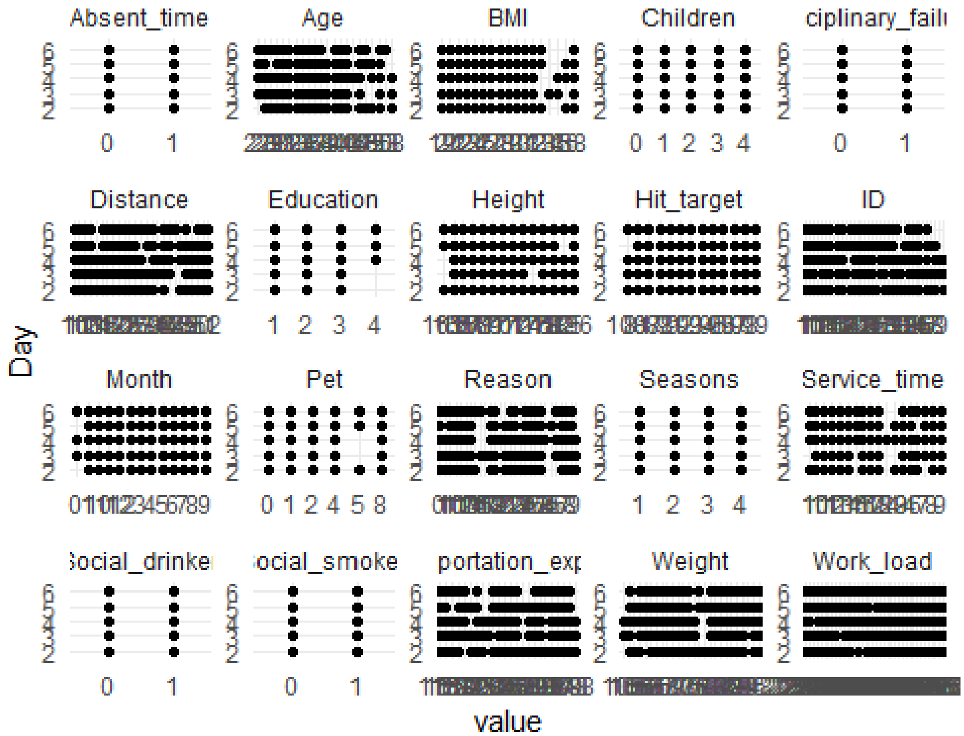


Day

```
#frequency table by Day of Absence
dat %>%
  count(Day)

## # A tibble: 5 x 2
##   Day       n
##   <ord> <int>
## 1 2       161
## 2 3       154
## 3 4       156
## 4 5       125
## 5 6       144

#bar chart
dat %>%
  ggplot(aes(x=Month)) +
  geom_bar() +
  theme_minimal()
```




```
dat %>%
  gather(-Day, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Day)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

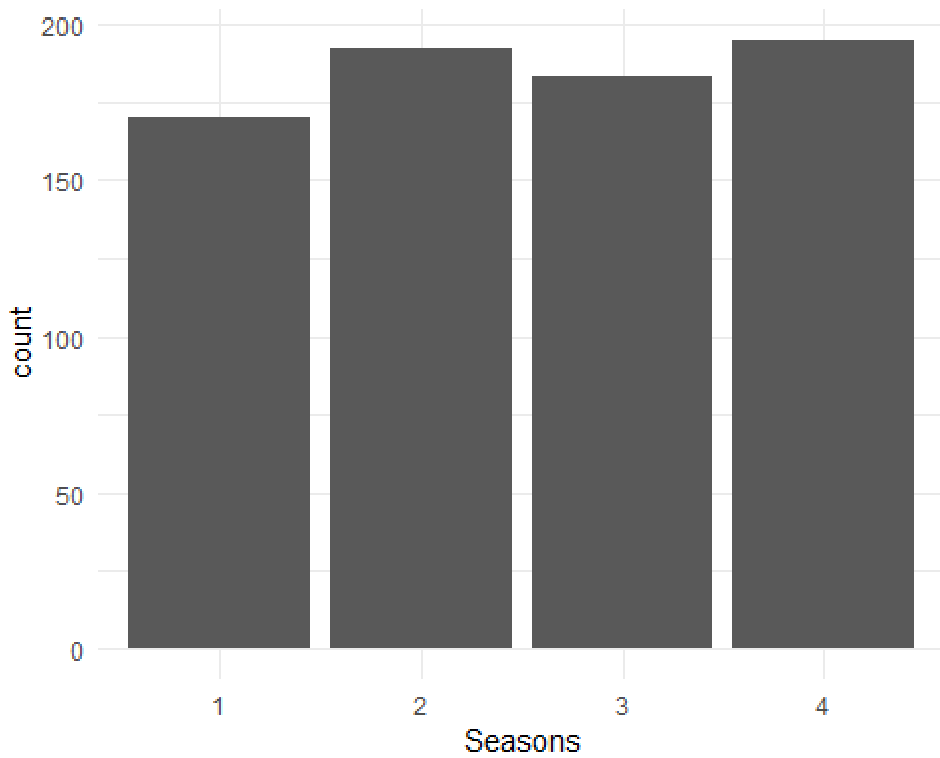


Seasons

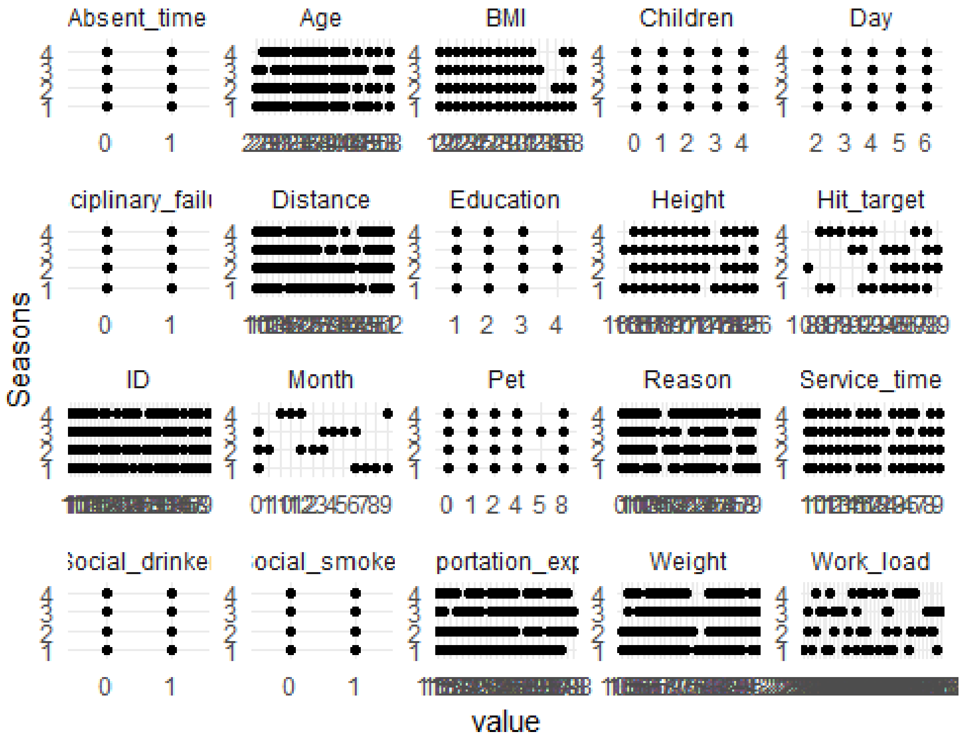
```r
#frequency table by Season of Absence
dat %>%
  count(Seasons)
```

```
## # A tibble: 4 x 2
##   Seasons     n
##   <ord>   <int>
## 1 1         170
## 2 2         192
## 3 3         183
## 4 4         195
```

```r
#bar chart
dat %>%
  ggplot(aes(x=Seasons)) +
  geom_bar() +
  theme_minimal()
```

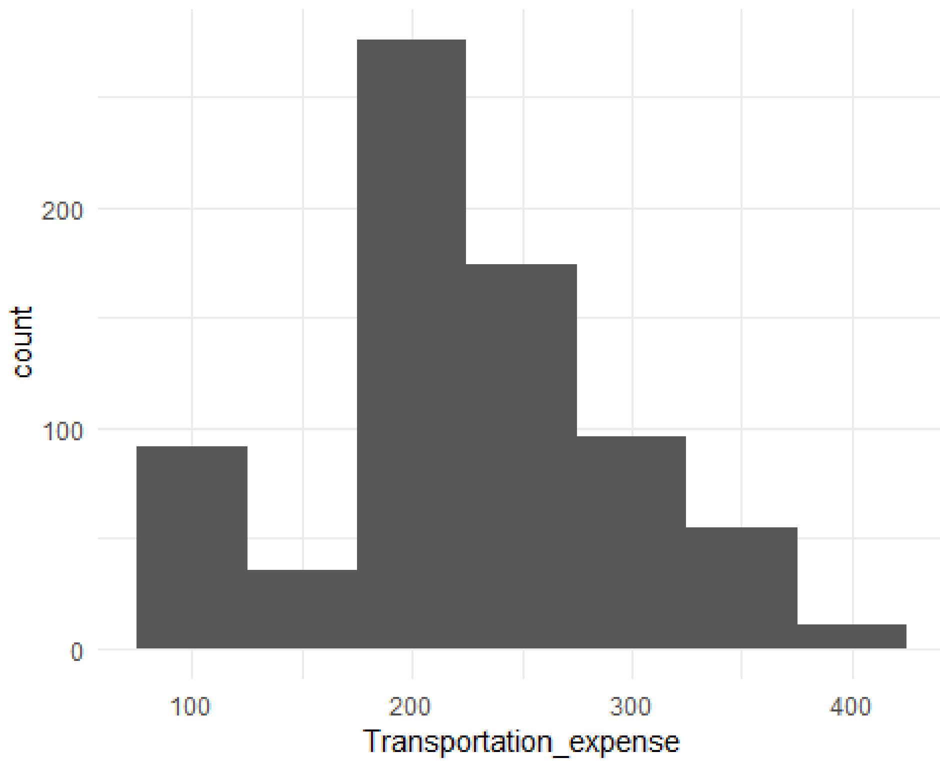



```r
#Scatterplots for variable 'Seasons'
dat %>%
  gather(-Seasons, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Seasons)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

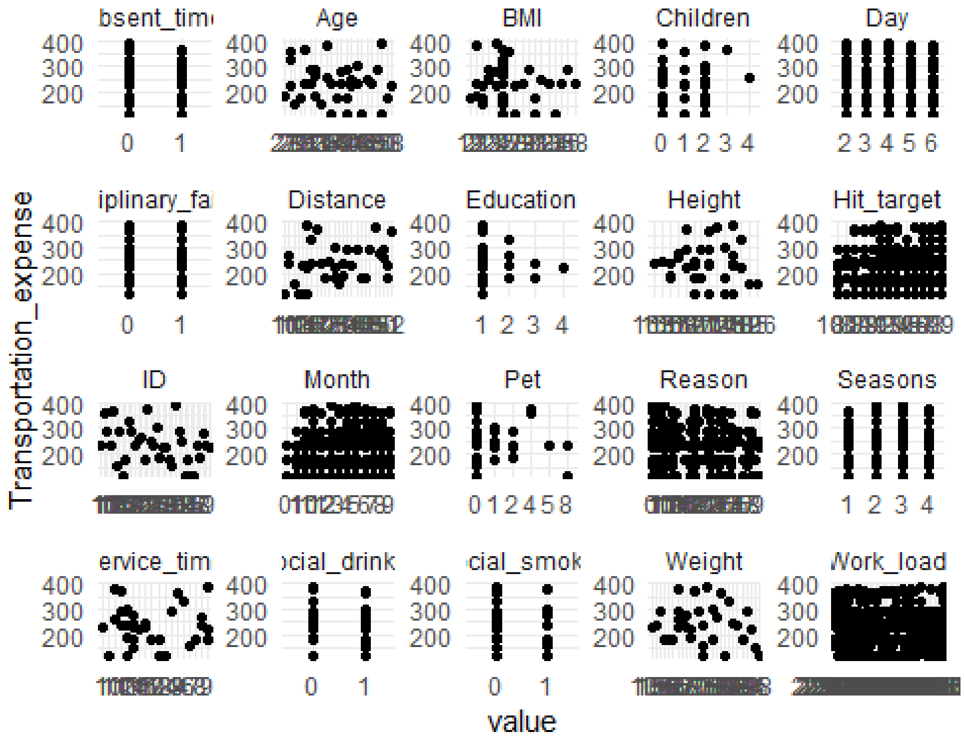


Transportation Expense

```r
#summary of transportation expenses
summary(dat$Transportation_expense)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   118.0   179.0   225.0   221.3   260.0   388.0

#histograph
ggplot(data = dat,
       aes(x = Transportation_expense)) +
  geom_histogram(binwidth = 50) +
  theme_minimal()
```



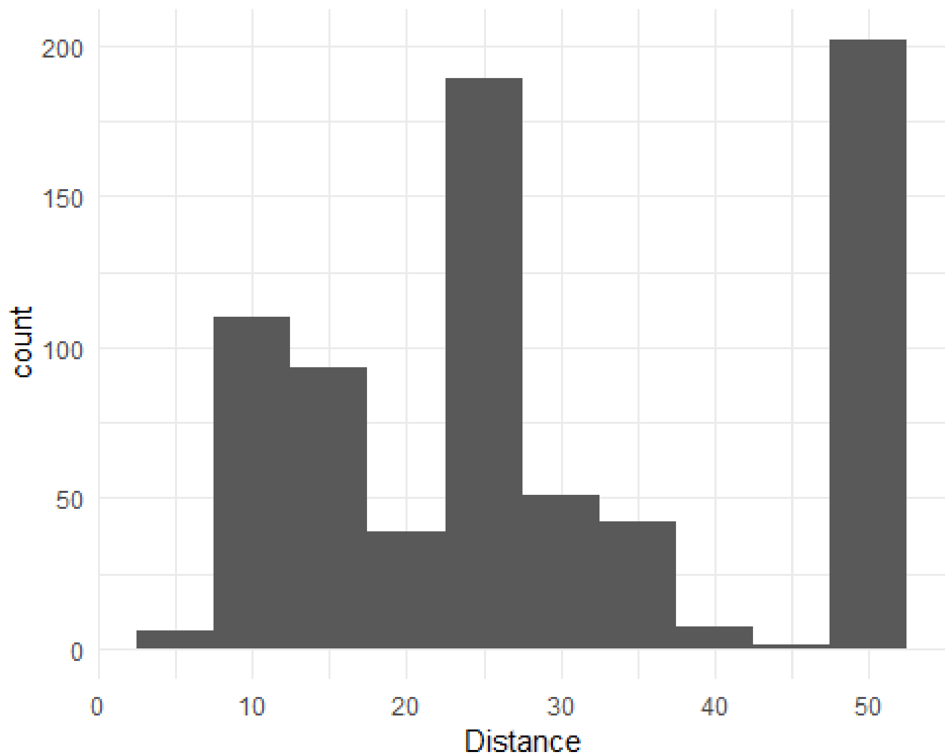

```
#Scatterplots for variable 'Transportation_expense'
dat %>%
  gather(-Transportation_expense, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Transportation_expense)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



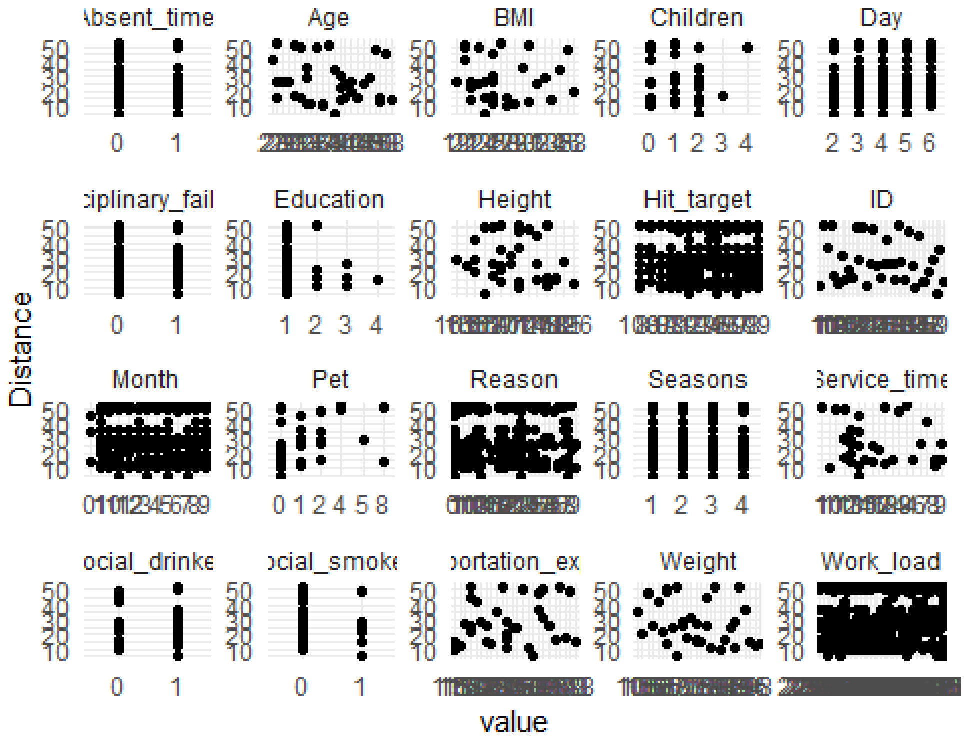

```r
# Possible positive correlation seen between distance and
Transportation_expense
```

Distance

```r
#summary of distance
summary(dat$Distance)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00   16.00   26.00   29.63   50.00   52.00

#histogram
ggplot(data = dat,
       aes(x = Distance)) +
  geom_histogram(binwidth = 5) +
  theme_minimal()
```



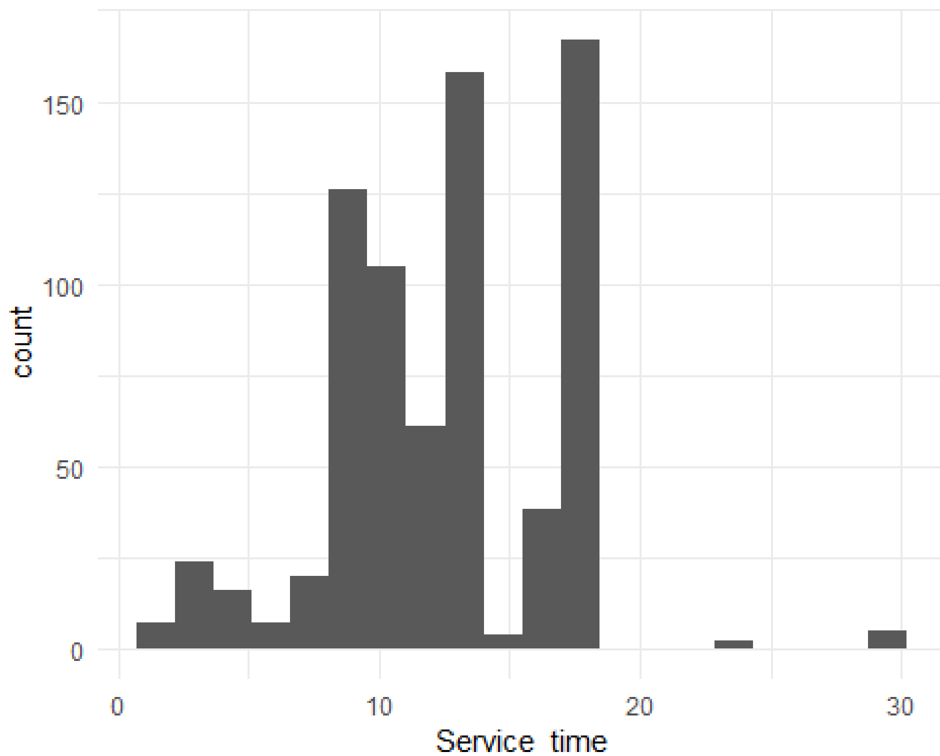

```
#Scatterplots for variable 'Distance'
dat %>%
  gather(-Distance, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Distance)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



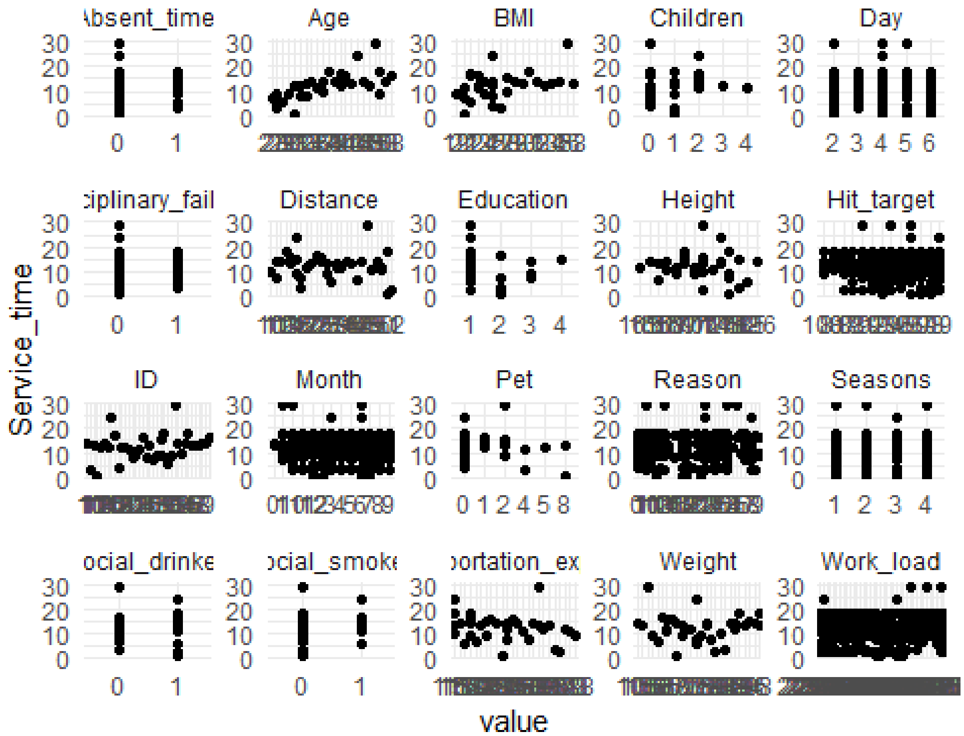

```
#Possible Positive correlation seen between distance and
Transportation_expense
```

Service Time

```
#summary for Service_time
summary(dat$Service_time)

##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.00    9.00   13.00   12.55   16.00   29.00

#histogram
ggplot(data = dat,
       aes(x = Service_time)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

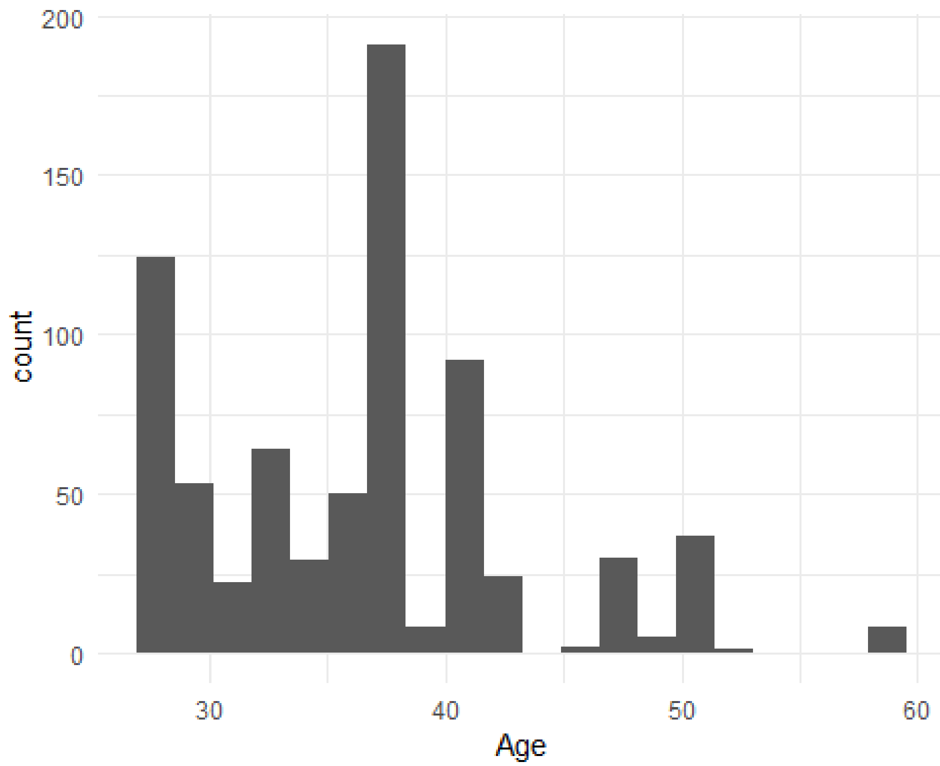



```r
#Scatterplots for variable 'Service_time'
dat %>%
  gather(-Service_time, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Service_time)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

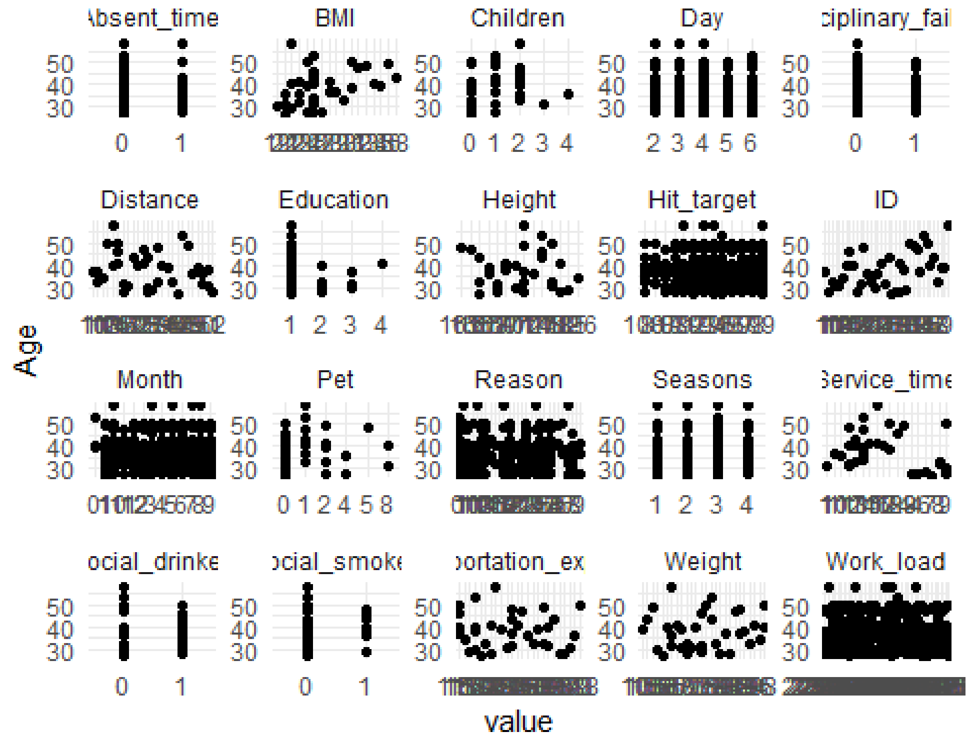


Age

```r
#summary for Age
summary(dat$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.00   31.00   37.00   36.45   40.00   58.00

#histogram
ggplot(data = dat,
       aes(x = Age)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

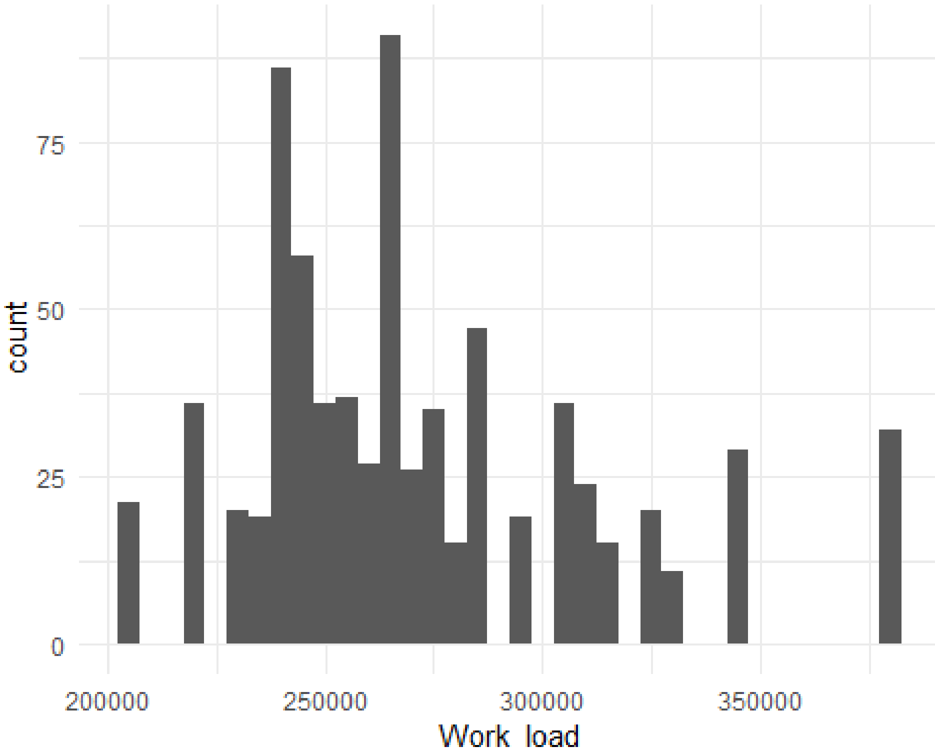



```r
#Scatterplots for variable 'Age'
dat %>%
  gather(-Age, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Age)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

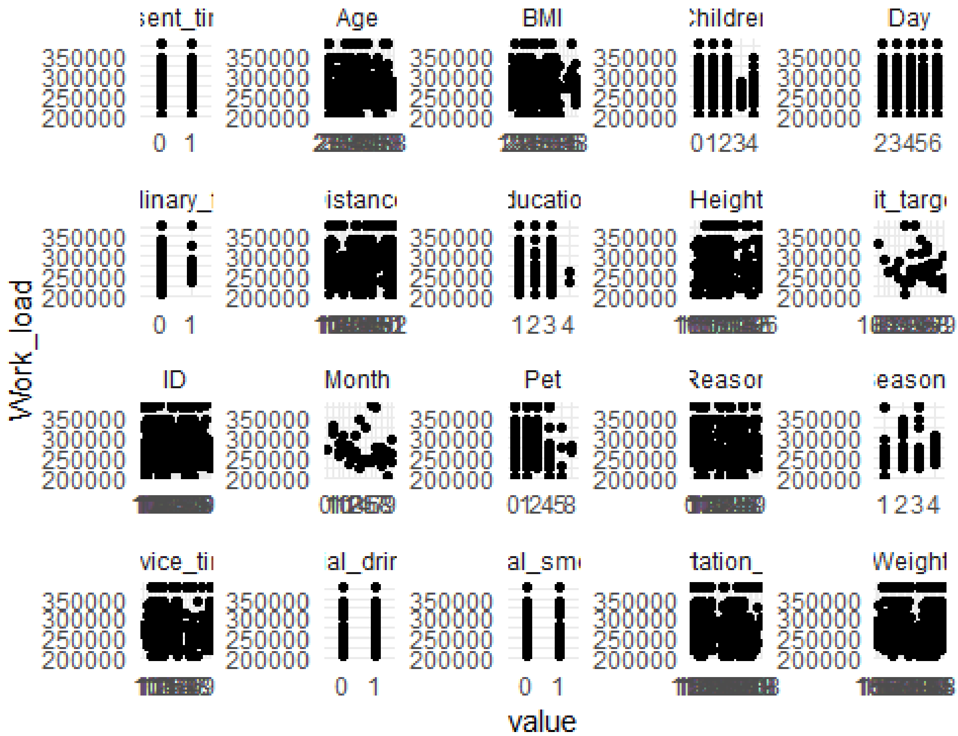


Workload

```r
#summary for work load
summary(dat$Work_load)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  205917  244387  264249  271490  294217  378884

#histogram
ggplot(data = dat,
       aes(x = Work_load)) +
  geom_histogram(binwidth = 5000) +
  theme_minimal()
```

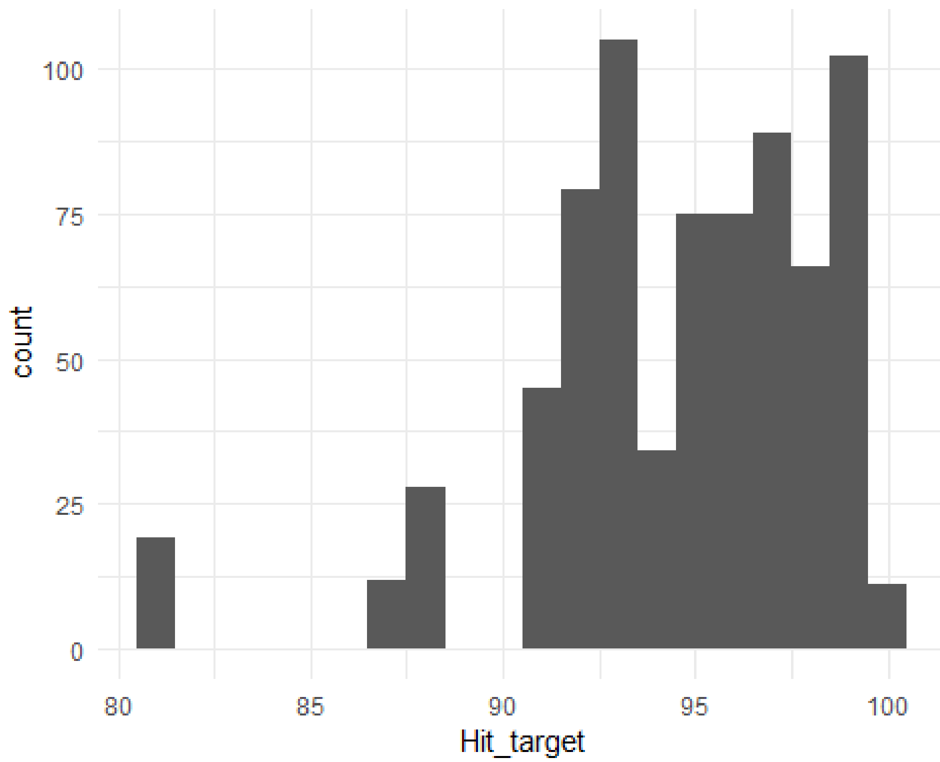



```r
#Scatterplots for variable 'Work_load'
dat %>%
  gather(-Work_load, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Work_load)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

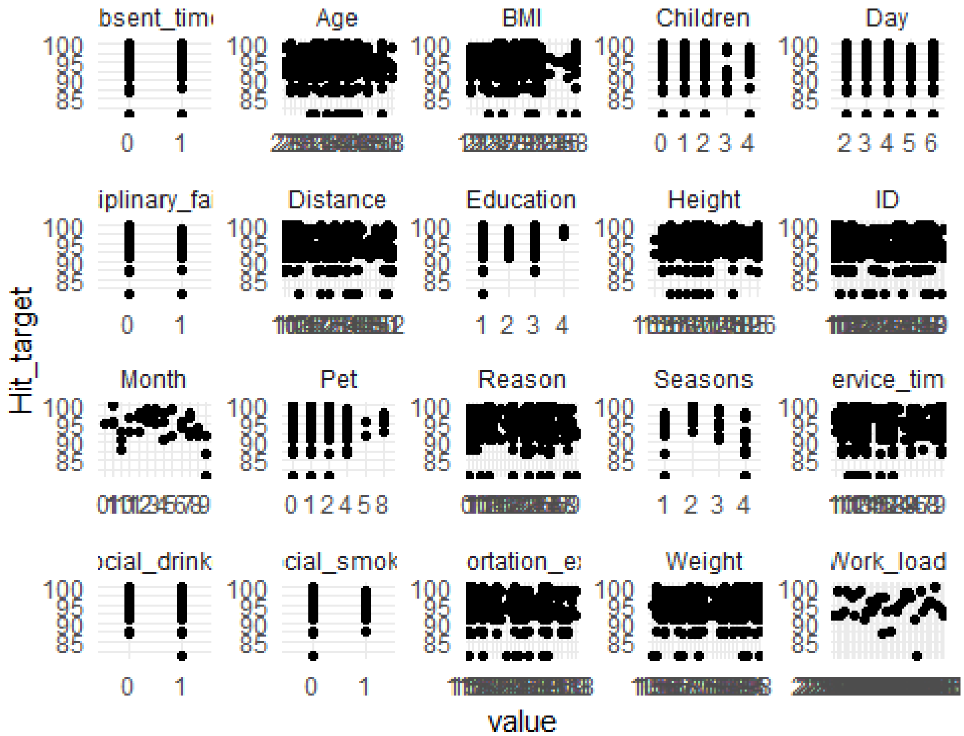


Hit Target

```r
#summary for hit target
summary(dat$Hit_target)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   81.00   93.00   95.00   94.59   97.00  100.00

#histogram
ggplot(data = dat,
       aes(x = Hit_target)) +
  geom_histogram(bins = 20) +
  theme_minimal()
```

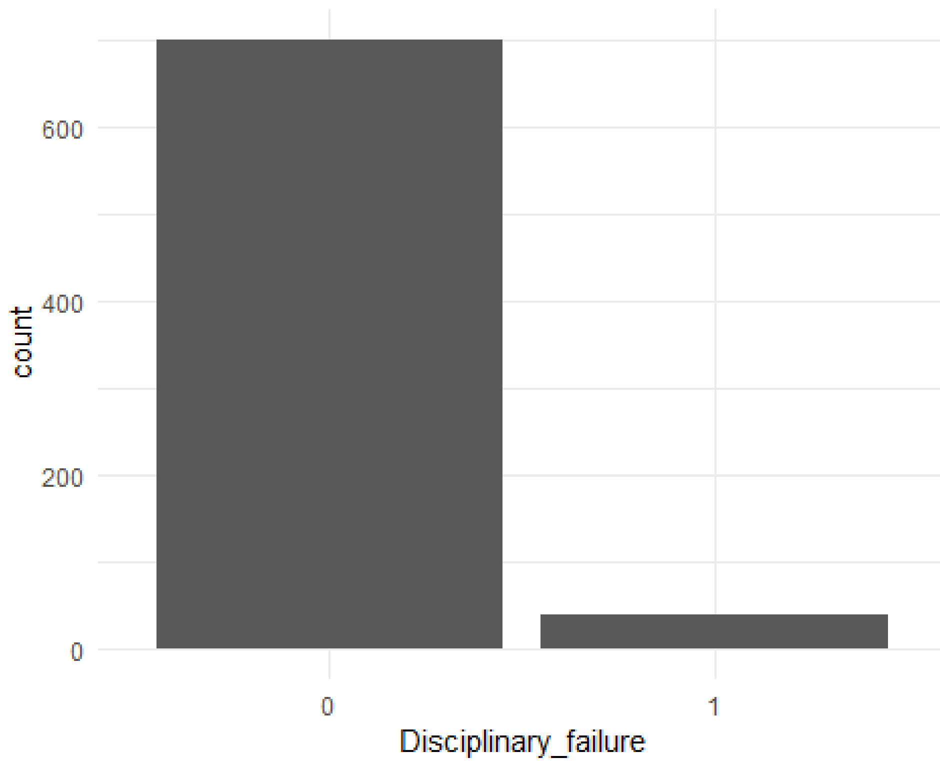



```
#Scatterplots for variable 'Hit_target'
dat %>%
  gather(-Hit_target, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Hit_target)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

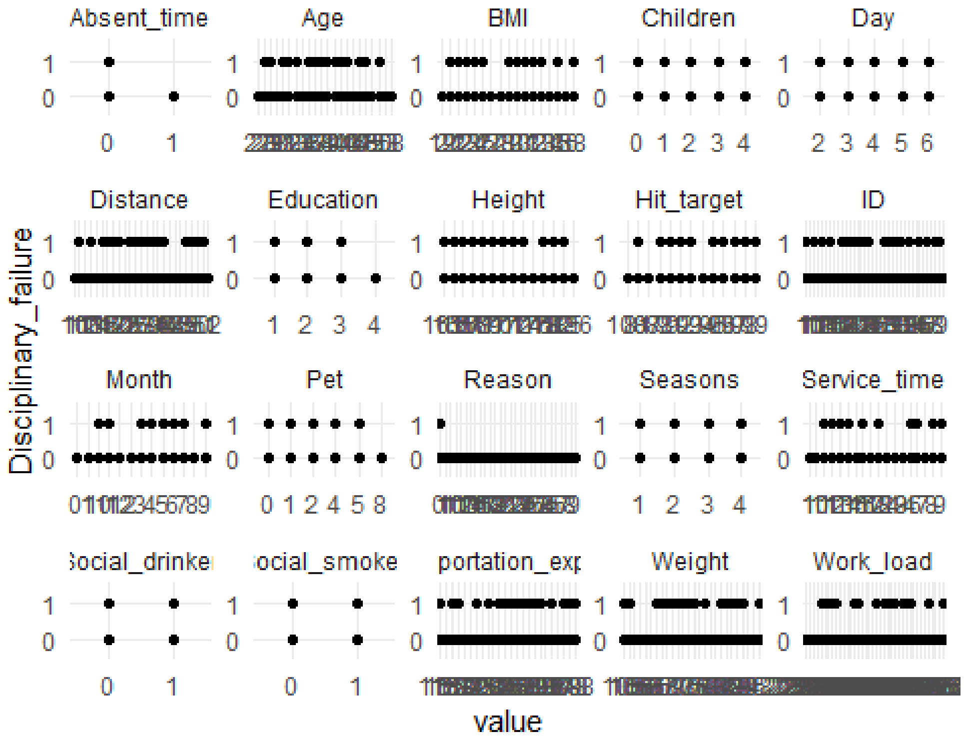


Disciplinary Failure

```
#table for disciplinary failure
dat %>%
  count(Disciplinary_failure)

## # A tibble: 2 x 2
##   Disciplinary_failure     n
##   <ord>                <int>
## 1 0                      700
## 2 1                       40

#bar chart
ggplot(data = dat,
       aes(x = Disciplinary_failure)) +
  geom_bar() +
  theme_minimal()
```

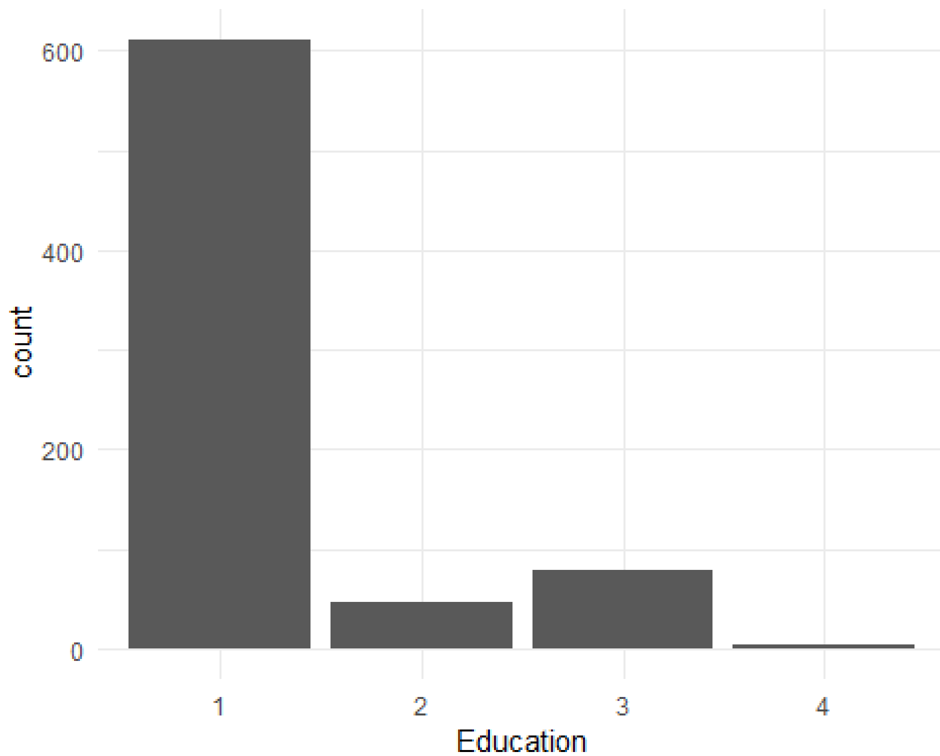



```r
#Scatterplots for variable 'Disciplinary_failure'
dat %>%
  gather(-Disciplinary_failure, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Disciplinary_failure)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

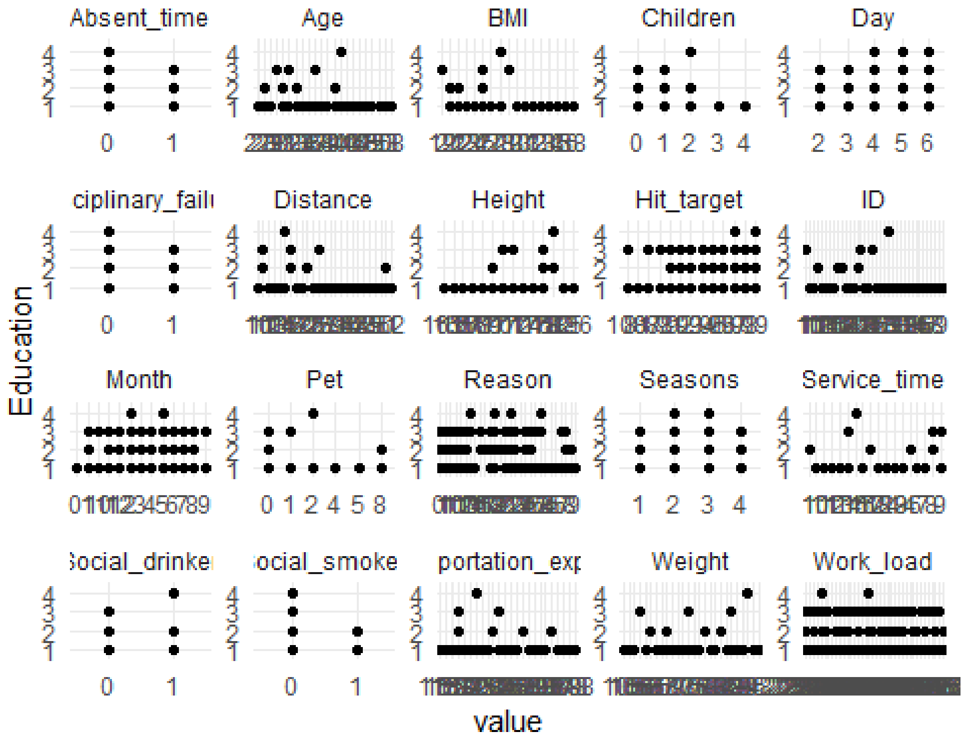


Education

```
#table for education
dat %>%
  count(Education)

## # A tibble: 4 x 2
##   Education     n
##   <ord>     <int>
## 1 1           611
## 2 2            46
## 3 3            79
## 4 4             4

#bar chart
ggplot(data = dat,
       aes(x = Education)) +
  geom_bar() +
  theme_minimal()
```

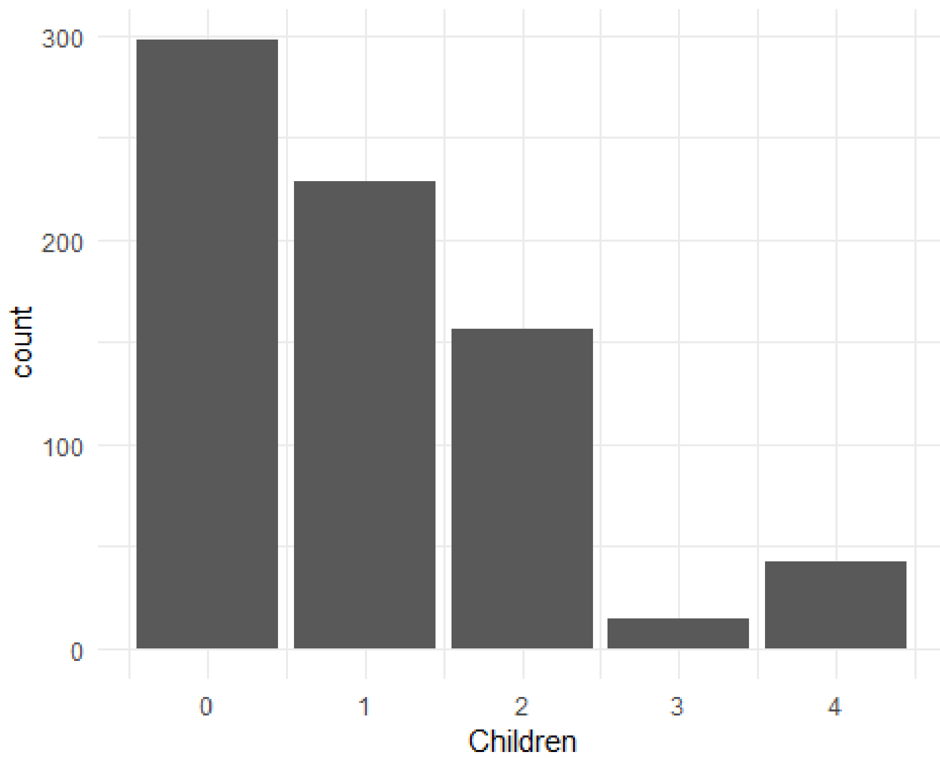



```
#Scatterplots for variable 'Education'
dat %>%
  gather(-Education, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Education)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

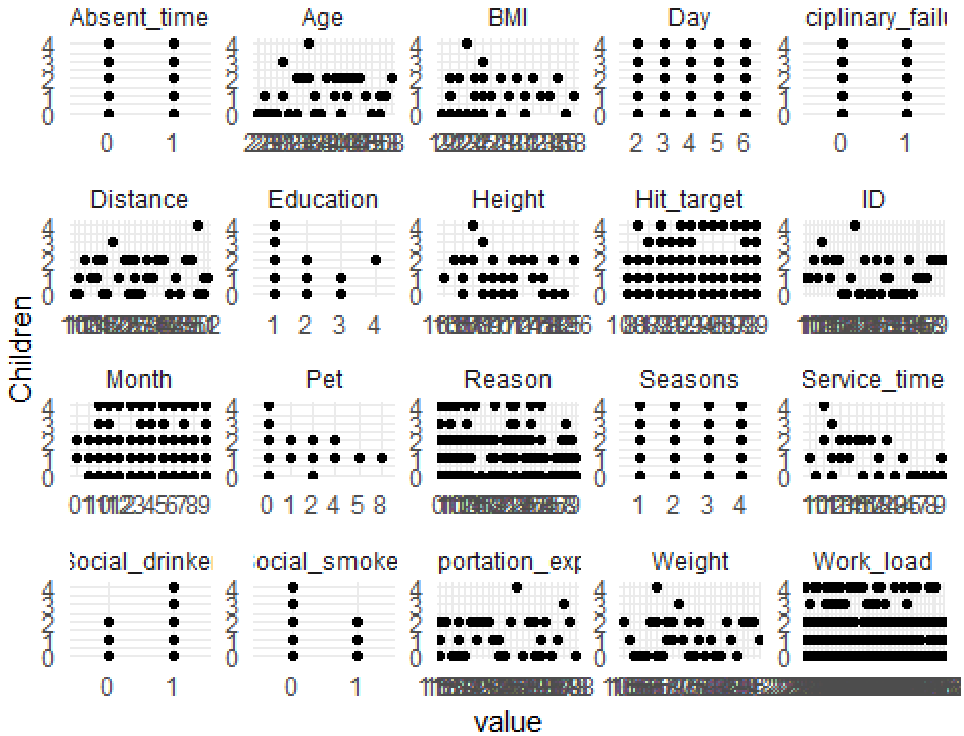


Children

```
#table for number of children
dat %>%
  count(Children)

## # A tibble: 5 x 2
##   Children     n
##      <dbl> <int>
## 1        0   298
## 2        1   229
## 3        2   156
## 4        3    15
## 5        4    42

#bar chart
ggplot(data = dat,
       aes(x = Children)) +
  geom_bar() +
  theme_minimal()
```

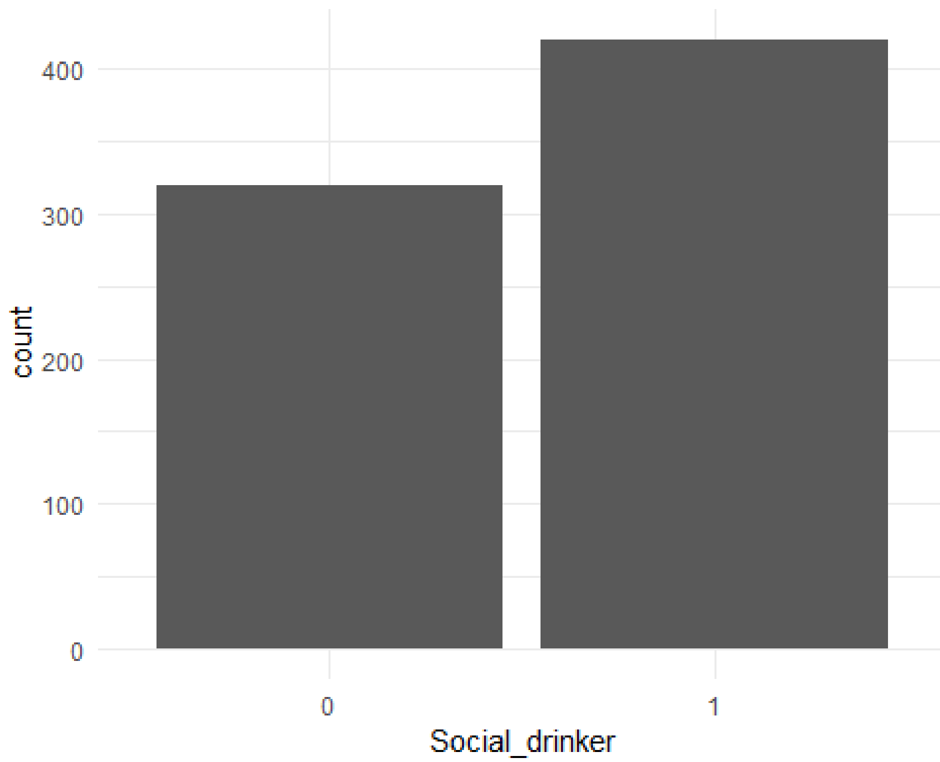



```
#Scatterplots for variable 'Children'
dat %>%
  gather(-Children, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Children)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

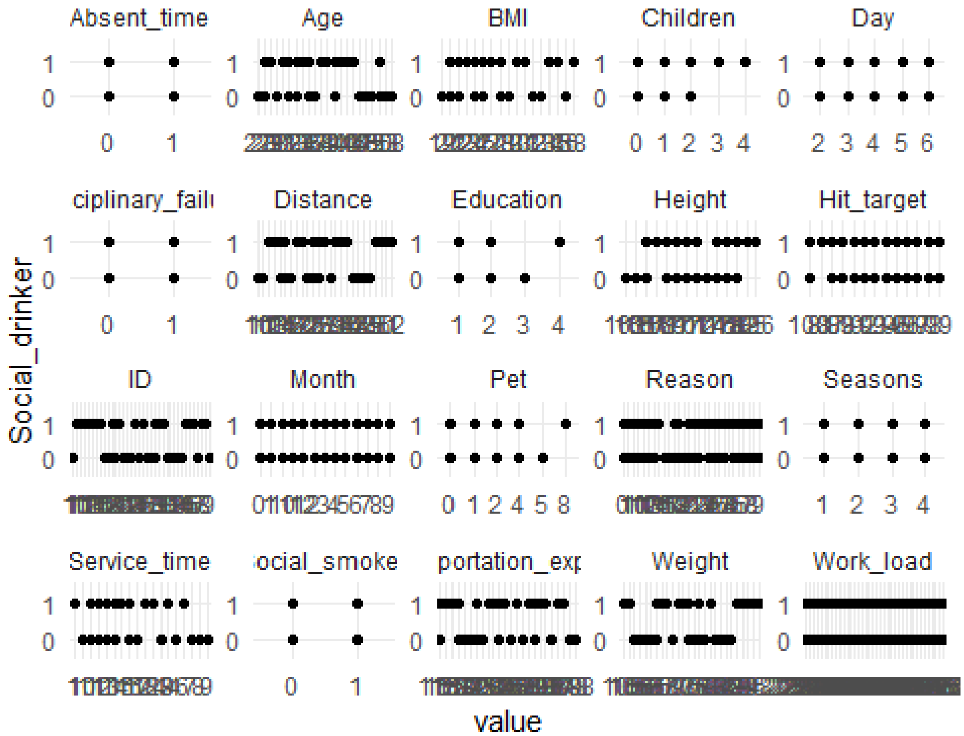


Social Drinker

```
#table for social drinking
dat %>%
  count(Social_drinker)

## # A tibble: 2 x 2
##   Social_drinker     n
##   <ord>          <int>
## 1 0                320
## 2 1                420

#bar chart
ggplot(data = dat,
       aes(x = Social_drinker)) +
  geom_bar() +
  theme_minimal()
```

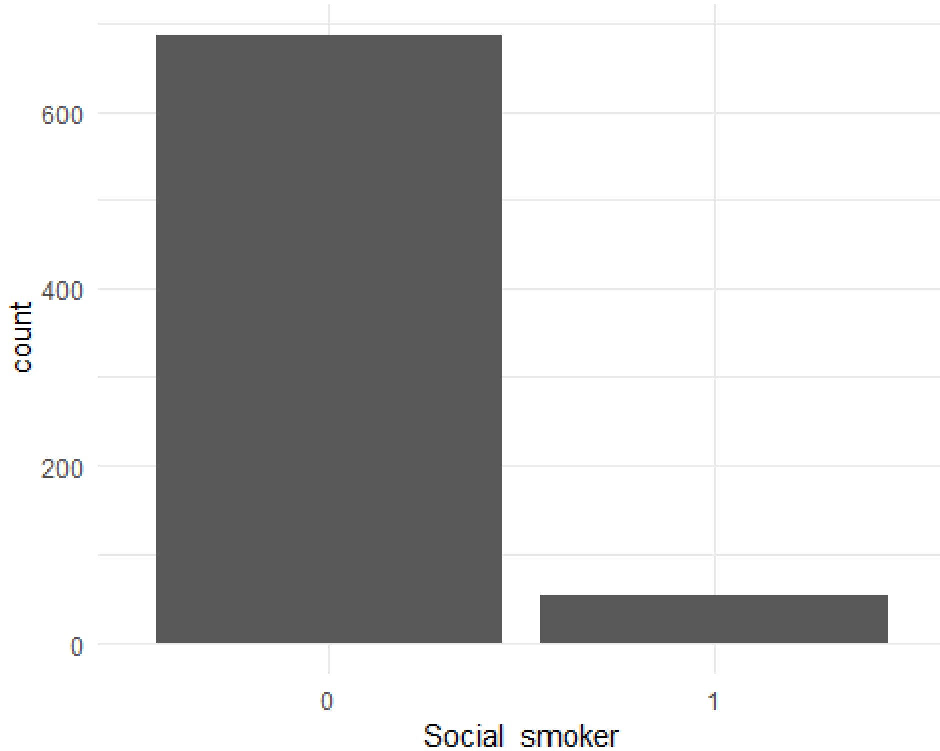



```r
#Scatterplots for variable 'Social_drinker'
dat %>%
  gather(-Social_drinker, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Social_drinker)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

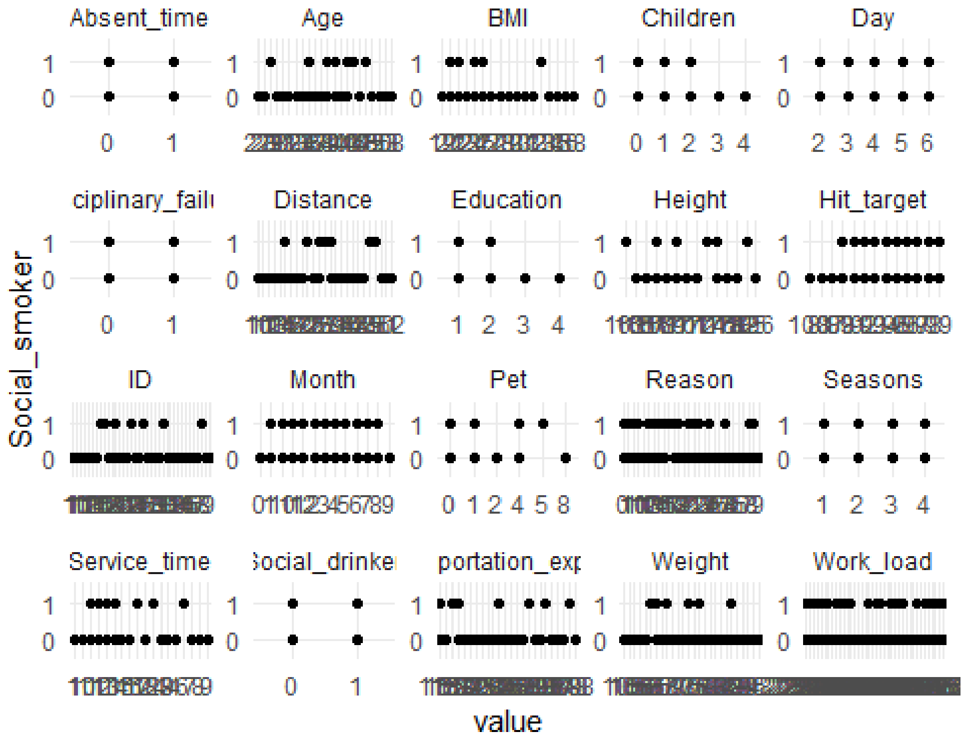


Social Smoker

```
#table for social smokers
dat %>%
  count(Social_smoker)

## # A tibble: 2 x 2
##   Social_smoker     n
##   <ord>         <int>
## 1 0               686
## 2 1                54

#bar chart
ggplot(data = dat,
       aes(x = Social_smoker)) +
  geom_bar() +
  theme_minimal()
```

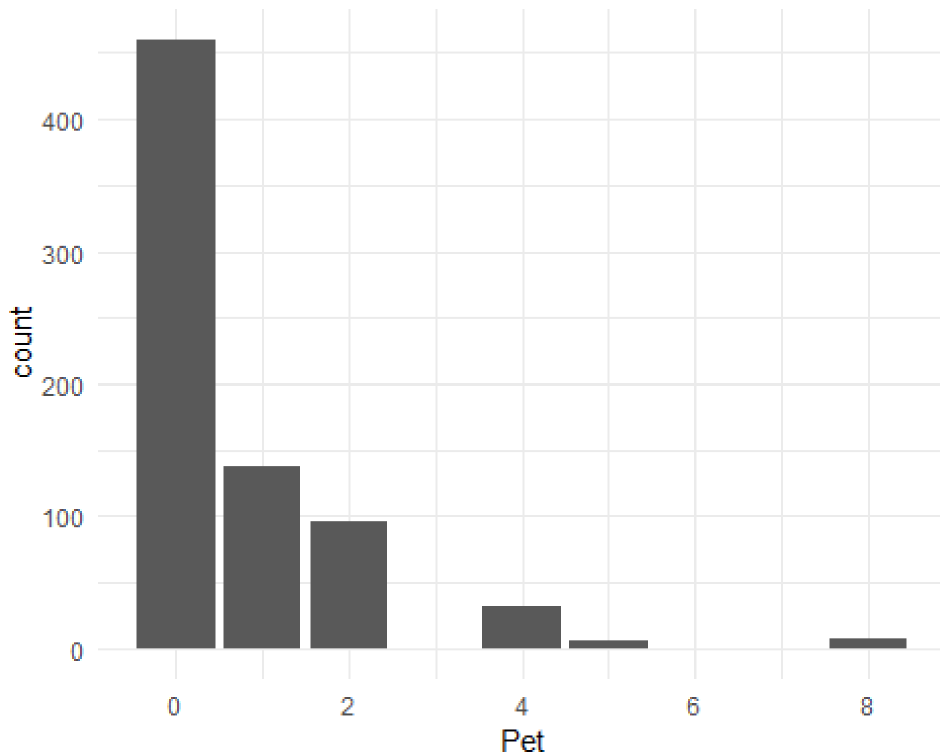



```r
#Scatterplots for variable 'Social_smoker'
dat %>%
  gather(-Social_smoker, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Social_smoker)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

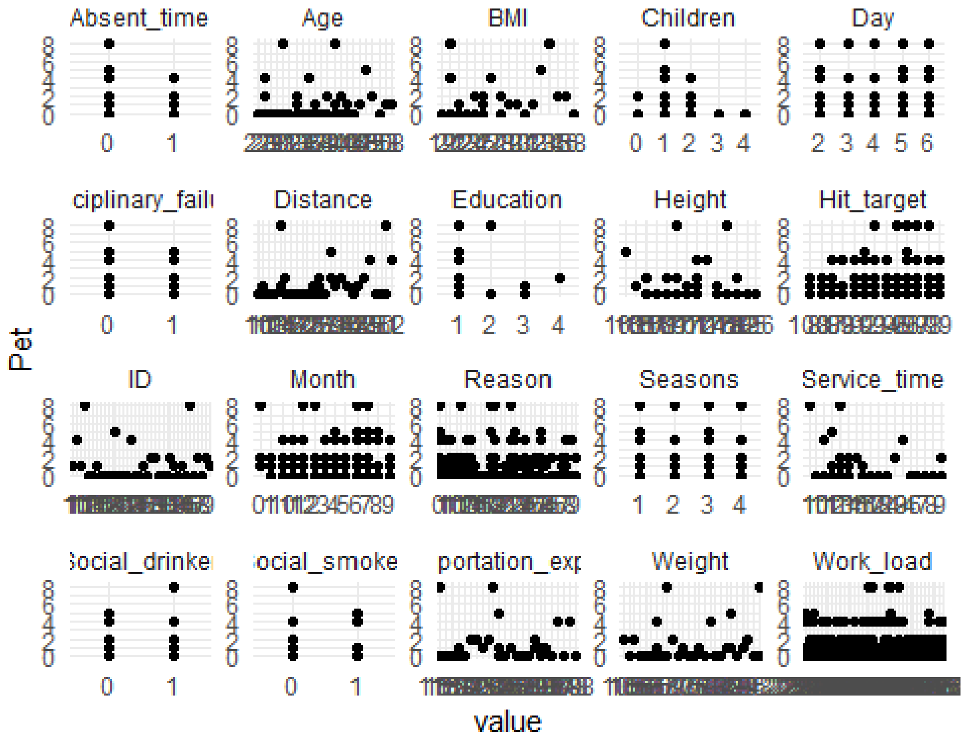


Pet

```
#summary for pets
summary(dat$Pet)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.7459  1.0000  8.0000

#histogram
ggplot(data = dat,
       aes(x = Pet)) +
  geom_bar() +
  theme_minimal()
```

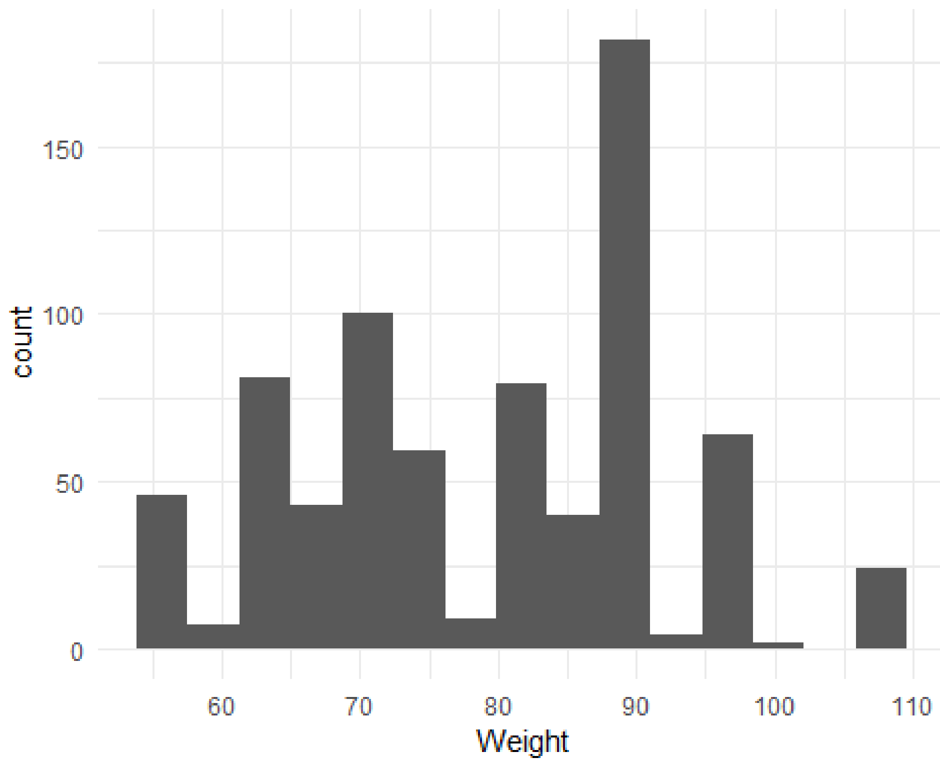



```
#Scatterplots for variable 'Pet'
dat %>%
  gather(-Pet, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Pet)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

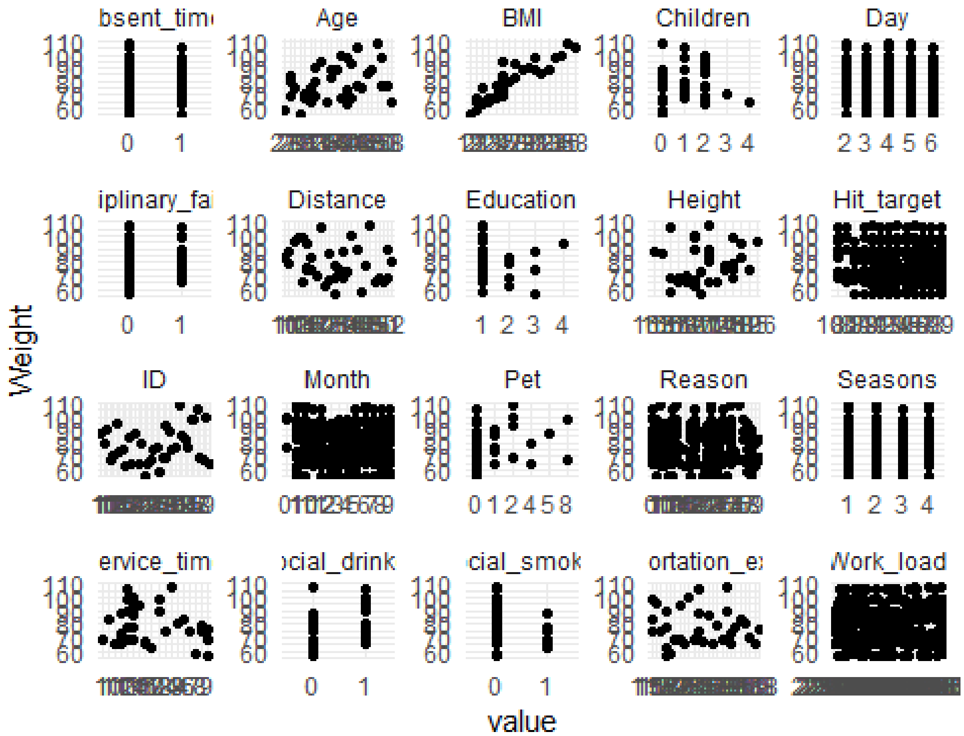


Weight

```
#summary of weight
summary(dat$Weight)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   56.00   69.00   83.00   79.04   89.00  108.00

#histogram
ggplot(data = dat,
       aes(x = Weight)) +
  geom_histogram(bins = 15) +
  theme_minimal()
```

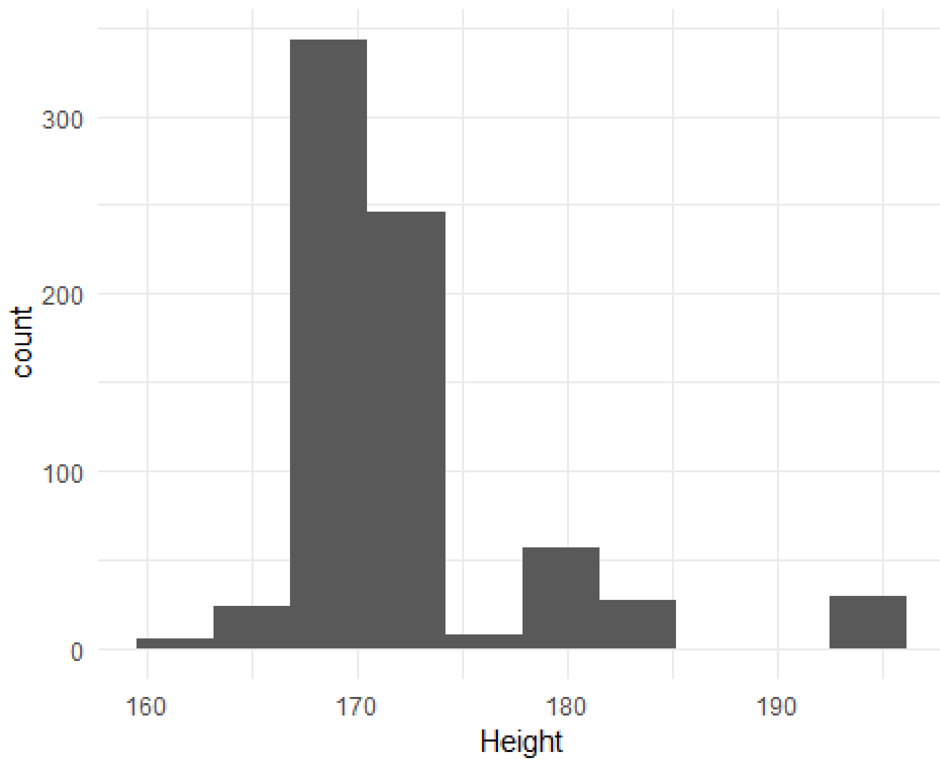



```
#Scatterplots for variable 'Weight'
dat %>%
  gather(-Weight, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Weight)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

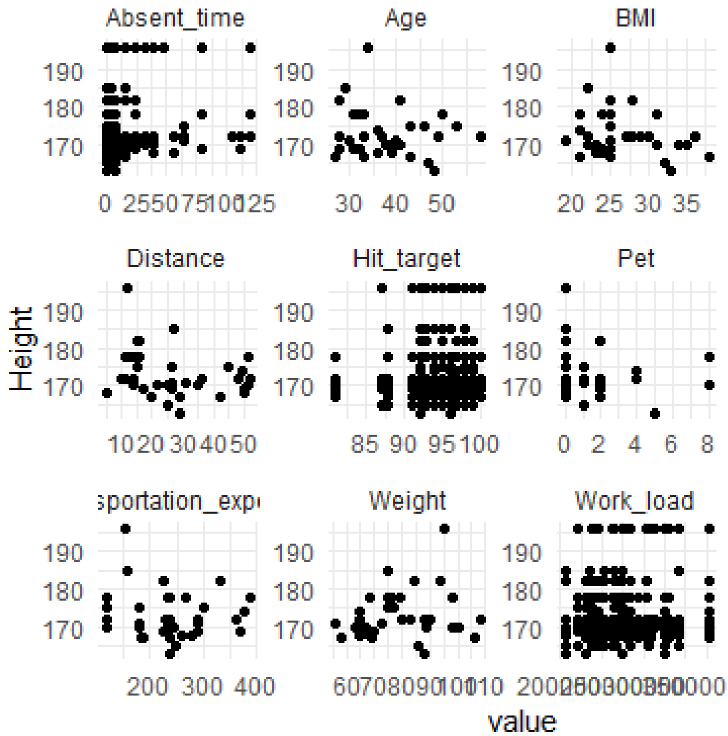


Height

```
#summary of height
summary(dat$Height)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   163.0   169.0   170.0   172.1   172.0   196.0

#histogram
ggplot(data = dat,
       aes(x = Height)) +
  geom_histogram(bins = 10) +
  theme_minimal()
```

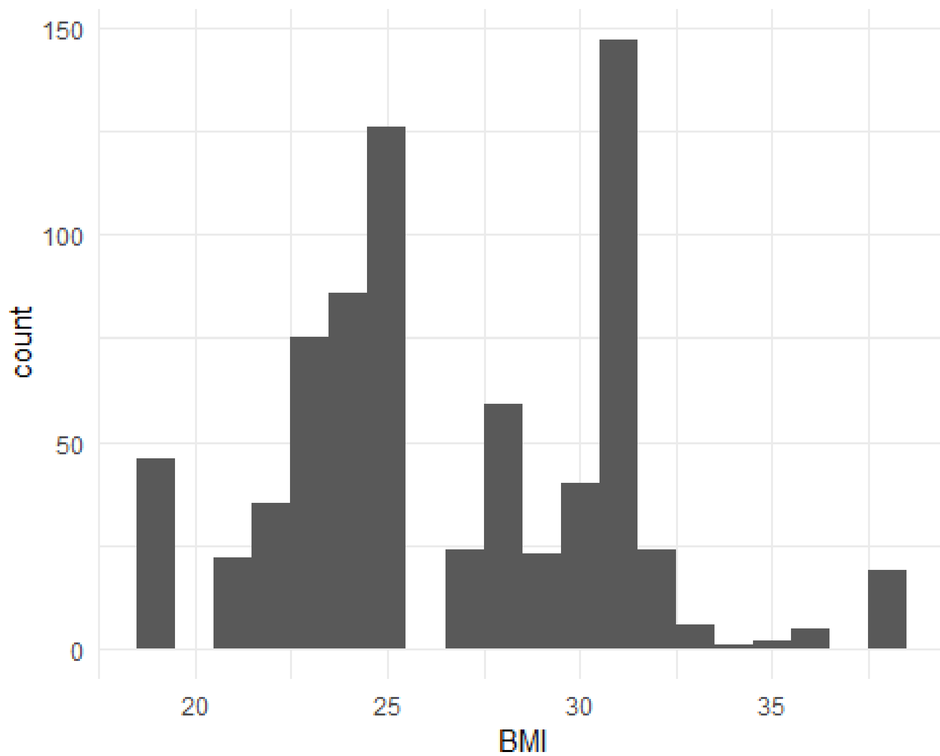



```r
#Scatterplots for variable 'Height'
dat.num %>%
  gather(-Height, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = Height)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```

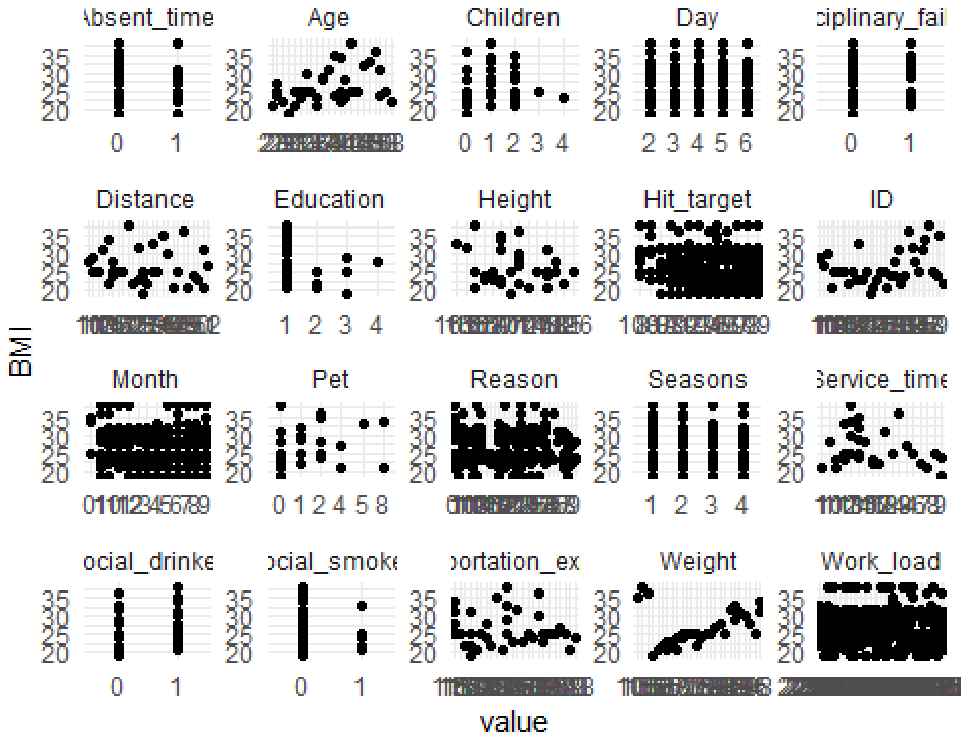


BMI

```
#summary for BMI
summary(dat$BMI)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   24.00   25.00   26.68   31.00   38.00

#histogram
ggplot(data = dat,
       aes(x = BMI)) +
  geom_histogram(binwidth = 1) +
  theme_minimal()
```




```r
#Scatterplots for variable 'BMI'
dat %>%
  gather(-BMI, key = "var_name", value = "value") %>%
  ggplot(aes(x = value, y = BMI)) +
  geom_point() +
  facet_wrap(~ var_name, scales = "free") +
  theme_minimal()
```



Additional Preprocessing

```
dat1 <- dat[-1]

#scale
scale <- sapply(dat1, is.numeric)
dat1[scale] <- lapply(dat1[scale],scale)
```

Initial Method Testing

```
R <- 50 # replications

# create the matrix to store values 1 row per model
err_matrix <- matrix(0, ncol=5, nrow=R)

sensitivity_matrix <- matrix(0, ncol=5, nrow=R)

fmeasure_matrix <- matrix(0, ncol=5, nrow=R)

gmean_matrix <- matrix(0, ncol=5, nrow=R)

# these are optional but I like to see how the model did each run so I can
check other output
KNNcm <- matrix(0, ncol=4, nrow=R)
glmcm <- matrix(0, ncol=4, nrow=R)
Treecm <- matrix(0, ncol=4, nrow=R)
rfcm <- matrix(0, ncol=4, nrow=R)
```

```
SVMcm <- matrix(0, ncol=4, nrow=R)

set.seed(1876)

for (r in 1:R){

# subsetting data to training and testing data
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat1), nrow(dat1)*p, replace=F)
data_train <-dat1[w,]
data_test <- dat1[-w,]

  ################################################################ knn

#Running the classifier

  knn <- knn(data_train[-20],
                       test = data_test[-20],
                       cl=data_train$Absent_time, k=2)

#predict doesn't work with KNN for factors
 knntable <- table(knn, data_test$Absent_time)

#generate confusion matrix
 cm_KNN <-  confusionMatrix(data = knntable, reference = data_test[,-20],
positive = "1")

 KNNcm [[r,1]] <-  cm_KNN$table[1,1]
 KNNcm [[r,2]] <-  cm_KNN$table[1,2]
 KNNcm [[r,3]] <-  cm_KNN$table[2,1]
 KNNcm [[r,4]] <-  cm_KNN$table[2,2]

 err_matrix [[r,1]] <-  (cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(
data_test)

  # store the errors (change the 1 to whichever model you have)

 sensitivity_matrix[[r, 1]] <- cm_KNN$byClass[1]

 fmeasure_matrix [[r, 1]] <- cm_KNN$byClass[7]

 gmean_matrix [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  ################################################################### GLM


  model_glm_1 = suppressWarnings(
    train(Absent_time ~ .,
```

```
                      data = data_train,
                      method = "glm",
                      family = 'binomial')
                      )

  yhat_glm = predict(model_glm_1, newdata = data_test[,-20])

  cm_glm = confusionMatrix(data = yhat_glm, reference = data_test[,20],
positive = "1")

  glmcm [[r,1]] <-  cm_glm$table[1,1]
  glmcm [[r,2]] <-  cm_glm$table[1,2]
  glmcm [[r,3]] <-  cm_glm$table[2,1]
  glmcm [[r,4]] <-  cm_glm$table[2,2]

  err_matrix [[r,2]] <-  (cm_glm$table[1,2]+cm_glm$table[2,1])/nrow(
data_test)

  # store the errors (change the 1 to whichever model you have)

  sensitivity_matrix[[r, 2]] <- cm_glm$byClass[1]

  fmeasure_matrix [[r, 2]] <- cm_glm$byClass[7]

  gmean_matrix [[r, 2]] <- sqrt(cm_glm$byClass[1]* cm_glm$byClass[2])

  #####################################################Decision Tree

  tree_mod = rpart(Absent_time ~ ., data = data_train)

  #prediction
  yhat_tree = predict(tree_mod, data_test, type = 'class')

  #generate confusion matrix
 cm_tree <-  confusionMatrix(data = table(yhat_tree, data_test$Absent_time),
reference = data_test[,-20], positive = "1")

 Treecm[[r,1]] <-  cm_tree$table[1,1]
 Treecm[[r,2]] <-  cm_tree$table[1,2]
 Treecm[[r,3]] <-  cm_tree$table[2,1]
 Treecm[[r,4]] <-  cm_tree$table[2,2]

  #store the errors
  err_matrix[r, 3] = mean(yhat_tree != data_test$Absent_time)

 sensitivity_matrix[[r, 3]] <- cm_tree$byClass[1]

 cm_tree$byClass[1]
```

```
fmeasure_matrix[[r, 3]] <- cm_tree$byClass[7]

gmean_matrix[[r, 3]] <- sqrt(cm_tree$byClass[1]* cm_tree$byClass[2])

#################################################### RF

  rf <- randomForest(Absent_time ~.,
                             data=data_train,
                             mtry=6,
                             ntree=50,
                             na.action=na.roughfix)

 yhat_rf = predict(rf, newdata = data_test, type= 'class')

 cm_rf = confusionMatrix(data = yhat_rf, reference = data_test[,20],
positive = "1")


 rfcm [[r,1]] <-  cm_rf$table[1,1]
 rfcm [[r,2]] <-  cm_rf$table[1,2]
 rfcm [[r,3]] <-  cm_rf$table[2,1]
 rfcm [[r,4]] <-  cm_rf$table[2,2]

err_matrix [[r,4]] <-  (cm_rf$table[1,2]+cm_rf$table[2,1])/nrow( data_test)

sensitivity_matrix[[r, 4]] <- cm_rf$byClass[1]

fmeasure_matrix[[r, 4]] <- cm_rf$byClass[7]

gmean_matrix[[r, 4]] <- sqrt(cm_rf$byClass[1]* cm_rf$byClass[2])

 ################################################################ SVM

 csvm_absent =   svm(Absent_time~., data=data_train,
                  type='C-classification')

 #prediction
 y_hat_csvm = predict(csvm_absent, data_test[,-20])

 cm_SVM = confusionMatrix(data = y_hat_csvm, reference = data_test[,20],
positive = "1")

 SVMcm [[r,1]] <-  cm_SVM$table[1,1]
 SVMcm [[r,2]] <-  cm_SVM$table[1,2]
 SVMcm [[r,3]] <-  cm_SVM$table[2,1]
 SVMcm [[r,4]] <-  cm_SVM$table[2,2]
```

```
  err_matrix[r,5] = (cm_SVM$table[1,2]+cm_SVM$table[2,1])/nrow(data_test)

  sensitivity_matrix[[r, 5]] <- cm_SVM$byClass[1]

  fmeasure_matrix [[r, 5]] <- cm_SVM$byClass[7]

  gmean_matrix [[r, 5]] <- sqrt(cm_SVM$byClass[1]* cm_SVM$byClass[2])

#statement indicates where in loop
  #cat("Finished Rep",r, "\n")
}
```

Change the matrix names to make easier to interpret

```
#rename the columns in the model
colnames(err_matrix) <- c("KNN","glm", "tree","RF", 'SVM')

colnames(sensitivity_matrix)<- c("KNN","glm", "tree","RF", 'SVM')

colnames(fmeasure_matrix) <- c("KNN","glm", "tree","RF", 'SVM')

colnames(gmean_matrix) <- c("KNN","glm", "tree","RF", 'SVM')


#rename the columns
colnames(KNNcm) <- c("True Negative","False Negative", "False Positive","True
Positive")

colnames(glmcm) <- c("True Negative","False Negative", "False Positive","True
Positive")

colnames(SVMcm) <- c("True Negative","False Negative", "False Positive","True
Positive")
```

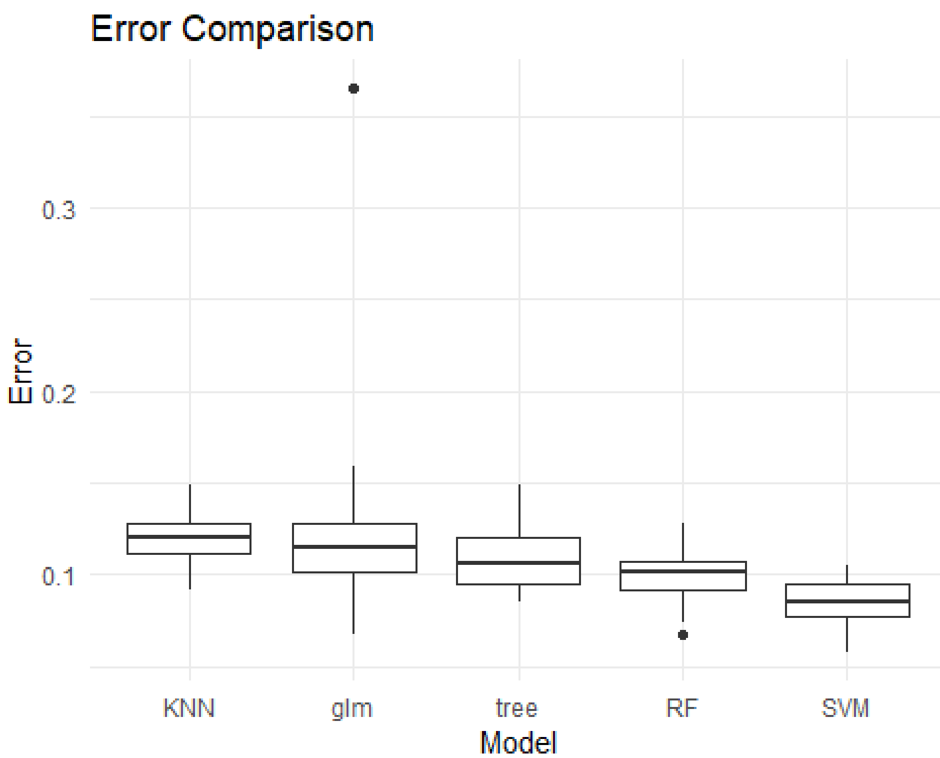
save output

```
save(err_matrix, file='errmatrix.RData')
save(sensitivity_matrix, file='sensmatrix.RData')
save(fmeasure_matrix, file='fmeasmatrix.RData')
save(gmean_matrix, file='gmeanmatrix.RData')
```

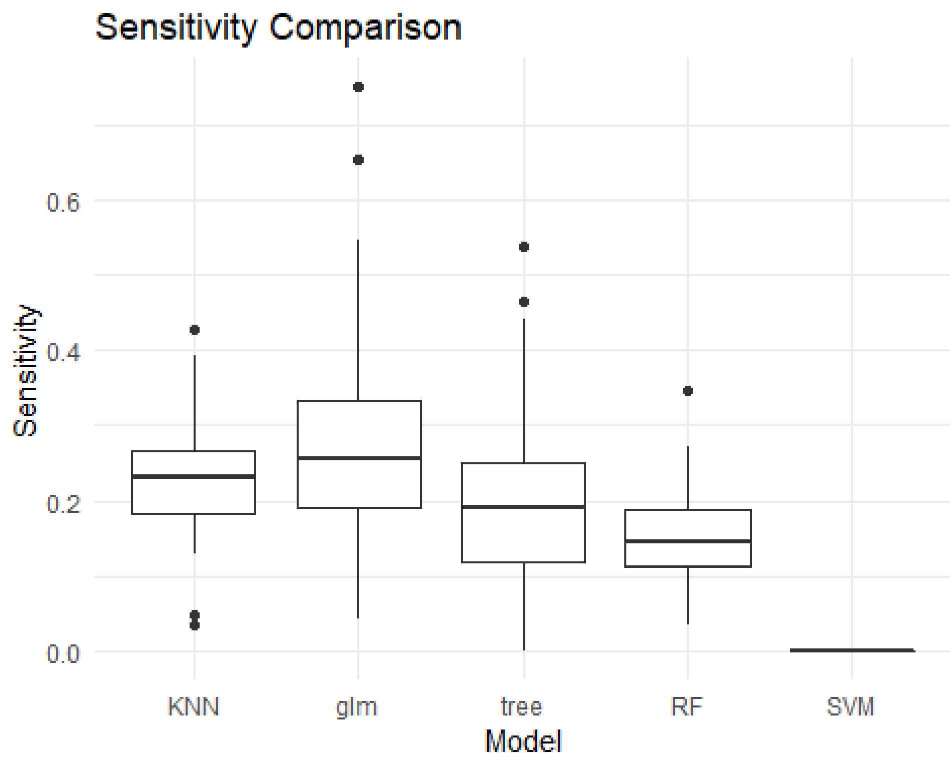
load output

```
load( file='errmatrix.RData')
load( file='sensmatrix.RData')
load( file='fmeasmatrix.RData')
load( file='gmeanmatrix.RData')

err_graph <- melt(err_matrix)
```

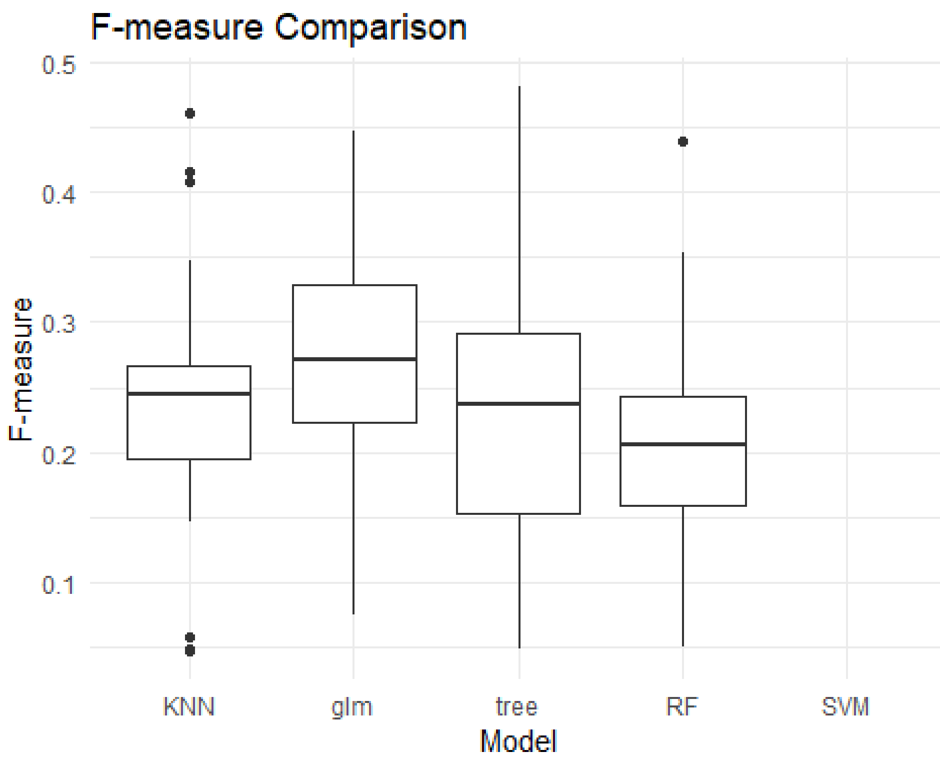
```r
ggplot(err_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal()+ labs(x= "Model", y= "Error", title="Error Comparison")
```



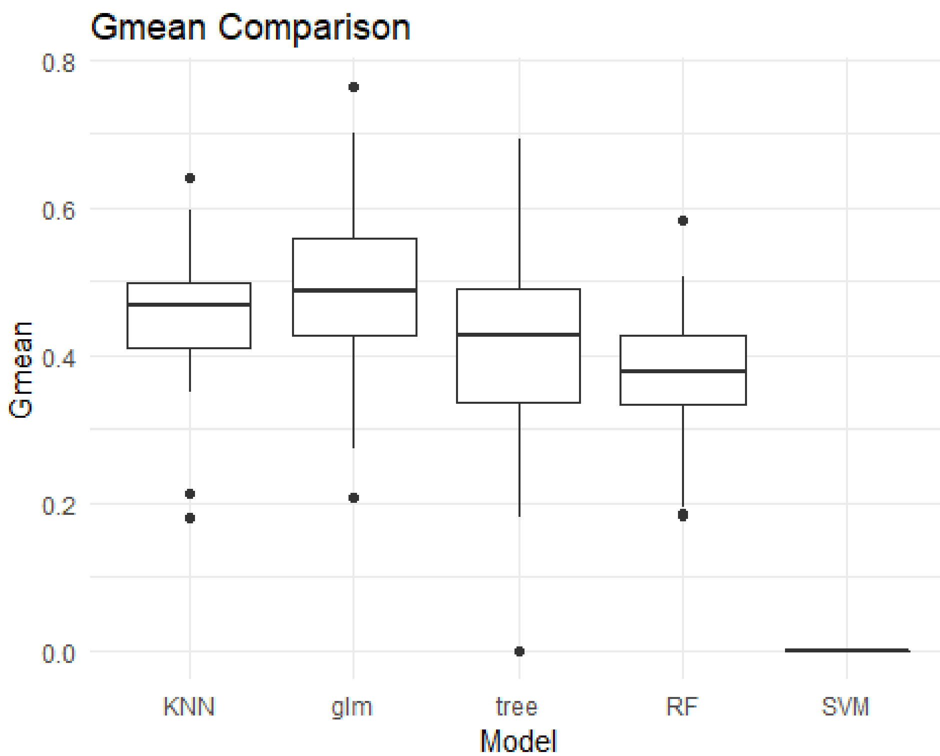

```r
sens_graph <- melt(sensitivity_matrix)

ggplot(sens_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal() + labs(x= "Model", y= "Sensitivity", title="Sensitivity
Comparison")
```




```
fmeas_graph <- melt(fmeasure_matrix)

ggplot(fmeas_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal() + labs(x= "Model", y= "F-measure", title="F-measure
Comparison")
```




```r
gmean_graph <- melt(gmean_matrix)

ggplot(gmean_graph,
       aes(x=Var2, y=value)) +
  geom_boxplot() +
  theme_minimal() + labs(x= "Model", y= "Gmean", title="Gmean Comparison")
```



From this we selected the KNN model

KNN Optimization

Initial Attempt: Optimize on Error

```
dat <- read_excel("Absenteeism_at_work.xls")
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
dat[col] <- lapply(dat[col], as.factor)
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")

#change variable represent missed time one day or greater
dat <- dat %>% mutate(Absent_time= ifelse(dat$Absent_time <=8,0,1))
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

###Optimizing the KNN

#For the tunning of the KNN model, we are going to create another
traning/test data sets.
```

```
#scaling the data:
dat_v <- dat #we are going to use dat_v for the manipulation
scale <- sapply(dat_v, is.numeric)
dat_v[scale] <- lapply(dat_v[scale],scale)

set.seed(1876)
#predicting class:
AB_class <- dat_v[, 21]
names(AB_class) <- c(1:nrow(dat_v))
dat_v$ID <- c(1:nrow(dat_v))

dat_v <- dat_v[1:737,]
rand_permute <- sample(x = nrow(dat_v), size = nrow(dat_v))

all_id_random <- dat_v[rand_permute, "ID"]
dat_v <- dat_v[,-1] #remove ID

set.seed(1876)
#random samples for training test
validate_id <- as.character(all_id_random[1:248])
training_id <- as.character(all_id_random[249:737])

dat_v_train <- dat_v[training_id, ]
dat_v_val <- dat_v[validate_id, ]
AB_class_train <- AB_class[training_id]
AB_class_val <- AB_class[validate_id]
table(AB_class_train)

## AB_class_train
##   0   1
## 448  41

set.seed(1876)
#Study significance of the variables
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat_v), nrow(dat_v)*p, replace=F)
data_train <-dat_v[w,]
data_test <- dat_v[-w,]


rf <- randomForest(Absent_time ~.,
                   data=data_train,
                   mtry=6,
                   ntree=50,
                   na.action=na.roughfix)

impfact <- importance(rf)

impfact <- as.list(impfact)
```

```r
names(impfact) <- colnames(dat_v[,-20])
impfact2 <- unlist(impfact)


most_sig_stats <- names(sort(desc(impfact2)))


#Re ordering variables by significance:

dat_v_train_ord <- dat_v_train[ c(most_sig_stats)]
str(dat_v_train_ord)

## 'data.frame':    489 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 1 23 25
25 24 18 27 2 28 26 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 4 8 6 4
9 4 3 11 5 7 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 3 4 1 3
2 2 2 1 3 5 ...
##  $ Work_load             : num [1:489, 1] -0.694 -0.818 -0.651 -1.262 -
1.679 ...
##  $ Hit_target            : num [1:489, 1] 0.903 0.638 1.167 1.167 -0.685
...
##  $ Age                   : num [1:489, 1] -0.841 -0.533 -0.996 3.326 -
0.533 ...
##  $ BMI                   : num [1:489, 1] -0.391 0.775 -1.791 -1.091 0.775
...
##  $ Weight                : num [1:489, 1] -0.701 0.851 -1.788 -1.089 0.851
...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 2 1 3 2 1 2
2 4 3 3 ...
##  $ Height                : num [1:489, 1] -0.516 -0.019 -0.185 -0.019 -
0.019 ...
##  $ Transportation_expense: num [1:489, 1] 2.2056 1.0107 -0.6322 0.0996
1.0107 ...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 1 1 2 1 2 1 2 1
...
##  $ Service_time          : num [1:489, 1] -0.126 0.102 -0.811 0.786 0.102
...
##  $ Children              : num [1:489, 1] 1.803 0.893 -0.928 0.893 0.893
...
##  $ Distance              : num [1:489, 1] -0.851 0.429 -0.245 -1.054 0.429
...
##  $ Pet                   : num [1:489, 1] -0.566 0.193 -0.566 0.193 0.193
...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 3 1 1 2
1 3 1 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1
...
```

```
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1
...

dat_v_val_ord <- dat_v_val[, names(dat_v_train_ord)]

#Monte Carlo Validation:
set.seed(1876)
size <- length(training_id)
red.size <- (2/3) * length(training_id)

training_family_L <- lapply(1:500, function(j) {
  perm <- sample(1:size, size = size, replace = F)
  shuffle <- training_id[perm]
  trn <- shuffle[1:326]
  trn
})

validation_family_L <- lapply(training_family_L,
                              function(x) setdiff(training_id, x))

#Finding an optimal set of variables and optimal k
set.seed(1876)
N <- seq(from = 2, to = 19, by = 1)

K <- seq(from = 1, to = 7, by = 1)
times <- 500 * length(N) * length(K)

set.seed(1876)
paramter_errors_df <- data.frame(mc_index = as.integer(rep(NA, times =
times)),
                                 var_num = as.integer(rep(NA, times =
times)),
                                 k = as.integer(rep(NA, times = times)),
                                 error = as.numeric(rep(NA, times = times)))

#Core knn_model:
# j = index, n = length of range of variables, k=k
core_knn <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)
  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])
  err <- (tbl[1, 2] + tbl[2, 1])/(tbl[1, 2] + tbl[2, 1]+tbl[1, 1] + tbl[2,
2])
  err
}

set.seed(1876)
param_df1 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df <- merge(param_df1, data.frame(k = K))
```

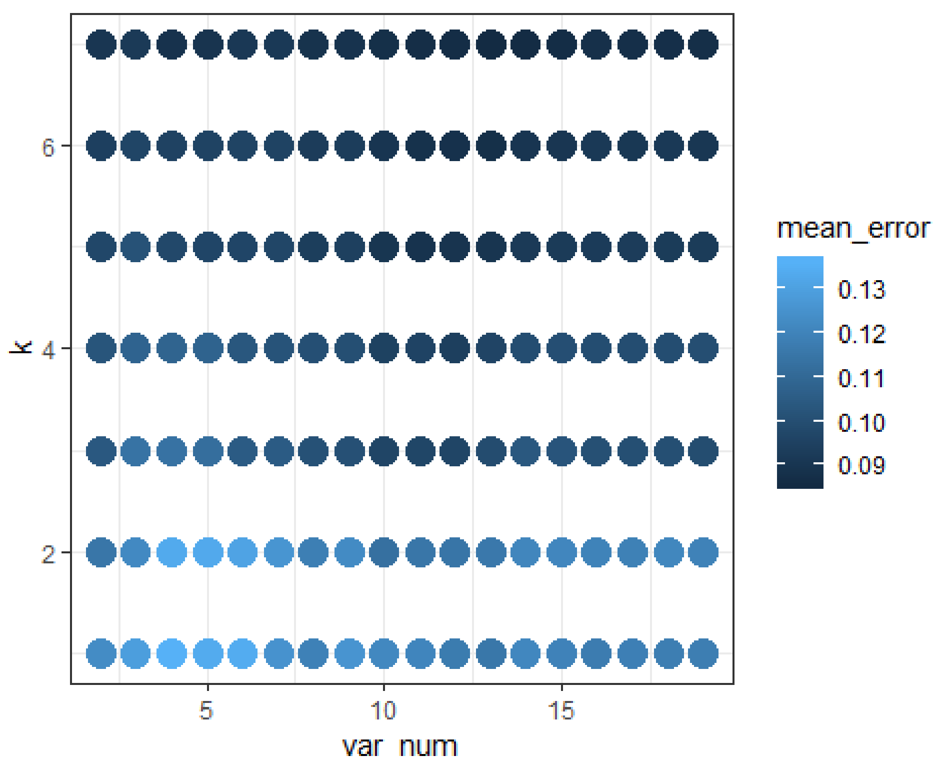
```
knn_err_est_df <- ddply(param_df[1:times, ], .(mc_index, var_num, k),
function(df) {
  err <- core_knn(df$mc_index[1], df$var_num[1], df$k[1])
  err
})

head(knn_err_est_df)
names(knn_err_est_df)[4] <- "error"

mean_errs_df <- ddply(knn_err_est_df, .(var_num, k), function(df)
mean(df$error))
head(mean_errs_df)
names(mean_errs_df)[3] <- "mean_error"
```

save output

```
save(mean_errs_df, file='mean_errs_df.RData')
```

load output

```
load( file='mean_errs_df.RData')

ggplot(data = mean_errs_df, aes(x = var_num, y = k, color = mean_error)) +
geom_point(size = 5) +
  theme_bw()
```



```r
#selects top model
mean_errs_df[which.min(mean_errs_df$mean_error), ]
```

```
##    var_num k mean_error
## 84      13 7 0.08592638
```

```r
#list in order of models
mean_errs_df %>% arrange(mean_error)
```

```
##     var_num k mean_error
## 1        13 7 0.08592638
## 2        14 7 0.08638037
## 3        12 7 0.08699387
## 4        15 7 0.08699387
## 5        11 7 0.08731288
## 6        19 7 0.08755828
## 7        17 7 0.08775460
## 8        18 7 0.08779141
## 9        10 7 0.08788957
## 10       13 6 0.08796319
## 11       16 7 0.08822086
## 12       12 6 0.08869939
## 13       11 6 0.08885890
## 14        4 7 0.08903067
## 15        8 7 0.08922699
## 16        9 7 0.08941104
## 17       11 5 0.08944785
## 18        5 7 0.08977914
## 19       12 5 0.08980368
## 20       13 5 0.09011043
## 21       10 6 0.09026994
## 22       14 6 0.09034356
## 23       10 5 0.09040491
## 24        2 7 0.09063804
## 25       15 6 0.09068712
## 26       19 6 0.09094479
## 27       17 6 0.09125153
## 28        6 7 0.09152147
## 29        7 7 0.09176687
## 30       16 6 0.09177914
## 31       18 6 0.09181595
## 32        3 7 0.09223313
## 33       14 5 0.09230675
## 34       15 5 0.09247853
## 35       16 5 0.09279755
## 36       19 5 0.09294479
## 37       18 5 0.09295706
## 38        8 6 0.09301840
## 39       17 5 0.09303067
## 40        9 6 0.09342331
```

```
## 41          8 5 0.09371779
## 42         12 4 0.09438037
## 43          2 6 0.09451534
## 44          9 5 0.09465031
## 45         10 4 0.09538650
## 46          4 6 0.09548466
## 47          5 6 0.09588957
## 48          7 6 0.09591411
## 49         11 4 0.09614724
## 50          6 6 0.09625767
## 51         13 4 0.09633129
## 52         12 3 0.09663804
## 53          3 6 0.09665031
## 54         10 3 0.09668712
## 55          6 5 0.09690798
## 56          5 5 0.09693252
## 57         11 3 0.09699387
## 58          7 5 0.09710429
## 59          2 5 0.09763190
## 60          4 5 0.09803681
## 61         13 3 0.09948466
## 62         19 3 0.09975460
## 63         16 4 0.09991411
## 64         17 4 0.10002454
## 65         18 4 0.10006135
## 66         14 4 0.10012270
## 67         19 4 0.10022086
## 68         15 4 0.10051534
## 69         17 3 0.10067485
## 70          9 4 0.10068712
## 71         18 3 0.10069939
## 72          8 4 0.10073620
## 73         16 3 0.10101840
## 74          9 3 0.10120245
## 75          8 3 0.10153374
## 76          3 5 0.10174233
## 77          7 4 0.10222086
## 78         15 3 0.10285890
## 79          2 4 0.10293252
## 80          6 4 0.10379141
## 81         14 3 0.10411043
## 82          2 3 0.10424540
## 83          7 3 0.10541104
## 84          6 3 0.10547239
## 85          5 4 0.10835583
## 86          3 4 0.10856442
## 87          4 4 0.10865031
## 88          5 3 0.11258896
## 89         10 2 0.11334969
## 90          4 3 0.11409816
```

```
## 91        3 3 0.11435583
## 92       12 2 0.11512883
## 93        2 2 0.11559509
## 94       11 2 0.11564417
## 95       13 1 0.11592638
## 96       13 2 0.11631902
## 97       12 1 0.11754601
## 98       16 1 0.11775460
## 99       18 1 0.11835583
## 100      19 1 0.11838037
## 101       8 2 0.11883436
## 102      17 1 0.11893252
## 103      17 2 0.11934969
## 104       8 1 0.11949693
## 105      19 2 0.11990184
## 106      15 1 0.12003681
## 107      16 2 0.12025767
## 108      11 1 0.12107975
## 109      14 2 0.12130061
## 110      15 2 0.12138650
## 111      18 2 0.12155828
## 112      14 1 0.12175460
## 113      10 1 0.12195092
## 114       3 2 0.12245399
## 115       9 2 0.12316564
## 116       2 1 0.12338650
## 117       7 1 0.12569325
## 118       9 1 0.12624540
## 119       7 2 0.12644172
## 120       3 1 0.12961963
## 121       6 2 0.13150920
## 122       5 2 0.13343558
## 123       4 2 0.13374233
## 124       5 1 0.13412270
## 125       6 1 0.13449080
## 126       4 1 0.13629448
```

The model predicts that error can be reduced by simply predicting a large K-value. This forces the models to chose all major class which explains why the error is ~8% in each model metric.

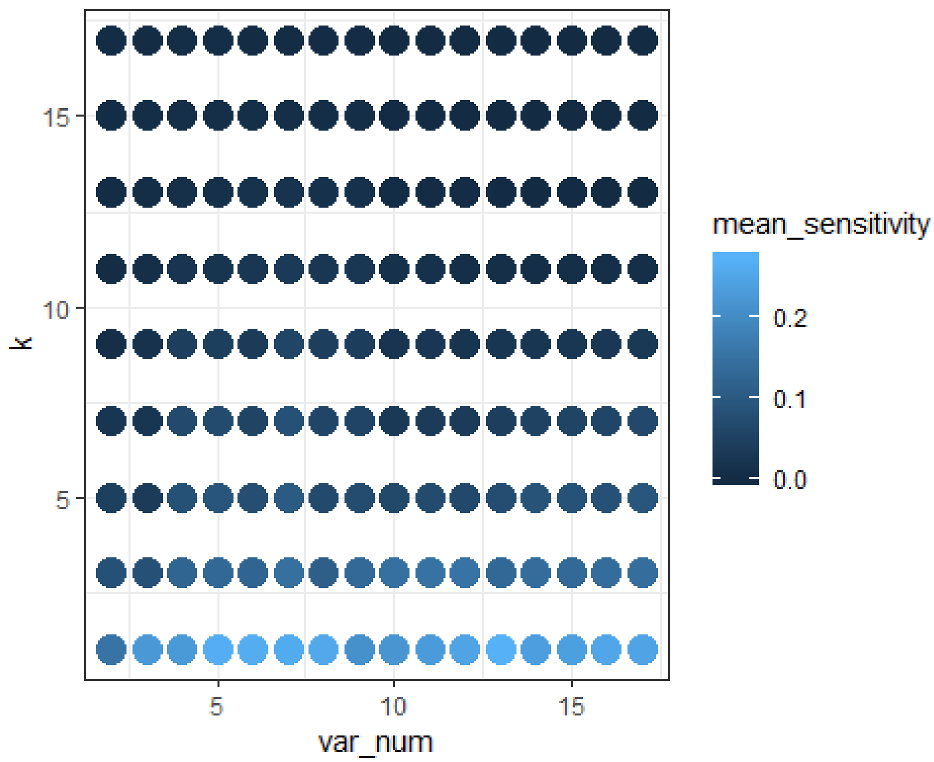
Second Attempt: Optimize on Sensitivity

```
N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 18.05547

K <- seq(from = 1, to = 7, by = 2)
times <- 500 * length(N) * length(K)
```

```r
core_knn_sen <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)

  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])

  #generate confusion matrix
  cm_KNN <-  confusionMatrix(data = tbl, reference
=AB_class_train[validation_family_L[[j]]], positive = "1")

  sen <- cm_KNN$byClass[1]
  sen
}

param_df1_2 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df_2 <- merge(param_df1_2, data.frame(k = K))

knn_err_est_df_2 <- ddply(param_df_2[1:times, ], .(mc_index, var_num, k),
function(df) {
  sen <- core_knn_sen(df$mc_index[1], df$var_num[1], df$k[1])
  sen
})

names(knn_err_est_df_2)[4] <- "Sensitivity"

mean_sens_df <- ddply(knn_err_est_df_2, .(var_num, k), function(df)
mean(df$Sensitivity))

names(mean_sens_df)[3] <- "mean_sensitivity"

save(mean_sens_df, file='mean_sens_df.RData')
```

load output

```r
load( file='mean_sens_df.RData')

ggplot(data = mean_sens_df, aes(x = var_num, y = k, color =
mean_sensitivity)) + geom_point(size = 5) +
  theme_bw()
```




```
mean_sens_df[which.max(mean_sens_df$mean_sensitivity), ]

##     var_num k mean_sensitivity
## 100      13 1        0.2716719

mean_sens_df %>% arrange(desc(mean_sensitivity))

##     var_num  k mean_sensitivity
## 1        13  1     0.2716718922
## 2         5  1     0.2666981059
## 3         6  1     0.2628591320
## 4         7  1     0.2605046872
## 5         8  1     0.2552407844
## 6        16  1     0.2517553543
## 7        12  1     0.2480793032
## 8        17  1     0.2480296623
## 9        15  1     0.2397691201
## 10       14  1     0.2365260921
## 11       11  1     0.2308062550
## 12        4  1     0.2286541914
## 13        3  1     0.2252105500
## 14       10  1     0.2196269051
## 15        9  1     0.2096249353
## 16        2  1     0.1545895840
## 17       12  3     0.1542561748
## 18       11  3     0.1507270254
## 19       10  3     0.1479739460
```

```
## 20         7  3      0.1469051222
## 21        17  3      0.1439008296
## 22        14  3      0.1407103771
## 23        16  3      0.1398890725
## 24         9  3      0.1333000615
## 25        15  3      0.1329005332
## 26        13  3      0.1318616785
## 27         5  3      0.1314689143
## 28         6  3      0.1257157165
## 29         4  3      0.1250362667
## 30         8  3      0.1135097359
## 31         7  5      0.1029324995
## 32        17  5      0.0951682128
## 33         5  5      0.0921849674
## 34        14  5      0.0888163290
## 35        15  5      0.0859068330
## 36         4  5      0.0850014887
## 37         7  7      0.0845910357
## 38         2  3      0.0837267779
## 39        16  5      0.0828803531
## 40         3  3      0.0826648706
## 41         6  5      0.0776177756
## 42        13  5      0.0760371061
## 43         9  5      0.0724752027
## 44        11  5      0.0715366975
## 45         5  7      0.0710006739
## 46         8  5      0.0701061563
## 47         4  7      0.0695262087
## 48        17  7      0.0670996216
## 49        12  5      0.0665066364
## 50        10  5      0.0652124881
## 51         8  7      0.0590302359
## 52         7  9      0.0587448537
## 53        16  7      0.0580783352
## 54         9  7      0.0571237870
## 55         6  7      0.0557918592
## 56        15  7      0.0551483043
## 57        14  7      0.0517686485
## 58         2  5      0.0473679592
## 59         5  9      0.0453139935
## 60         8  9      0.0443125481
## 61         4  9      0.0442821882
## 62        13  7      0.0425773248
## 63         9  9      0.0414619909
## 64        11  7      0.0372985831
## 65        12  7      0.0366790676
## 66         3  5      0.0361242991
## 67         6  9      0.0359191392
## 68         7 11      0.0347852958
## 69        10  7      0.0317932871
```

```
## 70        17  9     0.0313440351
## 71        16  9     0.0308590645
## 72        11  9     0.0280343252
## 73         8 11     0.0268810348
## 74        15  9     0.0256688937
## 75         6 11     0.0250697494
## 76         9 11     0.0250661199
## 77        13  9     0.0247272889
## 78         3  7     0.0242968605
## 79        14  9     0.0237716903
## 80         2  7     0.0231146342
## 81        12  9     0.0223681822
## 82         5 11     0.0221788938
## 83        10  9     0.0216155312
## 84         4 11     0.0215935200
## 85         7 13     0.0181307948
## 86         3  9     0.0165409038
## 87         6 13     0.0160554212
## 88         8 13     0.0149073185
## 89         9 13     0.0142500535
## 90        11 11     0.0132313608
## 91        10 11     0.0132233833
## 92         5 13     0.0121080481
## 93         4 13     0.0118604392
## 94         3 11     0.0104161169
## 95        17 11     0.0099627849
## 96         7 15     0.0098183963
## 97         5 15     0.0093792438
## 98         2  9     0.0092919394
## 99        12 11     0.0091332200
## 100        6 15     0.0086553094
## 101       15 11     0.0085188088
## 102       16 11     0.0083726568
## 103        4 15     0.0083231166
## 104       13 11     0.0082022169
## 105        5 17     0.0079976246
## 106       14 11     0.0077761245
## 107        9 15     0.0069795427
## 108        4 17     0.0067086136
## 109        3 13     0.0066638406
## 110        3 15     0.0066516036
## 111        8 15     0.0063581641
## 112        2 13     0.0054166823
## 113        2 17     0.0048990065
## 114        2 15     0.0045816267
## 115       10 13     0.0043574426
## 116        2 11     0.0042295629
## 117        3 17     0.0040418479
## 118        6 17     0.0037377511
## 119        7 17     0.0037298146
```

```
## 120      11 13     0.0034211566
## 121       8 17     0.0033093018
## 122       9 17     0.0028870796
## 123      12 13     0.0026548868
## 124      17 13     0.0021719122
## 125      14 13     0.0017992285
## 126      16 13     0.0015670857
## 127      15 13     0.0011293151
## 128      13 13     0.0010770063
## 129      10 15     0.0010506716
## 130      11 15     0.0008840049
## 131      17 15     0.0006172161
## 132      16 15     0.0005317460
## 133      17 17     0.0005023310
## 134      12 15     0.0004444444
## 135      15 17     0.0003356643
## 136      16 17     0.0003356643
## 137      15 15     0.0003095238
## 138      12 17     0.0002222222
## 139      13 15     0.0002222222
## 140      13 17     0.0002222222
## 141      14 15     0.0001428571
## 142      10 17     0.0000000000
## 143      11 17     0.0000000000
## 144      14 17     0.0000000000
```

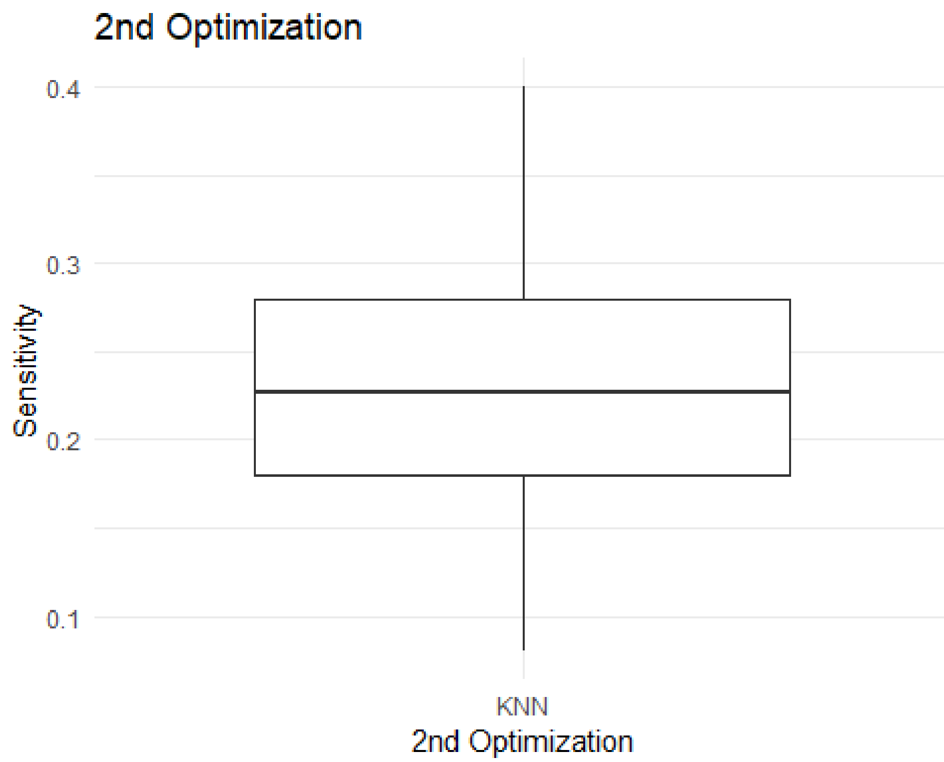
```
#Best KNN:

KNN_13_1 <- knn(train = dat_v_train_ord[, 1:13],
               dat_v_val_ord[, 1:13], AB_class_train,
               k = 1)

tbl_bm_val <- table(KNN_13_1, AB_class_val)
tbl_bm_val
```

```
##          AB_class_val
## KNN_13_1   0   1
##          0 213  16
##          1  13   6
```

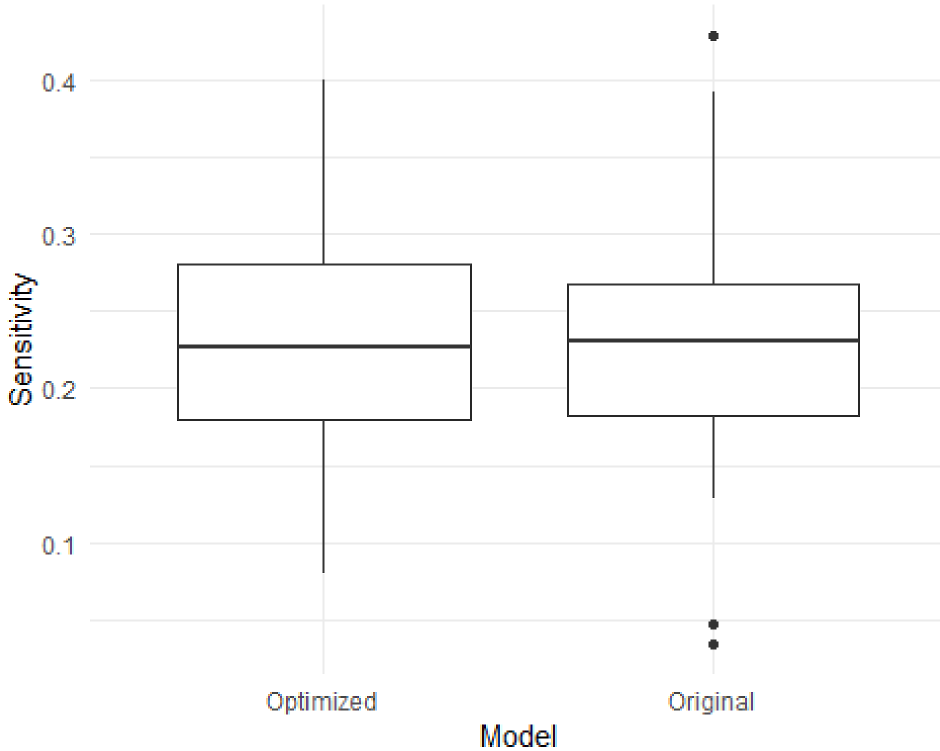
```
cm_KNN_opt <-  confusionMatrix(data = tbl_bm_val, reference = dat_v_val_ord[,
1:13], positive = "1")

R <- 50 # replications

# create the matrix to store values 1 row per model
err_matrix_opt <- matrix(0, ncol=1, nrow=R)

sensitivity_matrix_opt <- matrix(0, ncol=1, nrow=R)
```

```r
fmeasure_matrix_opt <- matrix(0, ncol=1, nrow=R)

gmean_matrix_opt <- matrix(0, ncol=1, nrow=R)

KNNcm <- matrix(0, ncol=4, nrow=R)

dat_smaller <- dat[, names(dat_v_train_ord)]
dat_smaller[,20] <- dat$Absent_time

dat_smaller <- dat_smaller[1:737,] # remove lines with non-meaningful data

scale <- sapply(dat_smaller, is.numeric)
dat_smaller[scale] <- lapply(dat_smaller[scale],scale)
head(dat_smaller)

##   Reason Month Day   Work_load Hit_target        Age        BMI     Weight
## 1     26     7   3 -0.8160263  0.6374158 -0.5292037  0.7818833  0.8561660
## 2      0     7   3 -0.8160263  0.6374158  2.1019046  1.0158452  1.4779119
## 3     23     7   4 -0.8160263  0.6374158  0.2446517  1.0158452  0.7784478
## 4      7     7   5 -0.8160263  0.6374158  0.3994228 -0.6218877 -0.8536352
## 5     23     7   5 -0.8160263  0.6374158 -0.5292037  0.7818833  0.8561660
## 6     23     7   6 -0.8160263  0.6374158  0.2446517  1.0158452  0.7784478
##   Seasons      Height Transportation_expense Social_drinker Service_time
## 1       1 -0.01930235              1.0078374              1    0.1025410
## 2       1  0.97319750             -1.5458897              1    1.2406839
## 3       1 -0.35013563             -0.6349110              1    1.2406839
## 4       1 -0.68096891              0.8584966              1    0.3301696
## 5       1 -0.01930235              1.0078374              1    0.1025410
## 6       1 -0.35013563             -0.6349110              1    1.2406839
##      Children   Distance        Pet Education Disciplinary_failure
## 1  0.89294976  0.4295322  0.2057297         1                    0
## 2 -0.01603363 -1.1199466 -0.5678559         1                    1
## 3 -0.92501702  1.4400619 -0.5678559         1                    0
## 4  0.89294976 -1.6588958 -0.5678559         1                    0
## 5  0.89294976  0.4295322  0.2057297         1                    0
## 6 -0.92501702  1.4400619 -0.5678559         1                    0
##   Social_smoker V20
## 1             0   0
## 2             0   0
## 3             0   0
## 4             1   0
## 5             0   0
## 6             0   0

set.seed(1876)


for (r in 1:R){

  # subsetting data to training and testing data
```

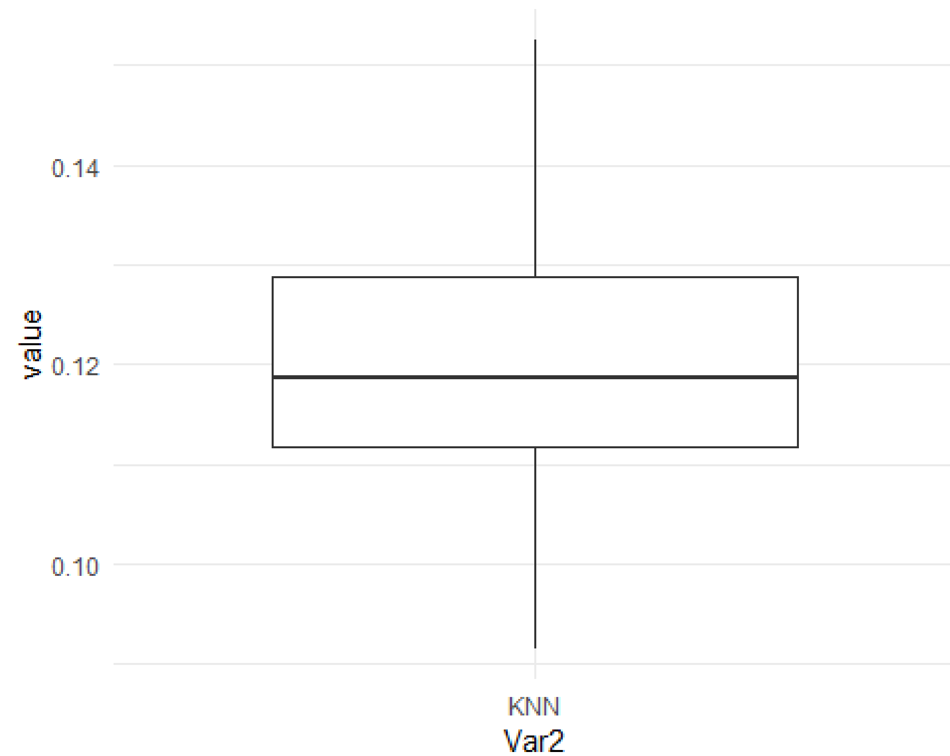
```r
  p <- .6 # proportion of data for training
  w <- sample(1:nrow(dat_smaller), nrow(dat_smaller)*p, replace=F)
  data_train <-dat_smaller[w,]
  data_test <- dat_smaller[-w,]

  ################################################################ knn

  #Running the classifier

  knn <- knn(data_train[,1:13],
             test = data_test[,1:13],
             cl=data_train[,20], k=1)

  #predict doesn't work with KNN for factors
  knntable <- table(knn, data_test[,20])

  #generate confusion matrix
  cm_KNN <-  confusionMatrix(data = knntable, reference = data_test[,1:2],
positive = "1")

  KNNcm [[r,1]] <-  cm_KNN$table[1,1]
  KNNcm [[r,2]] <-  cm_KNN$table[1,2]
  KNNcm [[r,3]] <-  cm_KNN$table[2,1]
  KNNcm [[r,4]] <-  cm_KNN$table[2,2]

  err_matrix_opt [[r,1]] <-  (cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(
data_test)

  sensitivity_matrix_opt[[r, 1]] <- cm_KNN$byClass[1]

  fmeasure_matrix_opt [[r, 1]] <- cm_KNN$byClass[7]

  gmean_matrix_opt [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  #cat("Finished Rep",r, "\n")
}
colnames(sensitivity_matrix_opt)<- "KNN"
graph_sens <- melt(sensitivity_matrix_opt)

graph <- ggplot(graph_sens,aes(x=Var2, y=value) )+ geom_boxplot() +
labs(x="2nd Optimization", y="Sensitivity", title ="2nd Optimization")+
theme_minimal()
graph
```




```
median(sensitivity_matrix_opt)

## [1] 0.2264957

#compare to initial model
comp_matrix_sens <- cbind(sensitivity_matrix_opt[,1], sensitivity_matrix[,1])

colnames(comp_matrix_sens)<- c("Optimized","Original")

graph_comparison <- melt(comp_matrix_sens)

ggplot(graph_comparison,aes(x=Var2, y=value) )+ geom_boxplot() +labs(x=
"Model", y= "Sensitivity") +
  theme_minimal()
```




```
colnames(err_matrix_opt)<- "KNN"
graph_err <- melt(err_matrix_opt)

graph <- ggplot(graph_err,aes(x=Var2, y=value) )+ geom_boxplot()+
theme_minimal()
graph
```



Predicted Model does not do that much better

Third Attempt: SMOTE

```
set.seed(1876)

dat <- read_excel("Absenteeism_at_work.xls")
col <- c("ID", "Reason for absence", "Month of absence", "Day of the week",
"Seasons", "Disciplinary failure", "Education", "Social drinker",   "Social
smoker")
dat[col] <- lapply(dat[col], as.factor)
colnames(dat) <- c("ID", "Reason", "Month", "Day", "Seasons",
"Transportation_expense", "Distance", "Service_time", "Age", "Work_load",
"Hit_target", "Disciplinary_failure", "Education", "Children",
"Social_drinker", "Social_smoker", "Pet", "Weight", "Height", "BMI",
"Absent_time")


nums <- unlist(lapply(dat, is.numeric))
dat.num <- dat[ , nums]

#change variable represent missed time one day or greater
dat <- dat %>% mutate(Absent_time= ifelse(dat$Absent_time <=8,0,1))
str(dat)

## Classes 'tbl_df', 'tbl' and 'data.frame':    740 obs. of  21 variables:
##  $ ID                          : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
```

```
7 11 3 10 20 14 1 ...
##  $ Reason               : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
##  $ Month                : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
##  $ Day                  : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
##  $ Seasons              : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
##  $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
...
##  $ Distance             : num  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service_time         : num  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                  : num  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work_load            : num  239554 239554 239554 239554 239554 ...
##  $ Hit_target           : num  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Education            : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
##  $ Children             : num  2 1 0 2 2 0 1 4 2 1 ...
##  $ Social_drinker       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Social_smoker        : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
##  $ Pet                  : num  1 0 0 0 1 0 4 0 0 1 ...
##  $ Weight               : num  90 98 89 68 90 89 80 65 95 88 ...
##  $ Height               : num  172 178 170 168 172 170 172 168 196 172
...
##  $ BMI                  : num  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absent_time          : num  0 0 0 0 0 0 0 0 1 0 ...
```

```
dat$Absent_time <- as.factor(dat$Absent_time)
#Transforming to Data Frame
dat <- as.data.frame(dat)

str(dat)
```

```
## 'data.frame':    740 obs. of  21 variables:
##  $ ID                   : Factor w/ 36 levels "1","2","3","4",..: 11 36 3
7 11 3 10 20 14 1 ...
##  $ Reason               : Factor w/ 28 levels "0","1","2","3",..: 26 1 23
8 23 23 22 23 20 22 ...
##  $ Month                : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 8 8 8 ...
##  $ Day                  : Factor w/ 5 levels "2","3","4","5",..: 2 2 3 4
4 5 5 5 1 1 ...
##  $ Seasons              : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
##  $ Transportation_expense: num  289 118 179 279 289 179 361 260 155 235
```

```
...
##  $ Distance             : num  36 13 51 5 36 51 52 50 12 11 ...
##  $ Service_time         : num  13 18 18 14 13 18 3 11 14 14 ...
##  $ Age                  : num  33 50 38 39 33 38 28 36 34 37 ...
##  $ Work_load            : num  239554 239554 239554 239554 239554 ...
##  $ Hit_target           : num  97 97 97 97 97 97 97 97 97 97 ...
##  $ Disciplinary_failure : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Education            : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 3 ...
##  $ Children             : num  2 1 0 2 2 0 1 4 2 1 ...
##  $ Social_drinker       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 1
...
##  $ Social_smoker        : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 1 1 1
...
##  $ Pet                  : num  1 0 0 0 1 0 4 0 0 1 ...
##  $ Weight               : num  90 98 89 68 90 89 80 65 95 88 ...
##  $ Height               : num  172 178 170 168 172 170 172 168 196 172
...
##  $ BMI                  : num  30 31 31 24 30 31 27 23 25 29 ...
##  $ Absent_time          : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1
...

###Optimizing the KNN

#For the tunning of the KNN model, we are going to create another
traning/test data sets.

#scaling the data:
dat_v <- dat #we are going to use dat_v for the manipulation
scale <- sapply(dat_v, is.numeric)
dat_v[scale] <- lapply(dat_v[scale],scale)
head(dat_v)

##   ID Reason Month Day Seasons Transportation_expense   Distance
## 1 11     26     7   3       1              1.0107248  0.4292653
## 2 36      0     7   3       1             -1.5433353 -1.1209354
## 3  3     23     7   4       1             -0.6322379  1.4402658
## 4  7      7     7   5       1              0.8613645 -1.6601356
## 5 11     23     7   5       1              1.0107248  0.4292653
## 6  3     23     7   6       1             -0.6322379  1.4402658
##   Service_time        Age  Work_load Hit_target Disciplinary_failure
## 1    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 2    1.2419848  2.0914456 -0.8176594  0.6382541                    1
## 3    1.2419848  0.2392429 -0.8176594  0.6382541                    0
## 4    0.3297577  0.3935931 -0.8176594  0.6382541                    0
## 5    0.1017010 -0.5325083 -0.8176594  0.6382541                    0
## 6    1.2419848  0.2392429 -0.8176594  0.6382541                    0
##   Education    Children Social_drinker Social_smoker       Pet     Weight
## 1         1  0.89311870              1             0  0.1927195  0.8510972
```

```
## 2         1 -0.01722267              1             0 -0.5658572  1.4720605
## 3         1 -0.92756405              1             0 -0.5658572  0.7734768
## 4         1  0.89311870              1             1 -0.5658572 -0.8565516
## 5         1  0.89311870              1             0  0.1927195  0.8510972
## 6         1 -0.92756405              1             0 -0.5658572  0.7734768
##        Height        BMI Absent_time
## 1 -0.01903313  0.7754078           0
## 2  0.97516826  1.0087554           0
## 3 -0.35043360  1.0087554           0
## 4 -0.68183407 -0.6246778           0
## 5 -0.01903313  0.7754078           0
## 6 -0.35043360  1.0087554           0
```

```
#predicting class:
AB_class <- dat_v[, 21]
names(AB_class) <- c(1:nrow(dat_v))
dat_v$ID <- c(1:nrow(dat_v))

dat_v <- dat_v[1:737,]
nrow(dat_v)
```

```
## [1] 737
```

```
rand_permute <- sample(x = nrow(dat_v), size = nrow(dat_v))

all_id_random <- dat_v[rand_permute, "ID"]
dat_v <- dat_v[,-1] #remove ID

#######

splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                  list = FALSE,
                                  times = 1)

trainSplit <- dat_v[ splitIndex,]
testSplit <- dat_v[-splitIndex,]

trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)

prop.table(table(trainSplit$Absent_time))
```

```
##
##   0   1
## 0.5 0.5
```

```
#######

#labels to make inserted code work
```

```
validate_id <- c(1:nrow(testSplit))
training_id <- c(1:nrow(trainSplit))

#rename to work with rest of code
dat_v_train <- trainSplit
dat_v_val <- testSplit
AB_class_train <- trainSplit$Absent_time
AB_class_val <- testSplit$Absent_time
#Confirms data comes out as expected
table(AB_class_train)

## AB_class_train
##  0  1
## 64 64

#Study significance of the variables

rf <- randomForest(Absent_time ~.,
                   data=dat_v_train,
                   mtry=6,
                   ntree=50,
                   na.action=na.roughfix)

impfact <- importance(rf)

impfact <- as.list(impfact)
names(impfact) <- colnames(dat_v[,-20])
impfact2 <- unlist(impfact)


most_sig_stats <- names(sort(desc(impfact2)))

#Re ordering variables by significance:

dat_v_train_ord <- dat_v_train[ c(most_sig_stats)]
str(dat_v_train_ord)

## 'data.frame':    128 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 28 23 1
8 23 23 14 8 19 7 ...
##  $ Work_load             : num [1:128, 1] -0.0761 -0.1657 -0.1657 -0.8663
-1.6789 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 12 11
11 6 9 11 7 8 13 8 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 3 1 5 1
1 5 2 1 1 1 ...
##  $ Hit_target            : num [1:128, 1] -0.42 -1.743 -1.743 1.167 -0.685
```

```
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Distance              : num [1:128, 1] -1.323 -0.649 -0.649 1.508 1.508
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Height                : num [1:128, 1] -0.019 -0.848 -0.848 -0.019 -
0.019 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Weight                : num [1:128, 1] 0.3078 2.093 2.093 0.0749 0.0749
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Age                   : num [1:128, 1] 0.0849 1.011 1.011 -1.3043 -
1.3043 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Transportation_expense: num [1:128, 1] -1.543 0.204 0.204 2.086 2.086
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Service_time          : num [1:128, 1] -0.582 0.102 0.102 -2.179 -2.179
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Pet                   : num [1:128, 1] -0.566 -0.566 -0.566 2.468 2.468
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ BMI                   : num [1:128, 1] 0.3087 2.6422 2.6422 0.0754
0.0754 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 4 4 4 3 1 4
1 1 4 1 ...
##  $ Children              : num [1:128, 1] -0.9276 -0.0172 -0.0172 -0.0172
-0.0172 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
```

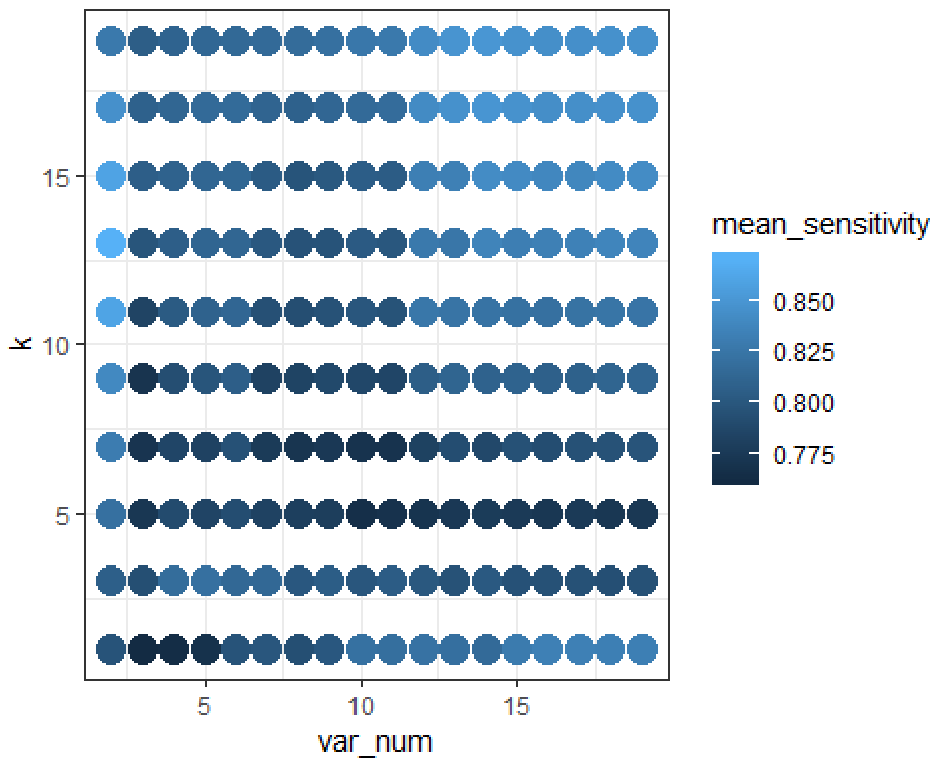
```
##   .. ..$ : NULL
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
3 2 2 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 1 1 1
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 1 2 1
...
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
```

```
dat_v_val_ord <- dat_v_val[, names(dat_v_train_ord)]
str(dat_v_val_ord)
```

```
## 'data.frame':    368 obs. of  19 variables:
##  $ Reason                : Factor w/ 28 levels "0","1","2","3",..: 26 1 22
23 20 22 2 11 19 28 ...
##  $ Work_load             : num [1:368, 1] -0.818 -0.818 -0.818 -0.818 -
0.818 ...
##  $ Month                 : Factor w/ 13 levels "0","1","2","3",..: 8 8 8 8
8 8 8 9 9 9 ...
##  $ Day                   : Factor w/ 5 levels "2","3","4","5",..: 2 2 5 5
1 1 2 3 1 3 ...
##  $ Hit_target            : num [1:368, 1] 0.638 0.638 0.638 0.638 0.638
...
##  $ Distance              : num [1:368, 1] 0.429 -1.121 1.508 1.373 -1.188
...
##  $ Height                : num [1:368, 1] -0.019 0.975 -0.019 -0.682 3.958
...
##  $ Weight                : num [1:368, 1] 0.8511 1.4721 0.0749 -1.0894
1.2392 ...
##  $ Age                   : num [1:368, 1] -0.5325 2.0914 -1.3043 -0.0695 -
0.3782 ...
##  $ Transportation_expense: num [1:368, 1] 1.011 -1.543 2.086 0.578 -0.991
...
##  $ Service_time          : num [1:368, 1] 0.102 1.242 -2.179 -0.354 0.33
...
##  $ Pet                   : num [1:368, 1] 0.193 -0.566 2.468 -0.566 -0.566
...
##  $ BMI                   : num [1:368, 1] 0.7754 1.0088 0.0754 -0.858 -
0.3913 ...
##  $ Seasons               : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1
1 1 1 1 ...
##  $ Children              : num [1:368, 1] 0.8931 -0.0172 -0.0172 2.7138
0.8931 ...
##  $ Education             : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 3
1 2 1 1 ...
##  $ Disciplinary_failure  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1
...
##  $ Social_drinker        : Factor w/ 2 levels "0","1": 2 2 2 2 2 1 2 1 2 2
...
```

```
##  $ Social_smoker         : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2
...

#######################
#######################

#Monte Carlo Validation:

size <- nrow(dat_v_train)
sub <- (2/3) * nrow(dat_v_train)

training_family_L <- lapply(1:500, function(j) {
  perm <- sample(1:size, size = size, replace = F)
  shuffle <- training_id[perm]
  trn <- shuffle[1:sub]
  trn
})

validation_family_L <- lapply(training_family_L,
                              function(x) setdiff(training_id, x))

#Finding an optimal set of variables and optimal k

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 9.219544

K <- seq(from = 1, to = 19, by = 2)
times <- 500 * length(N) * length(K)

#Execution of the test with loops

paramter_errors_df <- data.frame(mc_index = as.integer(rep(NA, times =
times)),
                                 var_num = as.integer(rep(NA, times =
times)),
                                 k = as.integer(rep(NA, times = times)),
                                 error = as.numeric(rep(NA, times = times)))


param_df1 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df <- merge(param_df1, data.frame(k = K))

N <- seq(from = 2, to = 19, by = 1)
sqrt(length(training_family_L[[1]]))

## [1] 9.219544
```

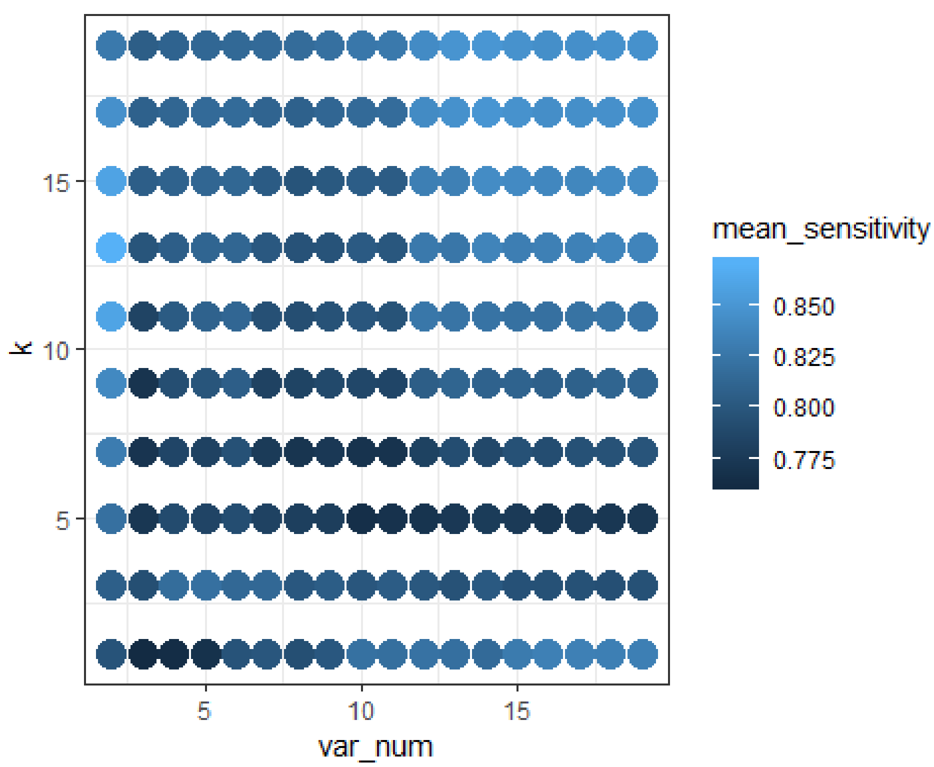
```r
K <- seq(from = 1, to = 19, by = 2)
times <- 500 * length(N) * length(K)

core_knn_sen <- function(j, n, k) {
  knn_predict <- knn(train = dat_v_train_ord[training_family_L[[j]], 1:n],
                     test = dat_v_train_ord[validation_family_L[[j]], 1:n],
                     cl = AB_class_train[training_family_L[[j]]],
                     k = k)

  tbl <- table(knn_predict, AB_class_train[validation_family_L[[j]]])

  sen <- (tbl[2, 2] )/(tbl[1, 2] + tbl[2, 2])
  sen
}


param_df1_2 <- merge(data.frame(mc_index = 1:500), data.frame(var_num = N))
param_df_2 <- merge(param_df1_2, data.frame(k = K))

knn_err_est_df_2 <- ddply(param_df_2[1:times, ], .(mc_index, var_num, k),
function(df) {
  sen <- core_knn_sen(df$mc_index[1], df$var_num[1], df$k[1])
  sen
})

head(knn_err_est_df_2)

##   mc_index var_num  k        V1
## 1        1       2  1 0.8571429
## 2        1       2  3 0.8571429
## 3        1       2  5 0.8571429
## 4        1       2  7 0.8571429
## 5        1       2  9 0.9047619
## 6        1       2 11 0.9047619

names(knn_err_est_df_2)[4] <- "Sensitivity"

mean_sens_df2 <- ddply(knn_err_est_df_2, .(var_num, k), function(df)
mean(df$Sensitivity))

names(mean_sens_df2)[3] <- "mean_sensitivity"

ggplot(data = mean_sens_df2, aes(x = var_num, y = k, color =
mean_sensitivity)) + geom_point(size = 5) +
  theme_bw()
```




```
mean_sens_df2[which.max(mean_sens_df2$mean_sensitivity), ]

##   var_num  k mean_sensitivity
## 7       2 13        0.8706989

order <- mean_sens_df2 %>% arrange(desc(mean_sensitivity))

save(mean_sens_df2, file='mean_sens_df2.RData')
save(order, file='order.RData')
```

load output

```
load( file='mean_sens_df2.RData')
load( file='order.RData')

ggplot(data = mean_sens_df2, aes(x = var_num, y = k, color =
mean_sensitivity)) + geom_point(size = 5) +
  theme_bw()
```



Repeat test with top 5 choices and resampled training/test data

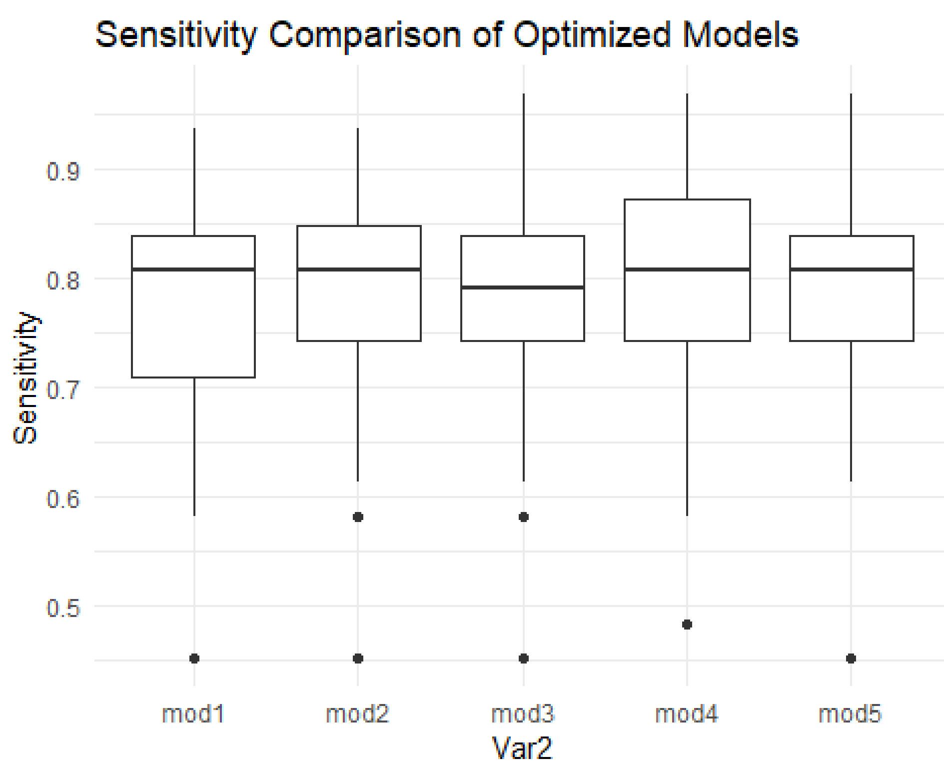
```
R <- 100 # replications


# create the matrix to store values 1 row per model
err_matrix_opt <- matrix(0, ncol=5, nrow=R)

sensitivity_matrix_opt <- matrix(0, ncol=5, nrow=R)

fmeasure_matrix_opt <- matrix(0, ncol=5, nrow=R)

gmean_matrix_opt <- matrix(0, ncol=5, nrow=R)

# these are optional but I like to see how the model did each run so I can
check other output
KNNcm <- matrix(0, ncol=4, nrow=R)
KNNcm2 <- matrix(0, ncol=4, nrow=R)
KNNcm3 <- matrix(0, ncol=4, nrow=R)
KNNcm4 <- matrix(0, ncol=4, nrow=R)
KNNcm5 <- matrix(0, ncol=4, nrow=R)

set.seed(1876)

for (r in 1:R){
```

```
  # subsetting data to training and testing data
  splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                    list = FALSE,
                                    times = 1)

  trainSplit <- dat_v[ splitIndex,]
  testSplit <- dat_v[-splitIndex,]

  trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
  trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)

  ################################################################ knn

  #Running the classifier

  #option 1

  knn <- knn(trainSplit[,1:order[1,1]],
             test = testSplit[,1:order[1,1]],
             cl=trainSplit[,20], k=order[1,2])

  #predict doesn't work with KNN for factors
  knntable <- table(knn, testSplit[,20])

  cm_KNN <-  confusionMatrix(data = knntable, reference = testSplit[,20],
positive = "1")

  KNNcm [[r,1]] <-  cm_KNN$table[1,1]
  KNNcm [[r,2]] <-  cm_KNN$table[1,2]
  KNNcm [[r,3]] <-  cm_KNN$table[2,1]
  KNNcm [[r,4]] <-  cm_KNN$table[2,2]

  err_matrix_opt [[r,1]] <-
(cm_KNN$table[1,2]+cm_KNN$table[2,1])/nrow(testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 1]] <- cm_KNN$byClass[1]

  fmeasure_matrix_opt [[r, 1]] <- cm_KNN$byClass[7]

  gmean_matrix_opt [[r, 1]] <- sqrt(cm_KNN$byClass[1]* cm_KNN$byClass[2])

  ######################
  #option 2
```

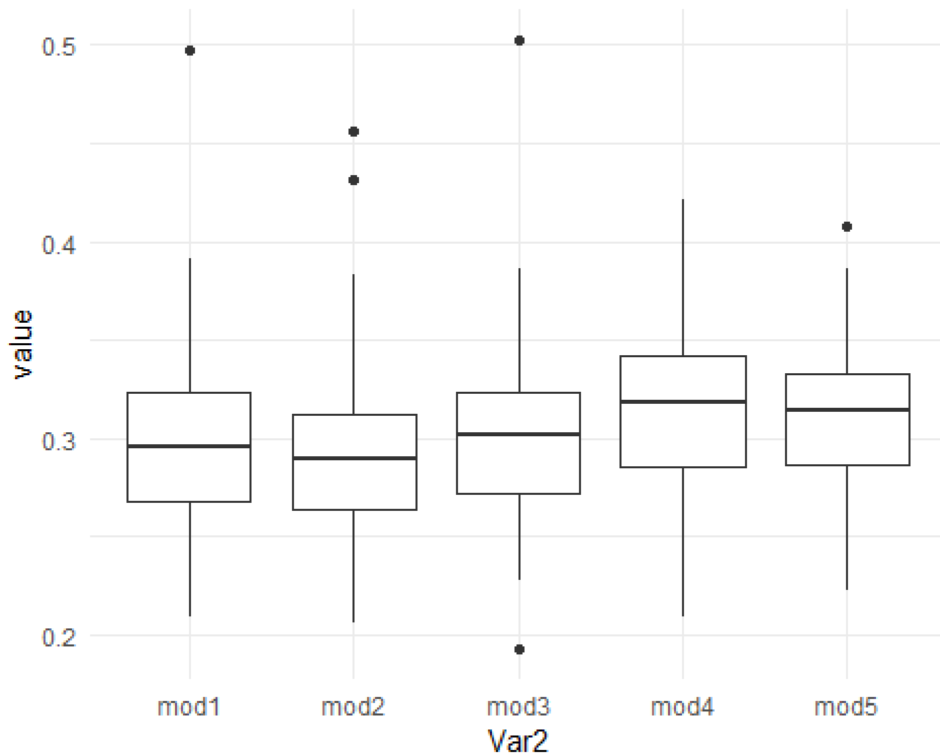
```
  knn <- knn(trainSplit[,1:order[2,1]],
             test = testSplit[,1:order[2,1]],
             cl=trainSplit[,20], k=order[2,2])

  #predict doesn't work with KNN for factors
  knntable2 <- table(knn, testSplit[,20])

  cm_KNN2 <-  confusionMatrix(data = knntable2, reference = testSplit[,20],
positive = "1")

  KNNcm2 [[r,1]] <-  cm_KNN2$table[1,1]
  KNNcm2 [[r,2]] <-  cm_KNN2$table[1,2]
  KNNcm2 [[r,3]] <-  cm_KNN2$table[2,1]
  KNNcm2 [[r,4]] <-  cm_KNN2$table[2,2]

  err_matrix_opt [[r,2]] <-
(cm_KNN2$table[1,2]+cm_KNN2$table[2,1])/nrow(testSplit)

  sensitivity_matrix_opt[[r, 2]] <- cm_KNN2$byClass[1]

  fmeasure_matrix_opt [[r, 2]] <- cm_KNN2$byClass[7]

  gmean_matrix_opt [[r, 2]] <- sqrt(cm_KNN2$byClass[1]* cm_KNN2$byClass[2])

  ##########
  #option 3

  knn <- knn(trainSplit[,1:order[3,1]],
             test = testSplit[,1:order[3,1]],
             cl=trainSplit[,20], k=order[3,2])

  #predict doesn't work with KNN for factors
  knntable <- table(knn, testSplit[,20])

  cm_KNN3 <-  confusionMatrix(data = knntable, reference = testSplit[,20],
positive = "1")

  KNNcm3 [[r,1]] <-  cm_KNN3$table[1,1]
  KNNcm3 [[r,2]] <-  cm_KNN3$table[1,2]
  KNNcm3 [[r,3]] <-  cm_KNN3$table[2,1]
  KNNcm3 [[r,4]] <-  cm_KNN3$table[2,2]

  err_matrix_opt [[r,3]] <-
(cm_KNN3$table[1,2]+cm_KNN3$table[2,1])/nrow(testSplit)

  sensitivity_matrix_opt[[r, 3]] <- cm_KNN3$byClass[1]

  fmeasure_matrix_opt [[r, 3]] <- cm_KNN3$byClass[7]
```

```
  gmean_matrix_opt [[r, 3]] <- sqrt(cm_KNN3$byClass[1]* cm_KNN3$byClass[2])

  ################
  #option 4

  knn <- knn(trainSplit[,1:order[4,1]],
             test = testSplit[,1:order[4,1]],
             cl=trainSplit[,20], k=order[4,2])

  #predict doesn't work with KNN for factors
  knntable4 <- table(knn, testSplit[,20])

  cm_KNN4 <-  confusionMatrix(data = knntable4, reference = testSplit[,20],
positive = "1")

  KNNcm4 [[r,1]] <-  cm_KNN4$table[1,1]
  KNNcm4 [[r,2]] <-  cm_KNN4$table[1,2]
  KNNcm4 [[r,3]] <-  cm_KNN4$table[2,1]
  KNNcm4 [[r,4]] <-  cm_KNN4$table[2,2]

  err_matrix_opt [[r,4]] <-
(cm_KNN4$table[1,2]+cm_KNN4$table[2,1])/nrow(testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 4]] <- cm_KNN4$byClass[1]

  fmeasure_matrix_opt [[r, 4]] <- cm_KNN4$byClass[7]

  gmean_matrix_opt [[r, 4]] <- sqrt(cm_KNN4$byClass[1]* cm_KNN4$byClass[2])

  #####################
  #option 5

  knn <- knn(trainSplit[,1:order[5,1]],
             test = testSplit[,1:order[5,1]],
             cl=trainSplit[,20], k=order[5,2])

  knntable5 <- table(knn, testSplit[,20])

  cm_KNN5 <-  confusionMatrix(data = knntable5, reference = testSplit[,20],
positive = "1")

  KNNcm5 [[r,1]] <-  cm_KNN5$table[1,1]
  KNNcm5 [[r,2]] <-  cm_KNN5$table[1,2]
  KNNcm5 [[r,3]] <-  cm_KNN5$table[2,1]
  KNNcm5 [[r,4]] <-  cm_KNN5$table[2,2]
```

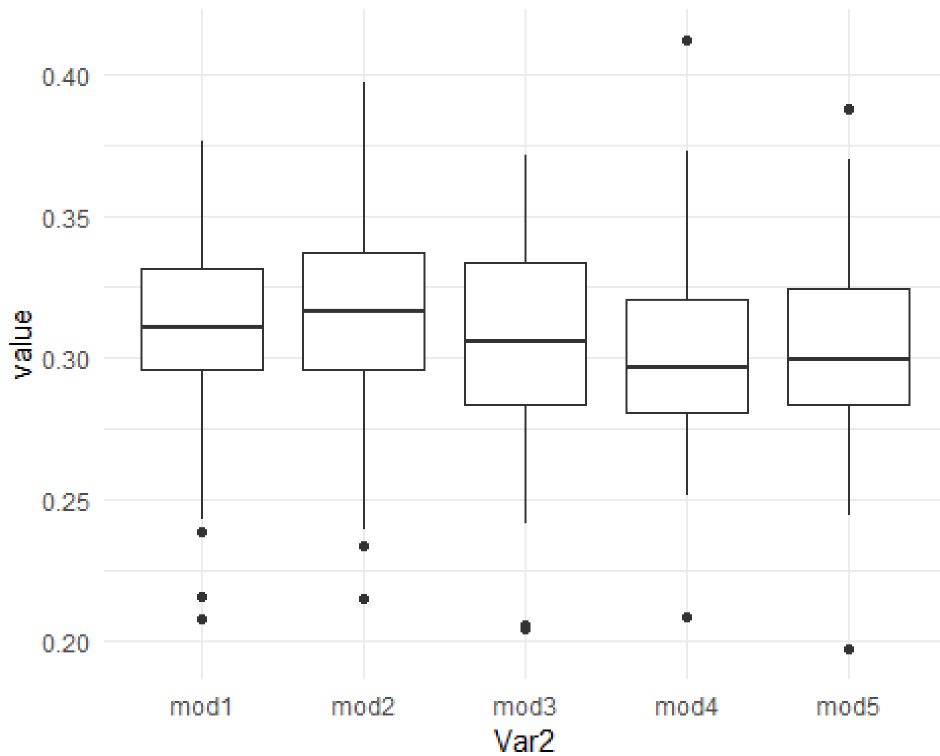
```
  err_matrix_opt [[r,5]] <-  (cm_KNN5$table[1,2]+cm_KNN5$table[2,1])/nrow(
testSplit)

  # store the errors

  sensitivity_matrix_opt[[r, 5]] <- cm_KNN5$byClass[1]

  fmeasure_matrix_opt [[r, 5]] <- cm_KNN5$byClass[7]

  gmean_matrix_opt [[r, 5]] <- sqrt(cm_KNN5$byClass[1]* cm_KNN5$byClass[2])

  #cat("Finished Rep",r, "\n")
}
colnames(sensitivity_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_sens <- melt(sensitivity_matrix_opt)

ggplot(graph_sens,aes(x=Var2, y=value) )+ geom_boxplot()+
labs(y="Sensitivity", title="Sensitivity Comparison of Optimized Models") +
theme_minimal()
```



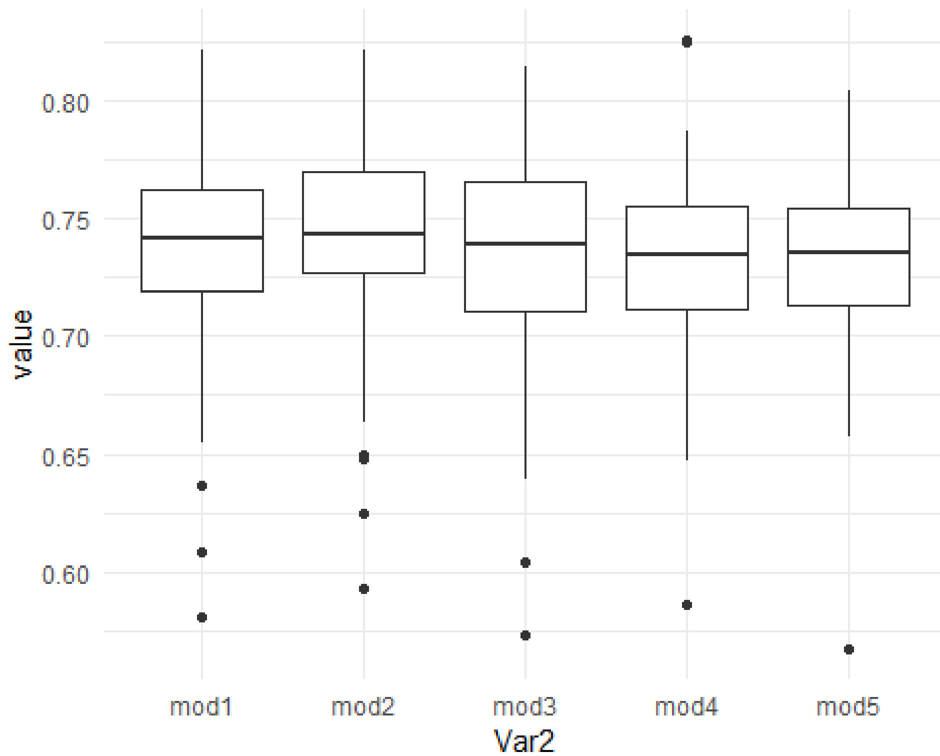

```
colnames(err_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_err <- melt(err_matrix_opt)

ggplot(graph_err,aes(x=Var2, y=value) )+ geom_boxplot() + theme_minimal()
```




```
colnames(fmeasure_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_fmeasure <- melt(fmeasure_matrix_opt)

ggplot(graph_fmeasure,aes(x=Var2, y=value) )+ geom_boxplot() +
theme_minimal()
```

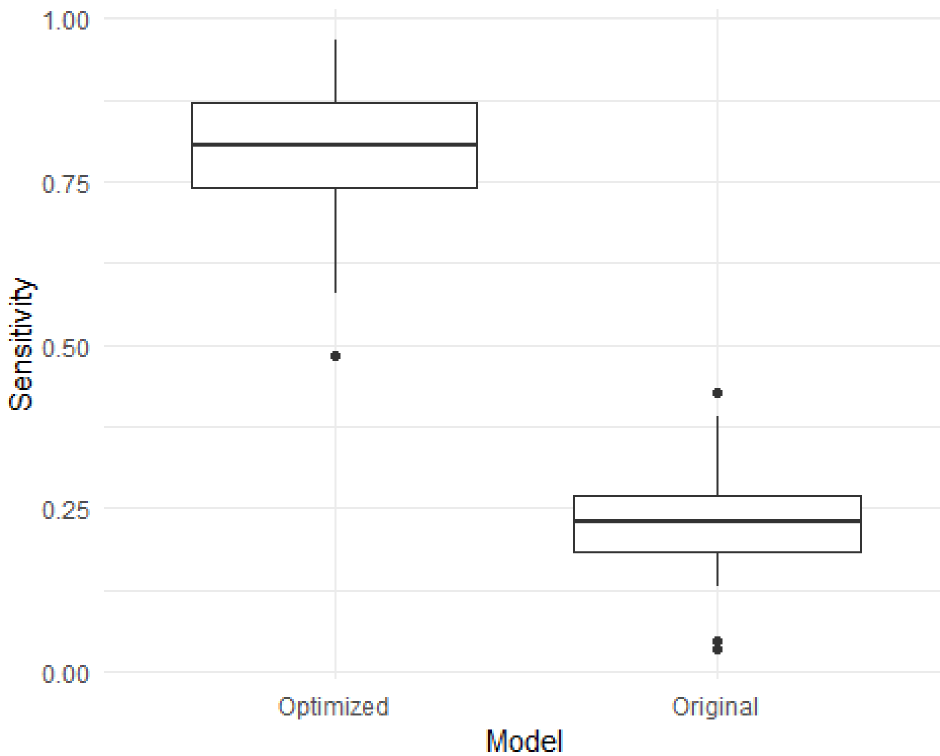



```
colnames(gmean_matrix_opt)<- c("mod1","mod2","mod3","mod4","mod5")
graph_gmean <- melt(gmean_matrix_opt)

ggplot(graph_gmean,aes(x=Var2, y=value) )+ geom_boxplot()+ theme_minimal()
```



Comparison to original model

```
#bind old and new model
comp_matrix_sens2 <- cbind(sensitivity_matrix_opt[,4],
sensitivity_matrix[,1])

colnames(comp_matrix_sens2)<- c("Optimized","Original")

graph_comparison <- melt(comp_matrix_sens2)

ggplot(graph_comparison,aes(x=Var2, y=value) )+ geom_boxplot() +labs(x=
"Model", y= "Sensitivity") +
  theme_minimal()
```



Confusion Matrix Opminized Model

```
set.seed(1976)
  splitIndex <- createDataPartition(dat_v$Absent_time, p = .50,
                                    list = FALSE,
                                    times = 1)

  trainSplit <- dat_v[ splitIndex,]
  testSplit <- dat_v[-splitIndex,]

  trainSplit$Absent_time <- as.factor(trainSplit$Absent_time)
  trainSplit <- SMOTE(Absent_time ~ ., trainSplit, perc.over = 100,
perc.under=200)

knn <- knn(trainSplit[,1:order[4,1]],
             test = testSplit[,1:order[4,1]],
             cl=trainSplit[,20], k=order[4,2])
  knntable4 <- table(knn, testSplit[,20])

  cm_KNN4 <-  confusionMatrix(data = knntable4, reference = testSplit[,20],
positive = "1")

cm_KNN4

## Confusion Matrix and Statistics
##
```

```
##
## knn   0   1
##   0 232  10
##   1 105  21
##
##                Accuracy : 0.6875
##                  95% CI : (0.6374, 0.7345)
##     No Information Rate : 0.9158
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.153
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.67742
##             Specificity : 0.68843
##          Pos Pred Value : 0.16667
##          Neg Pred Value : 0.95868
##              Prevalence : 0.08424
##          Detection Rate : 0.05707
##    Detection Prevalence : 0.34239
##       Balanced Accuracy : 0.68292
##
##        'Positive' Class : 1
##
```

Confusion Matrix Initial Model

```
set.seed(1976)
dat1 <- dat[-1]

#scale
scale <- sapply(dat1, is.numeric)
dat1[scale] <- lapply(dat1[scale],scale)
p <- .6 # proportion of data for training
w <- sample(1:nrow(dat1), nrow(dat1)*p, replace=F)
data_train <-dat1[w,]
data_test <- dat1[-w,]


#Running the classifier

  knn <- knn(data_train[-20],
                       test = testSplit[-20],
                       cl=data_train$Absent_time, k=2)

 knntable <- table(knn, testSplit$Absent_time)

#generate confusion matrix
 cm_KNN <-  confusionMatrix(data = knntable, reference = testSplit[,-20],
```

```
positive = "1")

cm_KNN

## Confusion Matrix and Statistics
##
##
## knn   0   1
##   0 320  20
##   1  17  11
##
##                Accuracy : 0.8995
##                  95% CI : (0.8641, 0.9282)
##     No Information Rate : 0.9158
##     P-Value [Acc > NIR] : 0.8868
##
##                   Kappa : 0.3184
##  Mcnemar's Test P-Value : 0.7423
##
##             Sensitivity : 0.35484
##             Specificity : 0.94955
##          Pos Pred Value : 0.39286
##          Neg Pred Value : 0.94118
##              Prevalence : 0.08424
##          Detection Rate : 0.02989
##    Detection Prevalence : 0.07609
##       Balanced Accuracy : 0.65220
##
##        'Positive' Class : 1
##
```