비정형데이터분석 기말과제

순천향대학교 빅데이터공학과

2017143 한태규

아래의 문서 URL : <https://www.notion.so/0494c0b1c6a1438a99c5f2d49d6467e9> github :

<https://github.com/TaegyuHan/Unstructured_data_analysis_finalTest_SCH>

소스의 구조를 파악하기 위해선 github을 참고

**code** 폴더 구조

│ README.md

│

├─draw # draw file

│ 이미지 설명.drawio

│ 프로세스.drawio

│

├─README\_IMG # 이미지 저장공간

│

└─src

│ .Rhistory

│ DataLoad.R # RData load 파일

│ library.R # 라이브러리 설치 및 load 파일

│ upSampling.R # Train Data upSampling 파일

│

├─TesT데이터만들기 # 새로운 데이터를 TEST로 만들고

│

│

│

│

│

│

│

│

│

│

│

# 모델 실행 파일

│ 산술통계\_TEST데이터\_만들기.R

│ 산술통계\_피크\_합친\_TEST\_만들기.R

│ 피크\_TEST데이터\_만들기.R

│

└─model # model 저장파일

RFModelPeak.rda RFModelPKST.rda

RFModelstatistic.rda

├─교수님\_word\_전처리

│ 교수님코드.R # 교수님 word code 파일

│

├─데이터\_하나로합치기

│ 데이터합치기.R # csv 파일 전처리 파일

│

└─분석 # 분석과정 저장 파일

산술\_통계.R 산술통계\_피크.R 피크.R

목차

code 폴더 구조 과제 목적

결과 표 최종 결론 Data 불러오기 Data 확인

Train, Test 데이터 UP 샘플링 산술 통계

산술 통계 분석 함수 sum ( 합 ) average ( 평균 ) min ( 최소값 ) max ( 최대값 )

geometric mean ( 기하 평균 ) median ( 중위값 )

Standard Deviation ( 표준 편차 ) skewness ( 왜도 )

산술 통계 데이터 분석 결과 모델링

RF statistic model Model Test

RF statistics Confusion Matrix

NB statistic model

피크분석

피크 추출 병렬처리 코드 Peak 추출 Data

저혈압, 정상혈압, Peak 추출수 비교

Test 데이터 전처리 모델링

RF Peak Model Test

RF Peak Confusion Matrix NB Peak Model Test

Statistic & Peak 모델

데이터 전처리

Model Trainning

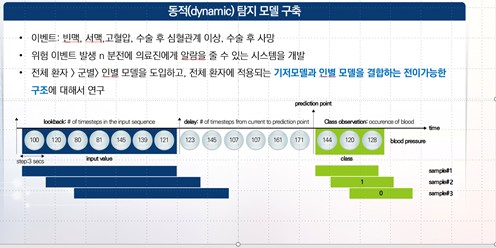
statistics & Peak Confusion Matrix

NB statistics & Peak model

NB statistics & Peak Confusion Matrix

## 과제 목적

아래의 과제의 목적은 환자의 혈압 데이터를 모델링 하여 미래의 저혈압을 예측하는 모델을 만드는 것이다.



위의 그림과 같이 input value의 기간 동안의 데이터를 가지고 class의 구간이 저혈압인지 정상혈압인지 예측 하는 모델을 만들어야한다.

class

1. : 정상혈압
2. : 저혈압

결과 표



## 최종 결론

statistic, Peak, statistic & Peak 3개의 RF 모델을 만들어 돌려보았지만 정상혈압을 예측하는것을 쉬 웠지만 저혈압을 예측하는것은 어려웠습니다.

NB 모델을 사용해서 저혈압 예측 성능이 RF보다 확실하게 향상하였습니다. 하지만 정상혈압예측률을 떨 어졌습니다.

모델을 딥러닝 케라스를 이용한 모델을 돌려보고 싶었지만 컴퓨터 사양이 부족한 관계로 시간이 부족하여 아쉬웠습니다.

**Data** 불러오기

데이터를 불러오는 코드 입니다.

# # # 데이터 합치기

# #

RDATA\_PATH <- ".DATA\_Rfile" DATA\_PATH <- "./DATA"

# 변수 선언

for (fn in list.files()){

# file name fn

# values name

val <- strsplit(strsplit(fn, split= "\_")[[1]][2], split=".csv")[[1]]

# 변수 선언

assign(val, read\_csv(fn))

}

# 변수 이름 반환 함수

makeDataList <- function()

{

setwd(DATA\_PATH)

dataNameList <- c() # 변수 리스트

for (fn in list.files()){

# values name

val <- strsplit(strsplit(fn, split= "\_")[[1]][2], split=".csv")[[1]]

dataNameList <- c(dataNameList, val)

}

return(dataNameList)

}

dataNameList <- makeDataList()

# # # BP DATA 추출

# #

# 혈압 추출 함수

findBloodPressure <- function(row)

{

strsplit(row, split=",")[[1]][2]

}

# signal 데이터 벡터형으로 추출

for (i in 2:length(dataNameList)) { print(dataNameList[i])

# 혈압 데이터 추출

tmp <- unlist(lapply(get(dataNameList[i])$signal[3:length(get(dataNameList[i])$signal)]

, findBloodPressure)) rm(list = dataNameList[i])

assign(paste0(dataNameList[i], "\_SignalData"), tmp)

}

tmp <- unlist(lapply(get(val)$signal[3:length(get(dataNameList[1])$signal)], findBloodPressure))

rm(list="tmp")

for ( val in dataNameList ){ print((val))

}

# # # 저장

setwd(RDATA\_PATH)

# save(slp01a\_SignalData, file = "slp01a\_SignalData.RData") # save(slp01b\_SignalData, file = "slp01b\_SignalData.RData") # save(slp02a\_SignalData, file = "slp02a\_SignalData.RData") # save(slp02b\_SignalData, file = "slp02b\_SignalData.RData") # save(slp03\_SignalData, file = "slp03\_SignalData.RData") # save(slp04\_SignalData, file = "slp04\_SignalData.RData") # save(slp14\_SignalData, file = "slp14\_SignalData.RData") # save(slp16\_SignalData, file = "slp16\_SignalData.RData") # save(slp32\_SignalData, file = "slp32\_SignalData.RData") # save(slp37\_SignalData, file = "slp37\_SignalData.RData") # save(slp41\_SignalData, file = "slp41\_SignalData.RData") # save(slp45\_SignalData, file = "slp45\_SignalData.RData") # save(slp48\_SignalData, file = "slp48\_SignalData.RData") # save(slp59\_SignalData, file = "slp59\_SignalData.RData") # save(slp60\_SignalData, file = "slp60\_SignalData.RData") # save(slp61\_SignalData, file = "slp61\_SignalData.RData") # save(slp66\_SignalData, file = "slp66\_SignalData.RData") # save(slp67x\_SignalData, file = "slp67x\_SignalData.RData")

# #

# # # 데이터 load

# load("slp01a\_SignalData.RData") # load("slp01b\_SignalData.RData") # load("slp02a\_SignalData.RData") # load("slp02b\_SignalData.RData") # load("slp03\_SignalData.RData") # load("slp04\_SignalData.RData") # load("slp14\_SignalData.RData") # load("slp16\_SignalData.RData")

# load("slp32\_SignalData.RData") # load("slp37\_SignalData.RData") # load("slp41\_SignalData.RData") # load("slp45\_SignalData.RData") # load("slp48\_SignalData.RData") # load("slp59\_SignalData.RData") # load("slp60\_SignalData.RData") # load("slp61\_SignalData.RData") # load("slp66\_SignalData.RData") # load("slp67x\_SignalData.RData")

# #

# # # 교수님 코드

ma <- function(x, n = 5)

{

stats::filter(x, rep(1 / n, n), sides = 2)

}

SRATE <- 250

MINUTES\_AHEAD <- 1

Data\_set <- list() # 샘플 생성후 저장할 공간

for (file in dataNameList){ print(paste0(file, "\_SignalData"))

IBP <- as.numeric(get(paste0(file, "\_SignalData"))) # IBP <- as.numeric(slp01a\_SignalData)

i <- 1

IBP\_data <- data.frame()

while (i < length(IBP) - SRATE\*(1+1+MINUTES\_AHEAD)\*60) { segx <- IBP[i:(i+SRATE\*1\*60-1)]

segy <- IBP[(i+SRATE\*(1+MINUTES\_AHEAD)\*60):(i+SRATE\*(1+1+MINUTES\_AHEAD)\*60-1)] segxd <- IBP[i:(i+SRATE\*(1+MINUTES\_AHEAD)\*60-1)]

if(is.na(mean(segx)) | is.na(mean(segy)) | max(segx)>200 | min(segx)<20 | max(segy)>200 | max(segy)<20 | max(segx) - min(segx) < 30 |

max(segy) - min(segy) < 30|(min(segxd,na.rm=T) <= 50)){

} else { #나머지의 경우

segy <- ma(segy, 2\*SRATE)

event <- ifelse(min(segy,na.rm=T) <= 50, 1, 0) # print(event)

IBP\_data<- rbind(IBP\_data, cbind(t(segx), event))

}

i <- i+1\*60\*SRATE

}

Data\_set[[file]] <- IBP\_data

}

# #

# # # Data 하나로 합치기

#Data\_set

for (fn in dataNameList){ print(paste0("Data\_set$", fn))

}

AllData <- rbind( Data\_set$slp01a,

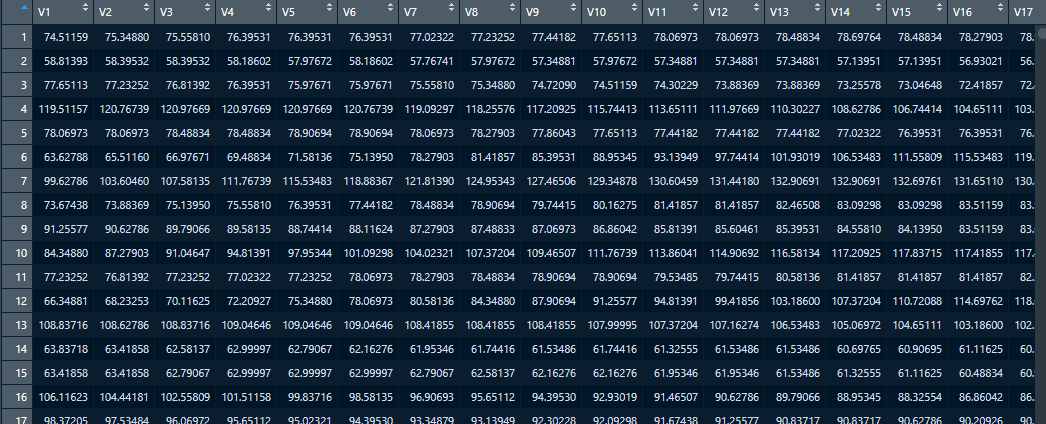
Data\_set$slp01b, Data\_set$slp02a, Data\_set$slp02b, Data\_set$slp03, Data\_set$slp04, Data\_set$slp14, Data\_set$slp16, Data\_set$slp32, Data\_set$slp37, Data\_set$slp41, Data\_set$slp45, Data\_set$slp48, Data\_set$slp59, Data\_set$slp60, Data\_set$slp61, Data\_set$slp66, Data\_set$slp67x )

# save(AllData, file = "AllData.RData") # AllData

# #

**Data** 확인

word로 받은 code를 작동시킨 후 결과



# Train, Test

데이터를 Train과 Test로 나눴습니다. 비율

Train : 2/3 Test : 1/2

#

# Train 데이터 Test데이터 나누기

#

trainIndex <- as.numeric(createDataPartition(AllData$event,p=2/3, list = F))

TrainData <- AllData[trainIndex, ] TestData <- AllData[-trainIndex, ]

TrainData %>% nrow() TrainData %>% ncol()

# > TrainData %>% nrow() # [1] 2662

# > TrainData %>% ncol() # [1] 15001

TestData %>% nrow() TestData %>% ncol()

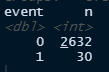
# > TestData %>% nrow() # [1] 1331

# > TestData %>% ncol() # [1] 15001

# #

데이터 **UP** 샘플링

class label의 데이터 분포가 한쪽으로 몰려있어서 up샘플링 해주었습니다.

정상혈압 : 2632

저혈압 : 30

#

# Up sampleling

TrainUPData <- groupdata2::upsample(

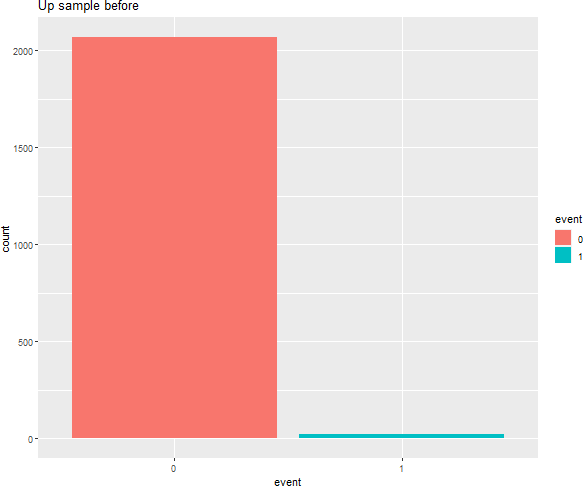
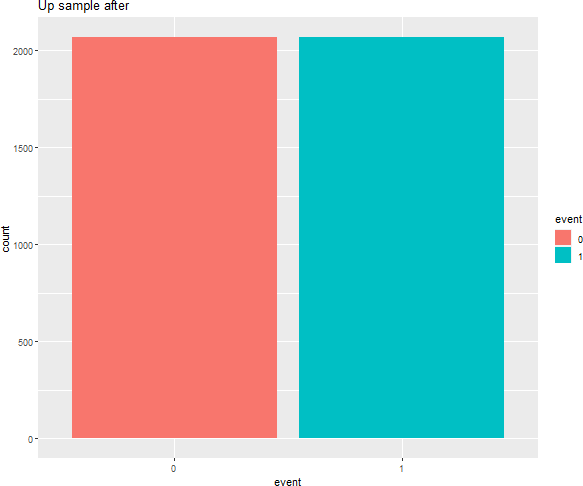
TrainData, # data cat\_col = "event"

)

#

#

#

산술 통계

#### 산술 통계 분석 함수

# # # 시각화 png 저장 함수

# 1920, 1017

saveggplot <- function(plot, fileName, width, height)

{

png(

filename=paste0(fileName, ".png"), width=width,

height=height, unit="px" )

print( plot )

dev.off()

}

# #

# # # 산술 통계 분석 함수

# 산술 통계 전처리 후

# hist 그래프로 이미지를 저장합니다.

ArithmeticStatFuc <- function( inputData = AllData,

AppliedFunction, StatisticalName )

{

# 데이터 전처리

tmpDataFrame <- data.frame(

tmpData = apply(inputData[,1:15000], 1, AppliedFunction), event = as.factor(inputData$event)

)

# 데이터 시각화

tmpDataPlot <- tmpDataFrame %>% ggplot( aes(x=tmpData, fill=event)) +

geom\_histogram(alpha=0.6, position = 'identity') + scale\_fill\_manual(values=c("#69b3a2", "#404080")) + theme\_ipsum() +

labs(fill="") + ggtitle(paste0(StatisticalName, "DataFrame"))

# 시각화 저장

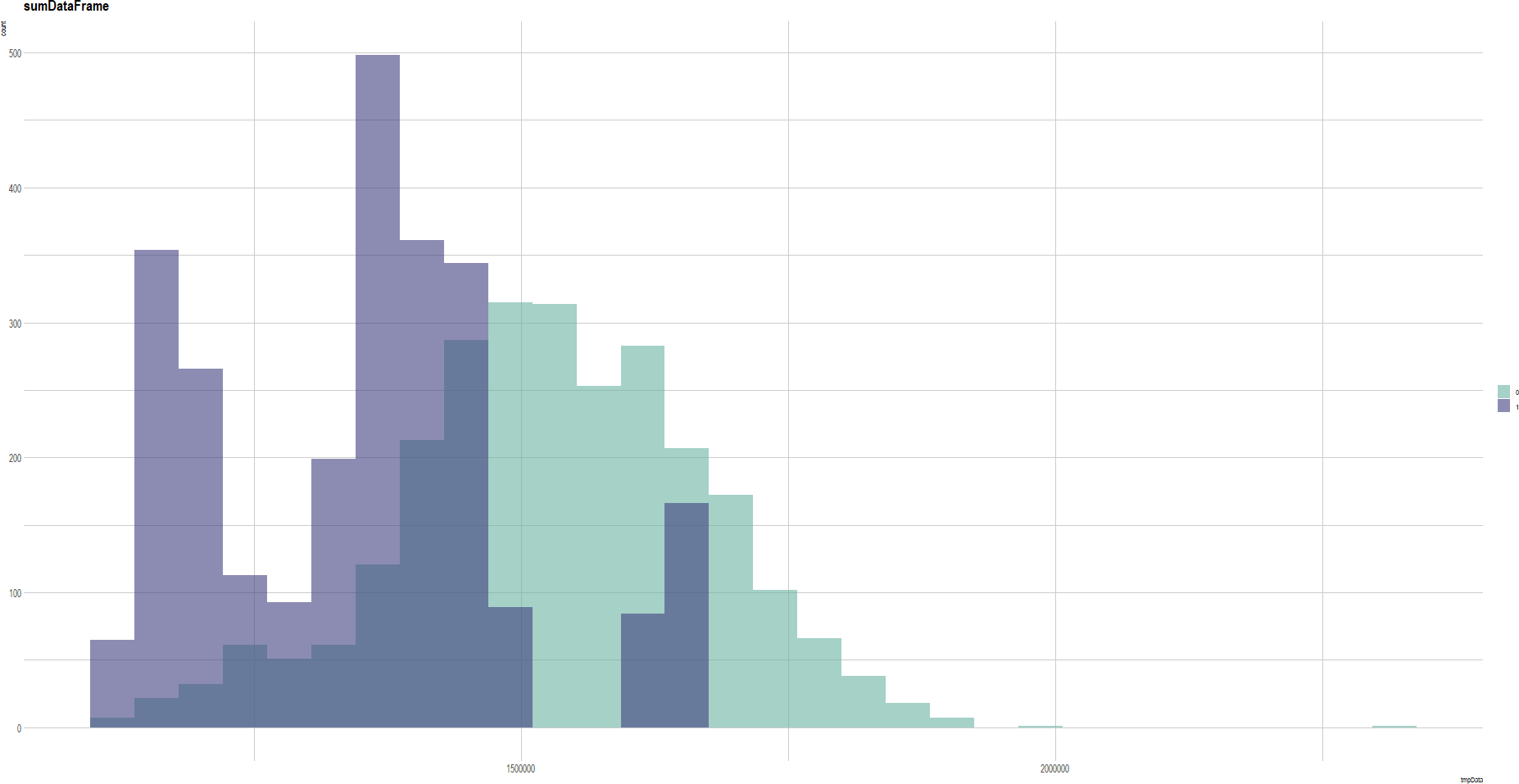
saveggplot(tmpDataPlot, StatisticalName, 1920, 1017)

}

# #

보라색 : 저혈압 초록색 : 정상혈압

**sum (** 합 **)**



# # # 데이터의 총 합 시각화 비교

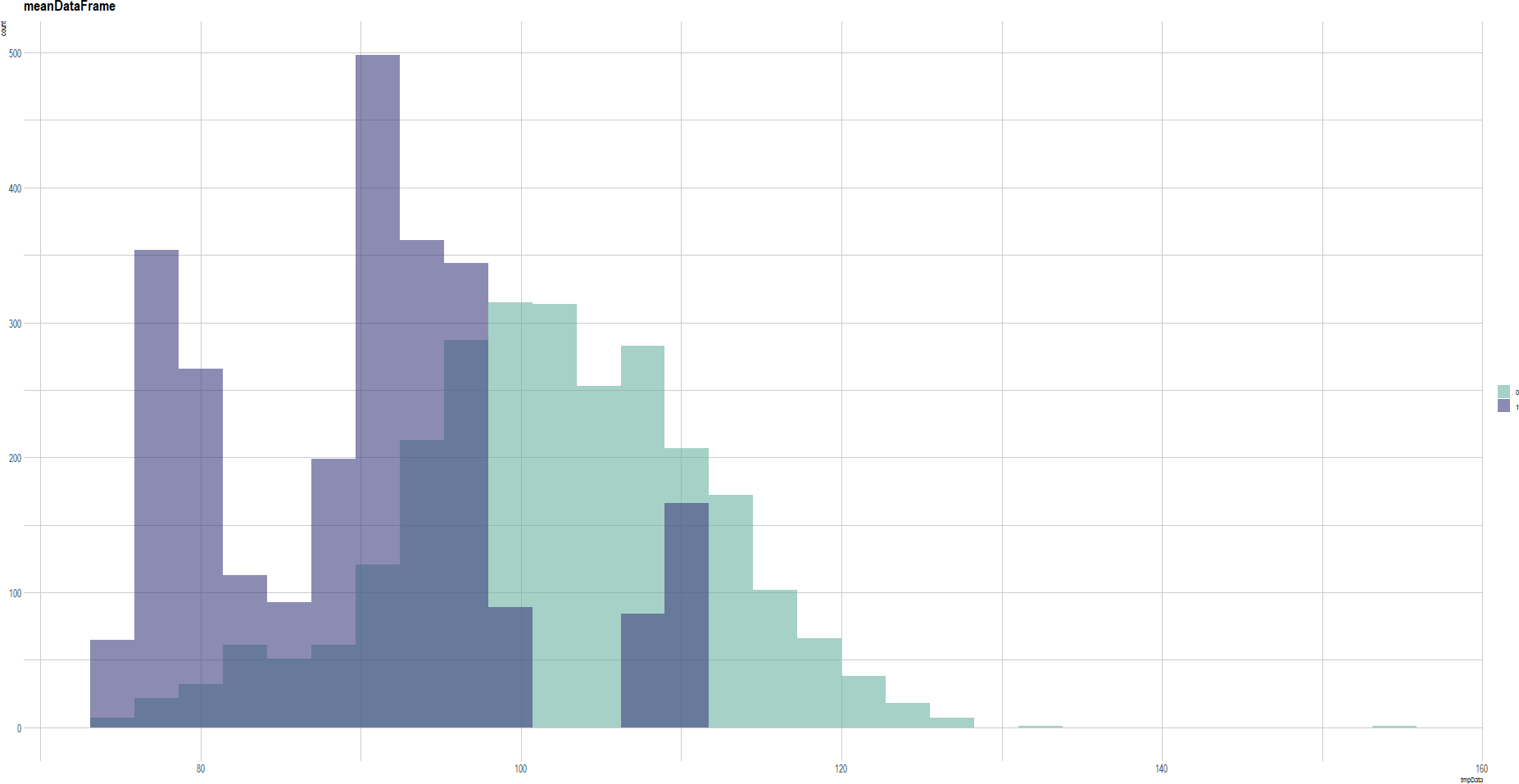
ArithmeticStatFuc( inputData = TrainUPData, sum,

"sum" )

# #

정상혈압의 분포의 50%이하 부분에 분포에 있는것을 알 수 있습니다. 이점으로 보아 Feature로 사용 가 능해 보입니다.

**average (** 평균 **)**



# # # 데이터의 총 평균 시각화 비교

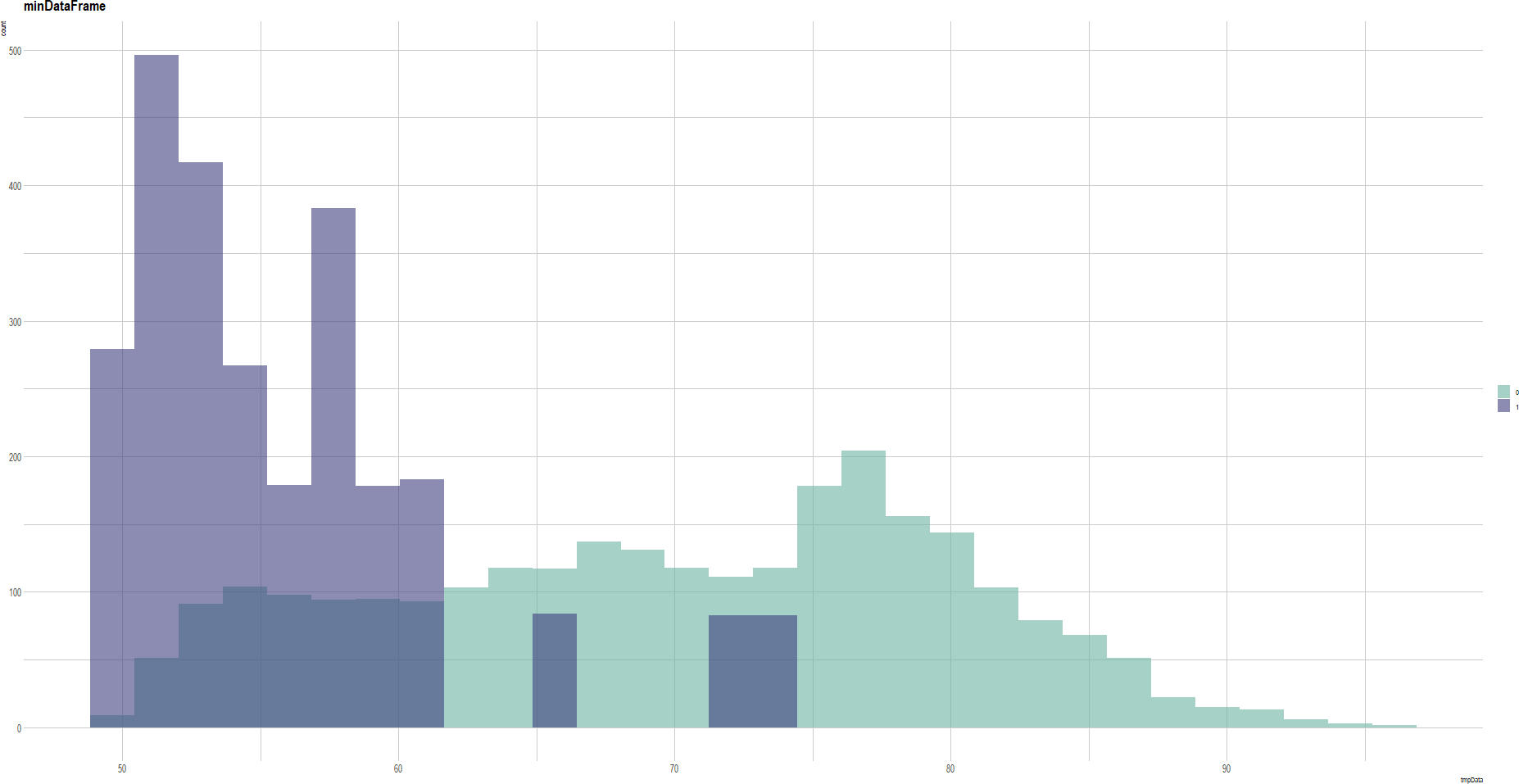
ArithmeticStatFuc( inputData = TrainUPData, mean,

"mean" )

# #

평균 그래프는 sum 그래프와 모양이 똑같이 나왔습니다.

**min (** 최소값 **)**



# # # 데이터의 총 최소값 시각화 비교

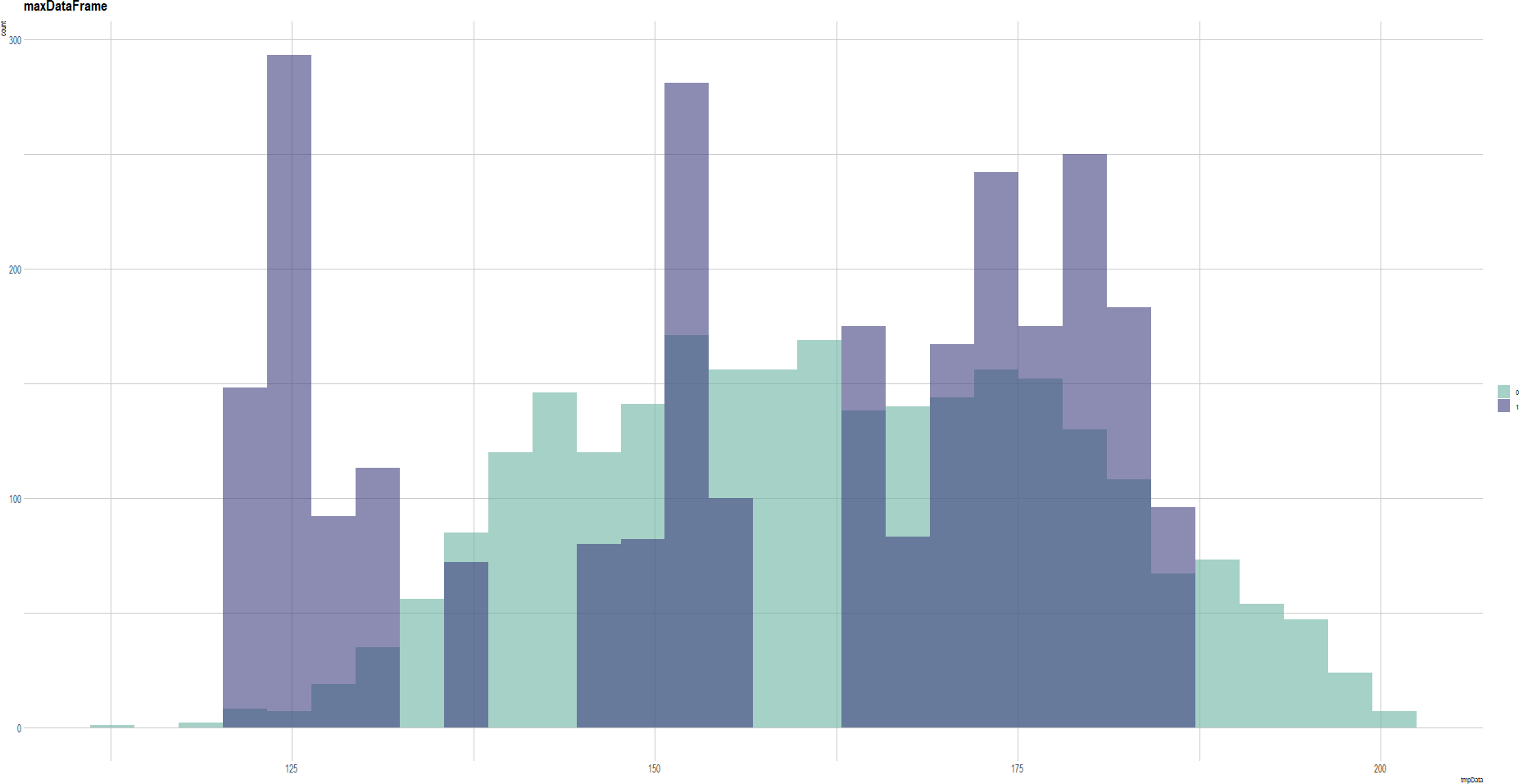
ArithmeticStatFuc( inputData = TrainUPData, min,

"min" )

# #

min 그래프는 합, 평균 보다 더 작은 쪽으로 몰려있다. 이 분포도 Feature로 사용이 가능해 보입니다.

**max (** 최대값 **)**



# # # 데이터의 총 최대값 시각화 비교

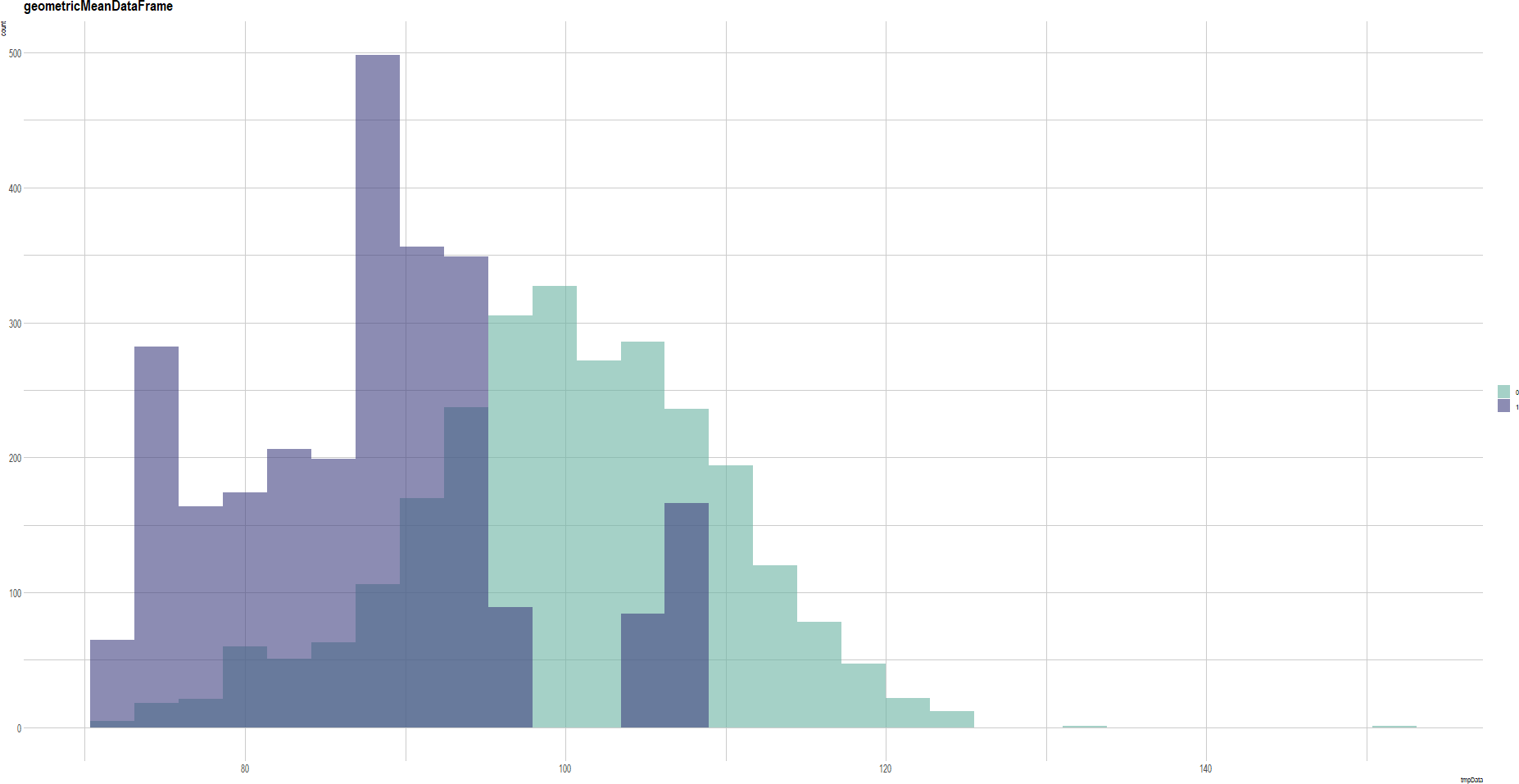
ArithmeticStatFuc( inputData = TrainUPData, max,

"max" )

# #

최대 값은 sum, average, min보다 저혈압의 데이터가 분포해 있어 feature로 사용이 불가능해 보입니 다.

**geometric mean (** 기하 평균 **)**



# # # 데이터의 총 기하 평균 시각화 비교

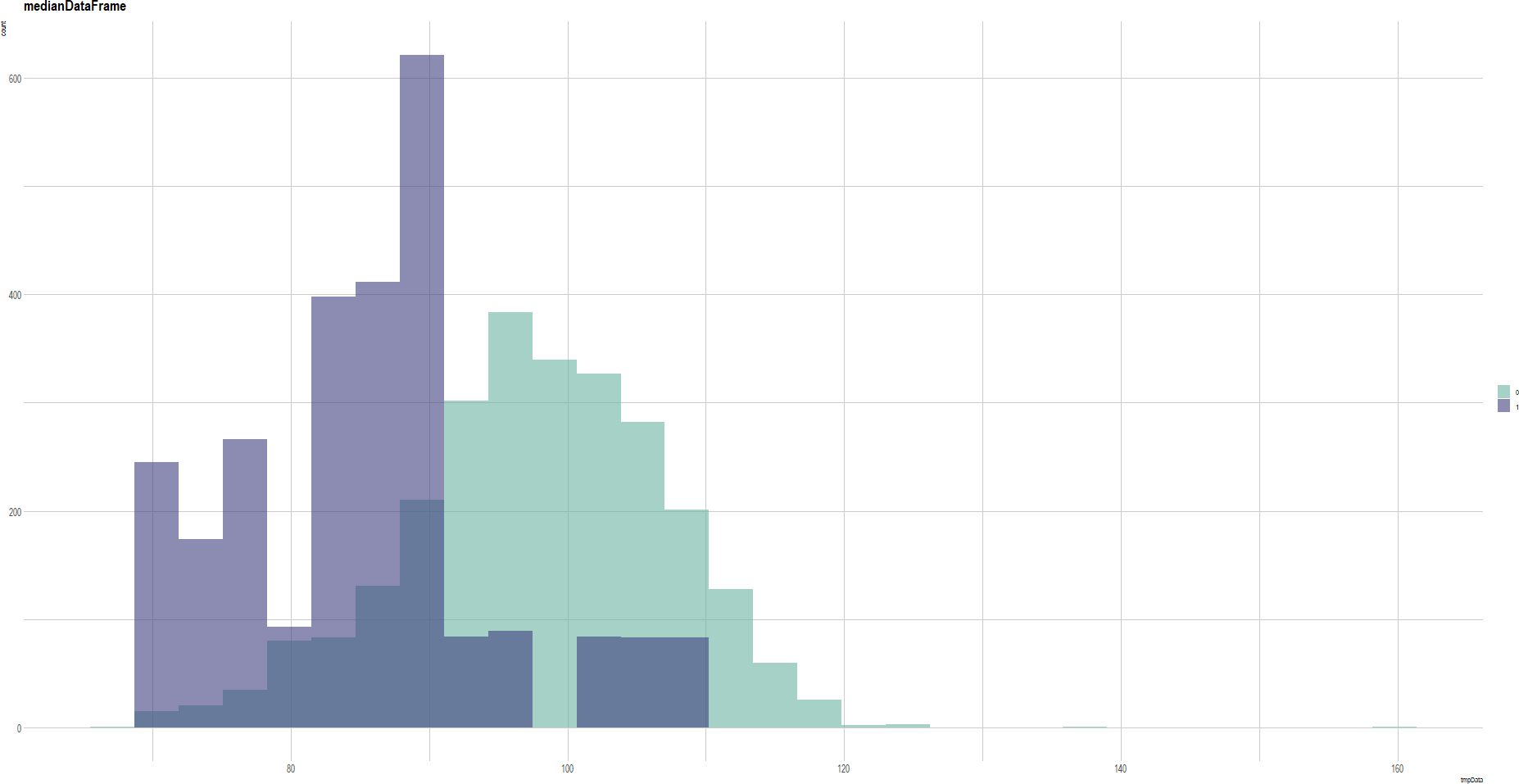
ArithmeticStatFuc( inputData = TrainUPData,

psych::geometric.mean, "geometricMean" )

# #

기하 평균은 산술 평균이랑 비슷하지만 약간의 차이가 있습니다. 정상 혈압의 중간 이하쪽에 분포해 있습니다.

**median (** 중위값 **)**



# # # 데이터의 총 중위값 시각화 비교

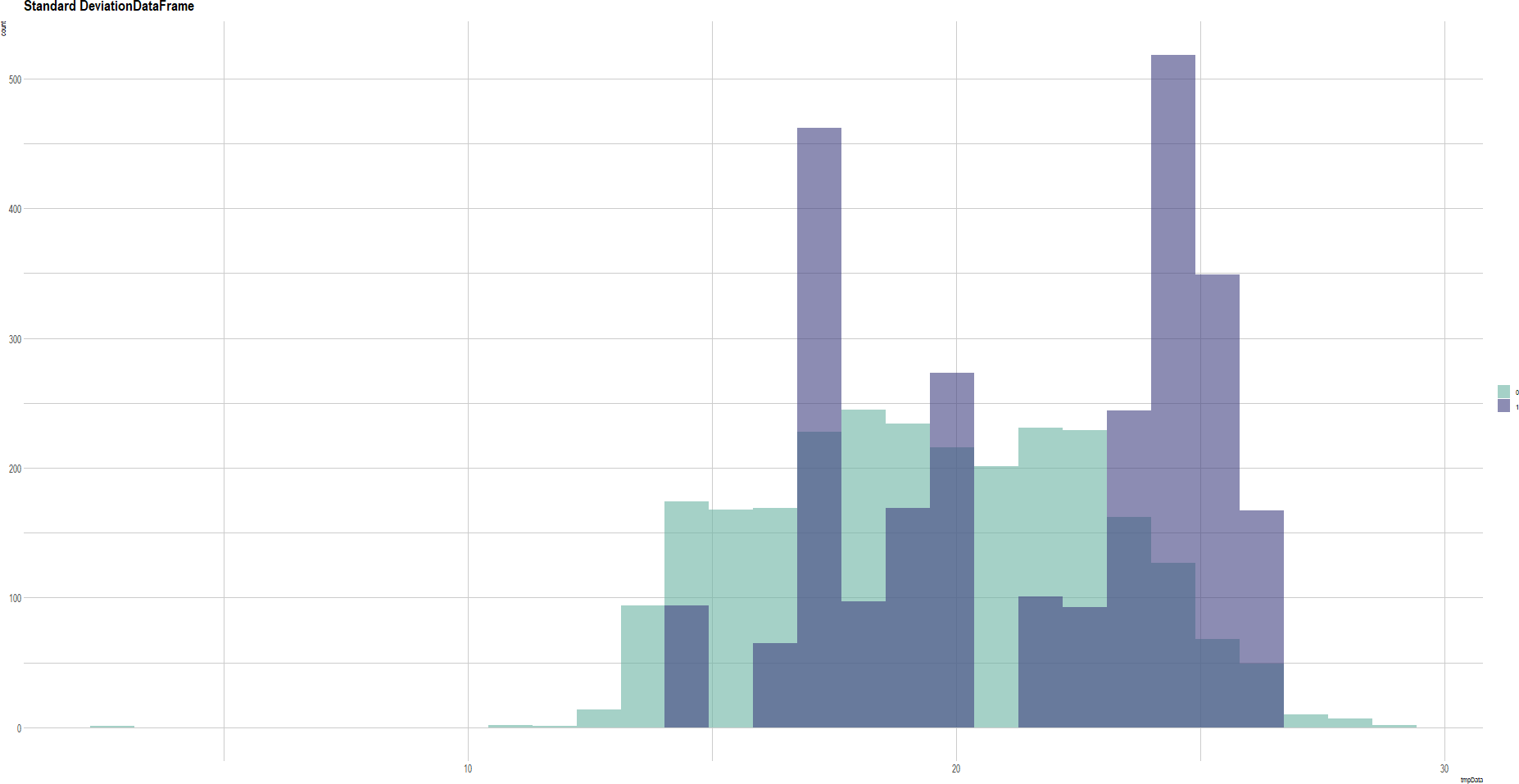
ArithmeticStatFuc( inputData = TrainUPData, median,

"median" )

# #

중위 값의 저혈압 데이터는 정상데이터의 그래프에서 주로 50% 미만에 속하지만 예외인 데이터가 존재합 니다.

**Standard Deviation (** 표준 편차 **)**



# # # 데이터의 총 표준편차 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,

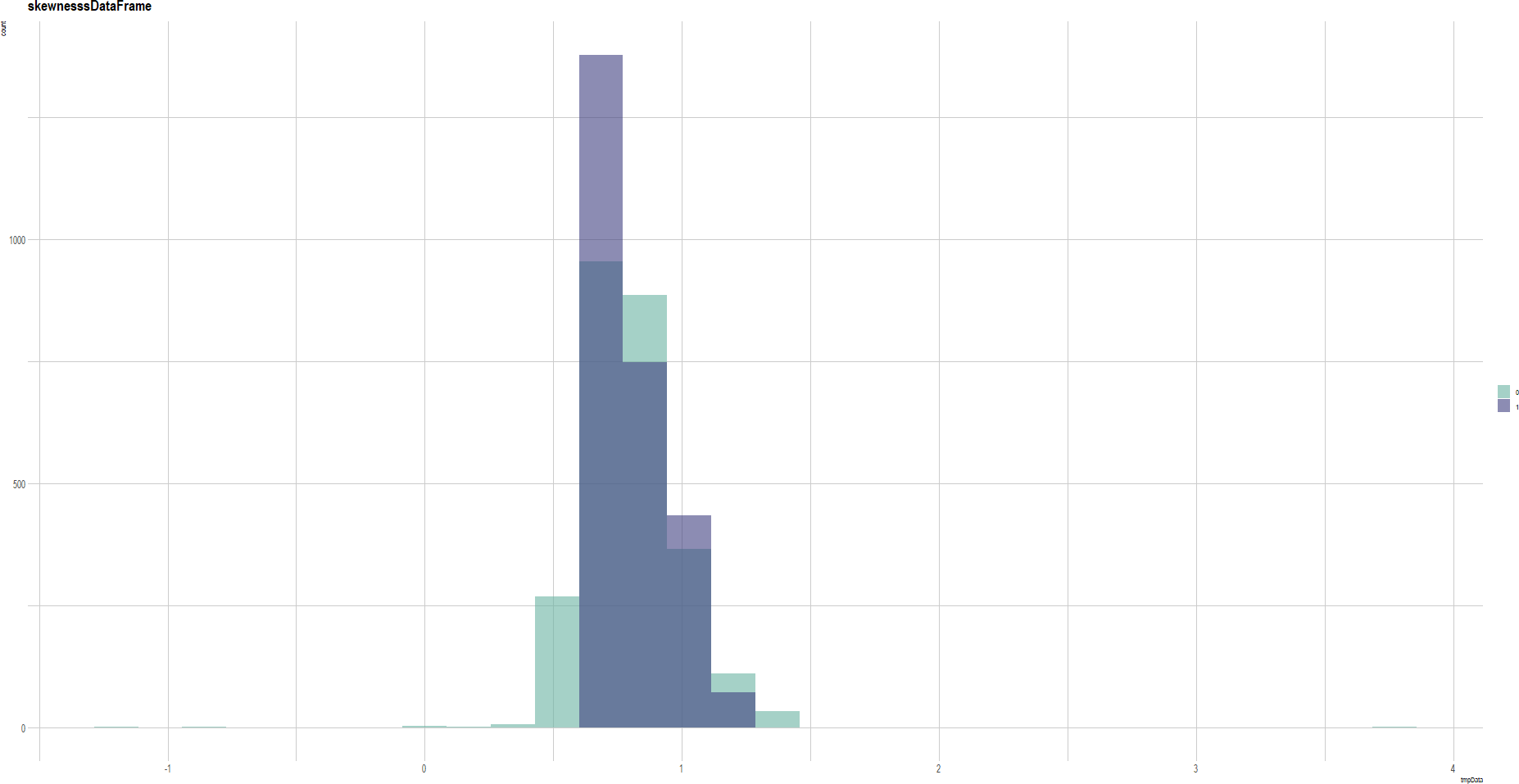
stats::sd,

"Standard Deviation" )

# #

표준 편차 분석은 저혈압 데이터가 히스토 그램에 골고루 분포해 있는것으로 보아 feature로 선택하기 부 적절해 보입니다.

**skewness (** 왜도 **)**



# # # 데이터의 총 왜도 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,

e1071::skewness, "skewnesss" )

# #

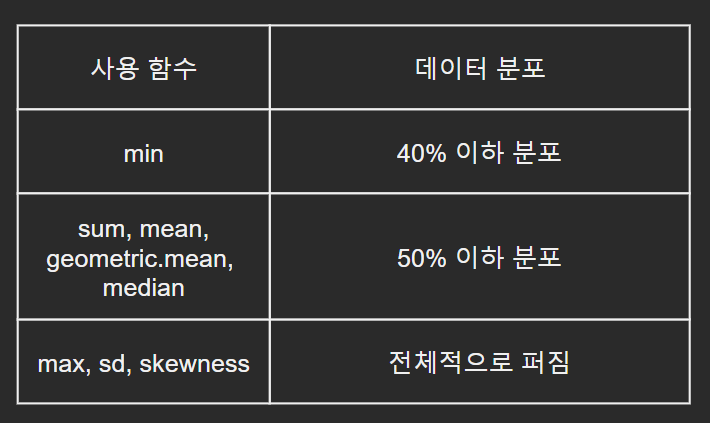
왜도 분석도 골고루 분포되어 있어서 feature로 선택하기 부적절해보입니다.

## 산술 통계 데이터 분석 결과

사용 통계 함수

sum, maen, min, max, geometric.mean, median, sd, skewness

데이터 분포 표



max, sd, skewness 저혈압 데이터가 전체적으로 분포해있어서 feature로 사용하기 힘들어 보인다.

나머지 데이터는 정상 데이터의 분포에서 50%이하 부분에 분포해있어 feature로 사용이 가능할것 같으나 정 상 데이터와 뚜렷하게 분류할 수 있는 값을 찾지 못해 다른 feature도 찾아봐야 할것으로 보인다.

## 모델링

# # # 모델 데이터

# train

statisticsPreProcessTrainData <- data.frame( sum = apply(TrainUPData[,1:15000], 1, sum), mean = apply(TrainUPData[,1:15000], 1, mean), min = apply(TrainUPData[,1:15000], 1, min), max = apply(TrainUPData[,1:15000], 1, max),

geometricMean = apply(TrainUPData[,1:15000], 1, psych::geometric.mean), median = apply(TrainUPData[,1:15000], 1, median),

sd = apply(TrainUPData[,1:15000], 1, stats::sd,),

skewness = apply(TrainUPData[,1:15000], 1, e1071::skewness), event = as.factor(TrainUPData$event)

)

# Test

statisticsPreProcessTestData <- data.frame( sum = apply(TestData[,1:15000], 1, sum), mean = apply(TestData[,1:15000], 1, mean), min = apply(TestData[,1:15000], 1, min), max = apply(TestData[,1:15000], 1, max),

geometricMean = apply(TestData[,1:15000], 1, psych::geometric.mean),

median = apply(TestData[,1:15000], 1, median), sd = apply(TestData[,1:15000], 1, stats::sd,),

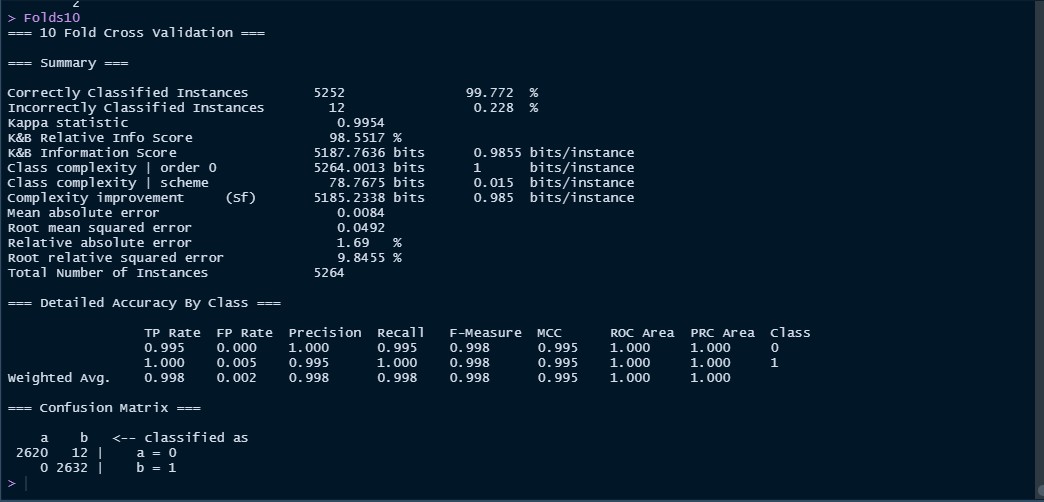
skewness = apply(TestData[,1:15000], 1, e1071::skewness), event = as.factor(TestData$event)

)

# #

모델 호출 및 Train

**RF statistic model**



# # # 모델 호출

RF <- RWeka::make\_Weka\_classifier("weka/classifiers/trees/RandomForest") RFModelstatistic <- RF(as.factor(event)~., data=statisticsPreProcessTrainData) summary(RFModelstatistic)

Folds10 <- evaluate\_Weka\_classifier(RFModelstatistic,

numFolds = 10, complexity = TRUE, class = TRUE)

# 모델 저장

# setwd(MODEL\_PATH)

# .jcache(RFModelstatistic$classifier)

# save(RFModelstatistic, file="RFModelstatistic.rda")

# 예측

predStatistic <- predict(RFModelstatistic, newdata = statisticsPreProcessTestData[1:8])

# #

**Model Test**

# # predShowConfusionMatrix <- function(TargetData, predData)

{

# # # confusionMatrix 시각화를 만드는 함수 입니다.

# #

confusionMatrixData <- tibble("target" = TargetData,

"prediction" = predData) basic\_table <- table(confusionMatrixData)

cfm <- as\_tibble(basic\_table)

cvms::plot\_confusion\_matrix(cfm,

target\_col = "target", prediction\_col = "prediction", counts\_col = "n",

palette = "Greens" )

}

AllDataCM <- predShowConfusionMatrix(statisticsPreProcessTestData$event, predStatistic) # 이미지 저장

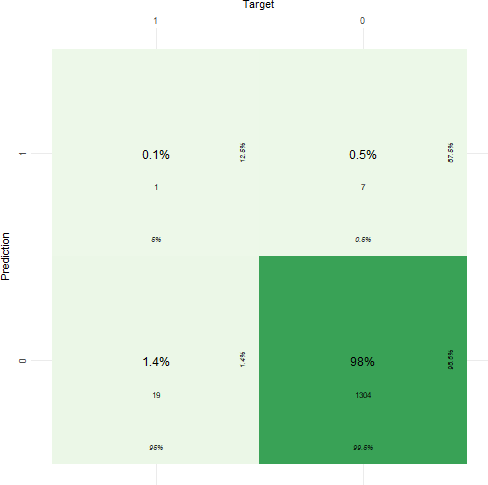
saveggplot( plot = AllDataCM, fileName = "AllDataCM", width = 600, height = 500) # #

사용 통계 함수

sum, maen, min, max, geometric.mean, median, sd, skewness

위의 산술 통계를 모두 feature로 잡아 모델을 Test한 결과

### RF statistics Confusion Matrix



결과로는 저혈압을 20개중에 1개를 예측했습니다.

### NB statistic model

# # # NB 모델

# 모델 호출

NBModelST <- naiveBayes(as.factor(event)~., data=statisticsPreProcessTrainData) summary(NBModelST)

# 모델 저장

# setwd(MODEL\_PATH)

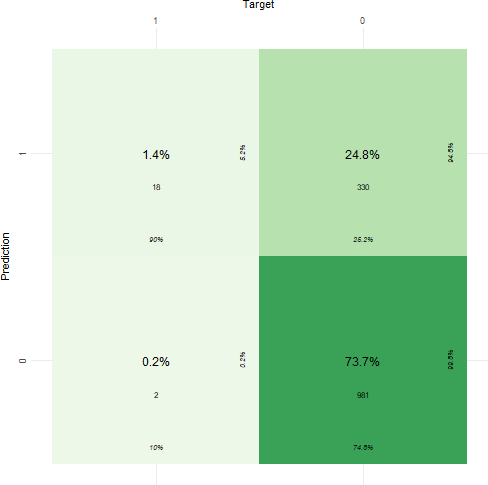
# save(NBModelST, file="NBModelST.rda")

# 예측

predNBST <- predict(NBModelST, newdata = statisticsPreProcessTestData[1:8]) showNBST <- predShowConfusionMatrix(statisticsPreProcessTestData$event, predNBST) setwd(IMG\_PATH)

# 이미지 저장

saveggplot( plot = showNBST, fileName = "showNBST", width = 600, height = 500) # #



RF 모델 보다 저혈압 예측 성능이 20건 중에서 18건으로 증가 하였다 하지만 정상혈압 예측률이 98%에서 73%로 떨어졌다.

## 피크분석

시각화를 위해서 데이터를 나눠줍니다.

# # # 저혈압 데이터 추출

LowBPData <- TrainData %>% filter( event == 1

)

# 저혈압 데이터 추출

NomalBPData <- TrainData %>% filter( event == 0

)

# #

정상혈압 데이터와 저혈압 데이터 시각화 비교

#

# 첫번째 행 데이터

#

# 저혈압

LowBPDataRowOne <- data.frame( xValue = seq(15000),

BP = as.numeric(LowBPData[1,1:15000])

)

# 정상혈압

NomalBPRowOne <- data.frame( xValue = seq(15000),

BP = as.numeric(NomalBPData[1,1:15000])

)

#

#

#

# 저혈압 데이터

LowBPPlot <- LowBPDataRowOne %>% ggplot( aes(x=xValue ,y=BP)) + geom\_line() +

ggtitle("LowBPPlot")

#

# 이미지 저장

saveggplot( plot = LowBPPlot, fileName = "LowBPPlot", width = 600, height = 500)

# #

#

# 정상 데이터 데이터

NomalBPPlot <- NomalBPData %>% ggplot( aes(x=xValue ,y=BP)) + geom\_line() +

ggtitle("NomalBPPlot")

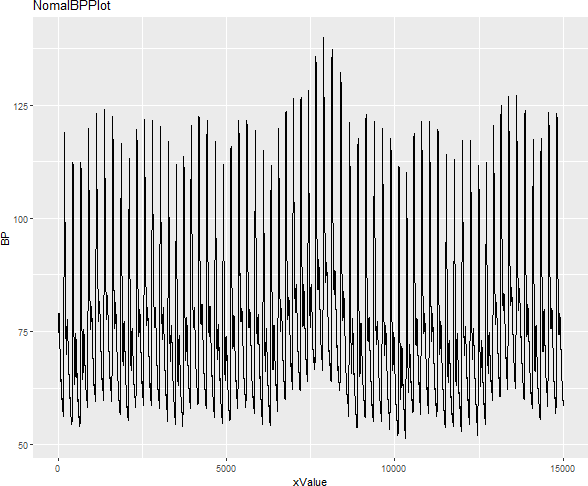
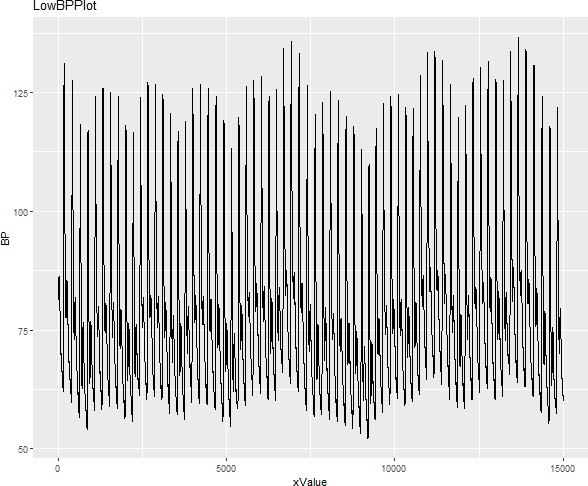
#

# 이미지 저장

saveggplot( plot = NomalBPPlot, fileName = "NomalBPPlot", width = 600, height = 500)

# #

Low blood pressure Normal blood pressure



둘의 시각화를 보았을 때 별로 차이가 없는 것을 알 수 있습니다.

피크 추출

피크 시각화 함수

# # # 피크 시각화 함수

MakePeakPlot <- function(LinePlotData, PointPlotData, filename)

{

# # # confusionMatrix 시각화를 만드는 함수 입니다.

# #

tmpPeakPlot <- LinePlotData %>% ggplot() + geom\_line(aes(x=xValue ,y=BP)) + geom\_point(data = PointPlotData,

aes(x=PointPlotData[,2], y=PointPlotData[,1]), color="red") +

ggtitle(paste0(filename))

saveggplot( plot = tmpPeakPlot,

fileName = filename, width = 600, height = 500)

}

# # # 모델 저장

# setwd(MODEL\_PATH)

# .jcache(RFModelPeak$classifier)

# save(RFModelPeak, file="RFModelPeak.rda")

peak 데이터를 추출후 시각화 합니다.

# # # 피크 추출

# 저혈압

LowPeak <- as.data.frame(pracma::findpeaks(LowBPDataRowOne$BP))

LowBPPeakPlot <- LowBPDataRowOne %>% ggplot() +

geom\_line(aes(x=xValue ,y=BP)) +

geom\_point(data = LowPeak, aes(x=LowPeak[,2], y=LowPeak[,1]), color="red") + ggtitle("Low BP Peak Plot")

saveggplot( plot = LowBPPeakPlot,

fileName = "LowBPPeakPlot", width = 600, height = 500)

# 정상 혈압

NomalPeak <- as.data.frame(pracma::findpeaks(NomalBPRowOne$BP))

NomalBPPeakPlot <- NomalBPRowOne %>% ggplot() +

geom\_line(aes(x=xValue ,y=BP)) +

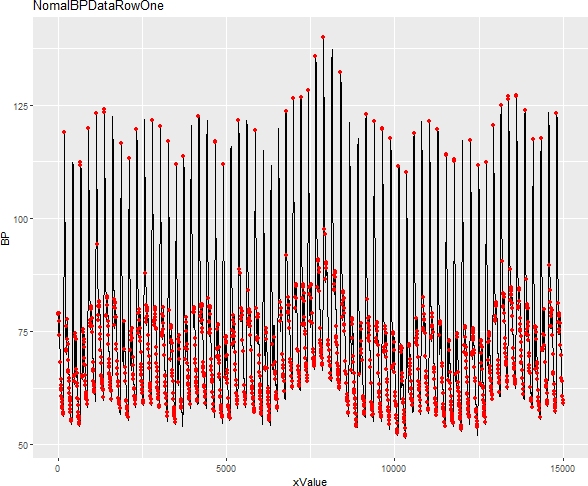
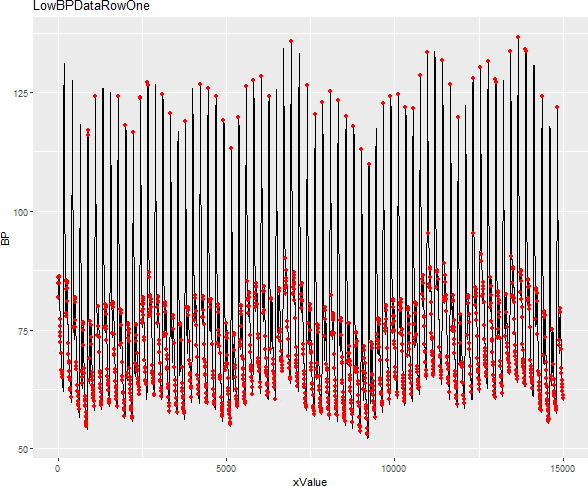
geom\_point(data = NomalPeak, aes(x=NomalPeak[,2], y=NomalPeak[,1]), color="red") + ggtitle("Nomal BP Peak Plot")

saveggplot( plot = NomalBPPeakPlot,

fileName = "NomalBPPeakPlot", width = 600, height = 500)

# #

Low blood pressure Normal blood pressure



# 피크의 수 LowPeak %>% nrow # [1] 1392

# 피크의 수 NomalPeak %>% nrow # [1] 1493

위의 피크를 보아 혈압이 높을 때 보다 낮을 때 피크가 많이 존재하는 것으로 보인다. 현재의 1분을 가지고 1분뒤의 혈압을 예측하는 것이니 현재 혈압이 낮으면

1분뒤의 혈압도 낮아 저혈압으로 나올것으로 추측

높은 피크보단 낮을 피클 추출한것을 feature로 학습 시켰을 경우 더 좋은 결과가 나올것으로 예측하여 데이터 의 위 아래를 뒤집어 Peak를 추출하였습니다.

# # # 피크 추출

# 시계열 데이터를 뒤집고

# Peak 추출한 시각화

# 저혈압

LowCHPeak <- as.data.frame(pracma::findpeaks(-LowBPDataRowOne$BP,

zero = "-", # 80이하인 피크만 추출

minpeakheight = -80))

LowBPCHDataRowOne <- data.frame( xValue = LowBPDataRowOne$xValue, BP = -LowBPDataRowOne$BP

)

MakePeakPlot(

LinePlotData = LowBPCHDataRowOne, PointPlotData = LowCHPeak, filename = "LowBPCHDataRowOne"

)

# 정상 혈압

NomalCHPeak <- as.data.frame(pracma::findpeaks(-NomalBPRowOne$BP,

zero = "-",

minpeakheight = -80))

NomalBPCHDataRowOne <- data.frame( xValue = NomalBPRowOne$xValue, BP = -NomalBPRowOne$BP

)

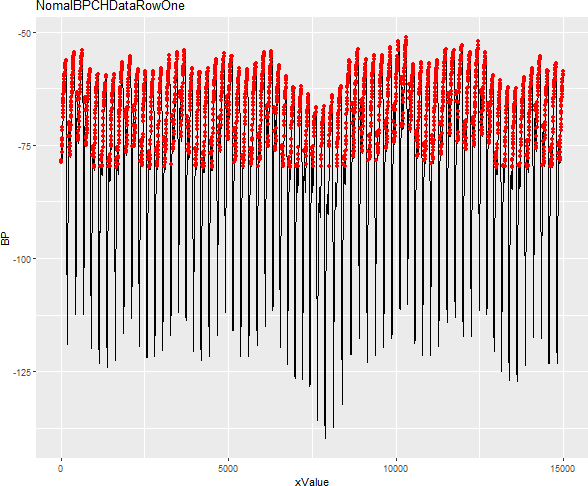
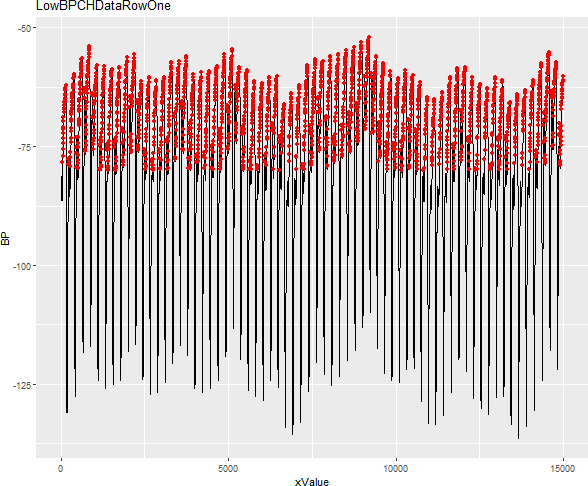
MakePeakPlot(

LinePlotData = NomalBPCHDataRowOne, PointPlotData = NomalCHPeak, filename = "NomalBPCHDataRowOne"

)

# #

Low blood pressure Normal blood pressure



# 피크의 수 NomalCHPeak %>% nrow # [1] 2664

# 피크의 수 LowCHPeak %>% nrow # [1] 2392

피크 수를 보았을 때 일반적으로 피크를 추출했을 떄 보다. 피크가 더 많이 추출 되었다. 저혈압과 정상혈압의 차이가 300peak정도 차이가 난다.

다른 피크들도 알아보기 위해 전처리를 실시하였다.

#### 병렬처리 코드

피크 전처리중 컴퓨팅 속도가 너무 느려 알아보니 병렬처리방법이 있어 사용하기로 하였다.

# # # 피크 전부 추출

# 병렬처리

# datalist <- list()

# 사용할 코어 갯수와 클러스터 선정

cores <- parallel::detectCores() - 1

cluster <- parallel::makeCluster(spec = cores)

# 클러스터를 등록

doParallel::registerDoParallel(cl = cluster) for ( num in 100:50 ){

print(num)

datalist[[paste0('v',num)]] <- foreach(i = 1:nrow(TrainUPData),

.packages=c('dplyr','pracma'),

.combine=rbind ) %dopar% {

tmp <- pracma::findpeaks(-as.numeric(TrainUPData[i,1:15000]),

zero = "-", # 80이하인 피크만 추출

minpeakheight = -num ) %>% nrow if(is.null(tmp)) {tmp <- -1}

c( i, tmp )

}

}

# 클러스터 해제

parallel::stopCluster(cluster)

# 데이터 합치기

# col 이름 수정

for (i in 100:50){

colnames(Peakdatalist[[paste0('v',i)]]) <- c("rowNumber", paste0('nrow',i))

}

# 데이터 합치기

Peakdataframe <- data.frame( rowNumber = seq(5264)

)

for (i in 100:50){

Peakdataframe <- inner\_join(Peakdataframe,

as.data.frame(Peakdatalist[[paste0('v',i)]]), by='rowNumber')

}

Peakdataframe[["evnet"]] <- TrainUPData$event # 데이터 -1 > 0으로 변경

for (i in 100:50){ Peakdataframe[[paste0('nrow',i)]][Peakdataframe[[paste0('nrow',i)]]==-1] <- 0

}

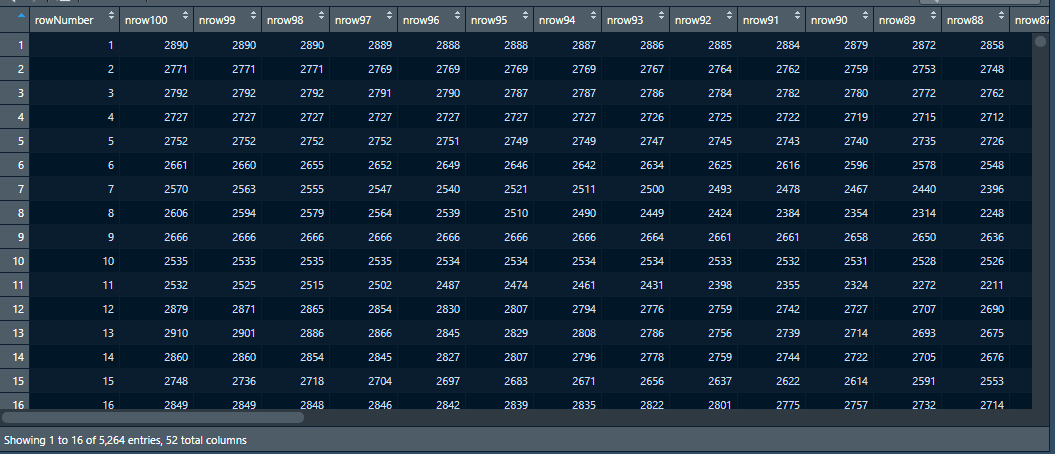
# 데이터 저장

# save(Peakdataframe, file = "Peakdatalist.RData")

# #

위의 코드는 혈압 100이하부터 50이하까지의 피크를 추출한 데이터 입니다.

**Peak** 추출 **Data**



저혈압**,** 정상혈압**, Peak** 추출수 비교

저혈압 데이터와 정상혈압 데이터의 차이를 비교하기 위해 혈압 100이하 부터 50이하 까지 Peak의 수를 추 출하여 평균 값을 시각화 했습니다.

# # # Peak 예시 시각화

# 데이터 만들기

LowPeakPlotData <- Peakdataframe %>%

filter(evnet ==1) %>% select(2:52) %>% apply(2,base::mean) %>% t

NomalPeakPlotData <- Peakdataframe %>%

filter(evnet ==0) %>% select(2:52) %>% apply(2,base::mean) %>% t

PeakLineData <- rbind(LowPeakPlotData, NomalPeakPlotData) %>% t %>%

as.data.frame

colnames(PeakLineData) <- c('LowPeakPlotData', 'NomalPeakPlotData') PeakLineData["xlabel"] <- 100:50

PeakLinePlot<- PeakLineData %>% ggplot(aes(x=xlabel)) + geom\_line(aes(y=LowPeakPlotData), color="red") + geom\_point(aes(y=LowPeakPlotData), color="red") + geom\_line(aes(y=NomalPeakPlotData), color="blue") +

geom\_point(aes(y=NomalPeakPlotData), color="blue") + xlim(100,50) +

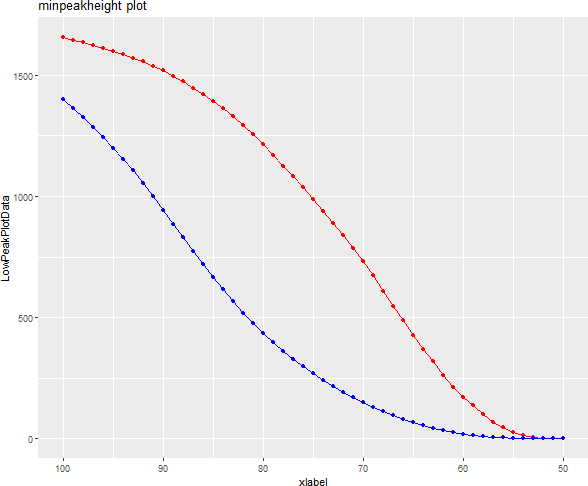
ggtitle("minpeakheight plot")

# 이미지 저장

saveggplot( plot = PeakLinePlot, fileName = "PeakLinePlot", width = 600, height = 500)

# #

저혈압 : 빨간색 정상혈압 : 파란색



위의 그래프로 보아서 저혈압이 혈압 100이하의 Peak를 추출하였을 때 정상혈압보다 더 많이 추출되는것을 알 수 있었습니다. 혈압 65이하 까지는 저혈압 Peak의 평균 수가 더 많았고 50이하로 내려가면 둘다 0으로 Peak가 추출 안되는것을 알 수 있었습니다.

둘의 차이가 보이는 것으로 보아 feature로 사용하면 Train이 잘 될것으로 판단 model에 돌려보기로 했습니 다.

**Test** 데이터 전처리

# # # Test 피크 전부 추출

TestData # 병렬처리

datalistTest <- list()

# 사용할 코어 갯수와 클러스터 선정

cores <- parallel::detectCores() - 1

cluster <- parallel::makeCluster(spec = cores)

# 클러스터를 등록

doParallel::registerDoParallel(cl = cluster) for ( num in 100:50 ){

print(num)

datalistTest[[paste0('v',num)]] <- foreach(i = 1:nrow(TestData),

.packages=c('dplyr','pracma'),

.combine=rbind ) %dopar% {

tmp <- pracma::findpeaks(-as.numeric(TestData[i,1:15000]),

zero = "-", # 80이하인 피크만 추출

minpeakheight = -num ) %>% nrow if(is.null(tmp)) {tmp <- -1}

c( i, tmp )

}

}

# 클러스터 해제

parallel::stopCluster(cluster)

# 데이터 합치기

# col 이름 수정

for (i in 100:50){

colnames(datalistTest[[paste0('v',i)]]) <- c("rowNumber", paste0('nrow',i))

}

# 데이터 합치기

PeakTestdataframe <- data.frame(

rowNumber = seq(nrow(datalistTest[["v100"]]))

)

for (i in 100:50){

PeakTestdataframe <- inner\_join(PeakTestdataframe,

as.data.frame(datalistTest[[paste0('v',i)]]), by='rowNumber')

}

PeakTestdataframe[["event"]] <- PeakTestdataframe$event # 데이터 -1 > 0으로 변경

for (i in 100:50){ PeakTestdataframe[[paste0('nrow',i)]][PeakTestdataframe[[paste0('nrow',i)]]==-1] <- 0

}

# 데이터 저장

# save(PeakTestdataframe, file = "PeakTestdataframe.RData")

# #

## 모델링

### RF Peak Model Test

# # # 모델 호출

RF <- RWeka::make\_Weka\_classifier("weka/classifiers/trees/RandomForest") RFModelPeak <- RF(as.factor(event)~., data=Peakdataframe[,2:53]) summary(RFModelPeak)

Folds10 <- evaluate\_Weka\_classifier(RFModelPeak,

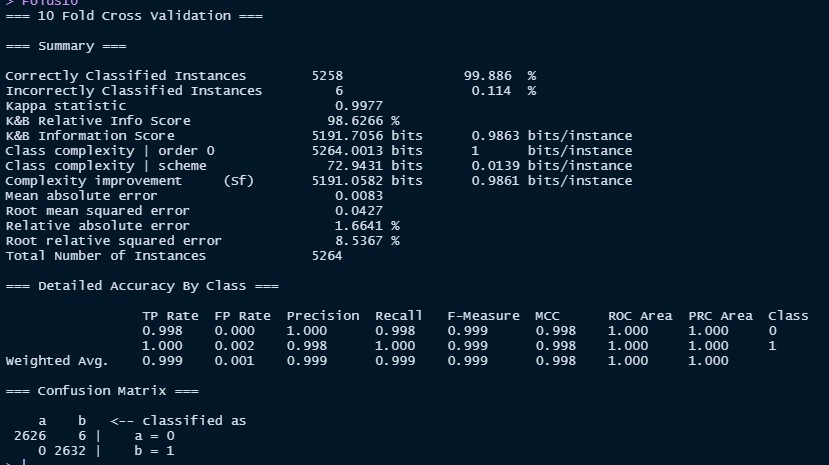
numFolds = 10, complexity = TRUE, class = TRUE)

# 모델 저장

# setwd(MODEL\_PATH)

# .jcache(RFModelPeak$classifier)

# save(RFModelPeak, file="RFModelPeak.rda")



**RF Peak Confusion Matrix**

# 예측

Peakpred <- predict(RFModelPeak, newdata = PeakTestdataframe[2:52])

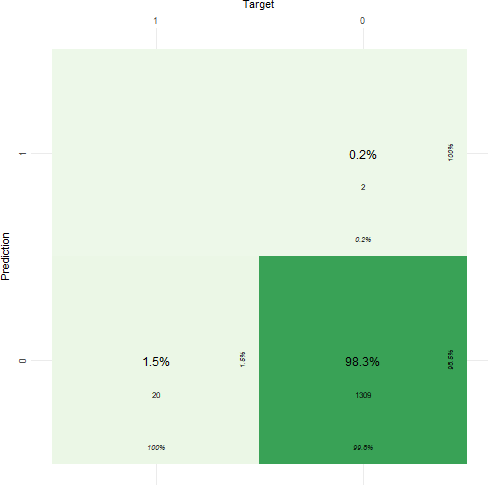
# #

# # PeackaCM <- predShowConfusionMatrix(PeakTestdataframe$event, Peakpred)

# 이미지 저장

saveggplot( plot = PeackaCM, fileName = "PeackaCM", width = 600, height = 500)

# #



Peak 데이터를 Train한 RF모델의 성능은 정상혈압 예측률이 0.3%더 상승했지만 산술통계 모델과 같이 저혈 압은 1개도 예측하지 못했습니다.

### NB Peak Model Test

# # # NB 모델

# 모델 호출

NBModelPK <- naiveBayes(as.factor(event)~., data=Peakdataframe[,2:53]) summary(NBModelPK)

# 모델 저장

# setwd(MODEL\_PATH)

# save(NBModelPK, file="NBModelPK.rda")

# 예측

predNBPK <- predict(NBModelPK, newdata = PeakTestdataframe[2:52]) showNBPK <- predShowConfusionMatrix(PeakTestdataframe$event, predNBPK)

# 이미지 저장

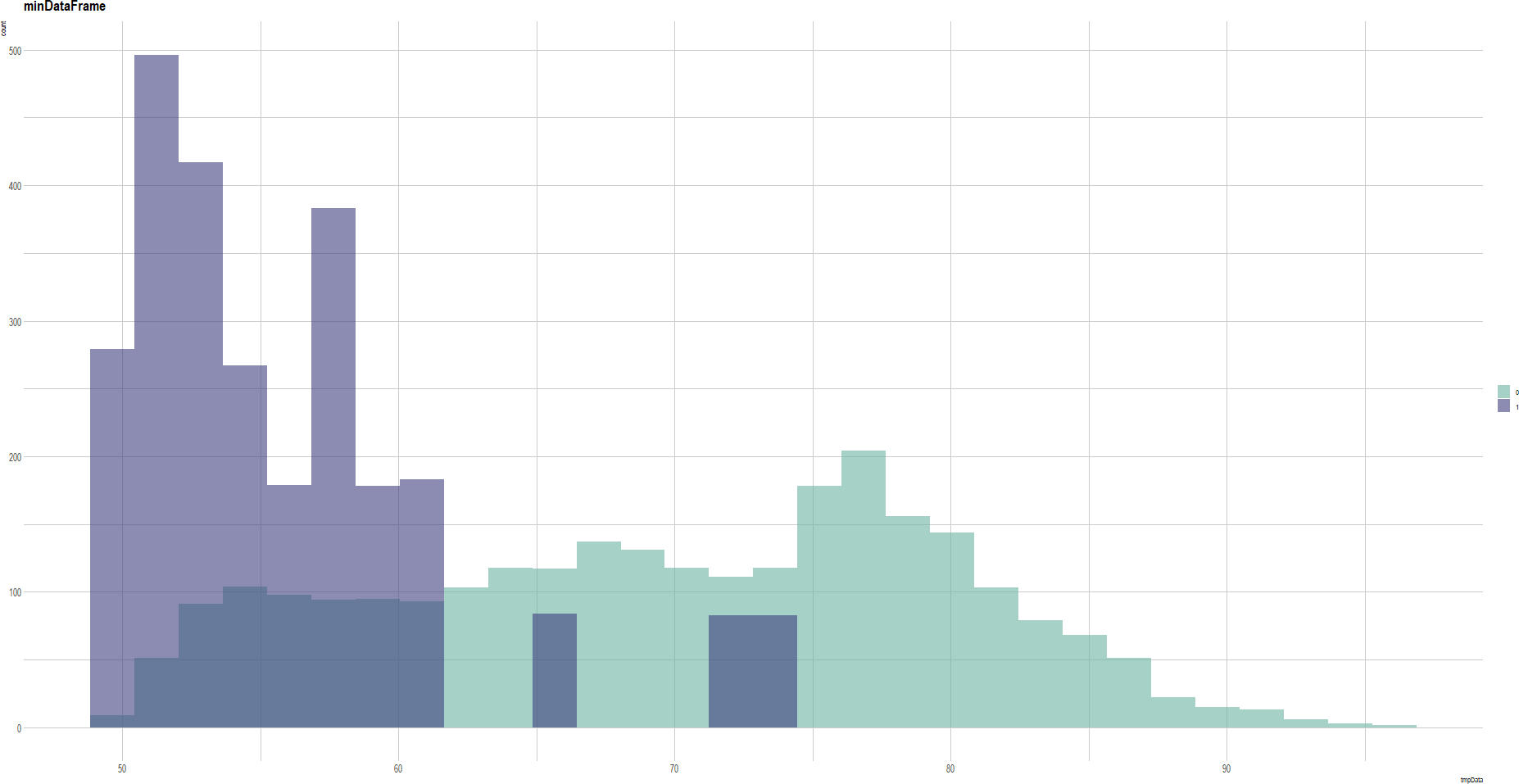
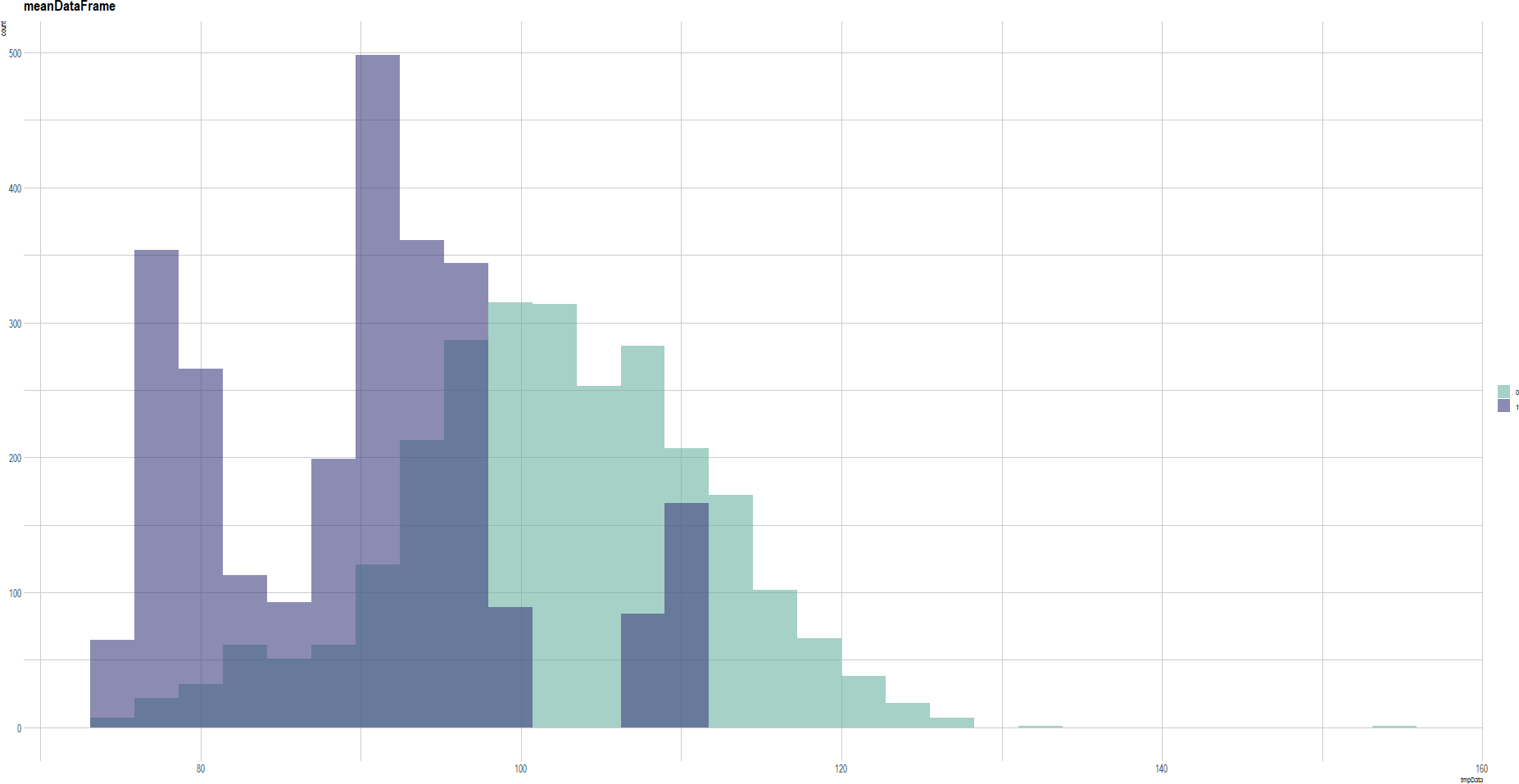
saveggplot( plot = showNBPK, fileName = "showNBPK", width = 600, height = 500) # #



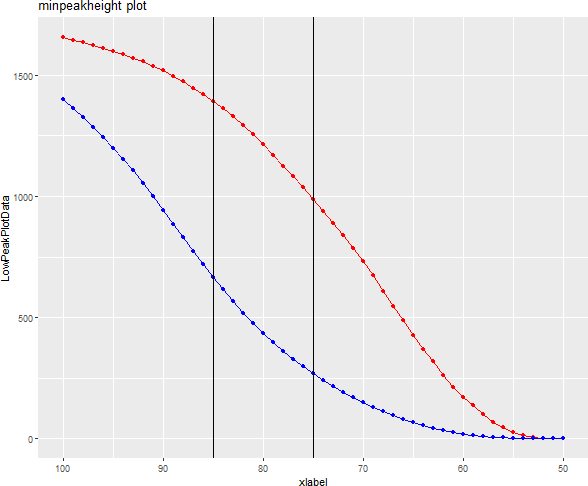
Peak feature 학습한 NB모델은 통계 모델 보다 저혈압 예측률은 떨어졌지만 정상혈압 예측률은 증가하였습 니다.

# Statistic & Peak 모델

평균 최소



Peak 85이하 ~ 75이하 데이터



정상혈압데이터와 저혈압데이터의 차이가 많이 나는 것들만 Feature로 정해 모델을 돌려보았습니다.

# # # 데이터 확인

nrow(statisticsPreProcessTrainData) nrow(statisticsPreProcessTestData)

nrow(Peakdataframe) nrow(PeakTestdataframe)

# > nrow(statisticsPreProcessTrainData) # [1] 5264

# > nrow(statisticsPreProcessTestData) # [1] 1331

# >

# > nrow(Peakdataframe) # [1] 5264

# > nrow(PeakTestdataframe) # [1] 1331

# #

데이터 전처리

# # #

nameslist <- c()

for (i in 85:75){

nameslist <- c(nameslist, paste0("nrow",i))

}

PeakstatisticTrainData <- cbind(statisticsPreProcessTrainData[c("mean", "min")],

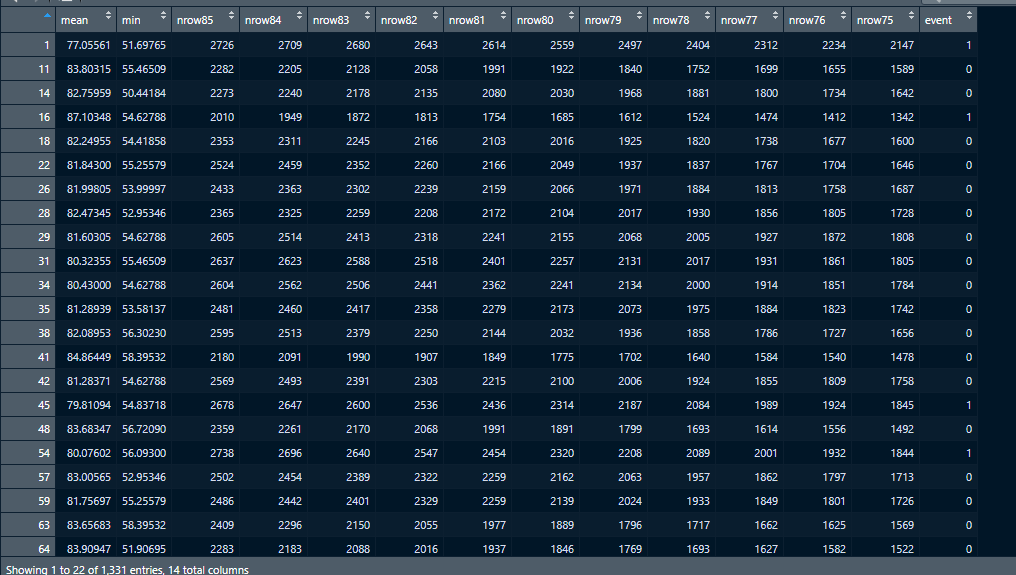
Peakdataframe[nameslist], event = Peakdataframe$event)

PeakstatisticTestData <- cbind(statisticsPreProcessTestData[c("mean", "min")],

PeakTestdataframe[nameslist], event = PeakTestdataframe$event)

# save(PeakstatisticTrainData, file = "PeakstatisticTrainData.RData") # save(PeakstatisticTestData, file = "PeakstatisticTestData.RData")

# #



**Model Trainning**

# # # 모델 호출

RF <- RWeka::make\_Weka\_classifier("weka/classifiers/trees/RandomForest") RFModelPKST <- RF(as.factor(event)~., data=PeakstatisticTrainData) summary(RFModelPKST)

Folds10 <- evaluate\_Weka\_classifier(RFModelPKST,

numFolds = 10, complexity = TRUE, class = TRUE)

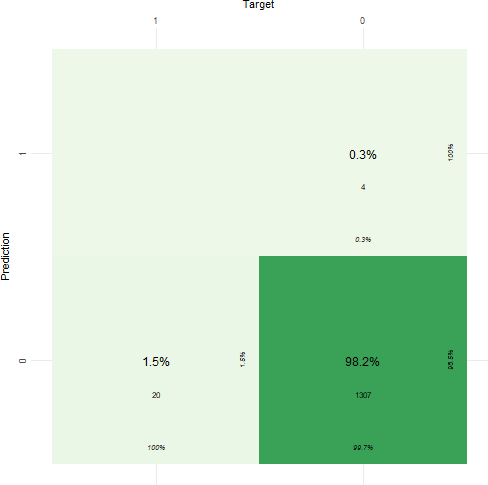
# 모델 저장

# setwd(MODEL\_PATH)

# .jcache(RFModelPKST$classifier)

# save(RFModelPKST, file="RFModelPKST.rda")

**statistics & Peak Confusion Matrix**



# 예측

predPKST <- predict(RFModelPKST, newdata = PeakstatisticTestData[1:13])

AllDataPKST <- predShowConfusionMatrix(PeakstatisticTestData$event, predPKST)

# 이미지 저장

saveggplot( plot = AllDataPKST, fileName = "AllDataPKST", width = 600, height = 500)

# #

산술통계 데이터의 mean, min feature와 Peak데이터의 minpeakheight 범위 7585 사이의 feature를 RF Model에 Train 시켰을때 static model보다는 정상혈압 예측률이 2 높아졌으며 Peak보다는 2 낮았다.

하지만 다른 RF model과 같이 저혈압 데이터는 예측하지 못했습니다.

### NB statistics & Peak model

# # # NB 모델

# 모델 호출

NBModelPKST <- naiveBayes(as.factor(event)~., data=PeakstatisticTrainData) summary(NBModelPKST)

# 모델 저장

# setwd(MODEL\_PATH)

# save(NBModelPKST, file="NBModelPKST.rda")

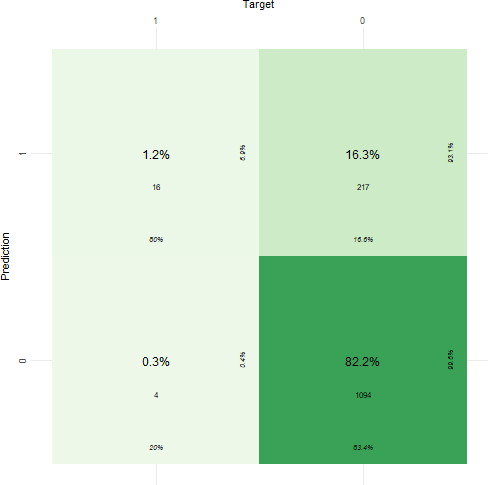
# 예측

predNBPKST <- predict(NBModelPKST, newdata = PeakstatisticTestData[1:13]) showNBPKST <- predShowConfusionMatrix(PeakstatisticTestData$event, predNBPKST)

# 이미지 저장

saveggplot( plot = showNBPKST, fileName = "showNBPKST", width = 600, height = 500) # #

**NB statistics & Peak Confusion Matrix**



저혈압 예측률이 조금 하양했지만 정상혈압 예측률이 82%로 Peak feature만 넣었을 때 보다 증가 하였습니 다.