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

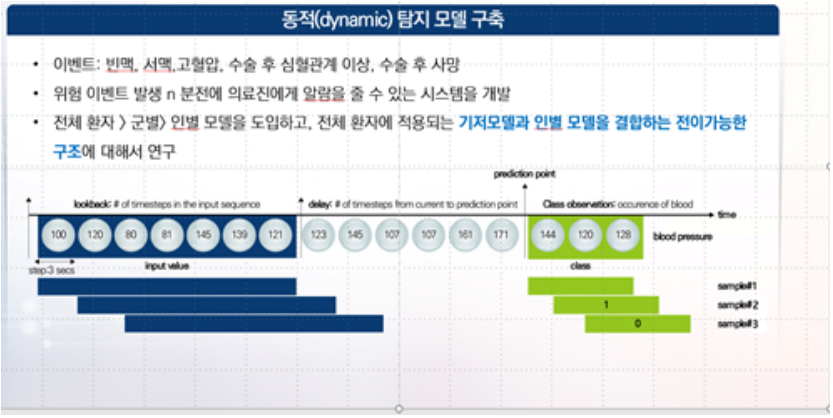
```
|  README.md
|
├─draw # draw file
|      이미지 설명.drawio
|      프로세스.drawio
|
├─README_IMG # 이미지 저장공간
|
└─src
    |  .Rhistory
    |  DataLoad.R # RData load 파일
    |  library.R # 라이브러리 설치 및 load 파일
    |  upSampling.R # Train Data upSampling 파일
    |
    ├─TesT데이터만들기 # 새로운 데이터를 TEST로 만들고
    |  |              #  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 예측하는 모델을 만들어야한다.

- class
  - 0 : 정상혈압
  - 1 : 저혈압

결과 표

전처리	사용 모델	정상혈압 예측률	저혈압 예측률
statistics	RF	1304 / 1311	1 / 20
	NB	981 / 1311	18 / 20
Peak	RF	1309 / 1311	0 / 20
	NB	1206 / 1311	13 / 20
statistics & Peak	RF	1307 / 1311	0 / 20
	NB	1094 / 1311	16 / 20

최종 결론

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

- NB 모델을 사용해서 저혈압 예측 성능이 RF보다 확실하게 향상하였습니다. 하지만 정상혈압예측률을 떨어졌습니다.
- 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)
{
```

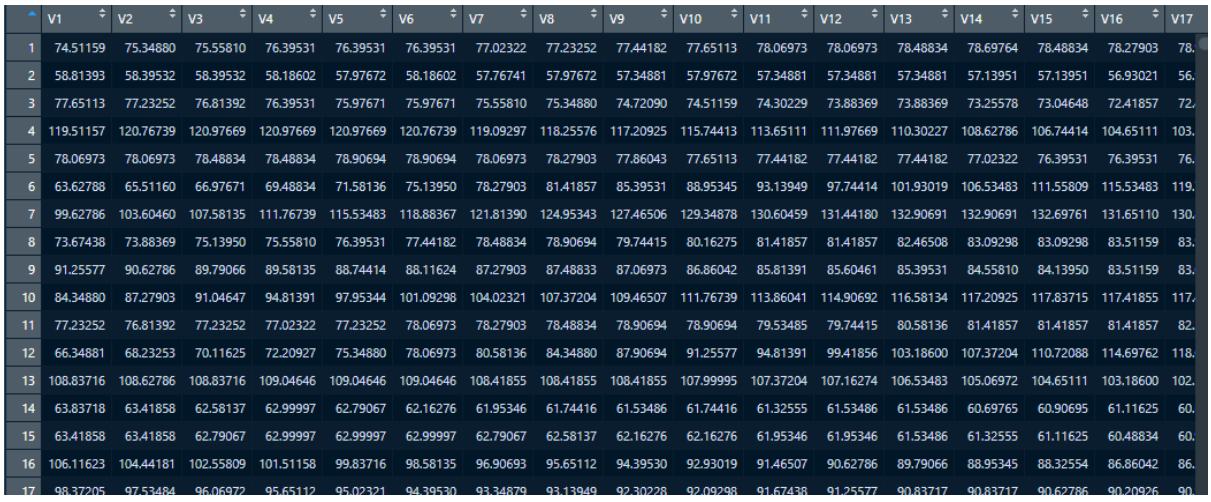
```
  strsplit(row, split=",")[[1]][2]
}


# 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



	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
1	74.51159	75.34880	75.55810	76.39531	76.39531	76.39531	77.02322	77.23252	77.44182	77.65113	78.06973	78.06973	78.48834	78.69764	78.48834	78.27903	78.
2	58.81393	58.39532	58.39532	58.18602	57.97672	58.18602	57.76741	57.97672	57.34881	57.97672	57.34881	57.34881	57.34881	57.13951	57.13951	56.93021	56.
3	77.65113	77.23252	76.81392	76.39531	75.97671	75.97671	75.55810	75.34880	74.72090	74.51159	74.30229	73.88369	73.88369	73.25578	73.04648	72.41857	72.
4	119.51157	120.76739	120.97669	120.97669	120.97669	120.76739	119.09297	118.25576	117.20925	115.74413	113.65111	111.97669	110.30227	108.62786	106.74414	104.65111	103.
5	78.06973	78.06973	78.48834	78.48834	78.90694	78.90694	78.06973	78.27903	77.86043	77.65113	77.44182	77.44182	77.44182	77.02322	76.39531	76.39531	76.
6	63.62788	65.51160	66.97671	69.48834	71.58136	75.13950	78.27903	81.41857	85.39531	88.95345	93.13949	97.74414	101.93019	106.53483	111.55809	115.53483	119.
7	99.62786	103.60460	107.58135	111.76739	115.53483	118.88367	121.81390	124.95343	127.46506	129.34878	130.60459	131.44180	132.90691	132.90691	132.69761	131.65110	130.
8	73.67438	73.88369	75.13950	75.55810	76.39531	77.44182	78.48834	78.90694	79.74415	80.16275	81.41857	81.41857	82.46508	83.09298	83.09298	83.51159	83.
9	91.25577	90.62786	89.79066	89.58135	88.74414	88.11624	87.27903	87.48833	87.06973	86.86042	85.81391	85.60461	85.39531	84.55810	84.13950	83.51159	83.
10	84.34880	87.27903	91.04647	94.81391	97.95344	101.09298	104.02321	107.37204	109.46507	111.76739	113.86041	114.90692	116.58134	117.20925	117.83715	117.41855	117.
11	77.23252	76.81392	77.23252	77.02322	77.23252	78.06973	78.27903	78.48834	78.90694	78.90694	79.53485	79.74415	80.58136	81.41857	81.41857	81.41857	82.
12	66.34881	68.23253	70.11625	72.20927	75.34880	78.06973	80.58136	84.34880	87.90694	91.25577	94.81391	99.41856	103.18600	107.37204	110.72088	114.69762	118.
13	108.83716	108.62786	108.83716	109.04646	109.04646	109.04646	108.41855	108.41855	108.41855	107.99995	107.37204	107.16274	106.53483	105.06972	104.65111	103.18600	102.
14	63.83718	63.41858	62.58137	62.99997	62.79067	62.16276	61.95346	61.74416	61.53486	61.74416	61.32555	61.53486	61.53486	60.69765	60.90695	61.11625	60.
15	63.41858	63.41858	62.79067	62.99997	62.99997	62.99997	62.79067	62.58137	62.16276	62.16276	61.95346	61.95346	61.53486	61.32555	61.11625	60.48834	60.
16	106.11623	104.44181	102.55809	101.51158	99.83716	98.58135	96.90693	95.65112	94.39530	92.93019	91.46507	90.62786	89.79066	88.95345	88.32554	86.86042	86.
17	98.37205	97.53484	96.06972	95.65112	95.02321	94.39530	93.34879	93.13949	92.30228	92.09298	91.67438	91.25577	90.83717	90.83717	90.62786	90.20926	90.

Train, Test

데이터를 Train과 Test로 나눴습니다.

비율

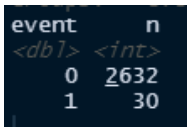
- Train : 2/3
- Test : 1/2

```
# ------------------------------------------------------------------------ #
# 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

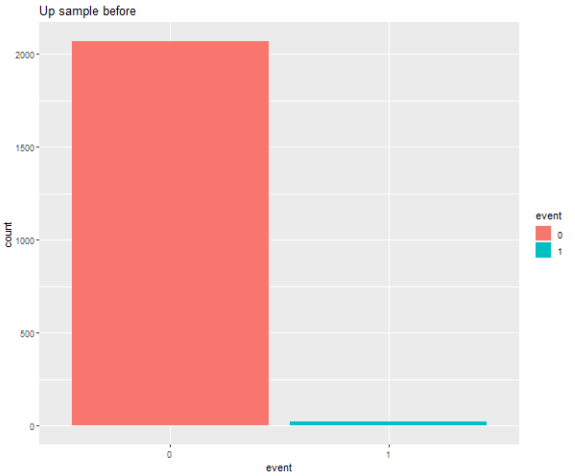
class label의 데이터 분포가 한쪽으로 몰려있어서 up샘플링 해주었습니다.

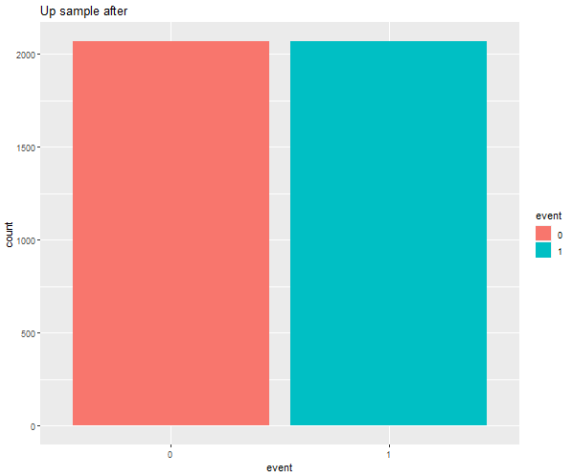
- 정상혈압 : 2632
- 저혈압 : 30



event	n
<dbl>	<int>
0	2632
1	30

```
# ------------------------------------------------------------------------ #
# Up sampleling
TrainUPData <- groupdata2::upsample(
  TrainData, # data
  cat_col = "event"
)
# ------------------------------------------------------------------------ #
```





산술 통계

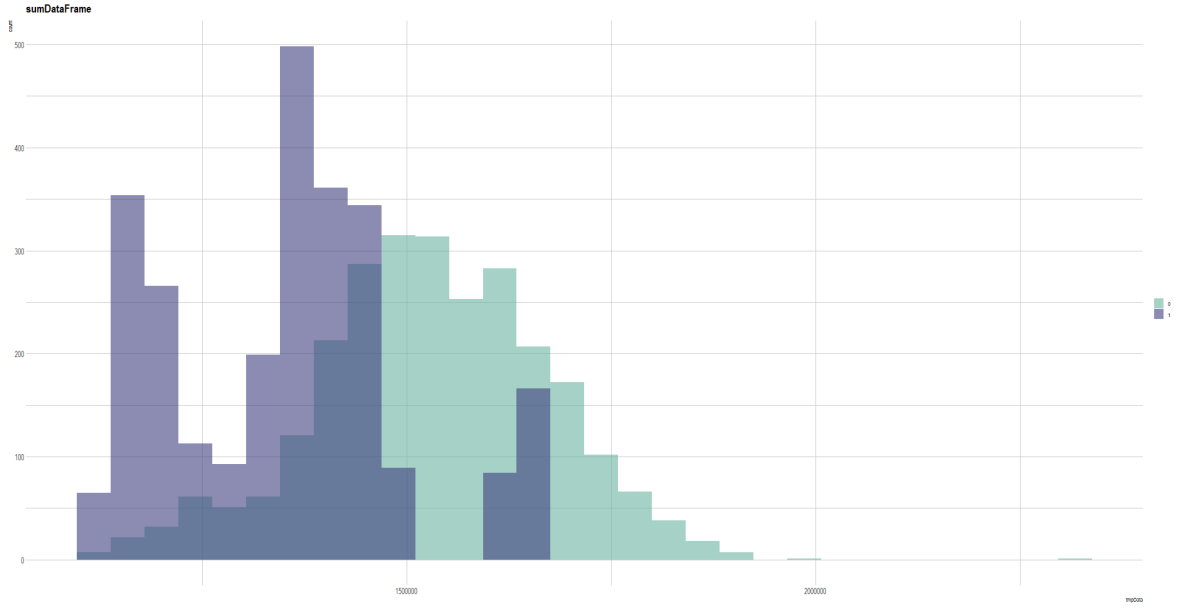
산술 통계 분석 함수

```
# ------------------------------------------------------------------------ #
# 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

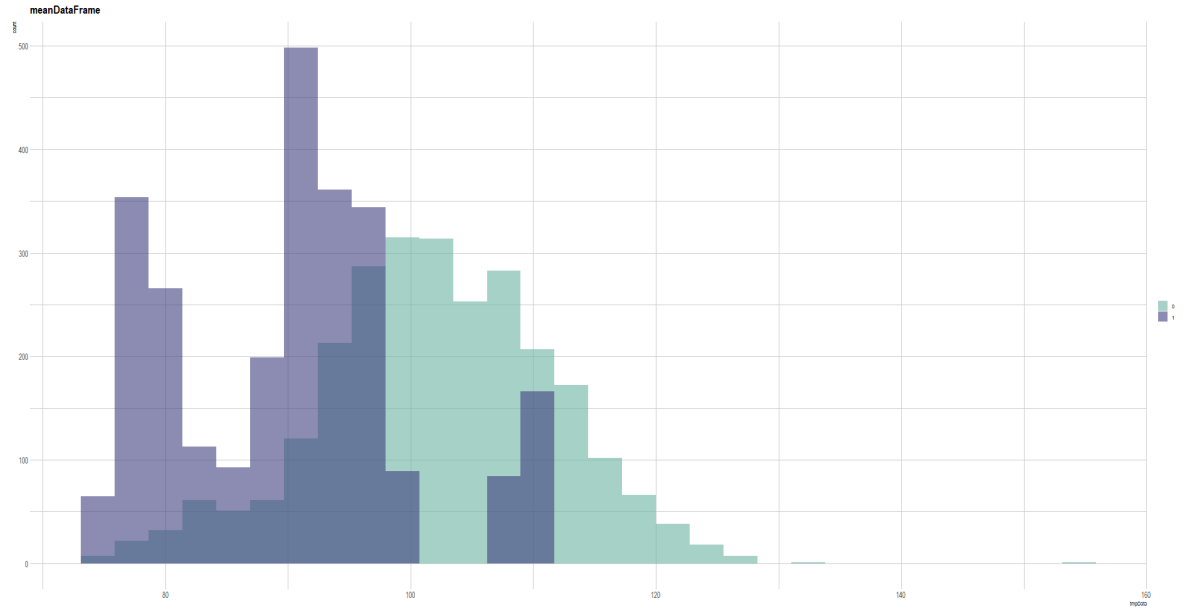
sum (합)




```
# ------------------------------------------------------------------------ #
# 데이터의 총 합 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,
                   sum,
                   "sum" )

# ------------------------------------------------------------------------ #
```

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

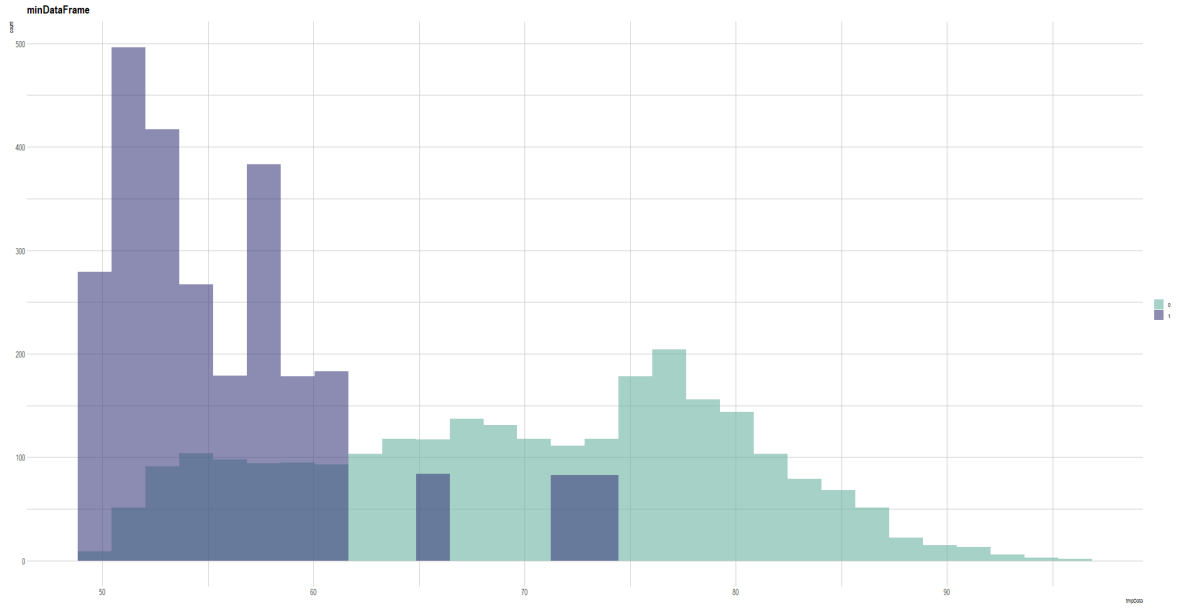
average (평균)




```
# ------------------------------------------------------------------------ #
# 데이터의 총 평균 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,
                   mean,
                   "mean" )

# ------------------------------------------------------------------------ #
```

- 평균 그래프는 sum 그래프와 모양이 똑같이 나왔습니다.

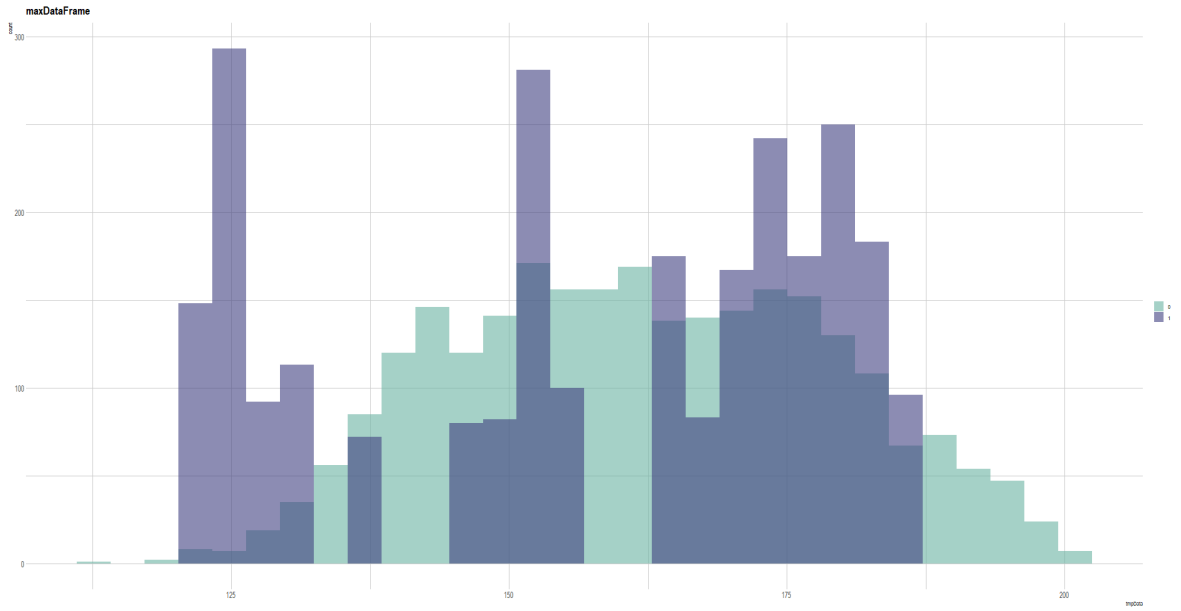
min (최소값)




```
# ------------------------------------------------------------------------ #
# 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.

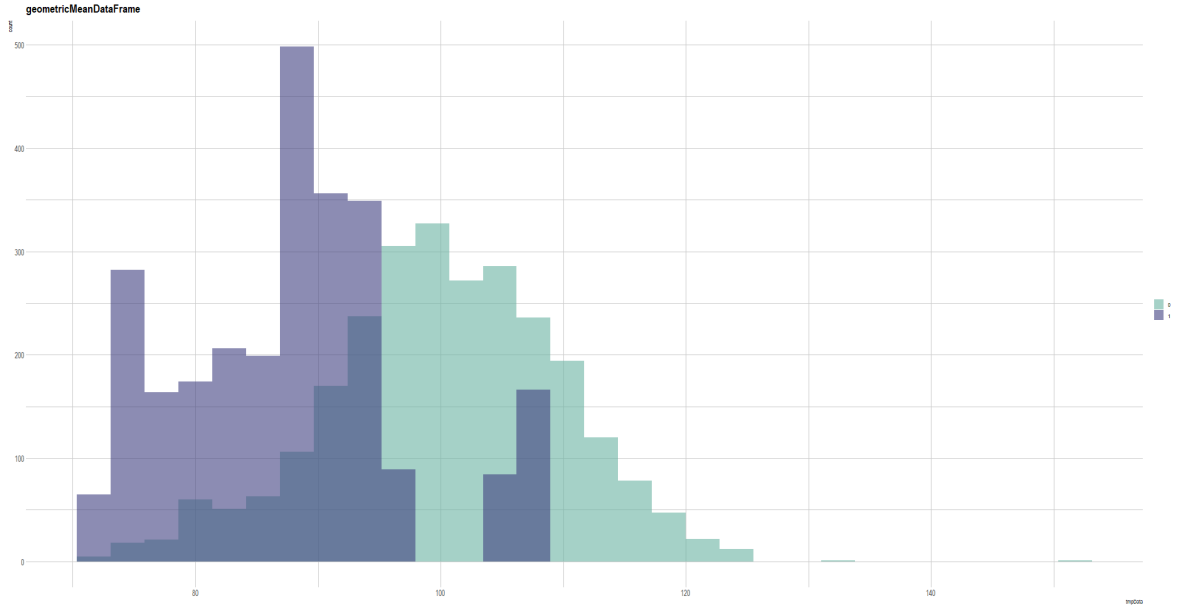
max (최대값)



```
# ------------------------------------------------------------------------ #
# 데이터의 총 최대값 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,
                   max,
                   "max" )

# ------------------------------------------------------------------------ #
```

- 최대 값은 sum, average, min보다 저혈압의 데이터가 분포해 있어 feature로 사용이 불가능해 보입니다.

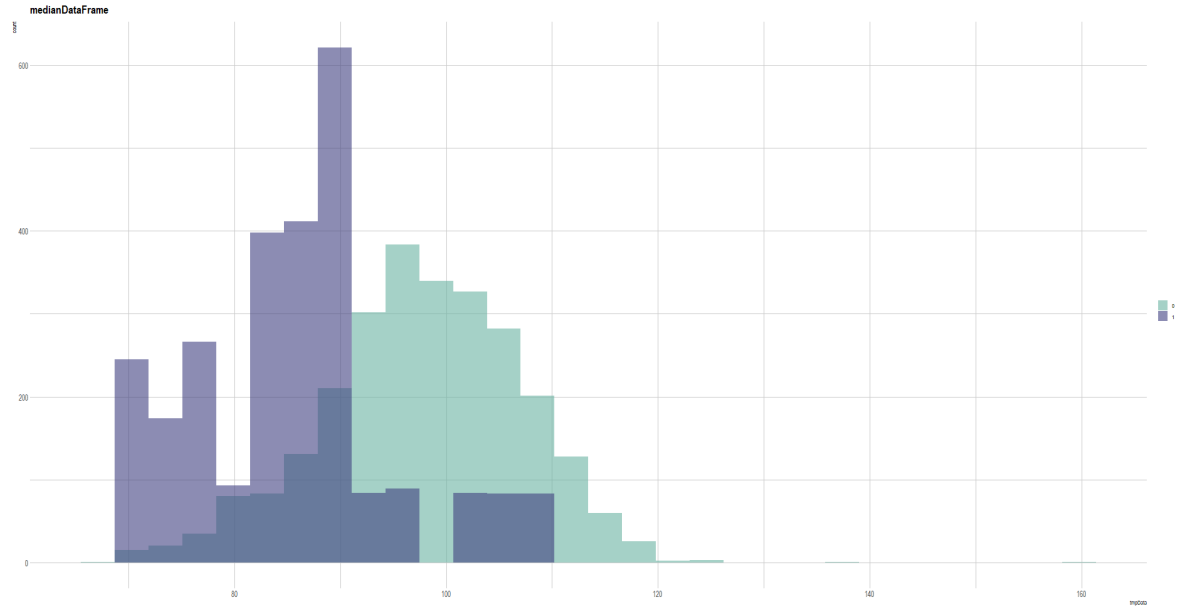
geometric mean (기하 평균)




```
# ------------------------------------------------------------------------ #
# 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.

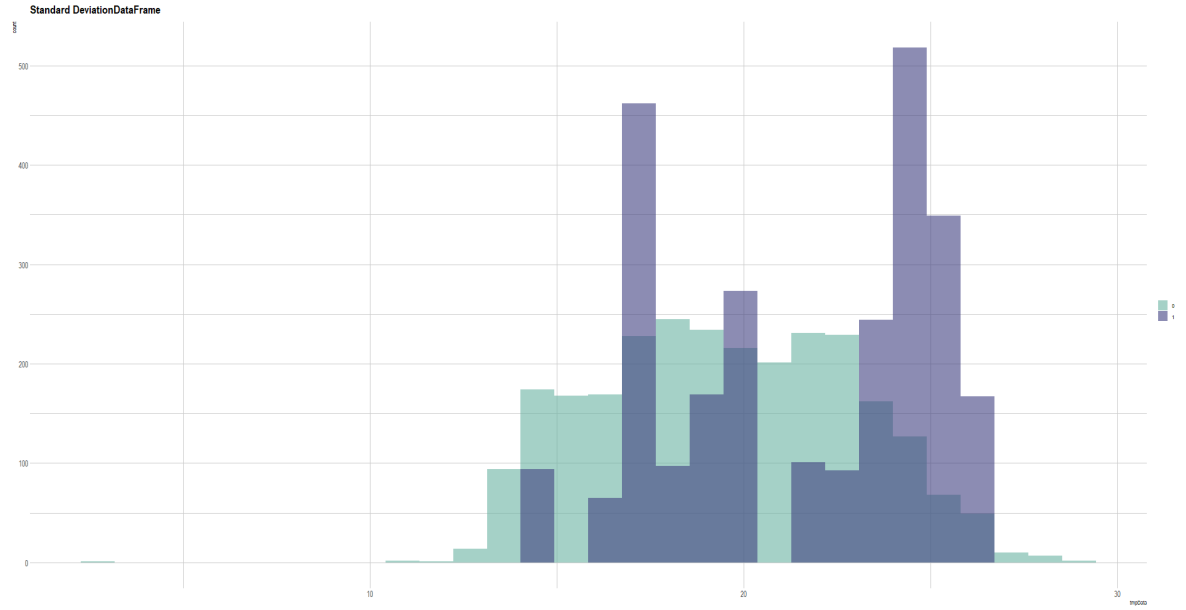
median (중위값)




```
# ------------------------------------------------------------------------ #
# 데이터의 총 중위값 시각화 비교

ArithmeticStatFuc( inputData = TrainUPData,
                   median,
                   "median" )

# ------------------------------------------------------------------------ #
```

- 중위 값의 저혈압 데이터는 정상데이터의 그래프에서 주로 50% 미만에 속하지만 예외인 데이터가 존재합니다.

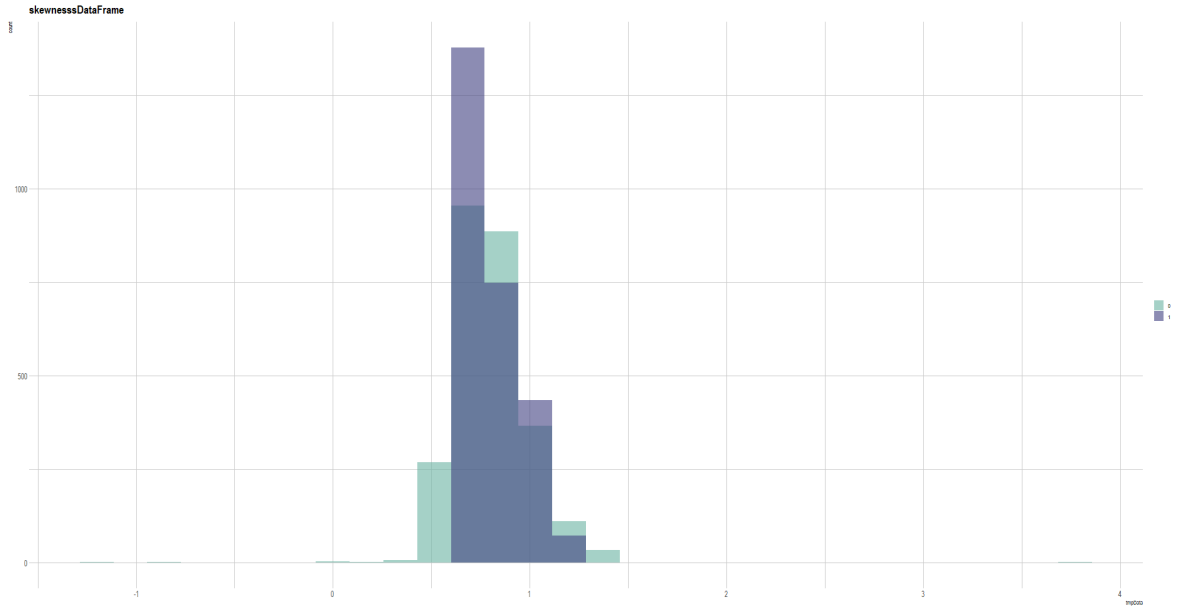
Standard Deviation (표준 편차)




```
# ------------------------------------------------------------------------ #
# 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 부적절해 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

사용 함수	데이터 분포
min	40% 이하 분포
sum, mean, geometric.mean, median	50% 이하 분포
max, sd, skewness	전체적으로 퍼짐

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

나머지 데이터는 정상 데이터의 분포에서 50%이하 부분에 분포해있어 feature로 사용이 가능할것 같으나 정상 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])

# ------------------------------------------------------------------------ #
```

```
> Folds10
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances        5252               99.772  %
Incorrectly Classified Instances        12                0.228  %
Kappa statistic                          0.9954
K&B Relative Info Score                 98.5517 %
K&B Information Score                 5187.7636 bits      0.9855 bits/instance
Class complexity | order 0            5264.0013 bits      1      bits/instance
Class complexity | scheme               78.7675 bits      0.015  bits/instance
Complexity improvement     (Sf)       5185.2338 bits      0.985  bits/instance
Mean absolute error                      0.0084
Root mean squared error                  0.0492
Relative absolute error                  1.69   %
Root relative squared error              9.8455 %
Total Number of Instances             5264

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.995    0.000    1.000      0.995    0.998      0.995    1.000     1.000     0
               1.000    0.005    0.995      1.000    0.998      0.995    1.000     1.000     1
Weighted Avg.  0.998    0.002    0.998      0.998    0.998      0.995    1.000     1.000

=== Confusion Matrix ===

    a    b   <-- classified as
 2620   12 |    a = 0
    0 2632 |    b = 1
>
```

Model Test

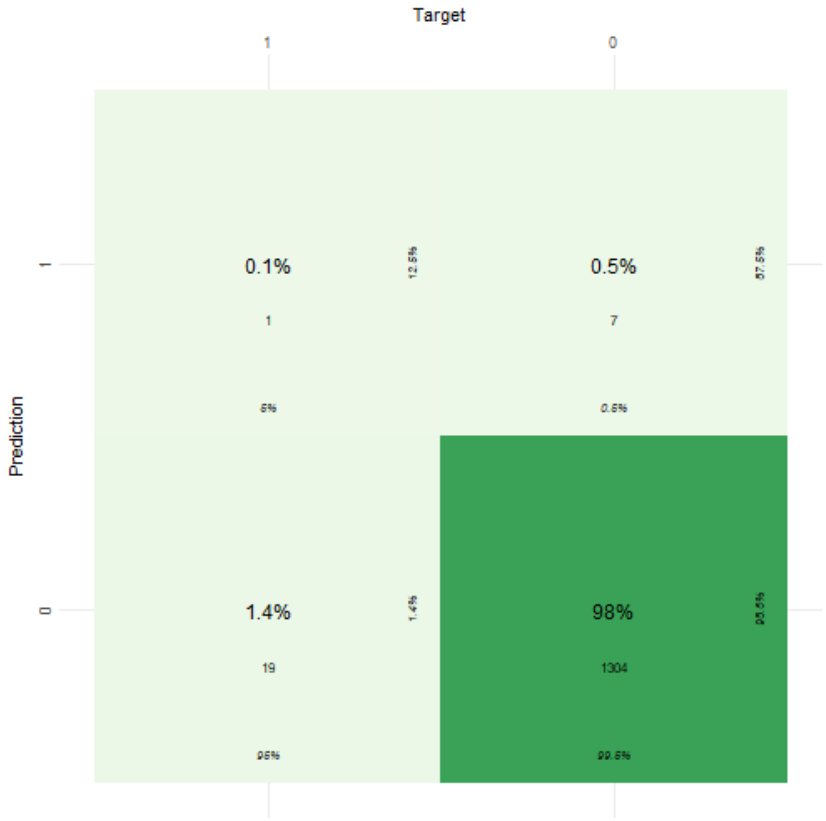
```
# ------------------------------------------------------------------------------ #

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

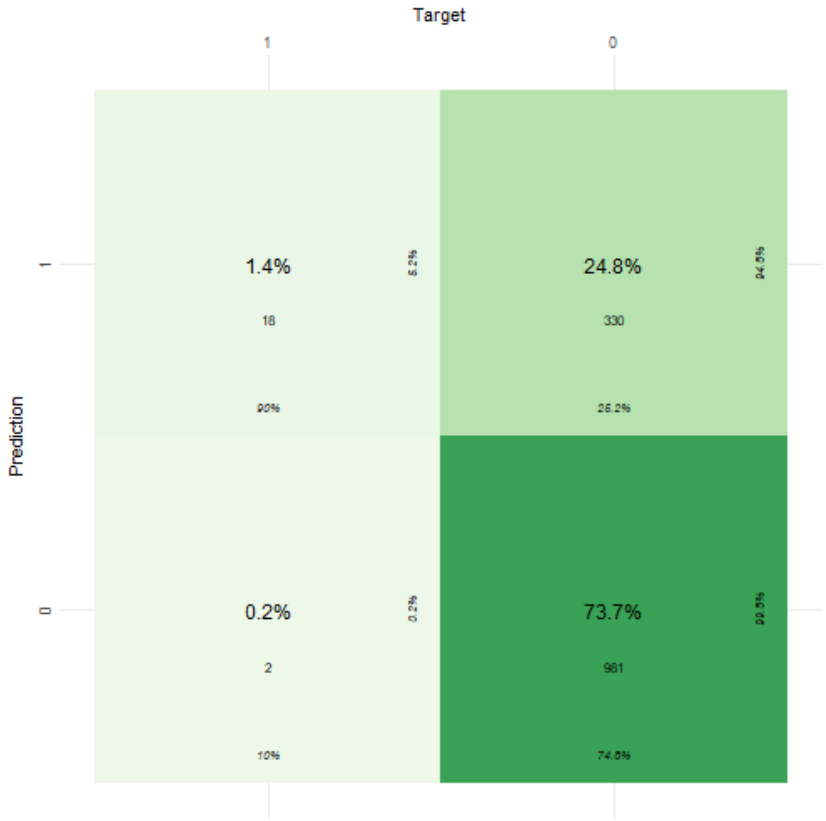
RF statistics Confusion Matrix



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

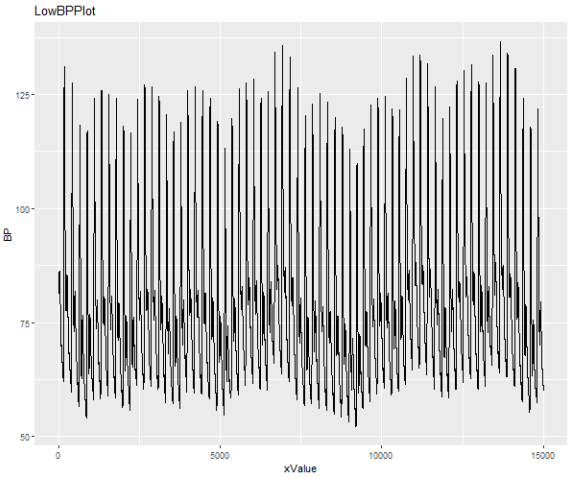
시각화를 위해서 데이터를 나눠줍니다.

```
# ------------------------------------------------------------------------ #
# 저혈압 데이터 추출
LowBPData <- TrainData %>% filter(
  event == 1
)

# 저혈압 데이터 추출
NomalBPData <- TrainData %>% filter(
  event == 0
)
# ------------------------------------------------------------------------ #
```

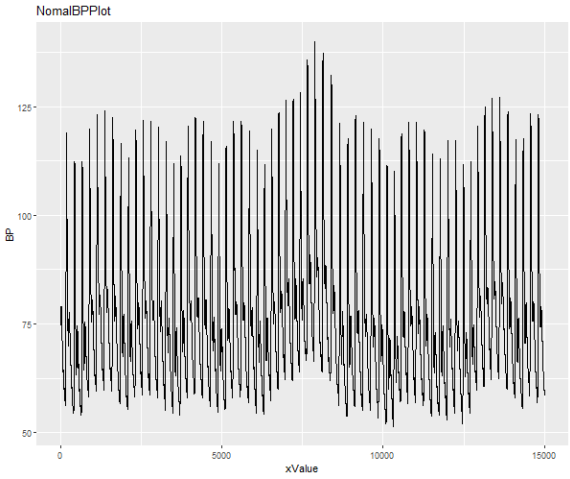
정상혈압 데이터와 저혈압 데이터 시각화 비교

```
# ------------------------------------------------------------------------ #
# 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

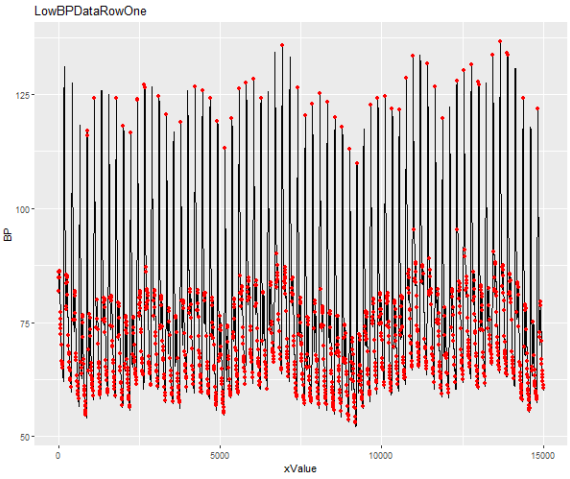
피크 시각화 함수

```
# ------------------------------------------------------------------------ #
# 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.

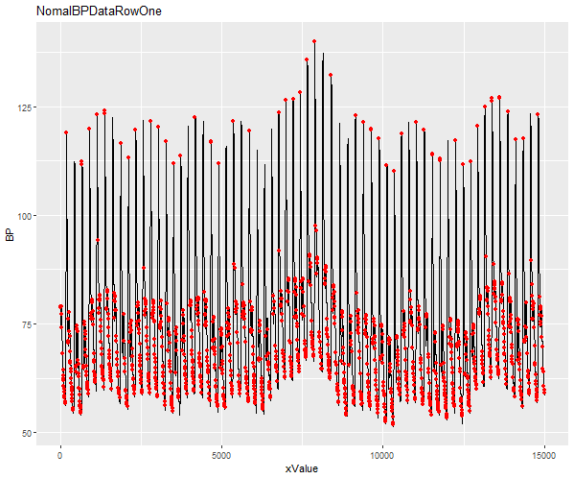
```
# ------------------------------------------------------------------------ #
# 피크 추출

# 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




```
# 피크의 수
LowPeak %>% nrow
# [1] 1392
```

Normal blood pressure




```
# 피크의 수
NomalPeak %>% nrow
# [1] 1493
```

위의 피크를 보아 혈압이 높을 때 보다 낮을 때 피크가 많이 존재하는 것으로 보인다.

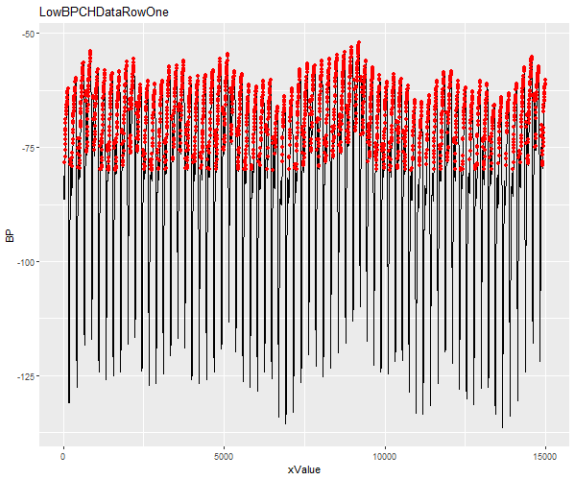
현재의 1분을 가지고 1분뒤의 혈압을 예측하는 것이니 현재 혈압이 낮으면
1분뒤의 혈압도 낮아 저혈압으로 나올것으로 추측

높은 피크보단 낮을 피클 추출한것을 feature로 학습 시켰을 경우 더 좋은 결과가 나올것으로 예측하여 데이터의 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 = "-",
```

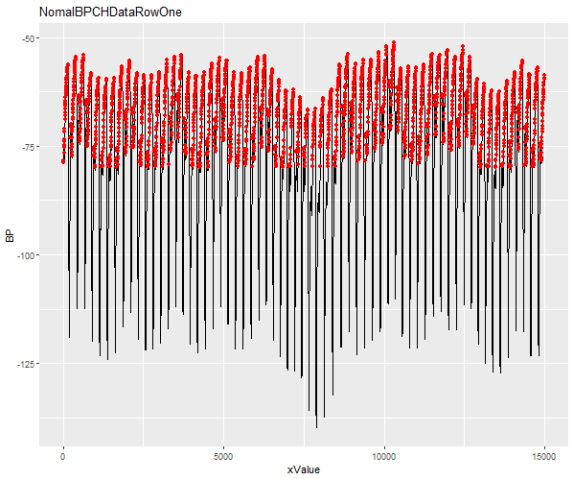
```
                                                minpeakheight = -80))
NomalBPCHDataRowOne <- data.frame(
  xValue = NomalBPRowOne$xValue,
  BP = -NomalBPRowOne$BP
)

MakePeakPlot(
  LinePlotData = NomalBPCHDataRowOne,
  PointPlotData = NomalCHPeak,
  filename = "NomalBPCHDataRowOne"
)

# ------------------------------------------------------------------------ #
```

Low blood pressure




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

Normal blood pressure




```
# 피크의 수
LowCHPeak %>% nrow
# [1] 2392
```

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

다른 피크들도 알아보기 위해 전처리를 실시하였다.

병렬처리 코드

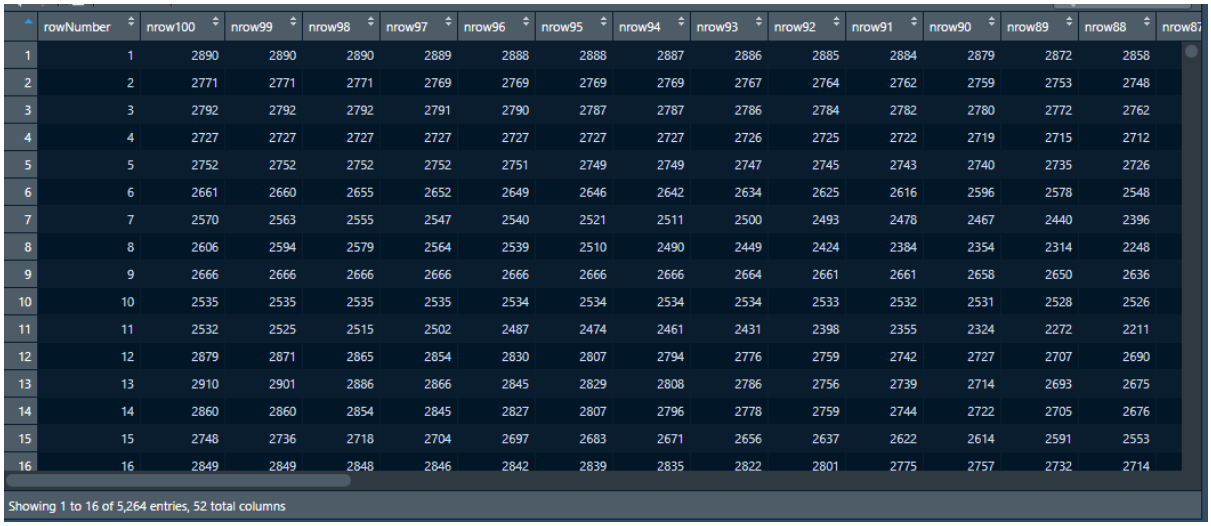
피크 전처리중 컴퓨팅 속도가 너무 느려 알아보니 병렬처리방법이 있어 사용하기로 하였다.

```
# ------------------------------------------------------------------------ #
# 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



	rowNumber	nrow100	nrow99	nrow98	nrow97	nrow96	nrow95	nrow94	nrow93	nrow92	nrow91	nrow90	nrow89	nrow88	nrow87
1	1	2890	2890	2890	2889	2888	2888	2887	2886	2885	2884	2879	2872	2858	
2	2	2771	2771	2771	2769	2769	2769	2769	2767	2764	2762	2759	2753	2748	
3	3	2792	2792	2792	2791	2790	2787	2787	2786	2784	2782	2780	2772	2762	
4	4	2727	2727	2727	2727	2727	2727	2727	2726	2725	2722	2719	2715	2712	
5	5	2752	2752	2752	2752	2751	2749	2749	2747	2745	2743	2740	2735	2726	
6	6	2661	2660	2655	2652	2649	2646	2642	2634	2625	2616	2596	2578	2548	
7	7	2570	2563	2555	2547	2540	2521	2511	2500	2493	2478	2467	2440	2396	
8	8	2606	2594	2579	2564	2539	2510	2490	2449	2424	2384	2354	2314	2248	
9	9	2666	2666	2666	2666	2666	2666	2666	2664	2661	2661	2658	2650	2636	
10	10	2535	2535	2535	2535	2534	2534	2534	2534	2533	2532	2531	2528	2526	
11	11	2532	2525	2515	2502	2487	2474	2461	2431	2398	2355	2324	2272	2211	
12	12	2879	2871	2865	2854	2830	2807	2794	2776	2759	2742	2727	2707	2690	
13	13	2910	2901	2886	2866	2845	2829	2808	2786	2756	2739	2714	2693	2675	
14	14	2860	2860	2854	2845	2827	2807	2796	2778	2759	2744	2722	2705	2676	
15	15	2748	2736	2718	2704	2697	2683	2671	2656	2637	2622	2614	2591	2553	
16	16	2849	2849	2848	2846	2842	2839	2835	2822	2801	2775	2757	2732	2714	

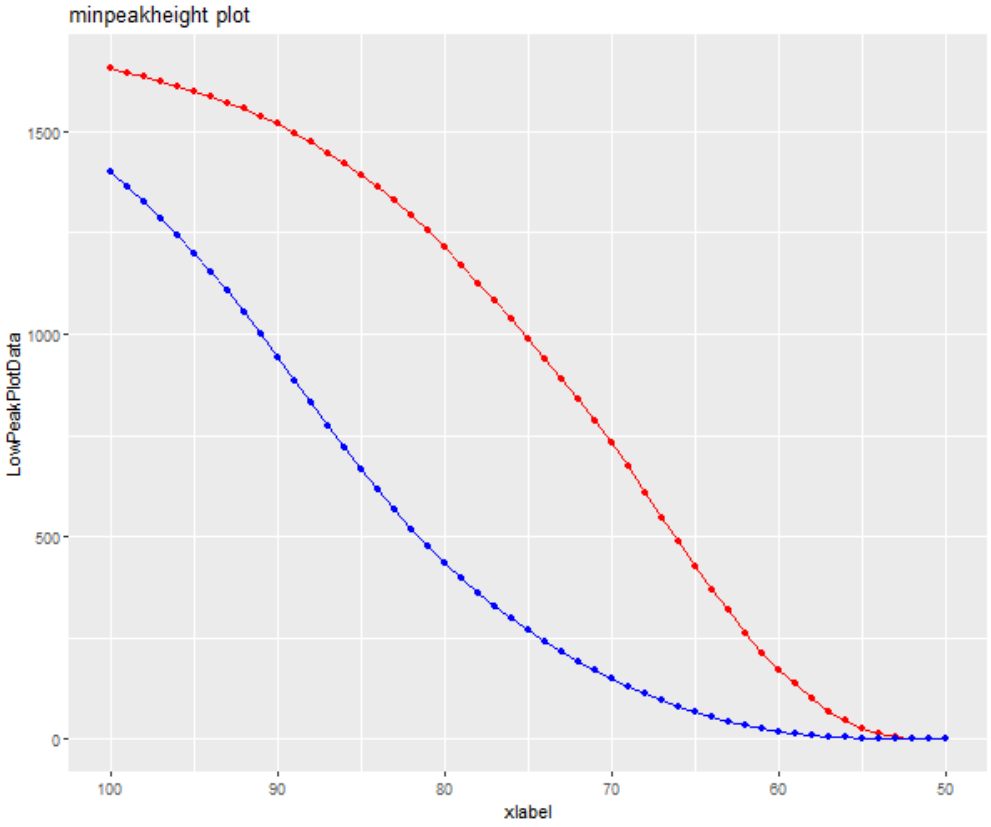
Showing 1 to 16 of 5,264 entries, 52 total columns

저혈압, 정상혈압, Peak 추출수 비교

저혈압 데이터와 정상혈압 데이터의 차이를 비교하기 위해 혈압 100이하 부터 50이하 까지 Peak의 수를 추출하여 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 했습니다.

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

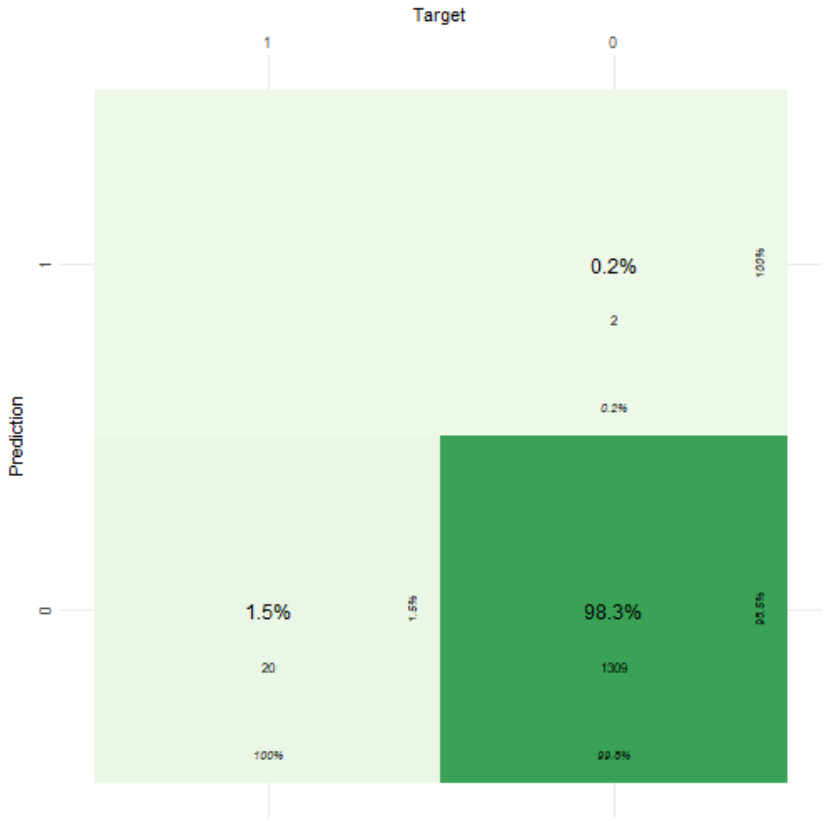
```
# ------------------------------------------------------------------------- #

# 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")
```

```
> Folds10
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances        5258               99.886  %
Incorrectly Classified Instances         6                0.114  %
Kappa statistic                          0.9977
K&B Relative Info Score                 98.6266 %
K&B Information Score                 5191.7056 bits      0.9863 bits/instance
Class complexity | order 0            5264.0013 bits      1      bits/instance
Class complexity | scheme               72.9431 bits      0.0139 bits/instance
Complexity improvement     (Sf)       5191.0582 bits      0.9861 bits/instance
Mean absolute error                      0.0083
Root mean squared error                  0.0427
Relative absolute error                  1.6641 %
Root relative squared error              8.5367 %
Total Number of Instances             5264

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.998    0.000    1.000      0.998    0.999      0.998    1.000     1.000     0
               1.000    0.002    0.998      1.000    0.999      0.998    1.000     1.000     1
Weighted Avg.  0.999    0.001    0.999      0.999    0.999      0.998    1.000     1.000

=== Confusion Matrix ===

    a    b   <-- classified as
 2626    6 |    a = 0
    0 2632 |    b = 1
```

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 저혈압은 1개도 예측하지 못했습니다.

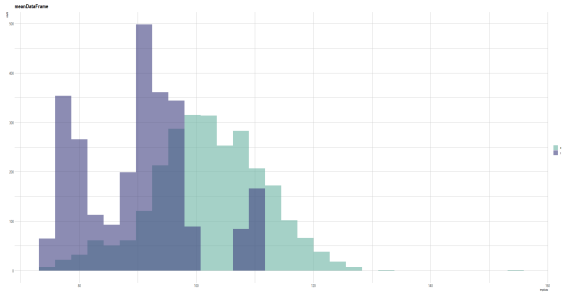
NB Peak Model Test

```
# ------------------------------------------------------------------------ #
# 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)

# ------------------------------------------------------------------------ #
```

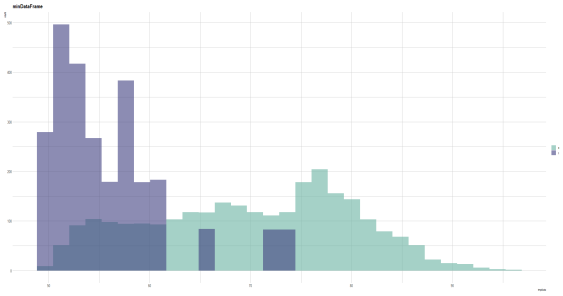
Peak feature 학습한 NB모델은 통계 모델 보다 저혈압 예측률은 떨어졌지만 정상혈압 예측률은 증가하였습니다.

Statistic & Peak 모델

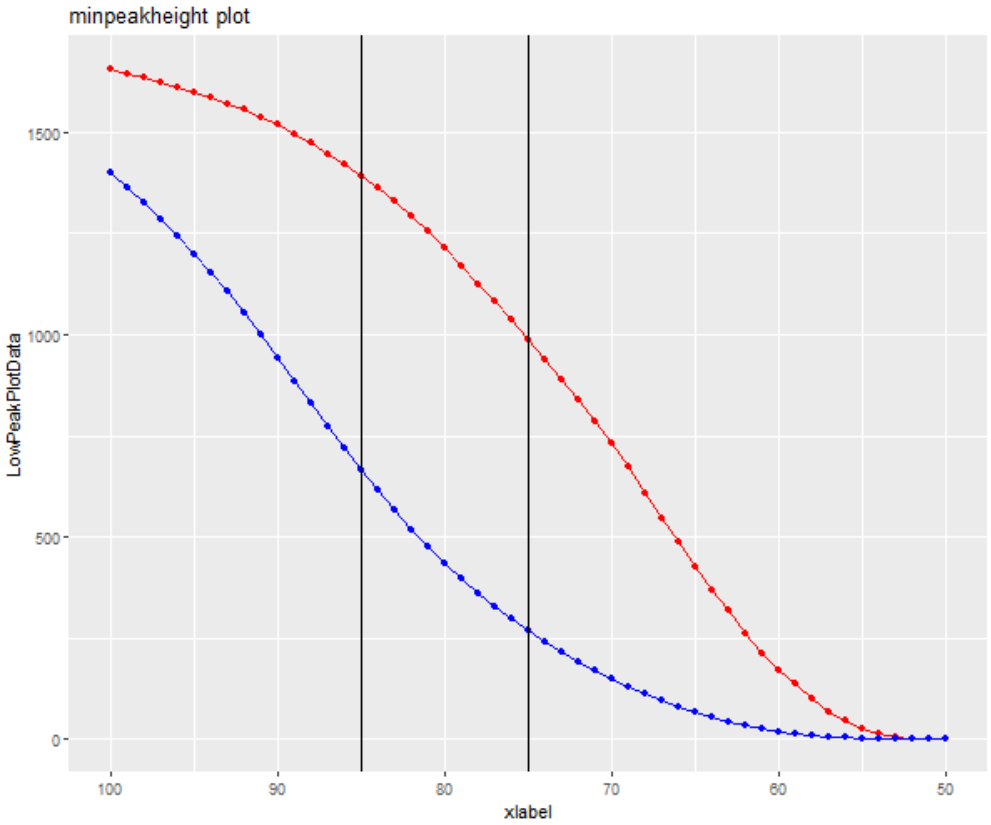
평균



최소



Peak 85이하 ~ 75이하 데이터



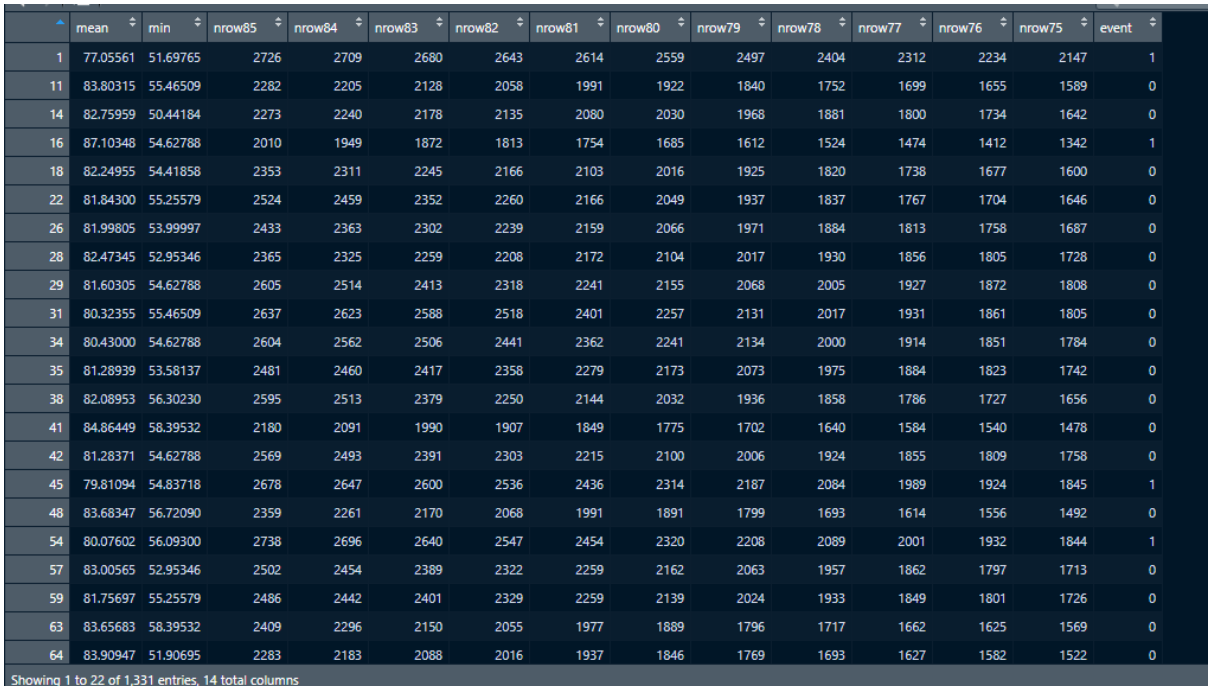
정상혈압데이터와 저혈압데이터의 차이가 많이 나는 것들만 Feature로 정해 모델을 돌려보았습니다.

```
# ------------------------------------------------------------------------ #
# 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")

# ------------------------------------------------------------------------ #
```



	mean	min	nrow85	nrow84	nrow83	nrow82	nrow81	nrow80	nrow79	nrow78	nrow77	nrow76	nrow75	event
1	77.05561	51.69765	2726	2709	2680	2643	2614	2559	2497	2404	2312	2234	2147	1
11	83.80315	55.46509	2282	2205	2128	2058	1991	1922	1840	1752	1699	1655	1589	0
14	82.75959	50.44184	2273	2240	2178	2135	2080	2030	1968	1881	1800	1734	1642	0
16	87.10348	54.62788	2010	1949	1872	1813	1754	1685	1612	1524	1474	1412	1342	1
18	82.24955	54.41858	2353	2311	2245	2166	2103	2016	1925	1820	1738	1677	1600	0
22	81.84300	55.25579	2524	2459	2352	2260	2166	2049	1937	1837	1767	1704	1646	0
26	81.99805	53.99997	2433	2363	2302	2239	2159	2066	1971	1884	1813	1758	1687	0
28	82.47345	52.95346	2365	2325	2259	2208	2172	2104	2017	1930	1856	1805	1728	0
29	81.60305	54.62788	2605	2514	2413	2318	2241	2155	2068	2005	1927	1872	1808	0
31	80.32355	55.46509	2637	2623	2588	2518	2401	2257	2131	2017	1931	1861	1805	0
34	80.43000	54.62788	2604	2562	2506	2441	2362	2241	2134	2000	1914	1851	1784	0
35	81.28939	53.58137	2481	2460	2417	2358	2279	2173	2073	1975	1884	1823	1742	0
38	82.08953	56.30230	2595	2513	2379	2250	2144	2032	1936	1858	1786	1727	1656	0
41	84.86449	58.39532	2180	2091	1990	1907	1849	1775	1702	1640	1584	1540	1478	0
42	81.28371	54.62788	2569	2493	2391	2303	2215	2100	2006	1924	1855	1809	1758	0
45	79.81094	54.83718	2678	2647	2600	2536	2436	2314	2187	2084	1989	1924	1845	1
48	83.68347	56.72090	2359	2261	2170	2068	1991	1891	1799	1693	1614	1556	1492	0
54	80.07602	56.09300	2738	2696	2640	2547	2454	2320	2208	2089	2001	1932	1844	1
57	83.00565	52.95346	2502	2454	2389	2322	2259	2162	2063	1957	1862	1797	1713	0
59	81.75697	55.25579	2486	2442	2401	2329	2259	2139	2024	1933	1849	1801	1726	0
63	83.65683	58.39532	2409	2296	2150	2055	1977	1889	1796	1717	1662	1625	1569	0
64	83.90947	51.90695	2283	2183	2088	2016	1937	1846	1769	1693	1627	1582	1522	0

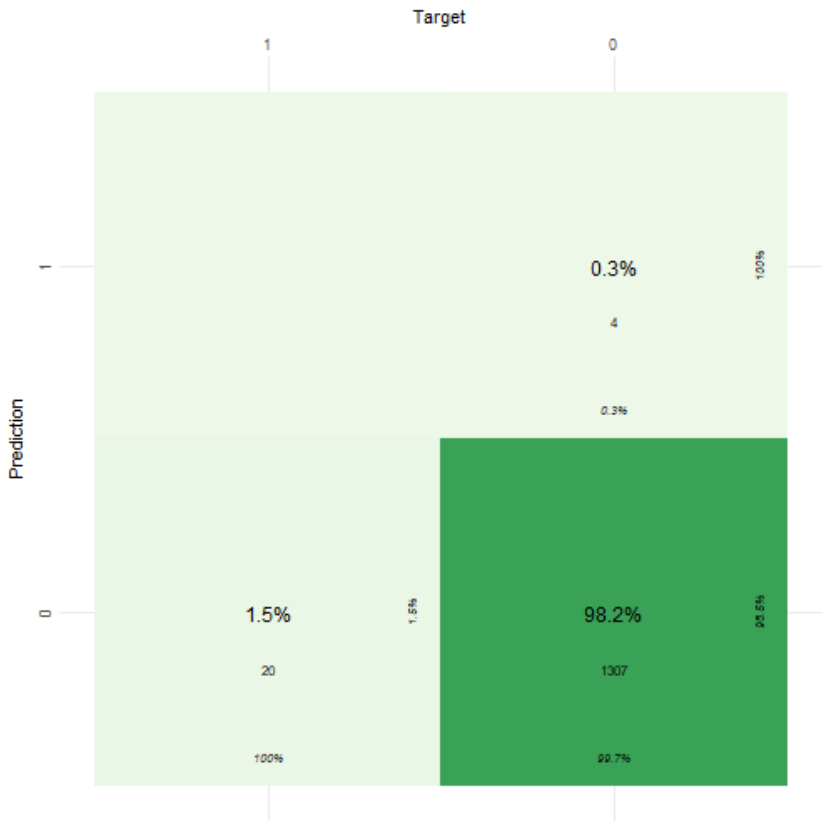
Showing 1 to 22 of 1,331 entries, 14 total columns

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 하였습니다.