데이터마이닝 기말프로젝트<K-Chip 데이터 분석>

모델 개발 및 최적화 과정 요약

본 연구에서는 기존 문헌을 기반으로 반응변수와 잠재적 설명변수를 선정하였습니다. 이를 바탕으로 본 데이터셋에서도 해당 변수 간의 관계와 적합성을 검증한 뒤, 최종적으로 모델링에 포함될 설명변수(step-wise:후진 제거법 이용)를 확정하였습니다.

또한, 양성 및 음성 데이터 간의 심각한 불균형 문제가 관찰됨에 따라, 이를 완화하고 모델 성능을 최적화하기 위해 가중치를 조정하는 방법을 채택하였습니다. 가중치 최적화를 위해 **k-fold 교차검증(k-fold cross-validation)** 방식을 활용하였으며, 가중치를 1에서 20까지 0.1 단위로 증가하여 총 200개의 모델을 생성하고 비교하였습니다. 각 모델의 성능 평가를 위해 다양한 지표를 활용하였으며, 다음의 제약 조건에 따라 최적 모델 후보군을 선정하였습니다.

최적 모델 후보군 선정 기준

1. **맥패든 R^2(McFadden's R^2)** 값이 가장 높은 모델
2. **AUC (Area Under the ROC Curve)** 값이 가장 높은 모델
3. **민감도(Sensitivity)**가 가장 높은 모델
4. **특이도(Specificity)**가 가장 높은 모델
5. **F1 Score**가 가장 높은 모델
6. 민감도가 **0.6 이상**이고 특이도가 **0.7 이상**일 때, AUC 값이 가장 높은 모델
7. 민감도가 **0.7 이상**이고 특이도가 **0.6 이상**일 때, AUC 값이 가장 높은 모델

최적 모델 선정 과정

각 후보군에 대해 모델 성능 지표(AUC, 민감도, 특이도, F1 Score, R^2)를 계산하고, 테스트 데이터셋에서의 **혼동 행렬(confusion matrix)**을 생성하여 성능을 비교하였습니다. 이를 바탕으로 연구 목적에 가장 부합하고, 실질적인 해석이 가능한 최적 모델을 최종적으로 선정하였습니다.

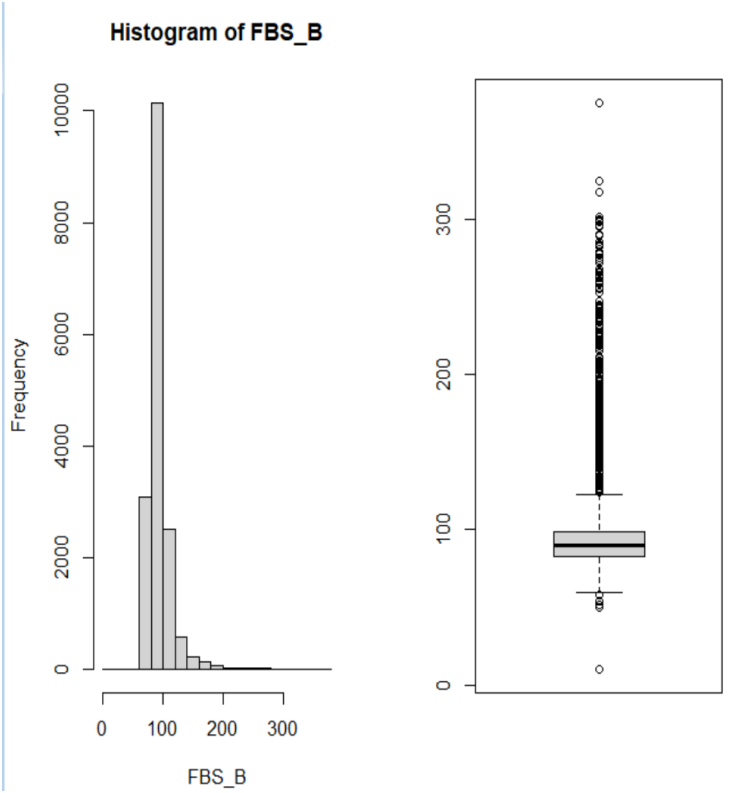
의의

이러한 접근은 데이터 불균형 문제를 효과적으로 처리하면서도 모델의 예측 정확도와 해석력을 동시에 고려하는 방식을 채택하였다는 점에서 차별화됩니다. 나아가 다양한 성능 지표를 활용하여 모델의 강점을 다각도로 평가함으로써, 현실적으로 신뢰할 수 있는 결과를 도출할 수 있었습니다.

K-Chip 원본 데이터 분석 전처리 및 반응변수, 설명 변수 선정

반응변수 선정 및 분석

대한당뇨병학회의 기준에 따르면 당뇨병의 진단에서 공복 혈당(Fasting Blood Sugar, FBS) 수치가 **126 mg/dL 이상**인 경우 당뇨병으로 진단합니다. K-Chip 데이터에서는 공복 혈당 수치를 FBS_B 컬럼을 통해 확인할 수 있습니다.



FBS_B 변수의 분포를 분석한 결과, 이는 정규분포 형태가 아니며, 다수의 이상치를 포함하고 있음을 확인하였습니다. 따라서, 반응 변수를 **로그 변환**하거나 이상치를 진단 후 제거하는 방안을 고려해야 했습니다. 그러나 이상치를 제거할 경우 데이터 손실이 클 것으로 판단하였습니다. 이에 따라 FBS_B 변수를 기준으로 **126 mg/dL 이상이면 "YES"**, 미만이면 "NO"로 범주형 변수를 생성하였습니다. 이러한 기준에 따라 로지스틱 회귀분석(Logistic Regression)을 진행하고자 합니다.

```
당뇨병 발병 YES 수 :  919
> cat("당뇨병 발병 NO 수 : ",당뇨병발병NO수,"\n")
당뇨병 발병 NO 수 :  15996
> cat("당뇨병 발병 비율 : ",당뇨병발병YES수/(당뇨병발병NO수+당뇨병발병YES수),"\n") #데이터간 불균형이 있음을 확인가능
당뇨병 발병 비율 :  0.05433048
> data$당뇨병유발 <- ifelse(data$FBS_B>=126, "YES",
+                       ifelse(data$FBS_B<126 , "NO", NA))
```

데이터 불균형:

질병관리청 국가건강정보포털에 따르면, 대한민국 내 당뇨병 유병률은 약 **13.8%**로 보고됩니다. 그러나 K-Chip 데이터에서는 당뇨병 "YES"의 비율이 **5%**로 확인되었으며, 이는 데이터 간 불균형이 존재함을 나타냅니다. 이러한 불균형은 모델 학습 시 **클래스 불균형 문제**를 초래할 가능성이 높습니다. 이를 해결하기 위해 당뇨병 "YES" 클래스에 **가중치(weight)**를 부여하는 방안을 채택하였습니다.

가중치 부여의 구체적인 산정 방식과 그 타당성은 모델 구축 과정에서 상세히 설명하겠습니다. 이와 같은 접근은 클래스 불균형 문제를 완화하고, 모델의 예측 성능과 신뢰도를 향상시키는 데 목적이 있습니다.

설명 변수 후보군 선정 :

우리는 당뇨병 예측 모델 구축을 위해 설명 변수의 후보군을 선정하고, 과제 명세에 따라 근거에 맞는 설명변수를 선택 하는 과정이 있었습니다. 설명 변수 후보군은 관련 문헌 및 데이터의 유의미성을 바탕으로 도출하였으며, 특히 한국인을 대상으로 한 당뇨병 위험 예측 모델에 관한 연구인 「**한국인에서 당뇨병 위험 예측 모델**」(연세대학교 의과대학 내분비내과 이용호 교수, 아주대학교 의과대학 내분비대사내과 김대중 교수, https://www.diabetes.or.kr/general/info/info_01.php?con=5)를 근거로 삼았습니다.

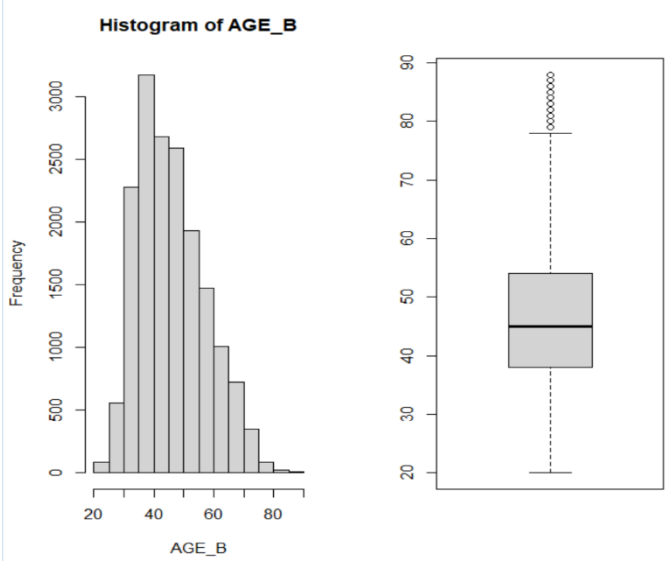
위 논문에서 제시된 당뇨병 위험 요인을 참고하여, K-Chip 데이터에서 활용 가능한 변수로 다음의 후보군을 선정하였습니다:

- **나이**
- **BMI(체질량지수)**
- **흡연 여부**
- **고혈압**
- **저밀도 콜레스테롤(LDL-C)**
- **중성지방(TG)**

이후, 후보 변수들이 K-Chip 데이터에서 당뇨병 예측에 있어 통계적으로 유의미한지 분석을 수행하였고, 각 변수의 중요도를 평가하였습니다. 평가 결과를 바탕으로, 모델에서 가장 유의미하고 기여도가 높은 변수를 최종적으로 설명 변수로 채택할 계획입니다.

이 과정은 단순히 변수의 선택에 그치지 않고, 변수 간 상관관계와 다중공선성을 고려하여 모델의 안정성을 높이는 방향으로 진행되었습니다. 또한, 이 접근법은 데이터 기반의 정량적 분석과 연구 기반의 정성적 근거를 통합하여 모델의 신뢰도를 극대화하는 것을 목표로 합니다.

설명 변수 후보군 선정 (나이)

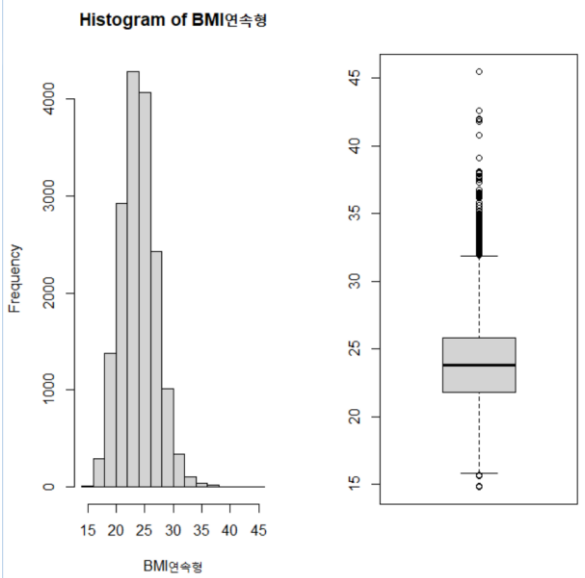



```
> par(mfrow=c(1,2))
> hist(AGE_B)
> boxplot(AGE_B)
> par(mfrow=c(1,2))
> hist(AGE_B)
> boxplot(AGE_B)
> cor(FBS_B,AGE_B,use = "complete.obs",method="spearman") #공복혈당과 나이의 hist가 정규분포하고 하$
[1] 0.3015539
> data$나이 <- AGE_B
>
```

Spearman 상관분석 결과, 나이와 공복 혈당(FBS_B) 간의 상관계수는 **0.3015539**로 나타났습니다. 이는 두 변수 간에 어느 정도의 **양의 상관관계**가 있음을 의미합니다. 즉, **나이가 증가할수록 공복 혈당 수치가 증가하는 경향**을 확인할 수 있습니다.

현재 모델 구축이 완료되지 않아 나이 변수가 로지스틱 회귀 모델에서 통계적으로 유의미한지 판단할 수 있는 **p값**은 확인되지 않았습니다. 그러나 나이 변수는 공복 혈당과의 상관성이 관찰되었기 때문에 **로지스틱 회귀 모델의 설명 변수 후보군**으로 선정하였습니다. 모델 구축 후, 나이 변수의 p값과 기여도를 검토하여 최종적으로 모델에서의 유효성을 평가할 계획입니다.

설명 변수 후보군 선정 (BMI)




```
> head(data$BMI연속형)
[1] 20.01503 23.04688 22.97578 25.27347 19.58821 26.49007
> sum(is.na(data$BMI연속형))
[1] 35
> str(data$BMI연속형)
 num [1:16955] 20 23 23 25.3 19.6 ...
> par(mfrow=c(1,2))
> hist(BMI연속형)
> cor(FBS_B,data$BMI연속형,use = "complete.obs",method="spearman")
[1] 0.2759934
>
> max(BMI연속형,na.rm=TRUE)
[1] 45.52294
> min(BMI연속형,na.rm=TRUE)
[1] 14.79184
> sum(BMI연속형<18.5,na.rm=TRUE)
[1] 482
> sum(BMI연속형>=18.5 & BMI연속형<=22.9,na.rm=TRUE)
[1] 5945
> sum(BMI연속형>22.9 & BMI연속형<=24.9,na.rm=TRUE)
[1] 4409
> sum(BMI연속형>24.9 & BMI연속형<=29.9,na.rm=TRUE)
[1] 5557
> sum(BMI연속형>29.9 & BMI연속형<=34.9,na.rm=TRUE )
[1] 485
> sum(BMI연속형>34.9,na.rm=TRUE)
[1] 42
>
> data$BMI구간 <- ifelse(data$BMI연속형<18.5 , "18.5미만",
+                        ifelse(BMI연속형>=18.5 & BMI연속형<=22.9, "18.5이상22.9이하",
+                               ifelse(BMI연속형>22.9 & BMI연속형<=24.9, "23.0이상24.9이하",
+                                      ifelse(BMI연속형>24.9 & BMI연속형<=29.9, "25.0이상29.9이하",
+                                             ifelse(BMI연속형>29.9, "30이상", NA)))))
> boxplot(BMI연속형)
>
```

K-Chip 데이터에는 **BMI** 항목이 존재하지 않으나, **신장(HT_B)**과 **체중(WT_B)** 컬럼을 활용하여 BMI를 계산하였습니다. BMI 계산법은 다음과 같습니다.

data$BMI연속형 <- WT_B / (HT_B * HT_B) * 10000

BMI는 체중(kg)을 신장(m)의 제곱으로 나눈 값입니다. 그러나 K-Chip 데이터에서 신장은 **cm 단위**로 제공되므로, 이를 **m 단위로 변환하기 위해 10,000을 곱하여 보정**하였습니다.

분석 결과, **BMI가 증가할수록 공복 혈당(FBS_B)이 증가하는 경향**이 관찰되었습니다. 그러나 BMI 변수에서 이상값이 다수 존재하였기에, 이상값으로 인한 정보 손실을 방지하고자 **BMI를 구간화하여 범주형 변수**로 변환하였습니다. 이 구간화의 기준은 **서울아산병원**에서 제공한 BMI 구간을 참고하였으며, 총 5개의 구간으로 나누었습니다.

설명 변수 후보군 선정 (흡연)

```
> sum(is.na(data$SMOK_B))
[1] 177
> sum(data$SMOK_B==1,na.rm=TRUE)
[1] 7948
> sum(data$SMOK_B==2,na.rm=TRUE)
[1] 3352
> sum(data$SMOK_B==3,na.rm=TRUE)
[1] 5478
> sum((data$SMOK_B==1|data$SMOK_B==2) & data$[illegible]=="YES",na.rm=TRUE)/(sum((data$SMOK_B==1|data$
[1] 0.0502352
> sum(data$SMOK_B==3 & data$[illegible]=="YES",na.rm=TRUE)/(sum(data$SMOK_B==3 & data$[illegible]=="YE$
[1] 0.06231725
> data$[illegible] <- ifelse(data$SMOK_B == 1 | data$SMOK_B == 2, "[illegible]",
+                     ifelse(data$SMOK_B == 3, "[illegible]", NA))
>
```

K-Chip 데이터에서 **흡연 여부**를 나타내는 컬럼의 자료형은 int**형**으로 설정되어 있습니다. 이 데이터는 값이 1과 2로 구분되어 있으며, R에서는 이를 연속형 변수로 처리할 가능성이 있습니다. 그러나 해당 변수는 범주형 변수로 사용되어야 하므로, **비흡연자**와 **흡연자**를 명확히 구분하는 새로운 범주형 변수로 조정하였습니다.

이를 위해 원래 컬럼의 값을 기반으로 1은 "비흡연자", 2는 "흡연자"로 변환한 뒤, 로지스틱 회귀 분석에서 이 변수를 **범주형 변수로 인식**할 수 있도록 처리하였습니다. 이러한 변환은 변수의 본질을 반영하면서 모델링의 정확성과 해석 가능성을 높이는 데 기여합니다.

이와 같은 변수 조정은 데이터의 특성을 고려한 사전 처리로, 모델 구축 과정에서 중요한 데이터 전처리 단계 중 하나로 볼 수 있습니다.

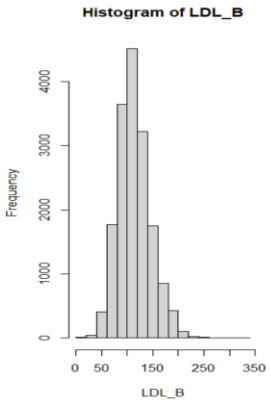
설명 변수 후보군 선정 (고혈압)

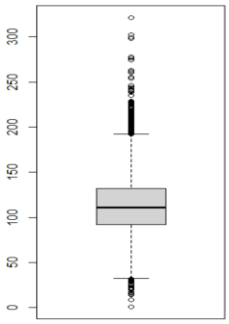
```
> data$고혈압여부 <- ifelse((data$SBP_B>=140 | data$DBP_B>=90),"고혈압자","비고혈압자")
> sum((data$고혈압여부=="고혈압자" & data$당뇨병유발=="YES"),na.rm=TRUE)/(sum((data$고혈압여부=="고$
[1] 0.09704473
> sum((data$고혈압여부=="비고혈압자" & data$당뇨병유발=="YES"),na.rm=TRUE)/(sum((data$고혈압여부=="$
[1] 0.2773109
>
```

고혈압은 수축기 혈압과 이완기 혈압 (SBP_B와 DBP_B)에 대한 컬럼을 이용하였습니다.

고혈압에 대한 컬럼은 존재하지 않기에 수축기 혈압이 140이상 혹은 이완기 혈압이 90 이상이라면 고혈압 컬럼에 고혈압자 그렇지 않다면 비고혈압자라는 컬럼 값을 넣었습니다.

설명 변수 후보군 선정 (저밀도콜레스테롤)

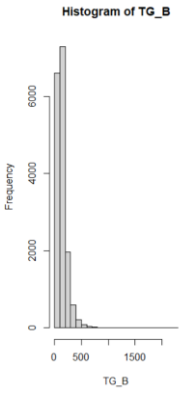


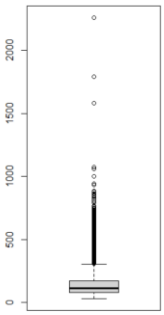



```
> sum(is.na(data$LDL_B))
[1] 165
> par(mfrow=c(1,2))
> hist(LDL_B)
> boxplot(LDL_B)
> cor(FBS_B,LDL_B,use = "complete.obs",method="spearman")
[1] 0.1006103
> cor(FBS_B,LDL_B,use = "complete.obs",method="pearson")
[1] 0.06938123
> data$저밀도콜레스테롤<-data$LDL_B
```

저밀도콜레스테롤도 어느정도 공복혈당과 양의 선형관계가 있다고 보이기는 합니다. 그렇기에 일단 후보군으로 채택한 후 타 변수와의 중요도를 비교하고자 합니다.

설명 변수 후보군 선정 (중성지방)






```
> par(mfrow=c(1,2))
> hist(TG_B)
> boxplot(TG_B)
> par(mfrow=c(1,2))
> hist(TG_B)
> boxplot(TG_B)
> cor(FBS_B,TG_B,use = "complete.obs",method="spearman")
[1] 0.2496177
> cor(FBS_B,TG_B,use = "complete.obs",method="pearson")
[1] 0.2192338
> data$중성지방 <- data$TG_B
```

중성지방도 어느정도 공복혈당과 양의 선형관계가 있다고 보이기는 합니다. 그렇기에 일단 후보군으로 채택한 후 타 변수와의 중요도를 비교하고자 합니다.

우리는 총 6개의 설명변수 후보군을 채택하였고 반응변수와 설명변수에 대한 전처리를

진행하였습니다. 이를 하나의 데이터프레임으로 만들고 1차적인 모델을 만들어서 변수에 대한 중요도를 판단하고 변수가 모델에 유의한지 확인하려고 합니다.

```
mydata <- data[, c("당뇨병유발", "나이", "BMI구간","흡연여부","고혈압여부","저밀도콜레스테롤")]
head(mydata)
colSums(is.na(mydata))

mydata <-  na.omit(mydata)
colSums(is.na(mydata))
head(mydata)

write.csv(mydata,"C:\\Users\\user\\Desktop\\학업\\2024-2\\데이터마이닝\\기말프로젝트\\mydata후보군.csv",row.nan


#변수선택 중요도 확인
mydata$당뇨병유발 <- factor(mydata$당뇨병유발, levels = c("NO", "YES"))
mydata$BMI구간 <- factor(mydata$BMI구간, levels=c("18.5미만","18.5이상22.9이하","23.0이상24.9이하","25.0이상29
mydata$흡연여부 <-  factor(mydata$흡연여부, levels = c("비흡연자", "흡연자"))
mydata$고혈압여부 <-  factor(mydata$고혈압여부, levels = c("비고혈압자", "고혈압자"))
```

보다 더 정확한 모델을 위하여 결측값에 대한 정보는 평균값 대치 같은 방안을 삭제하지 않았고 결측값은 제외하였습니다. 총 16955행에서 300행의 데이터 손실이 발생 하였으나 큰 데이터 손실은 발생하지 않아 정확한 모델링을 위해 이 정도의 데이터 손실은 감수하였습니다.

```
# 전체에서 제거하는법을 통해 모델 학습
logit_model_full <- glm(당뇨병유발 ~ ., data = mydata, family = binomial())

# 단계적 변수 선택 (Backward Elimination)
logit_model_stepwise <- step(logit_model_full, direction = "backward")

# 최종 모델 확인
summary(logit_model_stepwise)
```

```
Start:  AIC=6119.29
당뇨병유발 ~ 나이 + BMI구간 + 흡연여부 + 고혈압여부 +
    저밀도콜레스테롤

                  Df Deviance    AIC
- 저밀도콜레스테롤  1   6103.0 6119.0
<none>                 6101.3 6119.3
- 고혈압여부        1   6118.6 6134.6
- 흡연여부          1   6145.4 6161.4
- BMI구간           4   6201.7 6211.7
- 나이              1   6447.4 6463.4

Step:  AIC=6118.98
당뇨병유발 ~ 나이 + BMI구간 + 흡연여부 + 고혈압여부

             Df Deviance    AIC
<none>            6103.0 6119.0
- 고혈압여부  1   6120.1 6134.1
- 흡연여부    1   6146.9 6160.9
- BMI구간     4   6201.9 6209.9
- 나이        1   6448.0 6462.0
```

설명변수에 중요도가 나이 BMI구간 흡연여부 고혈압여부 중성지방 저밀도콜레스테롤 순으로 중요합니다.

로지스틱 회귀 분석(Logistic Regression) 모델을 구축하는 과정에서 **Backward Elimination**(후진 소거법)을 사용하여 변수를 선택하는 절차를 사용하였습니다.

후진 소거법은 주어진 코드는 **로지스틱 회귀 분석(Logistic Regression)** 모델을 구축하는 과정에서 **Backward Elimination**(후진 소거법)을 사용하여 변수를 선택하는 절차를 보여줍니다.

이를 분석하면 다음과 같은 내용이 도출됩니다.이를 분석하면 다음과 같은 내용이 도출됩니다.

간결성의 원칙에 따라 근거에 맞는 한 설명변수에 개수를 줄이더라도 간결한 모델을 선택하고자 하였습니다.

```
Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -7.2389789  0.4173632 -17.345  < 2e-16 ***
나이                     0.0614762  0.0032844  18.718  < 2e-16 ***
BMI구간18.5이상22.9이하  0.1445529  0.3705768   0.390  0.69648
BMI구간23.0이상24.9이하  0.5987934  0.3682906   1.626  0.10398
BMI구간25.0이상29.9이하  0.6485056  0.3673500   1.765  0.07750 .
BMI구간30이상            1.2883055  0.3921658   3.285  0.00102 **
흡연여부흡연자           0.3522824  0.0784081   4.493 7.02e-06 ***
고혈압여부고혈압자       0.2578847  0.0856722   3.010  0.00261 **
저밀도콜레스테롤        -0.0009223  0.0010746  -0.858  0.39076
중성지방                 0.0040491  0.0002872  14.098  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6671.2  on 16416  degrees of freedom
Residual deviance: 5917.2  on 16407  degrees of freedom
AIC: 5937.2

Number of Fisher Scoring iterations: 6
```

저밀도콜레스테롤의 설명변수에 중요도는 중요도 또한 제일 낮고 p값을 보면 유의하지 않음을 알 수 있습니다. 그래서 저밀도콜레스테롤에 대한 설명변수는 제외 하였습니다.

앞서 후진제거법을 통해 우리는 BMI구간이 중요한 역할을 함을 알 수 있지만 P값이 만족하지 않아 구간화를 다시 진행하였습니다.

```
glm(formula = 당뇨병유발 ~ 나이 + BMI구간 + 흡연여부 +
    고혈압여부 + 중성지방, family = binomial(), data = mydata)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -7.205862   0.203402 -35.427  < 2e-16 ***
나이                      0.061471   0.003285  18.714  < 2e-16 ***
BMI구간23.0이상24.9이하  0.457565   0.104884   4.363 1.29e-05 ***
BMI구간25.0이상29.9이하  0.504464   0.099863   5.052 4.38e-07 ***
BMI구간30이상             1.142256   0.169932   6.722 1.79e-11 ***
흡연여부흡연자            0.351932   0.078387   4.490 7.13e-06 ***
고혈압여부고혈압자        0.256480   0.085625   2.995  0.00274 **
중성지방                  0.004060   0.000287  14.148  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6671.2  on 16416  degrees of freedom
Residual deviance: 5918.1  on 16409  degrees of freedom
AIC: 5934.1

Number of Fisher Scoring iterations: 6
```

그 결과 모든 변수가 유의함을 알 수 있었습니다. 이에 저희가 이제 제대로 예측 모델을 만들기 위한 최종 데이터 프레임은 다음과 같습니다.

반응변수 : 당뇨병여부 ("YES" or "NO") 범주형 변수

설명변수 : 나이(연속형 변수)

BMI(범주형 변수)

구간은 다음과 같습니다. "22.9이하","23.0이상24.9이하","25.0이상29.9이하","30이상")

흡연여부 (범주형 변수) – 비흡연자, 흡연자

고혈압여부 (범주형 변수) – 비고혈압자, 고혈압자

중성지방 (연속형 변수)

최종 데이터 프레임 및 모델 구축:

우리 팀은 최종 데이터 프레임을 당뇨병유발 컬럼을 반응 변수로, 나이와 BMI, 흡연여부, 고혈압여부, 중성지방컬럼을 설명 변수로 설정하여 구성하였습니다. 이를 기반으로 당뇨병 예측 모델을 구축하였으며, 본 팀의 조원 이건은 로지스틱 회귀 모델을 설계하였습니다.

최종 데이터 프레임

당뇨병유발	나이	BMI구간	흡연여부	고혈압여부	중성지방
NO	39	22.9이하	비흡연자	비고혈압자	38
NO	44	23.0이상2	비흡연자	비고혈압자	77
NO	46	23.0이상2	비흡연자	비고혈압자	64
NO	43	25.0이상2	비흡연자	비고혈압자	142
NO	50	25.0이상2	비흡연자	비고혈압자	71
NO	42	23.0이상2	흡연자	비고혈압자	275
NO	48	25.0이상2	흡연자	비고혈압자	265
NO	44	22.9이하	비흡연자	비고혈압자	191
NO	43	22.9이하	흡연자	비고혈압자	111
NO	56	25.0이상2	비흡연자	비고혈압자	76
NO	46	23.0이상2	비흡연자	비고혈압자	77
NO	71	22.9이하	비흡연자	비고혈압자	167
NO	49	23.0이상2	흡연자	비고혈압자	117
NO	72	22.9이하	흡연자	고혈압자	136
NO	65	23.0이상2	비흡연자	비고혈압자	62
NO	52	23.0이상2	비흡연자	비고혈압자	55
YES	46	30이상	흡연자	비고혈압자	247
NO	45	30이상	비흡연자	비고혈압자	421
NO	58	25.0이상2	흡연자	비고혈압자	356

데이터 불균형 문제:

원본 데이터 분석 결과, 당뇨병 양성 레이블(Positive Label)이 전체 데이터의 **5%**에 불과하여, 양성과 음성 간 데이터 불균형이 발생하였습니다. 이러한 불균형은 모델 학습 과정에서 편향된 학습을 초래할 가능성이 큽니다. 이를 해결하기 위해 양성 레이블에 가중치를 부여하는 방식을 채택하였습니다.

가중치는 1부터 20까지 0.1씩 증가시켜 총 200개의 모델을 생성한 후, 각 모델의 성능 지표를 비교하여 가장 우수한 모델을 선정하고자 하였습니다. 200개의 모델 중 다음의 6개 후보군을 선정하여 최종 모델 후보로 설정하였습니다.

로지스틱 회귀에서 Prediction R2 사용 불가 이유

- **선형성 가정의 부재**:
  전통적인 R2는 선형 회귀 모델에서 **독립 변수와 종속 변수 간의 선형 관계**를 기반으로 계산됩니다. 하지만 로지스틱 회귀 모델은 **이항 종속 변수**를 대상으로 하며, 예측 확률을 기반으로 학습하기 때문에 이러한 선형 관계를 가정하지 않습니다.

- **잔차(Residual)의 정의 차이**:
  선형 회귀에서는 잔차(실제 값과 예측 값의 차이)가 R2$R2$ 계산의 핵심입니다. 그러나 로지스틱 회귀에서는 잔차가 확률(0~1 범위)로 표현되며, 이러한 잔차는 선형 회귀에서의 잔차와 개념적으로 다릅니다.

- **모델 성능 평가 방식 차이**:
  로지스틱 회귀는 예측의 정확성을 평가하기 위해 AUC, 민감도, 특이도, 로그우도(Log-Likelihood) 등 다른 지표를 주로 사용합니다. 따라서 전통적인 R2$R2$는 로지스틱 모델의 성능을 충분히 설명하지 못합니다.

이에 저희는 **Prediction R2** 대신 **McFadden R2** *를 사용하였습니다.*

정의:
로그우도의 비율을 이용하여 계산하며, Null 모델과 비교하여 개선된 모델의 설명력을 나타냅니다.

$$R^2_{\text{McFadden}} = 1 - \frac{\ln(\text{L}_1)}{\ln(\text{L}_0)}$$

특징:

- McFadden R^2는 일반적으로 0.2~0.4 범위에서 높은 성능을 나타내는 것으로 간주됩니다.
- 값이 1에 가까울수록 모델의 적합도가 높음을 의미합니다.

```r
# 최적 모델 갱신
if (mean_results["R2"] > best_r2) {
  best_r2 <- mean_results["R2"]
  best_model_r2 <- model
  best_weight_r2 <- weight_value
  best_cm_r2 <- cm
}

# 다른 조건별 최적 모델 갱신 (기존 로직 유지)
if (mean_results["AUC"] > best_auc) {
  best_auc <- mean_results["AUC"]
  best_model_auc <- model
  best_weight_auc <- weight_value
  best_cm_auc <- cm
}
if (mean_results["Sensitivity"] > best_sens) {
  best_sens <- mean_results["Sensitivity"]
  best_model_sens <- model
  best_weight_sens <- weight_value
  best_cm_sens <- cm
}
if (mean_results["Specificity"] > best_spec) {
  best_spec <- mean_results["Specificity"]
  best_model_spec <- model
  best_weight_spec <- weight_value
  best_cm_spec <- cm
}
if (mean_results["F1_Score"] > best_f1) {
  best_f1 <- mean_results["F1_Score"]
  best_model_f1 <- model
  best_weight_f1 <- weight_value
  best_cm_f1 <- cm
}
if (mean_results["Sensitivity"] >= 0.6 && mean_results["Specificity"] >= 0.7 && mean_results["AUC"] > best_auc_my)
  best_auc_my <- mean_results["AUC"]
  best_model_my <- model
  best_weight_my <- weight_value
  best_cm_my <- cm
}
if (mean_results["Sensitivity"] >= 0.7 && mean_results["Specificity"] >= 0.6 && mean_results["AUC"] > best_auc_my2)
  best_auc_my2 <- mean_results["AUC"]
  best_model_my2 <- model
  best_weight_my2 <- weight_value
  best_cm_my2 <- cm
}
# 결과 출력 (R² 포함)
cat(sprintf("Weight: %.1f | AUC: %.4f | Sens: %.4f | Spec: %.4f | F1: %.4f | R2: %.4f\n",
            weight_value, mean_results["AUC"], mean_results["Sensitivity"],
            mean_results["Specificity"], mean_results["F1_Score"], mean_results["R2"]))
```

1. **Prediction R2** 대신 **McFadden R2** 이 최고 성능을 보이는 모델
2. AUC가 최고 성능을 보이는 모델
3. 민감도가 최고 성능을 보이는 모델
4. 특이도가 최고 성능을 보이는 모델
5. F1 score가 최고 성능을 보이는 모델
6. 민감도가 0.6이상이고 특이도가 0.7이상이라는 제약 조건을 건 후 AUC가 최고 성능인 모델
7. 민감도가 0.7이상이고 특이도가 0.6이상이라는 제약 조건을 건 후 AUC가 최고 성능인 모델

Train/Test Set 분할 및 K-Fold 교차 검증:

과제 명세에 따라 데이터를 **Train Set**과 **Test Set**으로 나누는 것이 필수적이며, 이를 보다 신뢰성 있게 수행하기 위해 **K-Fold 교차 검증(K-Fold Cross-Validation)**을 적용하였습니다.

- **K-Fold 교차 검증**은 데이터를 **K개의 부분 집합**으로 나누고, 각 부분을 번갈아 가며 테스트 데이터로 사용하여 모델의 일반화 성능을 평가하는 방법입니다.
- 이 방식은 데이터 수가 적은 상황에서도 **데이터 효율성을 극대화**할 수 있으며, 특히 **데이터 불균형 문제**를 완화하는 데 효과적입니다.
- 우리는 **K = 5**로 설정하였으며, 이는 데이터 규모와 복잡도, 계산 비용을 모두 고려한 결과입니다. 5-Fold 방식은 데이터가 적은 경우 과도한 분할로 인한 학습 데이터 손실을 방지하면서, 충분한 검증 기회를 제공합니다.

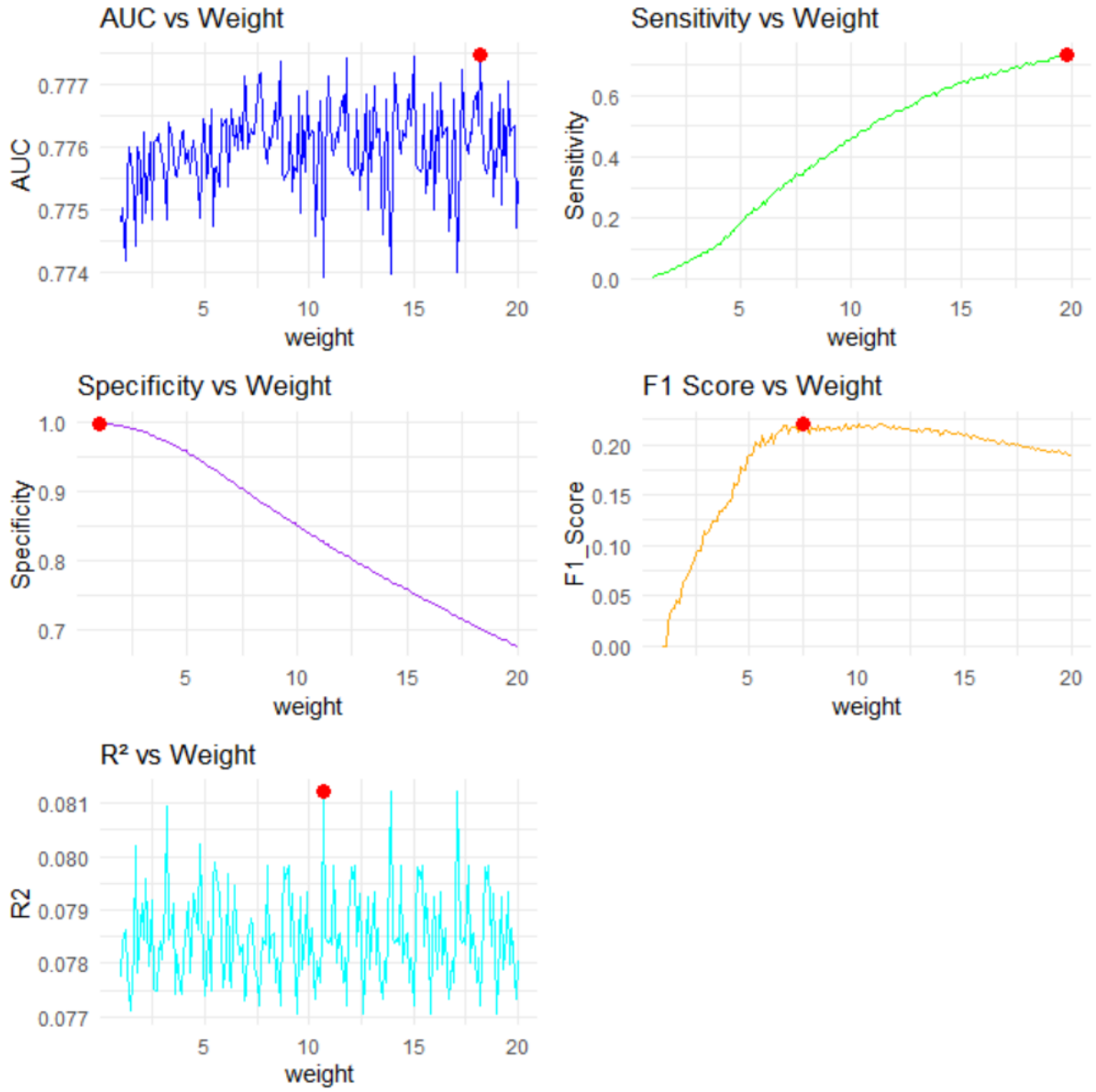
```
for (fold_index in seq_along(folds)) {
  test_idx <- folds[[fold_index]]
  train_data <- data[-test_idx, ]
  test_data <- data[test_idx, ]

  tryCatch({
    model <- train(
      당뇨병유발 ~ 나이 + BMI구간,
      data = train_data,
      method = "glm",
      family = "binomial",
      metric = "ROC",
      weights = train_data$weights,
      trControl = trainControl(method = "cv", number = 5, classProbs = TRUE, summaryFunction = twoClassSummary
    )

    # 테스트 데이터 예측
    pred_probs <- predict(model, test_data, type = "prob")
    pred_classes <- factor(ifelse(pred_probs$YES > 0.5, "YES", "NO"), levels = c("NO", "YES"))
    cm <- confusionMatrix(pred_classes, test_data$당뇨병유발, positive = "YES")

    # AUC 계산
    roc_curve <- roc(test_data$당뇨병유발, pred_probs$YES, levels = c("NO", "YES"))
    auc_value <- auc(roc_curve)

    # R² 계산 (pseudo R² for logistic regression)
    null_model <- glm(당뇨병유발 ~ 1, family = "binomial", data = test_data)
    full_model <- glm(당뇨병유발 ~ 나이 + BMI구간, family = "binomial", data = test_data)
    r2_value <- 1 - (logLik(full_model) / logLik(null_model))  # McFadden's R²
    r2_value <- as.numeric(r2_value)
```

모델 훈련 과정

```
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Weight: 14.9 | AUC: 0.7760 | Sens: 0.6399 | Spec: 0.7587 | F1: 0.2107 | R2: 0.0$
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Weight: 15.0 | AUC: 0.7754 | Sens: 0.6306 | Spec: 0.7582 | F1: 0.2076 | R2: 0.0$
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Weight: 15.1 | AUC: 0.7769 | Sens: 0.6387 | Spec: 0.7556 | F1: 0.2084 | R2: 0.0$
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
Setting direction: controls < cases
```

혼동행렬 및 모델 성능 평가:

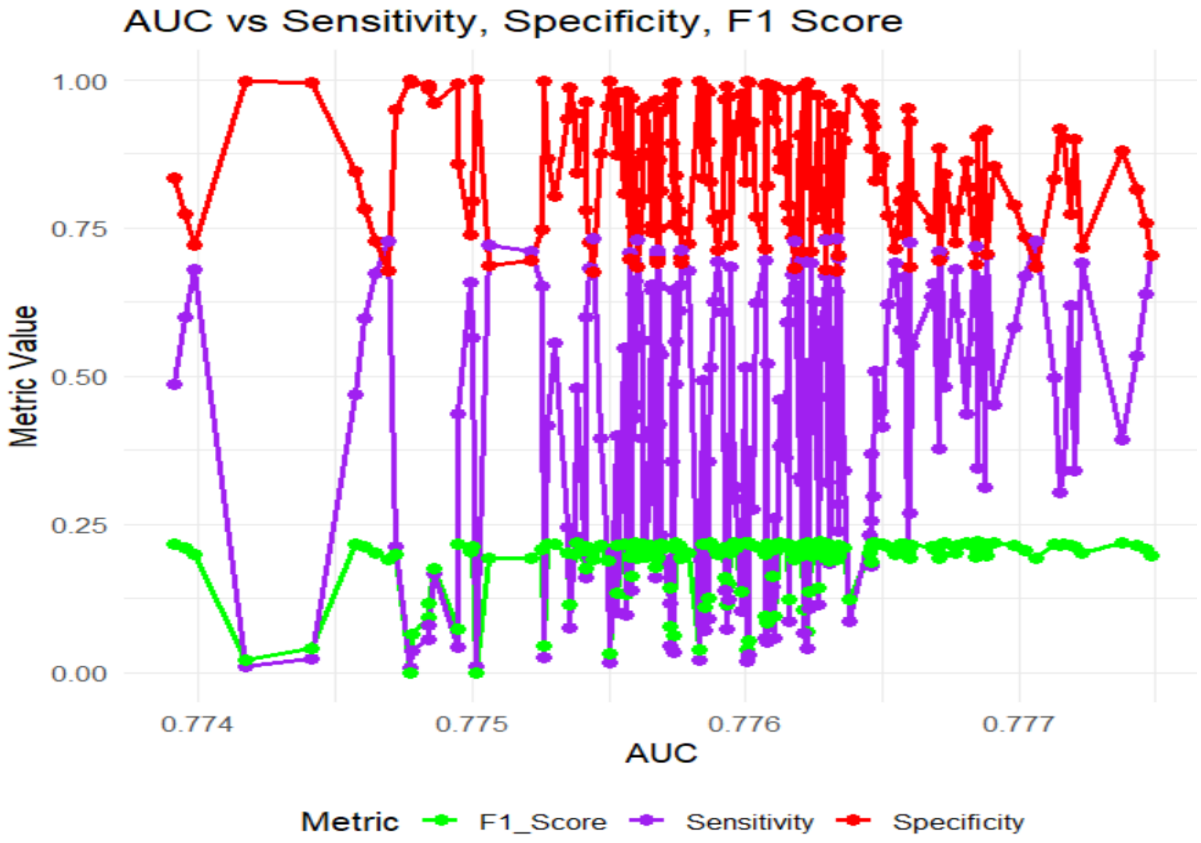
K-Fold 교차 검증 과정에서 각 Fold마다 **혼동행렬(Confusion Matrix)**을 생성하여 예측 결과와 실제 레이블을 비교하였습니다. 혼동행렬은 모델의 성능을 민감도(Sensitivity), 특이도(Specificity), F1 Score 등 다양한 관점에서 분석하는 데 사용되었습니다. 이를 통해 각 Fold에서의 성능 변화를 관찰하고, 최종적으로 평균 성능을 기반으로 최적의 모델을 선정하였습니다.



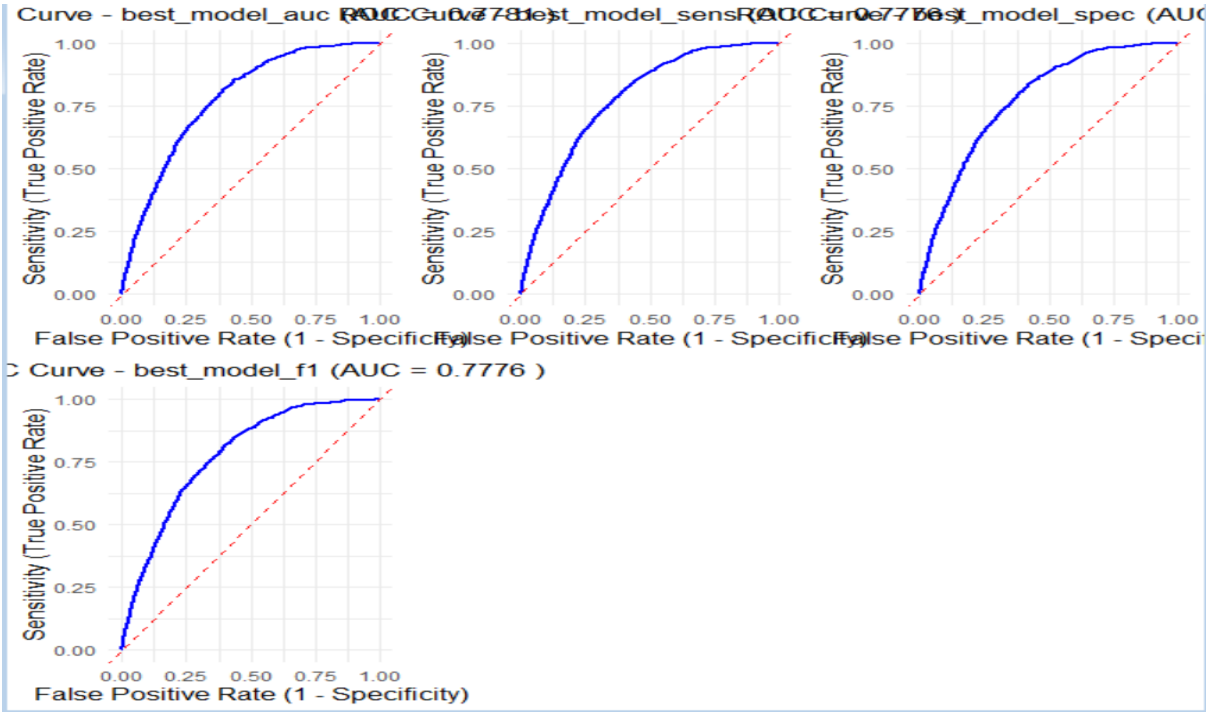
해당 그림은 가중치에 따른 AUC, 민감도, 특이도, F1_SCORE, R2의 변화하는 plot입니다. Auc와 R2은 weight에 따라 값의 변화가 일정한 규칙 없이 변하는 모습을 보입니다. 또한 최댓값과 최소값의 큰 차이가 없음을 알 수 있습니다. 민감도 같은경우 가중치를 증가 시킬수록 계속 증가하는 모습을 보입니다. 특이도 같은 경우 가중치를 증가시킬수록 계속 감소하는 경향을 보입니다. F1 score는 증가하다가 특정 지점부터 감소하면서 일정

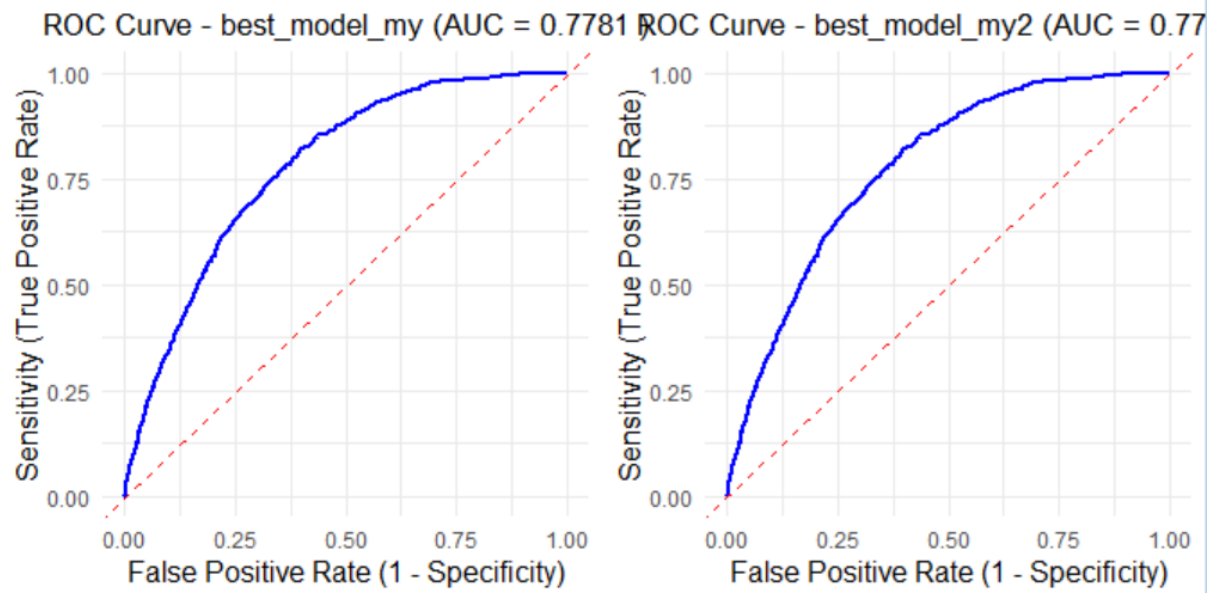
한 값으로 수렴하는 경향을 보입니다.

결론 : 모델 200개에서 auc와 r2의 변화 규칙은 불규칙적이며 최댓값과 최솟값의 큰 차이가 없음을 알 수 있습니다. 이에 저희는 auc와 r2 값에서 일부 손실을 보이더라도 명백한 근거하에 다른 모델이 우수하다고 판단되면 그 모델을 최종 모델로 선정하였습니다.



본 그림은 auc에 따른 민감도 특이도 f1 score의 변화를 의미합니다. 보시다 싶히 auc에 따른 민감도와 특이도의 변화 폭이 매우 큰 상황입니다.





후보군 모델 7개의 ROC 커브와 AUC 면적 값입니다. 보시다 싶히 AUC값의 큰 차이가 없기에 AUC가 가장 우수하다고 최종 모델로 선정하기에는 한계가 존재합니다. 이제

각 7개 모델의 대한 가중치 값 R2값 AUC값 민감도 특이도 F1 SCORE를 확인해보겠습니다.

1. R^2 이 최고 성능을 보이는 모델

```
Weight: 10.7 | AUC: 0.7739 | Sens: 0.4864 | Spec: 0.8355 | F1: 0.2162 | R2: 0.0812
Setting direction: controls < cases
```

. R^2이 최고 성능을 보이는 경우

가중치 : 10.7

AUC : 0.7739

R2 : 0.0812

민감도 : 0.4864

특이도 : 0.8355

F1 score : 0.2162

Accuracy : 0.8042

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2560   89
       YES  554   81

               Accuracy : 0.8042
                 95% CI : (0.7902, 0.8177)
    No Information Rate : 0.9482
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1302

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.47647
            Specificity : 0.82209
         Pos Pred Value : 0.12756
         Neg Pred Value : 0.96640
             Prevalence : 0.05177
         Detection Rate : 0.02467
   Detection Prevalence : 0.19336
      Balanced Accuracy : 0.64928

       'Positive' Class : YES
```

Test set으로 혼동행렬을 만들 결과 양성 레이블을 제대로 판단하지 못합니다. 거의 음성테이블로만 판단하는 경향을 보입니다. 데이터의 95%가 음성 테이블이기에 정확도라는 성능 지표에서는 높아 보이나 양성 테이블을 제대로 학습 하지 못하므로 해당 모델 후보군은 좋은 모델이라 판단을 하기에는 힘든 경향을 보입니다.

2. AUC 최고 성능을 보이는 모델

```
Weight: 18.2 | AUC: 0.7775 | Sens: 0.7051 | Spec: 0.7046 | F1: 0.1972 | R2: 0.0772
Setting direction: controls < cases
```

가중치 : 18.2

AUC : 0.7775

R2 : 0.0772

민감도 : 0.7775

특이도 : 0.7061

F1 score : 0.1972

Accuracy : 0.7103

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2201   38
       YES  913  131

               Accuracy : 0.7103
                 95% CI : (0.6945, 0.725
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1398

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.77515
            Specificity : 0.70681
         Pos Pred Value : 0.12548
         Neg Pred Value : 0.98303
             Prevalence : 0.05148
         Detection Rate : 0.03990
   Detection Prevalence : 0.31800
      Balanced Accuracy : 0.74098

       'Positive' Class : YES
```

과제 명세에 나온 평가 지표 중 AUC 측면에서는 최고 모델이기는 다른 모델과 지켜봐야 할 것 같습니다. 더 좋은 민감도와 특이도를 보일 수 있기 때문입니다. 첫 번째 모델에 비해 정확도 측면에서는 감소하였습니다.

3. 민감도 최고 성능을 보이는 모델

```
Weight: 19.8 | AUC: 0.7763 | Sens: 0.7332 | Spec: 0.6783 | F1: 0.1917 | R2: 0.0778
```

가중치 : 20

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2153   48
       YES  961  121

               Accuracy : 0.6927
                 95% CI : (0.6766, 0.7084)
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1146

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.71598
            Specificity : 0.69139
         Pos Pred Value : 0.11183
         Neg Pred Value : 0.97819
             Prevalence : 0.05148
         Detection Rate : 0.03686
   Detection Prevalence : 0.32958
      Balanced Accuracy : 0.70369

       'Positive' Class : YES
```

가중치 : 19.8

AUC : 0.7763

R2 : 0.0078

민감도 : 0.7332

특이도 : 0.6783

F1 score : 0.1917

Accuracy : 0.6927

1번째 모델에 비해 민감도가 많은 개선이 되었으나 특이도와 정확도 측면에서는 많은 감소를 보입니다.

4. 특이도 최고 성능을 보이는 모델

```
Weight: 1.0 | AUC: 0.7748 | Sens: 0.0083 | Spec: 0.9992 | F1: 0.0000 | R2: 0.0777
```

가중치 : 1

AUC : 0.7748

R2 : 0.0777

민감도 : 0.0083

특이도 : 0.9992

F1 score : 0.0777

Accuracy : 0.9488

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  3113  167
       YES    1    2

               Accuracy : 0.9488
                 95% CI : (0.9407, 0.9561)
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 0.489

                  Kappa : 0.0215

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0118343
            Specificity : 0.9996789
         Pos Pred Value : 0.6666667
         Neg Pred Value : 0.9490854
             Prevalence : 0.0514773
         Detection Rate : 0.0006092
   Detection Prevalence : 0.0009138
      Balanced Accuracy : 0.5057566

       'Positive' Class : YES
```

이 모델은 모두 음성이라고 판단해버린다. TEST 셋의 95%가 음성 레이블이기에 높은 정확성을 보이나 음성에 대한 판단을 하지 못하기에 결코 좋은 모델이라 할 수 없습니다.

5. F1 score 최고 성능을 보이는 모델

```
Weight: 7.5 | AUC: 0.7768 | Sens: 0.3447 | Spec: 0.9041 | F1: 0.2215 | R2: 0.0777
```

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2846  114
       YES  268   56

               Accuracy : 0.8837
                 95% CI : (0.8722, 0.8945)
    No Information Rate : 0.9482
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1704

 Mcnemar's Test P-Value : 4.951e-15

            Sensitivity : 0.32941
            Specificity : 0.91394
         Pos Pred Value : 0.17284
         Neg Pred Value : 0.96149
             Prevalence : 0.05177
         Detection Rate : 0.01705
   Detection Prevalence : 0.09866
      Balanced Accuracy : 0.62167

       'Positive' Class : YES
```

가중치 : 7.5

AUC : 0.7768

R2 : 0.0777

민감도 : 0.3447

특이도 : 0.9041

F1 score : 0.2215

Accuracy:0.8837

특이도도 90%이상 민감도도 34%이상이고 정확도도 88%에 해당하는 만큼 기존 4개의 모델보다 우수하다고 판단이 되나 민감도가 50%도 안되기에 최종 모델로 선택하기에는 여러가지를 고려해보아야 겠다고 생각이 듭니다.

6. 민감도가 0.6이상이고 특이도가 0.7이상이라는 제약 조건을 건 후 AUC가 최고 성능인 모델

```
Weight: 18.2 | AUC: 0.7775 | Sens: 0.7051 | Spec: 0.7046 | F1: 0.1972 | R2: 0.0772
Setting direction: controls < cases
```

가중치 : 18.2

AUC : 0.7775

R2 : 0.0772

민감도 : 0.7775

특이도 : 0.7061

F1 score : 0.1972

Accuracy : 0.7103

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2201   38
       YES  913  131

               Accuracy : 0.7103
                 95% CI : (0.6945, 0.725
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1398

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.77515
            Specificity : 0.70681
         Pos Pred Value : 0.12548
         Neg Pred Value : 0.98303
             Prevalence : 0.05148
         Detection Rate : 0.03990
   Detection Prevalence : 0.31800
      Balanced Accuracy : 0.74098

       'Positive' Class : YES
```

2번 모델과 같은 모델이 나왔습니다.

7. 민감도가 0.7이상이고 특이도가 0.6이상이라는 제약 조건을 건 후 AUC가 최고 성능인 모델

```
Weight: 18.2 | AUC: 0.7775 | Sens: 0.7051 | Spec: 0.7046 | F1: 0.1972 | R2: 0.0772
Setting direction: controls < cases
```

가중치 : 18.2

AUC : 0.7775

R2 : 0.0772

민감도 : 0.7775

특이도 : 0.7061

F1 score : 0.1972

Accuracy : 0.7103

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2201   38
       YES  913  131

               Accuracy : 0.7103
                 95% CI : (0.6945, 0.725
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1398

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.77515
            Specificity : 0.70681
         Pos Pred Value : 0.12548
         Neg Pred Value : 0.98303
             Prevalence : 0.05148
         Detection Rate : 0.03990
   Detection Prevalence : 0.31800
      Balanced Accuracy : 0.74098

       'Positive' Class : YES
```

2번 6번 모델과 같은 모델이 나왔습니다.

최종 모델 선정

최종 모델로 2,6,7번 모델을 선정하였습니다. 비록 2,6,7번 모델이 7개의 모델 후보군 중에서 가장 높은 AUC나 R2*R2* f1score 값을 기록한 것은 아니지만, 그 선택은 단순히 성능 지표만을 기준으로 한 결정이 아니었습니다. 1번 은 음성 클래스에 대한 정확한 예측을 하지 못하는 한계를 보였습니다. 특히, 데이터셋의 불균형이 심각하게 나타나는 상황에서, 95%가 음성인 데이터에서 모든 예측을 음성으로 판단하는 모델은 높은 정확도 지표를 보일 수 있지만, 실제로는 잘못된 예측을 반복하며 신뢰할 수 없습니다. 이러한 이유로, 모델의 정확도뿐만 아니라 민감도(Sensitivity)와 특이도(Specificity)를 종합적으로 고려해야 한다는 점을 간과할 수 없습니다.

따라서, 6번 모델은 AUC와 R2*R2* 측면에서 1번 모델이나 2번 모델에 비해 일부 성능 저하가 있긴 하지만(AUC 측면에서는 0.0001419의 손실을 보고 1번 모델에 비해서는 R2값은 0.003449)**, 민감도와 음성 클래스에 대한 예측을 더 잘 반영하며 불균형한 데이터셋에서도 유의미한 예측을 할 수 있다는 강점을 보입니다. 특히, 민감도는 모델이 음성 클래스에 대해 얼마나 잘 판별하는지를 평가하는 중요한 지표로, 6번 모델이 불균형 데이터 문제를 해결하는 데 있어 우수한 성능을 발휘했음을 알 수 있습니다.**

결론적으로, 단순한 성능 지표나 지표 간의 차이를 넘어서, 데이터의 불균형을 고려한 민감도와 예측의 전반적인 신뢰성을 반영한 결과 6번 모델을 최종 모델로 선정하였습니다. 이 모델이 실제 현장 문제를 해결하는 데 있어 더 큰 신뢰를 제공할 것으로 기대합니다.

```
> summary(best_model_my)

Call:
NULL

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)             -5.109097   0.083820 -60.953  < 2e-16 ***
나이                     0.073313   0.001442  50.823  < 2e-16 ***
BMI구간23.0이상24.9이하  0.403993   0.038700  10.439  < 2e-16 ***
BMI구간25.0이상29.9이하  0.483480   0.036996  13.069  < 2e-16 ***
BMI구간30이상            1.290766   0.075047  17.199  < 2e-16 ***
흡연여부흡연자           0.419179   0.030914  13.559  < 2e-16 ***
고혈압여부고혈압자       0.299835   0.036580   8.197 2.47e-16 ***
중성지방                 0.005269   0.000157  33.553  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34468  on 13133  degrees of freedom
Residual deviance: 28385  on 13126  degrees of freedom
AIC: 28175

Number of Fisher Scoring iterations: 6
```

로지스틱 회귀 모델에서 최종 성능 지표

```
Weight: 18.2 | AUC: 0.7775 | Sens: 0.7051 | Spec: 0.7046 | F1: 0.1972 | R2: 0.0772
Setting direction: controls < cases
```

가중치 : 18.2

AUC : 0.7775

R2 : 0.0772

민감도 : 0.7775

특이도 : 0.7061

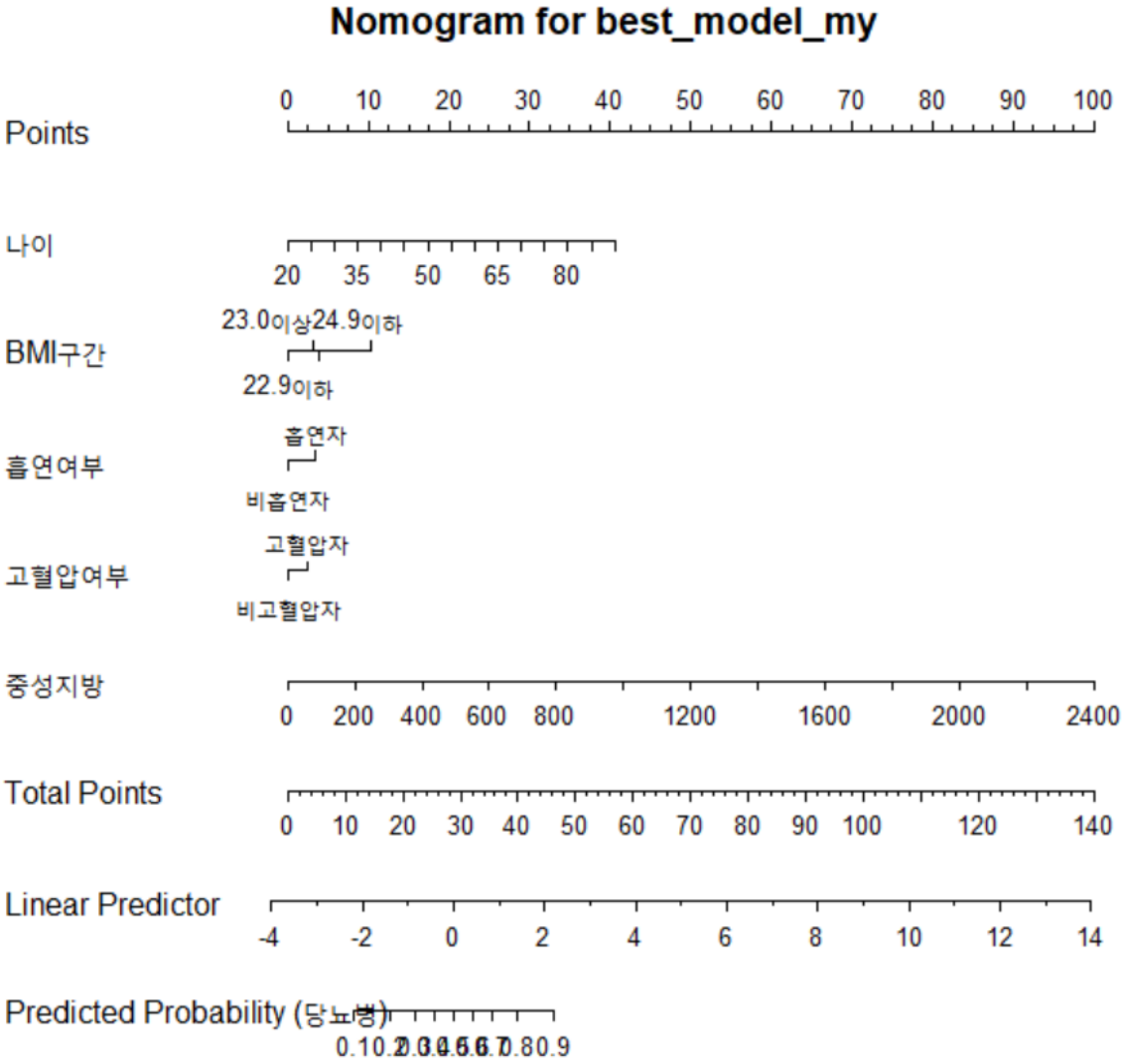
F1 score : 0.1972

Accuracy : 0.7103

```
Confusion Matrix and Statistics

          Reference
Prediction   NO  YES
       NO  2201   38
       YES  913  131

               Accuracy : 0.7103
                 95% CI : (0.6945, 0.725
    No Information Rate : 0.9485
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1398

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.77515
            Specificity : 0.70681
         Pos Pred Value : 0.12548
         Neg Pred Value : 0.98303
             Prevalence : 0.05148
         Detection Rate : 0.03990
   Detection Prevalence : 0.31800
      Balanced Accuracy : 0.74098

       'Positive' Class : YES
```

과제 명세와 별개로 노모그램까지 제작 해보았습니다.

노모그램(Nomogram)은 통계학과 의료 분야에서 많이 사용되는 도구로, 여러 변수들의 관계를 시각적으로 나타내어 특정 결과나 예측을 빠르고 직관적으로 추정할 수 있도록 도와주는 그래픽적인 도표입니다. 보통 환자 개개인의 예후를 예측하거나, 다양한 변수들의 영향을 시각적으로 이해할 때 사용됩니다.


Nomogram for best_model_my
Points
0 10 20 30 40 50 60 70 80 90 100
나이
20 35 50 65 80
BMI구간
23.0이상24.9이하
22.9이하
흡연여부
흡연자
비흡연자
고혈압여부
고혈압자
비고혈압자
중성지방
0 200 400 600 800 1200 1600 2000 2400
Total Points
0 10 20 30 40 50 60 70 80 90 100 120 140
Linear Predictor
-4 -2 0 2 4 6 8 10 12 14
Predicted Probability (당뇨병)
0.1 0.8 0.9

```
data=read.csv("C:₩₩Users₩₩user₩₩Desktop₩₩학업₩₩2024-2₩₩데이터마이닝₩₩기말프로젝트₩₩phenotype_csv.csv",fileEncoding='UTF-8')

head(data)

names(data)

attach(data)

summary(data) #16955행이 존재한다

str(data)

colSums(is.na(data))


#공복혈당에 대한 정보

sum(is.na(data$FBS_B))

sum(data$FBS_B,na.rm=TRUE)

par(mfrow=c(1,2))

hist(FBS_B)

boxplot(FBS_B)

당뇨병발병YES수 <- sum(data$FBS_B>=126,na.rm=TRUE)

당뇨병발병NO수 <- sum(data$FBS_B<126,na.rm=TRUE)

cat("당뇨병 발병 YES 수 : ",당뇨병발병YES수,"₩n")

cat("당뇨병 발병 NO 수 : ",당뇨병발병NO수,"₩n")

cat("당뇨병 발병 비율 : ",당뇨병발병YES수/(당뇨병발병NO수+당뇨병발병YES수),"₩n") #데이터간 불균형이 있음을 확인가능

data$당뇨병유발 <- ifelse(data$FBS_B>=126, "YES",
                          ifelse(data$FBS_B<126 , "NO", NA))

sum(data$FBS_B,na.rm=TRUE)
```

```
#나이에 대한 정보

sum(is.na(data$AGE_B))

sum(data$FBS_B,na.rm=TRUE)

par(mfrow=c(1,2))

hist(AGE_B)

boxplot(AGE_B)

cor(FBS_B,AGE_B,use = "complete.obs",method="spearman") #공복혈당과 나이의 hist가 정규분포라고 하기에는 보기 어려워 스피어맨 방식 사용, 양의 상관 관계가 존재

data$나이 <- AGE_B



#BMI에 대한 정보

data$BMI연속형 <- WT_B/(HT_B*HT_B)*10000 #bmi 계산법은 몸무게(wg단위)/(키(m단위)*2) 단위이다. data에서 키 정보는 cm단위이기에 10000을 곱해서 보정

attach(data)

head(data$BMI연속형)

sum(is.na(data$BMI연속형))

str(data$BMI연속형)

par(mfrow=c(1,2))

hist(BMI연속형)

boxplot(BMI연속형)

cor(FBS_B,data$BMI연속형,use = "complete.obs",method="spearman")


max(BMI연속형,na.rm=TRUE)

min(BMI연속형,na.rm=TRUE)
```

```
sum(BMI연속형<18.5,na.rm=TRUE)

sum(BMI연속형>=18.5 & BMI연속형<=22.9,na.rm=TRUE)

sum(BMI연속형>22.9 & BMI연속형<=24.9,na.rm=TRUE)

sum(BMI연속형>24.9 & BMI연속형<=29.9,na.rm=TRUE)

sum(BMI연속형>29.9 & BMI연속형<=34.9,na.rm=TRUE )

sum(BMI연속형>34.9,na.rm=TRUE)


data$BMI구간  <- ifelse(data$BMI연속형<18.5 , "18.5미만",
                         ifelse(BMI연속형>=18.5 & BMI연속형<=22.9, "18.5이상22.9이하",
                                ifelse(BMI연속형>22.9 & BMI연속형<=24.9, "23.0이상24.9
이하",
                                       ifelse(BMI연속형>24.9 & BMI연속형<=29.9, "25.0이
상29.9이하",
                                              ifelse(BMI연속형>29.9, "30이상", NA)))))


sum(!is.na(data$BMI구간))




#흡연

data$SMOK_B

sum(is.na(data$SMOK_B))

sum(data$SMOK_B==1,na.rm=TRUE)

sum(data$SMOK_B==2,na.rm=TRUE)

sum(data$SMOK_B==3,na.rm=TRUE)

sum((data$SMOK_B==1|data$SMOK_B==2) & data$당뇨병유발
```

```
=="YES",na.rm=TRUE)/(sum((data$SMOK_B==1|data$SMOK_B==2) & data$당뇨병유발
=="YES",na.rm=TRUE)+sum((data$SMOK_B==1|data$SMOK_B==2) & data$당뇨병유발
=="NO",na.rm=TRUE))

sum(data$SMOK_B==3 & data$당뇨병유발=="YES",na.rm=TRUE)/(sum(data$SMOK_B==3
& data$당뇨병유발=="YES",na.rm=TRUE)+sum(data$SMOK_B==3 & data$당뇨병유발
=="NO",na.rm=TRUE))

data$흡연여부 <- ifelse(data$SMOK_B == 1 | data$SMOK_B == 2, "비흡연자",
                    ifelse(data$SMOK_B == 3, "흡연자", NA))


#고혈압

sum(is.na(data$SBP_B))

sum(is.na(data$DBP_B))

data$고혈압여부 <- ifelse((data$SBP_B>=140 | data$DBP_B>=90),"고혈압자","비고혈압자
")

sum((data$고혈압여부=="고혈압자" & data$당뇨병유발
=="YES"),na.rm=TRUE)/(sum((data$고혈압여부=="고혈압자" & data$당뇨병유발
=="YES"),na.rm=TRUE)+sum((data$고혈압여부=="고혈압자" & data$당뇨병유발
=="NO"),na.rm=TRUE))

sum((data$고혈압여부=="비고혈압자" & data$당뇨병유발
=="YES"),na.rm=TRUE)/(sum((data$고혈압여부=="비고혈압자" & data$당뇨병유발=="비
고혈압자"),na.rm=TRUE)+sum((data$고혈압여부=="고혈압자" & data$당뇨병유발
=="NO"),na.rm=TRUE))



#저밀도콜레스테롤

data$LDL_B

sum(is.na(data$LDL_B))

par(mfrow=c(1,2))
```

```
hist(LDL_B)

boxplot(LDL_B)

cor(FBS_B,LDL_B,use = "complete.obs",method="spearman")

cor(FBS_B,LDL_B,use = "complete.obs",method="pearson")

data$저밀도콜레스테롤<-data$LDL_B


#중성지방

data$TG_B

sum(is.na(data$TG_B))

par(mfrow=c(1,2))

hist(TG_B)

boxplot(TG_B)

cor(FBS_B,TG_B,use = "complete.obs",method="spearman")

cor(FBS_B,TG_B,use = "complete.obs",method="pearson")

data$중성지방  <- data$TG_B


#새로운 데이터프레임 만들기

mydata <- data[, c("당뇨병유발", "나이", "BMI구간","흡연여부","고혈압여부","저밀도콜레스테롤","중성지방")]

head(mydata)

colSums(is.na(mydata))


mydata <-    na.omit(mydata)

colSums(is.na(mydata))

head(mydata)
```

```
write.csv(mydata,"C:₩₩Users₩₩user₩₩Desktop₩₩학업₩₩2024-2₩₩데이터마이닝₩₩기말프로젝트₩₩mydata후보군.csv",row.names=FALSE,fileEncoding="EUC-KR")


#변수선택 중요도 확인

mydata$당뇨병유발 <- factor(mydata$당뇨병유발, levels = c("NO", "YES"))

mydata$BMI구간 <- factor(mydata$BMI구간, levels=c("18.5미만","18.5이상22.9이하","23.0이상24.9이하","25.0이상29.9이하","30이상"))

mydata$흡연여부 <- factor(mydata$흡연여부, levels = c("비흡연자", "흡연자"))

mydata$고혈압여부 <- factor(mydata$고혈압여부, levels = c("비고혈압자", "고혈압자"))


# 전체에서 제거하는법을 통해 모델 학습

logit_model_full <- glm(당뇨병유발 ~ ., data = mydata, family = binomial())

summary(logit_model_full)

# 단계적 변수 선택 (Backward Elimination)

logit_model_stepwise <- step(logit_model_full, direction = "backward")


# 최종 모델 확인

summary(logit_model_stepwise)


logit_model_full <- glm(당뇨병유발 ~ ., data = mydata, family = binomial())

summary(logit_model_full)


sum(mydata$BMI구간=="18.5이상22.9이하",na.rm=TRUE)
```

```
sum(mydata$BMI구간=="18.5미만",na.rm=TRUE)

mydata$BMI구간 <- as.character(mydata$BMI구간)

mydata$BMI구간 <- ifelse(mydata$BMI구간 == "18.5이상22.9이하" | mydata$BMI구간 == "18.5미만", "22.9이하", mydata$BMI구간)



sum(mydata$BMI구간=="22.9이하",na.rm=TRUE)

mydata$BMI구간

mydata$BMI구간 <- factor(mydata$BMI구간, levels=c("22.9이하","23.0이상24.9이하","25.0이상29.9이하","30이상"))


names(mydata)

logit_model_full <- glm(당뇨병유발 ~ 나이+BMI구간 +흡연여부+고혈압여부+중성지방, data = mydata, family = binomial())

summary(logit_model_full)

final <- mydata[, c("당뇨병유발", "나이", "BMI구간","흡연여부","고혈압여부","중성지방")]

write.csv(final,"C:₩₩Users₩₩user₩₩Desktop₩₩학업₩₩2024-2₩₩데이터마이닝₩₩기말프로젝트₩₩finaldata.csv",row.names=FALSE,fileEncoding="EUC-KR")



#데이터 훈련

library(pROC)

library(caret)

library(ggplot2)

library(gridExtra)

library(dplyr)
```

```
library(tidyr)

data <- read.csv("C:₩₩Users₩₩user₩₩Desktop₩₩학업₩₩2024-2₩₩데이터마이닝₩₩기말프로젝트₩₩finaldata.csv",fileEncoding="EUC-KR")

head(data)

names(data)

attach(data)

str(data)

data$당뇨병유발 <- factor(data$당뇨병유발, levels = c("NO", "YES"))

data$BMI구간 <- factor(data$BMI구간, levels=c("22.9이하","23.0이상24.9이하","25.0이상29.9이하","30이상"))

data$흡연여부 <- factor(data$흡연여부, levels = c("비흡연자", "흡연자"))

data$고혈압여부<- factor(data$고혈압여부, levels = c("비고혈압자","고혈압자"))

class(data$당뇨병유발)

class(data$나이)

class(data$BMI구간)

class(data$흡연여부)

class(data$고혈압여부)

class(data$중성지방)

levels(data$당뇨병유발)

levels(data$나이)

levels(data$BMI구간)

colSums(is.na(data))


# 가중치 튜닝 범위

weights_range <- seq(1, 20, by = 0.1)
```

```
# 결과 저장 벡터 (리스트로 변경)
result_list <- list()

# 모델과 혼동 행렬을 저장할 리스트
model_list <- list()
cm_list <- list()

# 최적 모델 초기화
best_auc <- 0
best_model_auc <- NULL
best_weight_auc <- NULL
best_cm_auc <- NULL

# 추가된 R² 관련 변수
best_r2 <- 0
best_model_r2 <- NULL
best_weight_r2 <- NULL
best_cm_r2 <- NULL

# 최적 모델 초기화
best_sens <- 0
best_model_sens <- NULL
best_weight_sens <- NULL
best_cm_sens <- NULL
```

```
# 최적 모델 초기화
best_spec <- 0
best_model_spec <- NULL
best_weight_spec <- NULL
best_cm_spec <- NULL

# 최적 모델 초기화
best_f1 <- 0
best_model_f1 <- NULL
best_weight_f1 <- NULL
best_cm_f1 <- NULL

# my모델(제약조건 : 민감도>=0.6 특이도>=0.7)
best_auc_my <-0
best_model_my <- NULL
best_weight_my <- NULL
best_cm_my <- NULL

# my2모델(제약조건 : 민감도>=0.7 특이도>=0.6)
best_auc_my2 <-0
best_model_my2 <- NULL
best_weight_my2 <- NULL
best_cm_my2 <- NULL
```

```
data$weights <- ifelse(data$당뇨병유발 == "YES", weight_value, 30)

for (weight_value in weights_range) {
  data$weights <- ifelse(data$당뇨병유발 == "YES", weight_value, 1)

  # 5-fold 교차검증
  folds <- createFolds(data$당뇨병유발, k = 5, list = TRUE, returnTrain = FALSE)

  fold_results <- data.frame(
    AUC = numeric(),
    Sensitivity = numeric(),
    Specificity = numeric(),
    F1_Score = numeric(),
    R2 = numeric()  # R² 추가
  )

  for (fold_index in seq_along(folds)) {
    test_idx <- folds[[fold_index]]
    train_data <- data[-test_idx, ]
    test_data <- data[test_idx, ]

    # 모델 학습
    model <- train(
      당뇨병유발 ~ 나이 + BMI구간 + 흡연여부 + 고혈압여부 + 중성지방,
      data = train_data,
```

```
    method = "glm",
    family = "binomial",
    metric = "ROC",
    weights = train_data$weights,
    trControl = trainControl(
      method = "cv",
      number = 5,
      classProbs = TRUE,
      summaryFunction = twoClassSummary,
      savePredictions = "final"
    )
  )

  # 테스트 데이터 예측
  pred_probs <- predict(model, test_data, type = "prob")
  pred_classes <- factor(ifelse(pred_probs$YES > 0.5, "YES", "NO"), levels = c("NO",
"YES"))
  cm <- confusionMatrix(pred_classes, test_data$당뇨병유발, positive = "YES")

  # AUC 계산
  roc_curve <- roc(test_data$당뇨병유발, pred_probs$YES, levels = c("NO", "YES"))
  auc_value <- auc(roc_curve)

  # R² 계산 (pseudo R² for logistic regression)
  null_model <- glm(당뇨병유발 ~ 1, family = "binomial", data = test_data)
```

```
    full_model <- glm(당뇨병유발 ~ 나이 + BMI구간, family = "binomial", data =
test_data)
    r2_value <- 1 - (logLik(full_model) / logLik(null_model))  # McFadden's R²
    r2_value <- as.numeric(r2_value)

    # F1-Score 계산
    specificity_value <- cm$byClass["Specificity"]
    precision_value <- cm$byClass["Precision"]
    recall_value <- cm$byClass["Sensitivity"]
    if (is.na(precision_value)) precision_value <- 0.01
    if (is.na(recall_value)) recall_value <- 0.01
    f1_value <- 2 * (precision_value * recall_value) / (precision_value + recall_value)

    # 결과 저장
    fold_results <- rbind(fold_results, data.frame(
      AUC = auc_value,
      Sensitivity = cm$byClass["Sensitivity"],
      Specificity = cm$byClass["Specificity"],
      F1_Score = f1_value,
      R2 = r2_value
    ))
  }

  # 각 폴드의 평균 결과 계산
  mean_results <- colMeans(fold_results)
```

```
  # 결과 저장
  result_list[[as.character(weight_value)]] <- data.frame(
    weight = weight_value,
    AUC = mean_results["AUC"],
    Sensitivity = mean_results["Sensitivity"],
    Specificity = mean_results["Specificity"],
    mean_results["F1_Score"] <- ifelse(is.na(mean_results["F1_Score"]), 0,
mean_results[["F1_Score"]]),
    F1_Score = mean_results["F1_Score"],
    R2 = mean_results["R2"]
  )

  # 최적 모델 갱신
  if (mean_results["R2"] > best_r2) {
    best_r2 <- mean_results["R2"]
    best_model_r2 <- model
    best_weight_r2 <- weight_value
    best_cm_r2 <- cm
  }

  if (mean_results["AUC"] > best_auc) {
    best_auc <- mean_results["AUC"]
    best_model_auc <- model
    best_weight_auc <- weight_value
```

```
      best_cm_auc <- cm
    }
    if (mean_results["Sensitivity"] > best_sens) {
      best_sens <- mean_results["Sensitivity"]
      best_model_sens <- model
      best_weight_sens <- weight_value
      best_cm_sens <- cm
    }
    if (mean_results["Specificity"] > best_spec) {
      best_spec <- mean_results["Specificity"]
      best_model_spec <- model
      best_weight_spec <- weight_value
      best_cm_spec <- cm
    }
    if (mean_results["F1_Score"] > best_f1) {
      best_f1 <- mean_results["F1_Score"]
      best_model_f1 <- model
      best_weight_f1 <- weight_value
      best_cm_f1 <- cm
    }
    if (mean_results["Sensitivity"] >= 0.6 && mean_results["Specificity"] >= 0.7 &&
mean_results["AUC"] > best_auc_my) {
      best_auc_my <- mean_results["AUC"]
      best_model_my <- model
      best_weight_my <- weight_value
```

```
    best_cm_my <- cm
  }
  if (mean_results["Sensitivity"] >= 0.7 && mean_results["Specificity"] >= 0.6 &&
mean_results["AUC"] > best_auc_my2) {
    best_auc_my2 <- mean_results["AUC"]
    best_model_my2 <- model
    best_weight_my2 <- weight_value
    best_cm_my2 <- cm
  }

  # 결과 출력 (R² 포함)
  cat(sprintf("Weight: %.1f | AUC: %.4f | Sens: %.4f | Spec: %.4f | F1: %.4f | R2: %.4f₩n",
              weight_value, mean_results["AUC"], mean_results["Sensitivity"],
              mean_results["Specificity"], mean_results["F1_Score"], mean_results["R2"]))
}



# 5-fold 교차검증
# 결과 리스트 확인
print(result_list)
# 모델과 혼동 행렬 리스트 출력 (원하는 대로 확인 가능)
print(model_list)
print(cm_list)

cat("best_weight_r2 : ",best_weight_r2,"₩n")
```

```
cat("best_r2 : ",best_r2,"₩n")
print("best_cm_r2")
print(best_cm_r2)

cat("best_weight_auc : ",best_weight_auc,"₩n")
cat("best_auc : ",best_auc,"₩n")
print("best_cm_auc")
print(best_cm_auc)

cat("best_weight_sens : ",best_weight_sens,"₩n")
cat("best_sens : ",best_sens,"₩n")
print("best_cm_sens")
print(best_cm_sens)

cat("best_weight_spec : ",best_weight_spec,"₩n")
cat("best_spec : ",best_spec,"₩n")
print("best_cm_spec")
print(best_cm_spec)

cat("best_weight_f1 : ",best_weight_f1,"₩n")
cat("best_f1 : ",best_f1,"₩n")
print("best_cm_f1")
print(best_cm_f1)
```

```
cat("best_weight_my : ",best_weight_my,"₩n")

cat("best_auc_my : ",best_auc_my,"₩n")

print("best_cm_my")

print(best_cm_my)


cat("best_weight_my2 : ",best_weight_my2,"₩n")

cat("best_auc_my2 : ",best_auc_my2,"₩n")

print("best_cm_my2")

print(best_cm_my2)




# 결과 데이터를 결합

results_df <- bind_rows(lapply(result_list, function(x) x), .id = "weight")


# 데이터를 numeric 타입으로 변환

results_df$weight <- as.numeric(results_df$weight)


# 각 값의 최대값 인덱스 찾기

peak_auc <- which.max(results_df$AUC)

peak_sens <- which.max(results_df$Sensitivity)

peak_spec <- which.max(results_df$Specificity)

peak_f1 <- which.max(results_df$F1_Score)
```

```
peak_r2 <- which.max(results_df$R2)


# 그래프 그리기
p1 <- ggplot(results_df, aes(x = weight, y = AUC)) +
  geom_line(color = "blue") +
  geom_point(aes(x = weight[peak_auc], y = AUC[peak_auc]), color = "red", size = 3) +
  annotate("text", x = results_df$weight[peak_auc], y = results_df$AUC[peak_auc],
           label = paste("Max AUC:", round(results_df$AUC[peak_auc], 4), "₩nWeight:",
results_df$weight[peak_auc]),
           vjust = -1, color = "blue") +
  labs(title = "AUC vs Weight") +
  theme_minimal()

p2 <- ggplot(results_df, aes(x = weight, y = Sensitivity)) +
  geom_line(color = "green") +
  geom_point(aes(x = weight[peak_sens], y = Sensitivity[peak_sens]), color = "red", size =
3) +
  annotate("text", x = results_df$weight[peak_sens], y = results_df$Sensitivity[peak_sens],
           label = paste("Max Sens:", round(results_df$Sensitivity[peak_sens], 4),
"₩nWeight:", results_df$weight[peak_sens]),
           vjust = -1, color = "green") +
  labs(title = "Sensitivity vs Weight") +
  theme_minimal()

p3 <- ggplot(results_df, aes(x = weight, y = Specificity)) +
  geom_line(color = "purple") +
```

```
  geom_point(aes(x = weight[peak_spec], y = Specificity[peak_spec]), color = "red", size = 3) +
  annotate("text", x = results_df$weight[peak_spec], y = results_df$Specificity[peak_spec],
           label = paste("Max Spec:", round(results_df$Specificity[peak_spec], 4), "₩nWeight:", results_df$weight[peak_spec]),
           vjust = -1, color = "purple") +
  labs(title = "Specificity vs Weight") +
  theme_minimal()
p4 <- ggplot(results_df, aes(x = weight, y = F1_Score)) +
  geom_line(color = "orange") +
  geom_point(aes(x = weight[peak_f1], y = F1_Score[peak_f1]), color = "red", size = 3) +
  annotate("text", x = results_df$weight[peak_f1], y = results_df$F1_Score[peak_f1],
           label = paste("Max F1:", round(results_df$F1_Score[peak_f1], 4), "₩nWeight:", results_df$weight[peak_f1]),
           vjust = -1, color = "orange") +
  labs(title = "F1 Score vs Weight") +
  theme_minimal()
p5 <- ggplot(results_df, aes(x = weight, y = R2)) +
  geom_line(color = "cyan") +
  geom_point(aes(x = weight[peak_r2], y = R2[peak_r2]), color = "red", size = 3) +
  annotate("text", x = results_df$weight[peak_r2], y = results_df$R2[peak_r2],
           label = paste("Max R²:", round(results_df$R2[peak_r2], 4), "₩nWeight:", results_df$weight[peak_r2]),
           vjust = -1, color = "cyan") +
  labs(title = "R² vs Weight") +
  theme_minimal()
```

```
# 그래프 출력
grid.arrange(p1, p2, p3, p4,p5, ncol = 2)

# 1. result_list를 데이터 프레임으로 변환
data_list <- lapply(result_list, function(x) data.frame(x))
result_df <- do.call(rbind, data_list)

# weight 열 추가 (각 row에 weight 부여)
result_df$weight <- rep(sapply(result_list, function(x) x$weight), each =
nrow(result_list[[1]]))

# 데이터 구조 확인
#print(result_df)

# 2. Tidy 형태로 변환
# 'AUC', 'Sensitivity', 'Specificity', 'F1_Score'를 모아 Long 형식으로 변환
result_long <- result_df %>%
  pivot_longer(cols = c(Sensitivity, Specificity, F1_Score),
               names_to = "Metric",
               values_to = "Value")

# 3. ggplot을 사용하여 AUC 값에 따른 Metric 값 시각화
ggplot(result_long, aes(x = AUC, y = Value, color = Metric)) +
  geom_line(size = 1.2) +                # 선 그래프
```

```
  geom_point(size = 3) +                    # 각 포인트 추가
  scale_color_manual(values = c("green", "purple", "red")) +   # 색상 지정
  labs(title = "AUC vs Sensitivity, Specificity, F1 Score",
       x = "AUC",
       y = "Metric Value",
       color = "Metric") +                  # 범례 이름 지정
  theme_minimal(base_size = 15) +       # 깔끔한 테마
  theme(legend.position = "bottom")   # 범례 아래로 배치


# Plot ROC Curve 함수 정의
plot_roc_auc <- function(best_model, data) {
  # 모델 변수 이름 자동 추출
  model_name <- deparse(substitute(best_model))

  # 예측 확률을 얻기
  pred_probs <- predict(best_model, data, type = "prob")   # 클래스 확률 예측

  # ROC 곡선 계산 (positive class는 "YES")
  roc_curve <- roc(data$당뇨병유발, pred_probs$YES, levels = c("NO", "YES"))

  # AUC 값 계산
  auc_value <- auc(roc_curve)

  # ROC 데이터 프레임 만들기
```

```
  roc_data <- data.frame(
    FPR = 1 - roc_curve$specificities,  # False Positive Rate (1 - Specificity)
    Sensitivity = roc_curve$sensitivities  # Sensitivity (True Positive Rate)
  )

  # ROC 곡선 플로팅 (x축: FPR, y축: Sensitivity)
  p <- ggplot(roc_data, aes(x = FPR, y = Sensitivity)) +
    geom_line(color = "blue", size = 1) +  # ROC 곡선
    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  # y = x 대각선
    labs(
      title = paste("ROC Curve -", model_name, "(AUC =", round(auc_value, 4), ")"),
      x = "False Positive Rate (1 - Specificity)",
      y = "Sensitivity (True Positive Rate)"
    ) +
    theme_minimal() +
    theme(
      plot.title = element_text(hjust = 0.5),
      axis.title = element_text(size = 12)
    )
  cat("AUC for", model_name, ": ", auc_value, "₩n")
  # 플롯 반환
  return(p)
}
```

```
{
  # 각각의 플롯 생성
  auc_plot1 <- plot_roc_auc(best_model_auc, data)
  auc_plot2 <- plot_roc_auc(best_model_sens, data)
  auc_plot3 <- plot_roc_auc(best_model_spec, data)
  auc_plot4 <- plot_roc_auc(best_model_f1,data)
  auc_plot5 <- plot_roc_auc(best_model_r2,data)

  # 여러 플롯을 하나의 화면에 배치
  grid.arrange(auc_plot1, auc_plot2, auc_plot3, auc_plot4,   nrow = 2, ncol = 3)
}

{
  # 각각의 플롯 생성
  auc_plot6 <- plot_roc_auc(best_model_my, data)
  auc_plot7 <- plot_roc_auc(best_model_my2, data)

  # 여러 플롯을 하나의 화면에 배치
  grid.arrange(auc_plot6, auc_plot7, nrow = 2, ncol = 2)
}

library("rms")
best_model <- best_model_my
summary(best_model)
ddist <- datadist(data)
```

```
options(datadist = "ddist")

best_model_lrm <- lrm(당뇨병유발 ~ 나이 + BMI구간 + 흡연여부 + 고혈압여부 + 중성지방,
                      data = data, weights = data$weights)

coef(best_model_lrm)

best_model_lrm$coefficients <- coef(best_model$finalModel)

coef(best_model_lrm)


nom_auc <- nomogram(best_model_lrm, fun = function(x) 1 / (1 + exp(-x)), funlabel = "Predicted Probability (당뇨병)")

plot(nom_auc)

title(main = "Nomogram for best_model_my")

nom_auc



summary(best_model_my)
```

반응 변수 연속형 모델 – 연속형 변수 수축기 혈압 예측 모델 구축 및 분석

모델 설명 : 사용했던 10가지 모델들 중에서 부스팅을 사용한 모델들만 결정계수가 0.6을 넘어섰습니다. 사실 부스팅모델도 하이퍼파라미터 튜닝을 통해 모델 성능을 극대화하고자 했으나, 0.6을 넘지 못하고 0.55~0.57사이의 값으로 분포했었습니다. 그래서 모델분석 후 변수중요도 분석 결과에서 중요도가 비교적 낮게 나왔던 흡연여부(SMOK_B)와 음주여부(ALCO_B)를 제거했습니다. 그리고 설명변수를 혈압에 영향을 미칠 것으로 가정되는 주요 생리학적 변수(AGE_B, DBP_B, WAIST_B 등)와 중요도가 어느정도 있던 변수들을 기반으로 구성하였고, 이완기혈압과 나이, 체중과 같은 변수 간의 상호작용 효과(DBP_AGE, WAIST_AGE, DBP_WT)를 추가하여 변수 간 관계를 심화시켰습니다. 또한, 비선형적 특성을 반영하기 위해 제곱(DBP_B_sq, WAIST_B_sq)이나 로그 변환(log_WAIST_B, log_DBP_B), 제곱근 변환(sqrt_AGE_B) 등을 적용하여 데이터 분포를 정규화하고 모델 학습을 안정화했습니다. 마지막으로, BMI와 TG_FBS_ratio와 같은 대사적 상태를 나타내는 파생 변수를 포함하여, 혈압 변화와 관련된 중요한 생리적 특성을 반영했습니다. 최종적으로 부스팅모델들 중 Boosting Ensemble 모델이 최적의 RMSE와 R2 값을 보이기에 최종모델로 선택했습니다.

#전처리

데이터 중 1600개 미만의 데이터를 가진 열들은 제거했다. 추가로 FEV1과 FVC는 둘 다 자료가 있거나 둘 다 없거나 하다. FEV1열(과 FVC)열이 SBP_B와 유의미한 영향을 주지 않았고, NA값이 많아 제거하였다.

```
#datamining

data = read.table("C:₩₩Users₩₩yorion01₩₩Documents₩₩00_Current₩₩2024-2_데이터마이닝₩₩phenotype.txt",header = T,sep = "₩t")


head(data,1)

colSums(is.na(data))

str(data)

#데이터의 총 열 = 16955이므로 결측치가 이와 비슷한 열은 제외한다.

nData =
data[,c("FID","IID","AGE_B","SMOK_B","SMOKA_MOD_B","ALCO_B","ALCO_AMOUNT_B","EXER_B","HT_B","WT_B","WAIST_B",
                    "SBP_B","DBP_B","CHO_B","LDL_B","TG_B","HDL_B","FBS_B","GOT_B","GPT_B","GGT_B","URIC_B","FEV1","FVC",

"BIL","WBC","CREAT","STOMA","COLON","LIVER","LUNG","PROST","THROI","BREAC","RECTM","SCOLON","SRECTM",
                    "SPROST","STHROI","SBREAC","SLUNG","SSTOMA","SLIVER","SEX1","CRC","SCRC")]
```

```
str(nData)

colSums(is.na(nData))

nData$SMOKA_MOD_B

table(nData$SMOK_B)

#1600개 미만의 결측치를 가진 행은 제외

nData = nData[!is.na(data["SMOK_B"]),]

nData = nData[!is.na(nData["ALCO_AMOUNT_B"]),]

nData = nData[(!is.na(nData["WBC"])),]

nData = nData[(!is.na(nData["URIC_B"])),]

nData = nData[(!is.na(nData["BIL"])),]

nData = nData[(!is.na(nData["ALCO_B"])),]

nData = nData[(!is.na(nData["EXER_B"])),]

nData = nData[(!is.na(nData["WAIST_B"])),]

nData = nData[(!is.na(nData["LDL_B"])),]

nData = nData[(!is.na(nData["HT_B"])),]

nData = nData[(!is.na(nData["WT_B"])),]

nData = nData[(!is.na(nData["SBP_B"])),]

nData = nData[(!is.na(nData["DBP_B"])),]

nData = nData[(!is.na(nData["TG_B"])),]

nData = nData[(!is.na(nData["HDL_B"])),]

#흡연자가 아닌데, 하루 흡연 개수가 na이면 0으로 대체

nData[nData["SMOK_B"] != 3 & is.na(nData["SMOKA_MOD_B"]),"SMOKA_MOD_B"] = 0

#흡연자인데 , 하루 흡연 개수가 na인 개수 14개 행 삭제

sum(nData["SMOK_B"] == 3 & is.na(nData["SMOKA_MOD_B"]))

nData = nData[!(nData["SMOK_B"] == 3 & is.na(nData["SMOKA_MOD_B"])),]

colSums(is.na(nData))

write.csv(nData,file="C:₩₩Users₩₩yorion01₩₩Documents₩₩00_Current₩₩2024-2_데이터마이닝
₩₩DMDataByChanho.csv",row.names = F)
```

-연속형 분석

변수선택

<반응변수>

- **SBP_B: 수축기 혈압**

<설명변수>

원본 데이터에서 가져온 변수 :

- **AGE_B: 나이**
- **DBP_B: 이완기 혈압**
- **WAIST_B: 허리둘레**
- **WT_B: 체중**
- **CHO_B: 총콜레스트롤**
- **TG: 중성지방**
- **FBS_B: 공복 혈당**

```
       SBP_B         DBP_B       WAIST_B          WT_B         AGE_B       URIC_B          TG_B
FBS_B         GGT_B
  1.000000000   0.696841384   0.392092401   0.342276717   0.279381200   0.220955866   0.212837001
0.210499882   0.189968174
       CREAT           WBC          FEV1         CHO_B           FVC ALCO_AMOUNT_B   SMOKA_MOD_B
GPT_B          HT_B
  0.169253601   0.168809934   0.141012876   0.140555151   0.138210198   0.123958380   0.117751356
0.114986014   0.110370272
       GOT_B        SMOK_B         LDL_B         PROST        ALCO_B           BIL          LUNG
SBREAC        STHROI
  0.108314091   0.107218561   0.097927790   0.071902506   0.059688064   0.041946076   0.041819327
0.033730188   0.033196756
         CRC         RECTM         COLON         LIVER         STOMA        SSTOMA        SLIVER
SCOLON          SCRC
  0.017881806   0.015156266   0.011965006   0.008022205   0.004837540   0.004623866  -0.013550102
-0.013914331  -0.017226697
       SRECTM         SLUNG        SPROST         THROI        EXER_B             X         BREAC
HDL_B          SEX1
 -0.019413824  -0.030846749  -0.049427588  -0.054209541  -0.056951832  -0.089992959  -0.094116215
-0.144782456  -0.258499186
```

> 위는 수치형데이터로 구성된 데이터들 중 반응변수에 대한 설명변수들의 상관계수를 내림차순으로 정렬한 결과로서, 높은 상관성을 보이는 변수(DBP_B, WAIST_B, WT_B, AGE_B)와 혈압(CHO_B, TG_B, FBS_B)과 직접적인 관계를 가진다는 도메인 지식을 기반으로 선정하게 되었습니다. 그 외 초기에 흡연여부(SMOK_B)와 음주여부(ALCO_B)를 포함하고 전처리하여 모델을 테스트했으나 모델 성능결과가 좋지 못했고, 변수 중요도 분석에서 중요도가 낮아 제거한 후 데이터셋을 구성했습니다.

추가로, 모델 성능을 높이기 위해 다음과 같은 변수들이 추가되었습니다.

<추가 설명변수>

상호작용 변수

- **DBP_AGE: 이완기혈압 × 나이**

- **WAIST_AGE: 허리둘레 × 나이**
- **DBP_WT: 이완기혈압 × 체중**

원본 데이터에서 가져온 변수 :

- **DBP_B_sq: 이완기혈압의 제곱**
- **WAIST_B_sq: 허리둘레의 제곱**
- **log_WAIST_B: 허리둘레의 로그 변환**
- **log_DBP_B: 이완기혈압의 로그 변환**
- **sqrt_AGE_B: 나이의 제곱근 변환**

파생 변수

- **BMI: 체질량지수 (체중 ÷ 허리둘레²)**
- **TG_FBS_ratio: 중성지방 ÷ 공복 혈당 비율**

> 원본 데이터에서만 가져온 변수들로만 구성하여 모델을 훈련했을 때, 모델의 구성을 지속적으로 변경하여 훈련해도 R^2값이 0.55~0.57로 0.6을 넘지 않는 결과를 보였습니다. 모델의 예측력을 향상시키기 위해 변수 간 상호작용 효과를 반영하기 위해 나이와 이완기 혈압(DBP_AGE), 나이와 허리둘레(WAIST_AGE), 체중과 이완기 혈압(DBP_WT)과 같은 상호작용 변수를 생성하였으며, 허리둘레 및 이완기 혈압의 제곱(DBP_B_sq, WAIST_B_sq), 로그(log_WAIST_B, log_DBP_B), 나이의 제곱근(sqrt_AGE_B) 등 비선형성을 반영하는 변환을 수행하였습니다. 추가적으로, 체질량지수(BMI)와 중성지방-공복 혈당 비율(TG_FBS_ratio)과 같은 파생 변수를 통해 비만 및 대사 건강이 혈압에 미치는 영향을 포함했습니다.

<데이터 표준화>

AGE_B	DBP_B	WAIST_B	WT_B	CHO_B	TG_B	FBS_B	SBP_B	DBP_AGE
-0.07352	-0.79109	-0.88167	-0.70488	-0.93085	-0.6959	-0.40299	111	-0.44091
0.119809	-1.08303	-0.29083	-0.06675	-0.81134	-0.83189	0.038597	108	-0.44497
-0.17019	-0.0126	0.72973	0.881825	0.204513	-0.01595	1.314284	116	-0.1598
-0.26686	0.473967	0.514876	0.830085	0.234391	1.375322	0.283921	125	-0.02279
0.313141	-1.08303	0.837156	0.899072	1.549026	1.270715	-0.45205	103	-0.31507
-0.07352	-2.93197	-1.41881	-1.74831	0.294148	0.496622	-0.20673	93	-1.4233
-0.17019	0.084717	-0.23711	-0.24784	0.025245	-0.34023	-0.25579	114	-0.11616
0.119809	-0.69378	0.085168	-0.4893	-0.24366	-0.6959	0.480181	99	-0.25824

> 또한 부스팅 모델을 사용하기 때문에 높은 성능과 안정적인 모델 훈련을 위하여, 위와 같이 반응 변수인 SBP_B를 제외한 모든 설명변수들을 표준화를 통해 데이터의 분포를 정규화 했습니다.

모델 훈련

<랜덤포레스트>

```
Random Forest

9706 samples
  17 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 7765, 7765, 7764, 7766, 7764
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
  2     9.751533  0.5626511  7.497529
  3     9.761943  0.5617755  7.503408
  4     9.767510  0.5614679  7.505016
  5     9.771943  0.5612191  7.513006
  6     9.773864  0.5610907  7.509791

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

```
> # 예측 및 성능 평가
> rf_predictions <- predict(rf_model, newdata = test_data)
> rf_rmse <- sqrt(mean((rf_predictions - y_test)^2))
> rf_r2 <- cor(rf_predictions, y_test)^2
> cat("랜덤포레스트 RMSE:", rf_rmse, "\n")
랜덤포레스트 RMSE: 9.368118
> cat("랜덤포레스트 R²:", rf_r2, "\n")
랜덤포레스트 R²: 0.5898787
```

> 이번 학습에서는 하이퍼파라미터 튜닝을 통해 트리의 개수(ntree)를 500으로 고정하고, 각 트리에서 랜덤하게 선택할 변수 개수(mtry)를 2, 3, 4, 5, 6으로 설정하여 최적의 값을 탐색하였습니다. 또한, 5-fold 교차검증을 활용하여 데이터를 다섯 개로 나누어 학습과 검증을 반복하며, 데이터 편향을 최소화하고 모델의 일반화 성능을 평가하였습니다. 교차검증 결과 가장 낮은 RSME는 9.751이고, R^2=0.5627로 그 결과 최적의 mtry =2 이다. 테

스트 데이터 셋의 R^2은 대략 0.59로 종속 변수의(SBP_B) 변동성의 약 59%를 설명했습니다. 0.6에 가깝지만, 결국에는 0.6을 넘지 못해 일부 복잡한 패턴은 설명이 불가할 수 있습니다.

<GBM>

```
> # 최적 트리 수
> best_iter <- gbm.perf(gbm_model, method = "cv")
> cat("GBM 최적 트리 수:", best_iter, "\n")
GBM 최적 트리 수: 83
>
> # 예측 및 성능 평가
> gbm_predictions <- predict(gbm_model, newdata = test_data, n.trees = best_iter)
> gbm_rmse <- sqrt(mean((gbm_predictions - y_test)^2))
> gbm_r2 <- cor(gbm_predictions, y_test)^2
> cat("GBM RMSE:", gbm_rmse, "\n")
GBM RMSE: 9.23536
> cat("GBM R²:", gbm_r2, "\n")
GBM R²: 0.6015096
```

> GBM(Gradient Boosting Machine) 모델은 부스팅 알고리즘을 활용한 모델로서, 테스트 데이터에서 RMSE 9.235와 R^2 0.6015의 성능을 보이며 랜덤포레스트보다 더 높은 예측 정확도와 설명력을 보여주었습니다. 교차검증을 통해 최적 트리 개수(83개)를 선택하여 과적합을 방지했고, 데이터의 약 60% 변동성을 설명할 수 있었습니다. 이는 혈압(SBP_B)과 관련된 주요 변수와 상호작용을 잘 학습한 결과로, 모델이 복잡한 비선형성을 효과적으로 처리했음을 나타냅니다. 그러나 0.6을 간신히 넘어 마찬가지로 일부 패턴에 대해 설명이 불가능할 수 있습니다.

<XGBoost>

```
>
> xgb_predictions <- predict(xgb_model, newdata = as.matrix(x_test))
> xgb_rmse <- sqrt(mean((xgb_predictions - y_test)^2))
> xgb_r2 <- cor(xgb_predictions, y_test)^2
> cat("XGBoost RMSE:", xgb_rmse, "\n")
XGBoost RMSE: 9.218533
> cat("XGBoost R²:", xgb_r2, "\n")
XGBoost R²: 0.6028955
```

> XGBoost 모델은 테스트 데이터에서 RMSE 9.219와 R^2 0.603의 성능을 기록하며, 데이터의 약 60% 변동성을 설명할 수 있었습니다. 이 모델은 부스팅 알고리즘을 활용하여 최대 깊이 3의 트리를 100번 반복 학습하며 학습률(eta)을 0.1로 설정하여 과적합을 방

지하였습니다. XGBoost는 GBM과 비슷한 성능을 보였으나, 추가적인 하이퍼파라미터 튜닝이나 변수 생성 및 선택을 통해 성능을 더욱 개선할 가능성이 있습니다. 이는 데이터의 복잡한 비선형성과 상호작용을 효과적으로 학습했음을 나타내며, 혈압 예측에서 유의미한 모델로 평가됩니다.

<LightGBM>

```
> lgb_rmse <- sqrt(mean((lgb_predictions - y_test)^2))
> lgb_r2 <- cor(lgb_predictions, y_test)^2
> cat("LightGBM RMSE:", lgb_rmse, "\n")
LightGBM RMSE: 9.221304
> cat("LightGBM R²:", lgb_r2, "\n")
LightGBM R²: 0.6031961
```

> LightGBM 모델은 테스트 데이터에서 RMSE 9.221과 R^2 0.603의 성능을 기록하며, 데이터의 약 60.32% 변동성을 설명할 수 있었습니다. 이 모델은 learning rate 0.1, num_leaves 31, 반복 횟수nrounds 100으로 설정하여 훈련했습니다. XGBoost와 유사한 성능을 보이지만, LightGBM 특유의 빠른 계산 능력을 통해 효과적인 비선형성과 상호작용 학습이 가능합니다.

<앙상블>

```
> #부스팅 평균치 앙상블
> final_predictions <- (gbm_predictions + lgb_predictions + xgb_predictions) / 3
> final_r2 <- cor(final_predictions, y_test)^2
> cat("앙상블 모델 R²:", final_r2, "\n")
앙상블 모델 R²: 0.6056908
```

```
> #랜덤포레스트+부스팅
>
> final_predictions <- (rf_predictions + gbm_predictions + xgb_predictions + lgb_predictions) / 4
> final_r2 <- cor(final_predictions, y_test)^2
> cat("앙상블 R²:", final_r2, "\n")
앙상블 R²: 0.6047978
```

> 앙상블 모델은 개별 부스팅 모델(GBM, XGBoost, LightGBM)의 예측값을 평균한 경우 R^2 0.606을 기록하며, 데이터 변동성의 약 60.6%를 설명하였습니다. 랜덤포레스트와 부스팅 모델(GBM, XGBoost, LightGBM)을 결합한 경우 R^2 0.605로 약간의 성능 감소를 보였지만, 여전히 안정적인 예측력을 유지했습니다.

<선형 회귀>

Traindata set과 testdata set 으로 데이터를 나누었다. 데이터를 정규화하고 상관 행렬 시각화를 통해 변수간의 상관관계와 다중공선성에 대해 살펴보았다. Stepwise를 통해 변수를 선택하였다.

또한, 주성분 분석을 활용하여 선형 회귀 모델을 만들어 예측을 하였다.

Stepwise model의 R^2 값 : 0.562

Pca linear model의 R^2 값 : 0.284

```
# 데이터 로드 및 라이브러리 로드

nData <- read.csv("C:₩₩Users₩₩yorion01₩₩Documents₩₩00_Current₩₩2024-2_데이터마이닝₩₩DMDataByChanho.csv", header = TRUE)

str(nData)


library(caret)

library(glmnet)

library(MASS)

library(corrplot)


dt <- nData

dt <- dt[!is.na(dt$FEV1),]


colnames(dt)

# 변수 선택

x_vars <- colnames(dt[,-c(1,2,12,23,24)])

y_var <- "SBP_B"


set.seed(24120502)

trainIndex <- createDataPartition(dt$SBP_B, p = 0.7, list = FALSE)

trainData <- dt[trainIndex,]

testData <- dt[-trainIndex,]


# 데이터 정규화

preProc <- preProcess(trainData[, x_vars], method = c("center", "scale"))
```

```r
trainDataNorm <- predict(preProc, trainData[, x_vars])
testDataNorm <- predict(preProc, testData[, x_vars])
trainData <- data.frame(SBP_B = trainData$SBP_B, trainDataNorm)
testData <- data.frame(SBP_B = testData$SBP_B, testDataNorm)

# 상관 행렬 시각화
corr_matrix <- cor(trainData[, x_vars])
corrplot(corr_matrix, method = "circle", type = "upper", title = "Correlation Matrix", tl.cex = 0.5)
#시각화

# 변수 선택 (Stepwise Selection)
linear_reg_model <- lm(SBP_B ~ ., data = trainData[,c(x_vars,y_var)])
summary(linear_reg_model)
linear_pred = predict(linear_reg_model,newdata = testData[,x_vars])
length(linear_pred)
length(testData$SBP_B)
1-sum((linear_pred - testData$SBP_B)^2) / sum((testData$SBP_B - mean(testData$SBP_B))^2)
eval_model(testData$SBP_B,linear_pred)
stepwise_model <- stepAIC(linear_reg_model, direction = "both")

# 예측 수행 및 평가 함수 정의
eval_model <- function(true_values, predicted_values) {
  residuals <- true_values - predicted_values
  r_squared <- 1 - sum( residuals ^ 2 ) / sum( (true_values - mean(true_values))^2)
  rmse <- sqrt(mean(residuals^2))
  mae <- mean(abs(residuals))

  return(list(R2 = r_squared, RMSE = rmse, MAE = mae))
}

# 예측 수행 및 평가
```

```
stepwise_pred <- predict(stepwise_model, newdata = testData[,x_vars])
stepwise_eval <- eval_model(testData$SBP_B, stepwise_pred)
print(stepwise_eval)

# 주성분 분석 (PCA) 수행
pca_model <- prcomp(trainData[, x_vars], scale. = TRUE)
train_pca <- predict(pca_model, trainData[, x_vars])
test_pca <- predict(pca_model, testData[, x_vars])
pca_trainData <- data.frame(SBP_B = trainData$SBP_B, train_pca[, 1:10])
pca_testData <- data.frame(SBP_B = testData$SBP_B, test_pca[, 1:10])

# PCA를 활용한 선형 회귀 모델
pca_linear_model <- lm(SBP_B ~ ., data = pca_trainData)
pca_linear_pred <- predict(pca_linear_model, newdata = pca_testData)
pca_linear_eval <- eval_model(testData$SBP_B, pca_linear_pred)
print(pca_linear_eval)
```

<Ridge Model>

과적합을 방지하고 모델의 일반화 성능을 향상시키는 정규화 회귀 기법인 릿지 회귀 모델을 이용하여 예측 성능을 구하였다. 이 릿지 모델의 성능을 올리기 위해 최적의 lambda값을 구하였고, 교차 검능을 하였다.

첫번째 Ridge Model R^2 : 0.543

튜닝한 Ridge Model R^2 : 0.547

```
# 릿지 회귀
x_train = as.matrix(trainData[, x_vars])
y_train = trainData$SBP_B
x_test = as.matrix(testData[, x_vars])
y_test = testData$SBP_B

ridge_model= glmnet(x_train, y_train, alpha = 0)
```

```
optimal_lambda = cv.glmnet(x_train, y_train, alpha = 0)$lambda.min


ridge_pred = predict(ridge_model, s = optimal_lambda, newx = x_test)

ridge_eval = eval_model(testData$SBP_B, ridge_pred)


print(ridge_eval)

# 10-폴드 교차 검증 설정

ctrl <- trainControl(method = "cv", number = 10)

# 릿지 회귀 모델 학습 및 하이퍼파라미터 튜닝

set.seed(123)

ridge_tune = train(

  x = x_train,

  y = y_train,

  method = "glmnet",

  tuneGrid = expand.grid(alpha = 0, lambda = seq(0.001, 0.1, by = 0.001)),   # alpha = 0 for ridge regression

  trControl = ctrl,

  metric = "RMSE"

)


# 최적의 하이퍼파라미터 확인

best_lambda = ridge_tune$bestTune$lambda

print(best_lambda)


# 최적의 하이퍼파라미터로 모델 생성

ridge_model = glmnet(x_train, y_train, alpha = 0, lambda = best_lambda)


# 예측 수행

ridge_pred = predict(ridge_model, s = best_lambda, newx = x_test)


# 모델 평가

ridge_eval = evaluate_model(y_test, ridge_pred)
```

```
print(ridge_eval)
```

<SVM model>

Support vector machine model을 생성한다. 이 모델은 약 52%의 시험 데이터를 설명한다.

```
> #서포트 벡터 회귀 모델델

> svm_model = svm(SBP_B ~ ., data = trainData[, c(y_vars, x_vars)])

> svm_pred = predict(svm_model, newdata = testData[,x_vars])

> eval_model(testData$SBP_B,svm_pred)

$R2

[1] 0.516621
```

4가지 모델 중 최적의 모형은 Ridge Regression이다.

```
> results = data.frame(Model = c("Linear Regression","Pca Linear Regression","Ridge Regressio
n","SVM"),
+                      R2 = c(stepwise_eval$R2, pca_linear_eval$R2,ridge_eval$R2,svm_eval$R2)
+                      )
> print(results)
                  Model        R2
1     Linear Regression 0.5373614
2 Pca Linear Regression 0.5456958
3      Ridge Regression 0.5461090
4                   SVM 0.5166210
```

모델 성능 비교 요약:

```
> print(performance_summary)
                   Model     RMSE        R2
1          Random Forest 9.368118 0.5898787
2                    GBM 9.235360 0.6015096
3                XGBoost 9.218533 0.6028955
4               LightGBM 9.221304 0.6031961
5      Boosting Ensemble 9.199421 0.6056908
6 RF + Boosting Ensemble 9.199421 0.6047978
```

> 최종적으로 개별 부스팅 모델(GBM, XGBoost, LightGBM)의 예측값을 평균하여, 데이터의 복잡한 패턴과 상호작용을 효과적으로 학습한 결과 R2=0.6057로 가장 높은 설명력을 기록했습니다. 또한, 가장 작은 RMSE를 가져 둘 간에 균형을 고려하여 최적의 모델로 판단됩니다.

- SBP_B 분석 최종 모형: Boosting Ensemble

> 사용했던 10가지 모델들 중에서 부스팅을 사용한 모델들만 결정계수가 0.6을 넘어섰습니다. 사실 부스팅모델도 하이퍼파라미터 튜닝을 통해 모델 성능을 극대화하고자 했으나, 0.6을 넘지 못하고 0.55~0.57사이의 값으로 분포했었습니다. 그래서 모델분석 후 변수중요도 분석 결과에서 중요도가 비교적 낮게 나왔던 흡연여부(SMOK_B)와 음주여부(ALCO_B)를 제거했습니다. 그리고 설명변수를 혈압에 영향을 미칠 것으로 가정되는 주요 생리학적 변수(AGE_B, DBP_B, WAIST_B 등)와 중요도가 어느정도 있던 변수들을 기반으로 구성하였고, 이완기혈압과 나이, 체중과 같은 변수 간의 상호작용 효과(DBP_AGE, WAIST_AGE, DBP_WT)를 추가하여 변수 간 관계를 심화시켰습니다. 또한, 비선형적 특성을 반영하기 위해 제곱(DBP_B_sq, WAIST_B_sq)이나 로그 변환(log_WAIST_B, log_DBP_B), 제곱근 변환(sqrt_AGE_B) 등을 적용하여 데이터 분포를 정규화하고 모델 학습을 안정화했습니다. 마지막으로, BMI와 TG_FBS_ratio와 같은 대사적 상태를 나타내는 파생 변수를 포함하여, 혈압 변화와 관련된 중요한 생리적 특성을 반영했습니다. 최종적으로 부스팅모델들 중 Boosting Ensemble 모델이 최적의 RMSE와 R2값을 보이기에 최종모델로 선택했습니다.

<R 코드>

```
library(dplyr)

# 원본 데이터 불러오기
data <- read.csv("c:/Temp/data.csv")

# 원본 데이터에서 가져온 변수 선택
data <- data %>%
  select(AGE_B, DBP_B, WAIST_B, WT_B, CHO_B, TG_B, FBS_B, SBP_B)

# 생성된 상호작용 변수 추가
data$DBP_AGE <- data$DBP_B * data$AGE_B                    # 이완기혈압 × 나이
data$WAIST_AGE <- data$WAIST_B * data$AGE_B                # 허리둘레 × 나이
data$DBP_WT <- data$DBP_B * data$WT_B                      # 이완기혈압 × 체중

# 비선형 변수 변환
data$DBP_B_sq <- data$DBP_B^2                              # 이완기혈압의 제곱
data$WAIST_B_sq <- data$WAIST_B^2                          # 허리둘레의 제곱
data$log_WAIST_B <- log(data$WAIST_B + 1)                  # 허리둘레의 로그 변환
data$log_DBP_B <- log(data$DBP_B + 1)                      # 이완기혈압의 로그 변환
data$sqrt_AGE_B <- sqrt(data$AGE_B)                        # 나이의 제곱근 변환

# 파생 변수 추가
data$BMI <- data$WT_B / (data$WAIST_B^2)                   # 체질량지수 (체중 ÷ 허
```

리둘레²)

data$TG_FBS_ratio <- data$TG_B / (data$FBS_B + 1) # 중성지방 ÷ 공복 혈당 비율

데이터 표준화 (평균 0, 표준편차 1로 변환)

scaled_data <- data

scaled_data[, names(data) != "SBP_B"] <- scale(data[, names(data) != "SBP_B"])

head(scaled_data)

데이터 저장

write.csv(scaled_data, "c:/Temp/final_data.csv", row.names = FALSE)

패키지 불러오기

library(caret)

library(randomForest)

library(gbm)

library(xgboost)

library(lightgbm)

데이터 불러오기

data <- read.csv("c:/Temp/final_data.csv")

훈련/테스트 데이터셋 분리

set.seed(123)

train_index <- createDataPartition(data$SBP_B, p = 0.8, list = FALSE)

```
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

# 독립 변수와 종속 변수 분리
x_train <- train_data[, -which(names(train_data) == "SBP_B")]
y_train <- train_data$SBP_B
x_test <- test_data[, -which(names(test_data) == "SBP_B")]
y_test <- test_data$SBP_B

# 랜덤포레스트 하이퍼파라미터 튜닝
set.seed(123)
rf_grid <- expand.grid(
  mtry = c(2, 3, 4, 5, 6) # mtry 값 범위
)

rf_control <- trainControl(
  method = "cv",          # 교차 검증
  number = 5,             # 5-폴드
  search = "grid"         # 그리드 서치
)

rf_model <- train(
  SBP_B ~ .,
  data = train_data,
  method = "rf",
```

```
  metric = "RMSE",
  tuneGrid = rf_grid,
  trControl = rf_control
)

# 최적 모델 확인
print(rf_model)

# 예측 및 성능 평가
rf_predictions <- predict(rf_model, newdata = test_data)
rf_rmse <- sqrt(mean((rf_predictions - y_test)^2))
rf_r2 <- cor(rf_predictions, y_test)^2
cat("랜덤포레스트 RMSE:", rf_rmse, "₩n")
cat("랜덤포레스트 R²:", rf_r2, "₩n")



##########


# GBM 모델 훈련
set.seed(123)
gbm_model <- gbm(
  SBP_B ~ .,
  data = train_data,
```

```
  distribution = "gaussian",
  n.trees = 100,
  interaction.depth = 3,
  shrinkage = 0.1,
  cv.folds = 5,
  verbose = FALSE
)

# 최적 트리 수
best_iter <- gbm.perf(gbm_model, method = "cv")
cat("GBM 최적 트리 수:", best_iter, "₩n")

# 예측 및 성능 평가
gbm_predictions <- predict(gbm_model, newdata = test_data, n.trees = best_iter)
gbm_rmse <- sqrt(mean((gbm_predictions - y_test)^2))
gbm_r2 <- cor(gbm_predictions, y_test)^2
cat("GBM RMSE:", gbm_rmse, "₩n")
cat("GBM R²:", gbm_r2, "₩n")



#####
```

```
####### XGBoost #######
dtrain <- xgb.DMatrix(data = as.matrix(x_train), label = y_train)
dtest <- xgb.DMatrix(data = as.matrix(x_test), label = y_test)
xgb_model <- xgboost(
  data = dtrain,
  objective = "reg:squarederror",
  nrounds = 100,
  max_depth = 3,
  eta = 0.1,
  verbose = 0
)

xgb_predictions <- predict(xgb_model, newdata = as.matrix(x_test))
xgb_rmse <- sqrt(mean((xgb_predictions - y_test)^2))
xgb_r2 <- cor(xgb_predictions, y_test)^2
cat("XGBoost RMSE:", xgb_rmse, "₩n")
cat("XGBoost R²:", xgb_r2, "₩n")



######



# LightGBM 데이터 변환
lgb_train <- lgb.Dataset(data = as.matrix(x_train), label = y_train)
lgb_test <- lgb.Dataset(data = as.matrix(x_test), label = y_test)
```

```
# LightGBM 모델 훈련
set.seed(123)
params <- list(
  objective = "regression",
  learning_rate = 0.1,
  num_leaves = 31,
  metric = "rmse"
)

lgb_model <- lgb.train(
  params = params,                # 하이퍼파라미터 설정
  data = lgb_train,               # 훈련 데이터
  nrounds = 100,                  # 부스팅 반복 수
  valids = list(test = lgb_test), # 검증 데이터
  verbose = -1                    # 로그 최소화
)

# 예측 및 성능 평가
lgb_predictions <- predict(lgb_model, newdata = as.matrix(x_test))

# 성능 평가
lgb_rmse <- sqrt(mean((lgb_predictions - y_test)^2))
lgb_r2 <- cor(lgb_predictions, y_test)^2
cat("LightGBM RMSE:", lgb_rmse, "₩n")
```

```
cat("LightGBM R²:", lgb_r2, "₩n")
```

```
#####
#부스팅 평균치 앙상블
final_predictions <- (gbm_predictions + lgb_predictions + xgb_predictions) / 3
final_r2 <- cor(final_predictions, y_test)^2
cat("앙상블 모델 R²:", final_r2, "₩n")
```

```
#####

#랜덤포레스트+부스팅

final_predictions <- (rf_predictions + gbm_predictions + xgb_predictions +
lgb_predictions) / 4
final_r2 <- cor(final_predictions, y_test)^2
cat("앙상블 R²:", final_r2, "₩n")

# 성능 요약
```

```
performance_summary <- data.frame(
  Model = c("Random Forest", "GBM", "XGBoost", "LightGBM", "Boosting Ensemble", "RF + Boosting Ensemble"),
  RMSE = c(rf_rmse, gbm_rmse, xgb_rmse, lgb_rmse,
           sqrt(mean((final_predictions - y_test)^2)),  # Boosting 앙상블 RMSE
           sqrt(mean((final_predictions - y_test)^2))    RMSE
  ),
  R2 = c(rf_r2, gbm_r2, xgb_r2, lgb_r2,
         cor((gbm_predictions + lgb_predictions + xgb_predictions) / 3, y_test)^2,  # Boosting 앙상블 R^2
         cor((rf_predictions + gbm_predictions + xgb_predictions + lgb_predictions) / 4, y_test)^2  # RF + Boosting 앙상블 R^2
  )
)

# 출력
cat("모델 성능 비교 요약:₩n")
print(performance_summary)
```