다변량분석 5번째 과제

# Assignment 5

# ANN


학과 산업경영공학부

학번 2016170863

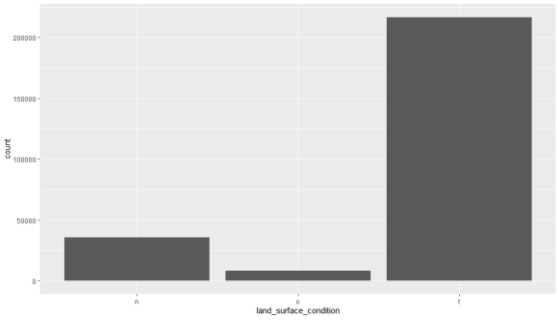
이름 추창욱

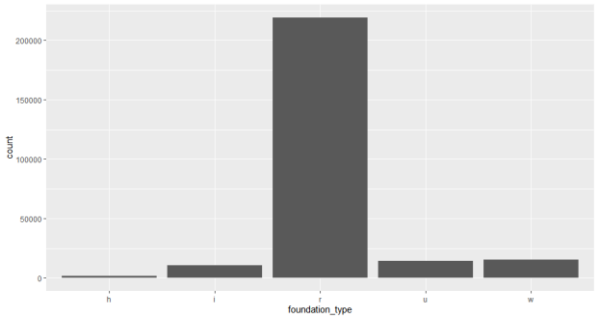
**전체 데이터 셋에 대해 다음 물음에 답하시오.**

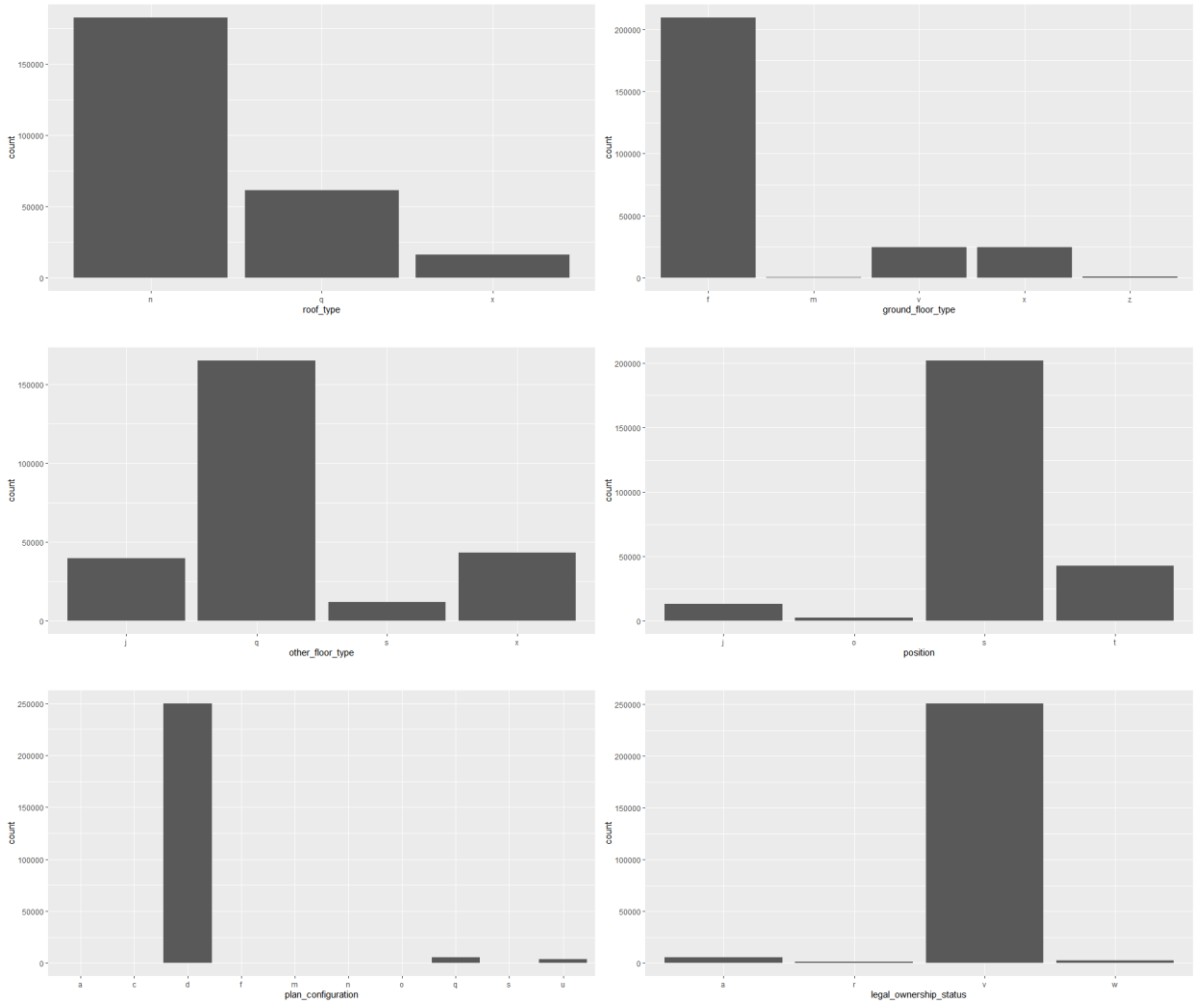
**[Q1] 입력 변수의 속성이 numeric 이 아닌 변수들을 모두 확인하고, 각 변수들의 요인 값들에 대한 Bar chart 를 도시하시오.**

```
$ land_surface_condition                : chr  "t" "o" "t" "t" ...
$ foundation_type                       : chr  "r" "r" "r" "r" ...
$ roof_type                             : chr  "n" "n" "n" "n" ...
$ ground_floor_type                     : chr  "f" "x" "f" "f" ...
$ other_floor_type                      : chr  "q" "q" "x" "x" ...
$ position                              : chr  "t" "s" "t" "s" ...
$ plan_configuration                    : chr  "d" "d" "d" "d" ...
$ has_superstructure_adobe_mud          : int  1 0 0 0 1 0 0 0 0 0 ...
$ has_superstructure_mud_mortar_stone   : int  1 1 1 1 0 1 1 0 1 0 ...
$ has_superstructure_stone_flag         : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_superstructure_cement_mortar_stone: int  0 0 0 0 0 0 0 0 0 0 ...
$ has_superstructure_mud_mortar_brick   : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_superstructure_cement_mortar_brick: int  0 0 0 0 0 0 0 1 0 1 ...
$ has_superstructure_timber             : int  0 0 0 1 0 0 0 1 1 0 ...
$ has_superstructure_bamboo             : int  0 0 0 1 0 0 0 0 0 0 ...
$ has_superstructure_rc_non_engineered  : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_superstructure_rc_engineered      : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_superstructure_other              : int  0 0 0 0 0 0 0 0 0 0 ...
$ legal_ownership_status                : chr  "v" "v" "v" "v" ...
$ count_families                        : int  1 1 1 1 1 1 1 1 1 1 ...
$ has_secondary_use                     : int  0 0 0 0 0 1 0 0 0 0 ...
$ has_secondary_use_agriculture         : int  0 0 0 0 0 1 0 0 0 0 ...
$ has_secondary_use_hotel               : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_secondary_use_rental              : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_secondary_use_institution         : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_secondary_use_school              : int  0 0 0 0 0 0 0 0 0 0 ...
$ has_secondary_use_industry            : int  0 0 0 0 0 0 0 0 0 0 ...
```

현재 가지고 있는 데이터의 변수들 중 numeric 형태가 아닌 값으로 이루어진 변수는 위에서 보는 바와 같이 총 8개라는 것을 확인 할 수 있다. 각각의 변수들에 바 플롯을 그려본 결과는 다음과 같다.






현재 위 structure을 보았을 때 숫자형이 아닌 변수들에 대해 바 플롯을 그려본 결과는 위와 같이 나왔으며 대부분 어느 특정 한 요인에 치우쳐져 있다는 것을 확인 할 수 있었다.

**[Q2] [Q1]에서 선택된 변수들에 대해 1-of-C coding (1-hot encoding) 방식을 통해 명목형(요인형) 변수를 범주의 개수만큼의 이진형(binary) 변수들로 구성되는 dummy variable 을 생성하시오.**

현재 위에서 선택한 numeric이 아닌 변수들에 대해 원핫인코딩을 진행해 본 결과, 변수의 개수가 68개로 늘었고, 더 이상 character타입의 데이터가 존재 하지 않는 다는 것을 확인 할 수 있었다.

**[Q3] Training 및 Validation 데이터셋을 바탕으로 실습 시간에 사용한 "nnet" package 를 사용하여 다음 hyper-parameter 조합들에 대한 Grid search 를 수행하여 최적의 조합을 찾아보시오 (합계 최소 21 가지 이상 탐색). 아래 두 가지 이외의 hyper-parameter 는 nnet package 의 default value 를 사용하시오. 성능이 가장 우수한 조합과 가장 열등한 조합과의 Accuracy 와 BCR 의 차이는 얼마인가?**

● **Number of hidden node: 최소 7 가지 이상 탐색**

● **maxit: 최소 3 가지 이상 탐색**

```
for(i in 1:nrow(a)){
  cat("Training ANN: the number of hidden nodes:",a$nH[i],",maxit:",a$maxit[i],"\n")
  evaluation <- c()

  # Training the model
  trn_input <- ANN_trn_input
  trn_target <- ANN_trn_target
  tmp_nnet <- nnet(trn_input,trn_target,size = a$nH[i],maxit = a$maxit[i],silent = TRUE,MaxNWts = 10000)

  #Evaluate the model
  val_input <- ANN_val_input
  val_target <- ANN_val_target

  real <- max.col(val_target)
  pred <- max.col(predict(tmp_nnet,val_input))
  evaluation <- rbind(evaluation,cbind(real,pred))
  #Confusion Matrix
  cfm <- matrix(0,nrow = 3, ncol = 3)
  cfm[1,1] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 1))
  cfm[1,2] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 2))
  cfm[1,3] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 3))
  cfm[2,1] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 1))
  cfm[2,2] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 2))
  cfm[2,3] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 3))

  cfm[3,1] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 1))
  cfm[3,2] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 2))
  cfm[3,3] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 3))

  val_perf[i,1] <- a$nH[i]
  val_perf[i,2] <- a$maxit[i]
  val_perf[i,3:4] <- t(perf_eval_multi(cfm))
}
```

저는 히든 노드는 3~30까지 4씩 증가시켜갔고, maxit은 100~300까지 100씩 증가 시켜 총 21가지의 조합을 만들어 돌려보았다. 내가 설정한 하이퍼 파라미터의 모든 조합을 구현하기 위해 for문을 구축하였고, confusion matrix를 설정하는 중 아예 0으로 나오면 에러가 뜨는 경우가 발생하였다. 슬랙에 비슷한 질문이 올라온 것을 확인하여 이를 참고하였고, 그 결과 3바이 3 매트릭스로 만들어 각 자리에 대해 손수 조건문을 넣어 돌려주었다.

위의 코드를 수행하는데 걸린 시간은

```
   사용자   시스템 elapsed
8303.95    7.15 8329.28
```

이였다.

우수한 조합과 열등한 조합들에 관해 BCR을 기준으로 상위 그리고 하위 몇 개를 캡처해서 가장 좋은 조합과 열등한 조합을 비교해 보았다.

| | nH | Maxit | ACC | BCR |
|---|---|---|---|---|
| [1,] | 7 | 300 | 0.64068 | 0.5510233 |
| [2,] | 27 | 100 | 0.63124 | 0.5428527 |
| [3,] | 27 | 300 | 0.66224 | 0.5201836 |
| [4,] | 15 | 200 | 0.66000 | 0.5154207 |
| [5,] | 11 | 300 | 0.66264 | 0.5152665 |
| [6,] | 27 | 200 | 0.66024 | 0.5133231 |
| [7,] | 19 | 200 | 0.65782 | 0.5130807 |
| [8,] | 19 | 300 | 0.65744 | 0.5084484 |
| [9,] | 3 | 300 | 0.62344 | 0.5032405 |

| | nH | Maxit | ACC | BCR |
|---|---|---|---|---|
| [1,] | 3 | 100 | 0.59102 | 0.0000000 |
| [2,] | 19 | 100 | 0.57208 | 0.0000000 |
| [3,] | 23 | 200 | 0.57208 | 0.0000000 |
| [4,] | 15 | 300 | 0.57208 | 0.0000000 |
| [5,] | 23 | 300 | 0.64832 | 0.0000000 |
| [6,] | 3 | 200 | 0.62098 | 0.4391983 |
| [7,] | 11 | 100 | 0.63510 | 0.4716161 |
| [8,] | 7 | 200 | 0.63956 | 0.4874571 |
| [9,] | 23 | 100 | 0.64384 | 0.4910388 |

파란 박스는 성능이 높게 나오는 상위 조합들을 추출한것이며 빨간 박스는 그 반대의 결과를 나타내는데, 위에서 알 수 있다시피, 가장 높은 BCR은 0.5510233 가장 낮은 BCR값을 가지는 조합은 0이였다는 것을 알 수있다. 또한 ACC의 경우 가장 BCR이 좋다고 판단되는 조합 기준 0.64068이라는 것을 확인 할 수 있는데, 이는 의외로 별로 큰 차이가 없다고 느껴졌다. 왜냐하면, BCR이 0으로 나온 조합들 기준으로 가장 낮은 정확도의 값은 0.57208이였기 때문이다. 좋은 성능과 나쁜 성능의 차이는 위의 두 박스를 번갈아가며 한눈에 비교를 할 수 있었다.

**[Q4] [Q3]에서 선택된 최적의 모델 구조에 nnet의 rang 옵션(default value = 0.7)을 세 가지 이상 변경해 가면서 성능 변화를 살펴보시오. [Q3]의 방식과 마찬가지로 Training/Validation 데이터 셋을 사용하여 탐색하시오.**

Q3에서 최적의 파라미터 조합이라 함은 현재 가장 BCR이 가장 높은 값을 가지는 조합이라고 생각하였다. 그렇기에 다음과 같은 코드를 실행해 주어 선택된 최적의 하이퍼 파라미터를 적용시켜주었다.

```
best_nH <- best_val_perf[1,1]
best_maxit <- best_val_perf[1,2]

rang = c(0.25,0.5,0.9)
val_perf_rang = matrix(0,length(rang),3)
```

또한, rang의 값은 0.25, 0.5, 0.9라는 세가지의 후보군으로 설정하였고, Q3에서 수행한 코드와 동일하게 수행하여 주었다.

| | rang | ACC | BCR |
|---|---|---|---|
| [1,] | 0.3 | 0.64990 | 0.5253607 |
| [2,] | 0.7 | 0.63416 | 0.4448523 |
| [3,] | 0.5 | 0.57208 | 0.0000000 |

Rang의 값을 0.3, 0.7, 0.5 defatult값을 포함하여 총 3개의 다른 값에 대하여 정확도와 BCR을 구해본 결과는 위와 같았다. 위에서 이전에 다른 값으로 수행을 하였을 때 0이라는 값이 나와 Confusion matrix연산 과정에서 에러가 떴기 때문에 3번 문제에서 사용한 방법과 같이 해결하였

다. 우선, 위에 있는 표는 최종 나온 결과표이고 각각에 대해 해석을 해보자면 0.3일 때 ACC와 BCR이 둘 다 가장 높다는 것을 확인 하였고, 0.7일 때 단순 정확도와 BCR의 값이 조금 적었으며, 0.5일 때는 ACC와 BCR의 관점에서 보았을 때, 성능이 높지 않다는 것을 확인 할 수 있었다.

```
for(i in 1:length(rang)){
  evaluation <- c()

  trn_input <- ANN_trn_input
  trn_target <- ANN_trn_target
  tmp_nnet <- nnet(trn_input,trn_target,size = best_nH, maxit = best_maxit, rang = rang[i], MaxNWts = 10000)

  val_input <- ANN_val_input
  val_target <- ANN_val_target
  evaluation <- rbind(evaluation, cbind(max.col(val_target),
                                        max.col(predict(tmp_nnet,val_input))))

  cfm <- matrix(0,nrow = 3, ncol = 3)
  cfm[1,1] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 1))
  cfm[1,2] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 2))
  cfm[1,3] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 3))
  cfm[2,1] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 1))
  cfm[2,2] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 2))
  cfm[2,3] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 3))
  cfm[3,1] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 1))
  cfm[3,2] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 2))
  cfm[3,3] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 3))

  val_perf_rang[i,1] <- rang[i]
  val_perf_rang[i,2:3] <- t(perf_eval_multi(cfm))
}
```

이는 이 과정을 위해 돌린 반복문 코드이다.

결론적으로 내가 설정한 rang 후보 값 세가지 중 최적의 rang 값은 0.3이라는 것을 확인 할 수 있었다.

**[Q5] [Q3]에서 확인된 최적의 hidden node의 수와 최적의 maxit, 그리고 [Q4]에서 확인된 가장 우수한 rang option으로 ANN 모델 구조를 확정하고 Training dataset과 Validation dataset을 결합한 데이터셋에 대해 학습을 수행하시오. 학습이 완료된 ANN 모델을 Test dataset에 적용하여 Accuracy와 BCR을 살펴보시오. 동일한 과정을 10회 반복 수행하여 수행 회차에 따라 정확도의 변동성은 어느 정도 되는지 확인 하시오.**

현재 최적의 하이퍼 파라미터 값들은 히든 노드는 7, maxit은 300, rang은 0.3이라는 것을 확인 하였고, 이를 가지고 ANN 구조를 training과 validation 셋을 결합한 데이터셋에 대해 학습을 수행하였다. 우선 총 학습을 하는데 걸린 시간은 5875.4 초 였다. 이 코드를 돌림으로써 각 수행마다 어떤 식으로 변하는 지를 확인 하기 위해 결과를 확인 해본 결과는 다음과 같았다.

| | ACC | BCR |
|---|---|---|
| [1,] | 0.3072301 | 0.3072301 |
| [2,] | 0.2188899 | 0.2188899 |
| [3,] | 0.3072301 | 0.3072301 |
| [4,] | 0.2188899 | 0.2188899 |
| [5,] | 0.3072301 | 0.3072301 |
| [6,] | 0.2188899 | 0.2188899 |
| [7,] | 0.3072301 | 0.3072301 |
| [8,] | 0.2188899 | 0.2188899 |
| [9,] | 0.3072301 | 0.3072301 |
| 10,] | 0.2188899 | 0.2188899 |

수행 회차에 따라 비교적 같은 값 두가지가 반복되며 추출된 것을 확인 할 수 있었다. 정확도와 BCR을 보여주고 있다는 것을 확인 할 수 있었다. 40%는 넘지 않으며, 대부분 20~40% 사이의 값을 산출 하고 있다. 가장 높은 ACC와 BCR의 값은 각각 0.3072301이라는 것을 확인 할 수 있었고, 이를 perf_summary table에 삽입하여 주었다.

| | ACC | BCR |
|---|---|---|
| ANN | 0.3072301 | 0.3072301 |

아래는 위의 결과를 산출하기 위해 이용된 코드이다.

```
ptm <- proc.time()
for(i in 1:10){
  evluation <- c()

  #Train the model
  Final_nnet <- nnet(nnet_trn, nnet_target, size = best_nH, maxit = best_maxit, rang = best_rang, MaxNWts = 10000)

  #Test and evaluate the model
  evaluation <- rbind(evaluation, cbind(max.col(tst_target), max.col(predict(Final_nnet,tst_input))))

  Final_cm <- matrix(0,nrow = 3, ncol = 3)
  Final_cm[1,1] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 1))
  Final_cm[1,2] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 2))
  Final_cm[1,3] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 3))
  Final_cm[2,1] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 1))
  Final_cm[2,2] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 2))
  Final_cm[2,3] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 3))
  Final_cm[3,1] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 1))
  Final_cm[3,2] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 2))
  Final_cm[3,3] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 3))

  val_perf_final[,1:2] <- t(perf_eval_multi(Final_cm))
}
proc.time() - ptm
```

**[Q6] [Q5]에서 사용된 모델 하이퍼파라미터를 고정시킨 상태에서 Genetic Algorithm을 이용한 변수 선택 모듈을 추가로 작성하여 최적으로 선택된 변수 조합을 확인해 보시오. Genetic Algorithm에서 변경 가능한 옵션의 경우 본인의 Assignment 3에서 설정했던 최적 값들을 그대로 사용하시오. Chromosome의 초기값을 다르게 설정하고 변수 선택을 3회 수행한 뒤, 동일한 변수 조합들이 선택되는지, 아니면 반복 회차마다 다른 변수들이 선택되는지 확인해보시오.**

과제 3을 수행하였을 당시 최적의 pop는 50, crossover rate는 0.01, mutation rate는 0.01(이는 0이 원래 최적의 값이었으나 그 당시에는 운이 좋았던 것으로 판단하여 작은 0.01 값을 선택), elitism은 1이였다는 것을 확인하였다. Nnet의 경우는 이전 문제 들에서 구한 최적의 하이퍼파라미터를 고정해주었다. 이를 토대로 chromosome의 초기 값을 다르게 하여 3회 수행하여 보았다. 이때 GA의 fit 함수로서 작성한 코드는 다음과 같다. 이때 사용한 지표값은 ACC를 사용하였는데 그 이유는 BCR의 경우는 이전에서도 보았듯이 아예 없는 값이 나오는 경우가 있었기에 ACC를 활용하도록 하였다.

```
fit_F1 <- function(string){

  sel_var_idx <- which(string == 1)
  # Use variables whose gene value is 1
  sel_x <- x[, sel_var_idx]
  xy <- data.frame(sel_x, y)
  # Training the model
  GA_nnet <- nnet(sel_x, y, size = 7, maxit = 300, rang = 0.3, MaxNWts = 10000)
  GA_nnet_prey <- predict(GA_nnet, tst_input)

  evaluation <- c()
  evaluation <- rbind(evaluation, cbind(max.col(tst_target), max.col(GA_nnet_prey)))
  Final_cm <- matrix(0,nrow = 3, ncol = 3)
  Final_cm[1,1] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 1))
  Final_cm[1,2] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 2))
  Final_cm[1,3] <- length(which(evaluation[,1] == 1 & evaluation[,2] == 3))
  Final_cm[2,1] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 1))
  Final_cm[2,2] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 2))
  Final_cm[2,3] <- length(which(evaluation[,1] == 2 & evaluation[,2] == 3))
  Final_cm[3,1] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 1))
  Final_cm[3,2] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 2))
  Final_cm[3,3] <- length(which(evaluation[,1] == 3 & evaluation[,2] == 3))
  GA_perf <- perf_eval_multi(Final_cm)
  return(GA_perf[1])
```

다음으로는 결과에 대한 해석을 진행해보도록 하겠다. 다음은 각각의 seed를 123, 12, 1234 였을 때, 선택된 변수들이다. 초반에 GA의 iteration을 100으로 하고 돌렸지만 8시간이 넘는 시간이 걸려 30으로 줄인 후 다시 실행해주었다.

```
> best_var_idx
 [1]  1  5  6  7  8  9 11 12 13 14 19 23 27 29 30 31 32 33 35 37 38 39 40 41 44 45 49 50 52 54 55 56 60 61 62
[36] 65 67 68
```

```
> best_var_idx
 [1]   1   4   6   7  10  11  13  17  20  21  24  28  29  31  35  36  37  39  40  41
[21]  42  43  45  46  49  50  52  54  60  61  65  67
```

```
> best_var_idx
 [1]   2   3   4   5   8  14  15  18  20  21  23  24  27  28  30  33  36  37  38  39
[21]  43  44  47  49  50  54  56  57  58  59  65
```

비슷하게 선택된 변수들도 많고 한번만 선택된 변수들도 존재한다. 하지만 변수는 정말 많이 줄어들었다는 것을 확인 할 수 있었다. 이때에 대략적으로 하나의 모델을 수행하는데 걸리는 시간은 다음과 같았다는 것을 확인 할 수 있었다.

시드 1개를 수행하였을 때 기준 =>

```
   사용자    시스템  elapsed
31627.08     35.31 31856.88
```

**[Q7] [Q6]에서 GA를 통해 한번 이상 주요 변수로 선택된 모든 변수들을 사용하여 나머지 설정은 [Q5]와 동일하게 하여 Test dataset에 대한 Accuracy와 BCR을 확인하고 모든 변수를 사용한 모델인 [Q5]의 결과와 비교해 보시오.**

위 문제에서 한번이라도 선택된 변수들의 인덱스를 전부 적어준 후 이를 데이터 셋에 적용 시킨 후 5번에서 실행한 과정을 다시 진행하여 보았다. 이를 진행하기 위해 걸린 시간은 다음과 같았다.

```
 사용자   시스템 elapsed
7272.75    27.48 7878.56
```

총 10번의 iteration을 돌려보았고 그에 따른 각각의 결과값은 다음과 같았다.

```
              ACC       BCR
 [1,] 0.3107539 0.3107539
 [2,] 0.1616837 0.1616837
 [3,] 0.3107539 0.3107539
 [4,] 0.1616837 0.1616837
 [5,] 0.3107539 0.3107539
 [6,] 0.1616837 0.1616837
 [7,] 0.3107539 0.3107539
 [8,] 0.1616837 0.1616837
 [9,] 0.3107539 0.3107539
[10,] 0.1616837 0.1616837
```

이전에 5번에서 진행한 결과와 비교를 해보았을 때 나오는 결과값의 범위는 0.15~0.35 사이 정도로 넓어졌다는 것을 확인할 수 있었다. 또한 알 수 있었던 점은 가장 높은 ACC와 BCR의 값이 이전에 비해 높아졌으며, 반대로 가장 낮은 값 또한 더 낮아졌다는 것을 확인 할 수 있었다.

위의 결과를 토대로 가장 성능이 높게 나온 결과는 다음과 같았다.

```
              ACC       BCR
New ANN 0.3107539 0.3107539
```

위에서도 언급하였지만, 결론적으로 바라보면 ACC와 BCR 값이 둘 다 증가하였고, 더 좋

은 모델의 기준을 더 적은 변수를 사용하고 시간이 더 적게 걸리며 성능이 높게 나오는 것이라고 하였을 때 문제 5번에서 진행한 모델보다 7번에서 진행한 모델이 더 좋은 모델이라는 결론을 내릴 수 있었다.

**[Q8] 동일한 Training/Validation 데이터셋을 이용하여 최적의 의사결정나무 하이퍼파라미터 조합을 탐색해 보시오. 이후 최적의 하이퍼파라미터 조합을 이용하여 Training/Validation 데이터셋을 결합한 데이터 셋으로 의사결정나무를 학습한 뒤, Test Dataset에 대한 Accuracy와 BCR을 산출해 보시오.**

현재 의사결정 나무를 수행해주기 위해 내가 선정한 의사결정나무 기법은 pre pruning 의사결정 나무이고, 각각의 하이퍼 파라미터는 다음과 같이 설정하여 여러 조합을 시도해보았다.

```
min_criterion = c(0.9, 0.95, 0.99)
min_split = c(10, 30, 50, 100)
max_depth = c(0, 10, 5)
```

위와 같이 총 36가지에 대한 조합을 살펴 보았고, 최적의 하이퍼 파라미터를 선정하기 위한 반복문을 돌린 후 ACC를 기준으로 정렬을 해본결과는 다음과 같았다.

| | min_criterion | min_split | max_depth | TPR | Precision | TNR | ACC | BCR | F1 |
|---|---|---|---|---|---|---|---|---|---|
| [1,] | 0.95 | 10 | 0 | 0.8505454 | 0.6683424 | 0.3661123 | 0.52180 | 0.5580279 | 0.7485155 |
| [2,] | 0.95 | 30 | 0 | 0.8505454 | 0.6683424 | 0.3661123 | 0.52180 | 0.5580279 | 0.7485155 |
| [3,] | 0.95 | 50 | 0 | 0.8505104 | 0.6683516 | 0.3661123 | 0.52178 | 0.5580164 | 0.7485078 |
| [4,] | 0.95 | 100 | 0 | 0.8508600 | 0.6682225 | 0.3623701 | 0.52162 | 0.5552713 | 0.7485621 |
| [5,] | 0.90 | 10 | 10 | 0.8583065 | 0.6551825 | 0.3089397 | 0.52074 | 0.5149417 | 0.7431140 |
| [6,] | 0.90 | 30 | 10 | 0.8583065 | 0.6551825 | 0.3089397 | 0.52074 | 0.5149417 | 0.7431140 |
| [7,] | 0.90 | 50 | 10 | 0.8583765 | 0.6551660 | 0.3085239 | 0.52074 | 0.5146160 | 0.7431295 |
| [8,] | 0.95 | 10 | 10 | 0.8580618 | 0.6549608 | 0.3093555 | 0.52064 | 0.5152147 | 0.7428796 |
| [9,] | 0.95 | 30 | 10 | 0.8580618 | 0.6549608 | 0.3093555 | 0.52064 | 0.5152147 | 0.7428796 |
| [10,] | 0.95 | 50 | 10 | 0.8580618 | 0.6549608 | 0.3093555 | 0.52064 | 0.5152147 | 0.7428796 |
| [11,] | 0.95 | 100 | 10 | 0.8579919 | 0.6550822 | 0.3097713 | 0.52064 | 0.5155398 | 0.7429315 |
| [12,] | 0.90 | 100 | 10 | 0.8582716 | 0.6552433 | 0.3079002 | 0.52062 | 0.5140642 | 0.7431400 |
| [13,] | 0.90 | 50 | 0 | 0.8471542 | 0.6713395 | 0.3698545 | 0.52022 | 0.5597533 | 0.7490688 |
| [14,] | 0.90 | 30 | 0 | 0.8469794 | 0.6713684 | 0.3704782 | 0.52018 | 0.5601673 | 0.7490184 |
| [15,] | 0.90 | 10 | 0 | 0.8469794 | 0.6713684 | 0.3700624 | 0.52014 | 0.5598529 | 0.7490184 |
| [16,] | 0.90 | 100 | 0 | 0.8476786 | 0.6711415 | 0.3646570 | 0.52002 | 0.5559783 | 0.7491503 |
| [17,] | 0.99 | 10 | 10 | 0.8490071 | 0.6570439 | 0.3089397 | 0.51542 | 0.5121445 | 0.7407916 |
| [18,] | 0.99 | 30 | 10 | 0.8490071 | 0.6570439 | 0.3089397 | 0.51542 | 0.5121445 | 0.7407916 |
| [19,] | 0.99 | 50 | 10 | 0.8490071 | 0.6570439 | 0.3089397 | 0.51542 | 0.5121445 | 0.7407916 |
| [20,] | 0.99 | 100 | 10 | 0.8489372 | 0.6571676 | 0.3093555 | 0.51542 | 0.5124680 | 0.7408436 |
| [21,] | 0.90 | 10 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [22,] | 0.90 | 30 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [23,] | 0.90 | 50 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [24,] | 0.90 | 100 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [25,] | 0.95 | 10 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [26,] | 0.95 | 30 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [27,] | 0.95 | 50 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [28,] | 0.95 | 100 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |
| [29,] | 0.99 | 10 | 5 | 0.8528178 | 0.6041558 | 0.2575884 | 0.51266 | 0.4686960 | 0.7072672 |

가장 높은 정확도는 0,5218이였고, 이에 해당하는 하이퍼파라미터는 0.95, 10, 0이였을 때라는 것을 확인 할 수 있었다. 위에서 구한 최적의 하이퍼 파라미터들로 다시 의사결정 모델을 구축한 후 test data 에 대해 돌려 본 결과는 다음과 같다. 우선은 confusion matrix를 살펴보겠다.

| | CART_pre_prediction | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 2490 | 3367 | 75 |
| 2 | 1455 | 28639 | 4129 |
| 3 | 120 | 10486 | 9840 |

사실 이를 수행하기 이전에 non-pruning방식과 post-pruning 방식을 수행해보았는데 그 때에는 1범주를 예측하는 경우가 하나도 없어 pre pruning을 선택하 된 것이다. 이에 대한 정확도와 BCR를 확인해보면 다음과 같다.

| | TPR | Precision | TNR | Accuracy | BCR | F1-Measure |
|---|---|---|---|---|---|---|
| Pre-Pruning | 0.8368349 | 0.6739857 | 0.4197572 | 0.5136714 | 0.5926782 | 0.7466336 |

정확도는 0.5136714가 나왔다는 것을 확인 할 수 있었고, BCR은 0.5926782가 나왔다는 것을 확인할 수 있었다.

**[Q9] Training dataset과 Validation dataset을 결합하여 Multinomial logistic regression을 학습하고, 이를 Test dataset에 적용하여 Accuracy와 BCR을 산출하고 이를 [Q5], [Q7], [Q8]의 결과와 비교해보시오.**

우선, 멀티 로지스틱 리그레션을 돌려보았을 때의 결과물에 대해 분석을 해보자면 다음과 같이 나왔다는 것을 확인할 수 있다.

```
> t(summary(ml_logit)$coefficients)
                                          2            3
(Intercept)                      2.2508623555  1.545972039
geo_level_1_id                  -0.2015044681 -0.048829617
geo_level_2_id                  -0.0367866143 -0.016263980
geo_level_3_id                  -0.0200526404 -0.015640412
count_floors_pre_eq              0.1212346008  0.417125623
age                              0.0153430258  0.013253141
area_percentage                  0.0636504738 -0.001592995
height_percentage                0.0451009554  0.042377108
land_surface_conditionn          0.0484541513  0.008835559
land_surface_conditiono          0.0318572741  0.065917341
land_surface_conditiont         -0.0594145951 -0.039076157
foundation_typeh                -0.1244040137 -0.072175540
foundation_typei                -0.0128656223 -0.013298475
foundation_typer                 0.0481209470  0.155678829
foundation_typeu                 0.0147269283  0.048520764
foundation_typew                -0.0391602775 -0.256481230
roof_typen                      -0.0253614507 -0.023119563
roof_typeq                       0.0726433727  0.171095995
roof_typex                      -0.0797843160 -0.257331601
ground_floor_typef               0.0461913507  0.115108617
ground_floor_typem              -0.0161267636 -0.012201804
ground_floor_typev              -0.0752894033 -0.207111153
ground_floor_typex               0.0224521377  0.062102183
ground_floor_typez              -0.0355024606 -0.045601456
other_floor_typej               -0.0564232446  0.102023825
other_floor_typeq                0.1012678666 -0.032684976
other_floor_types               -0.0965123117 -0.045071355
other_floor_typex               -0.0220515492 -0.030899093
positionj                        0.0453277405  0.007861076
positiono                        0.1929394025  0.138644722
positions                       -0.0992675382 -0.133884202
positiont                        0.0358089658  0.110773663
plan_configurationa              0.0141935257  0.013246825
plan_configurationc              0.0203631502  0.003119990
plan_configurationd              0.0010930185  0.007060862
plan_configurationf              0.2158839302  0.218432358
plan_configurationm              0.0321010219  0.086292737
plan_configurationn              0.1519510271  0.147090529
plan_configurationo              0.0140962777  0.021848000
plan_configurationq             -0.0309781038  0.035012903
plan_configurations             -0.0022907150 -0.006908699
plan_configurationu             -0.0115438934 -0.104230973
has_superstructure_adobe_mud     0.1920758053  0.205359602
has_superstructure_mud_mortar_stone  0.5070942637  0.592619754
has_superstructure_stone_flag    0.2144857917  0.304655590
has_superstructure_cement_mortar_stone  0.0546612885 -0.021685549
has_superstructure_mud_mortar_brick  0.3637181256  0.291884361
has_superstructure_cement_mortar_brick  0.0585612557 -0.235039548
has_superstructure_timber       -0.0690191104 -0.197782753
has_superstructure_bamboo        0.0187131526  0.017164195
has_superstructure_rc_non_engineered  -0.0806989300 -0.140480361
has_superstructure_rc_engineered -0.0997950729 -0.175195230
has_superstructure_other         -0.0612167801 -0.086744030
legal_ownership_statusa          -0.0133912596 -0.032946037
legal_ownership_statusr          -0.0220231544  0.012769213
legal_ownership_statusv          -0.0025743603 -0.014982048
legal_ownership_statusw           0.0403082103  0.065595554
count_families                    0.1202851824  0.157572699
has_secondary_use                -0.0857168050 -0.595545467
has_secondary_use_agriculture     0.1650067918  0.499310852
has_secondary_use_hotel           0.0044089515  0.230697190
has_secondary_use_rental         -0.0178125758  0.082694487
has_secondary_use_institution     0.0067657746  0.092284352
has_secondary_use_school          0.0237381118  0.043329360
has_secondary_use_industry        0.0081271136  0.062730703
has_secondary_use_health_post     0.0009279401  0.015765798
has_secondary_use_gov_office     -0.0840182176 -0.037432214
has_secondary_use_use_police      0.1277994546  0.128079364
has_secondary_use_other          -0.0007631844  0.030843293
```

위의 표를 보면 각 coefficient를 통해 변수들이 타겟 변수에 주는 영향도에 대해서 알아볼 수 있다. 또한 2,3이라는 것을 통해 1번 범주 대비 2번 범주는 몇이고, 1번 범주 대비 3번 범주는 몇인지를 알 수 있다.

| | ml_logit_pred | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 1648 | 4136 | 148 |
| 2 | 1118 | 29885 | 3220 |
| 3 | 107 | 16158 | 4181 |

혼동 행렬은 다음과 같이 나왔다는 것을 볼 수 있다. 1번 범주의 전체적으로 1범 범주인 것들은 2번 범주로 잘못 예측한 결과값의 빈도가 가장 높았고, 3번 범주 또한 2번 범주라고 잘 못 예측한 경우가 높았다. 그에 반해 2번 범주는 제대로 예측한 결과가 가장 높았다는 것을 알 수 있다.

여태까지 실행한 모든 모델들의 ACC와 BCR을 하나의 표로 작성하여 확인을 해보자면 아래와 같이 정리할 수 있다.

| Test Performance | ANN (all varibles) | ANN (Selected variables by GA) | Decision Tree | Multinomial Logistic Regression |
|---|---|---|---|---|
| **Accuracy** | 0.30720301 | 0.3107539 | 0.5136714 | 0.5893302 |
| **BCR** | 0.3072301 | 0.3107539 | 0.5926782 | 0.3674411 |

각각에 대해 비교를 해보자면 단순 정확도가 가장 높았던 모델은 multi logistic regression이였다. 하지만 이 이유에 대해서 생각을 해보자면, 혼동행렬을 통해 보았을 때, 아주 높은 비율로 2번 범주라고 예측을 하는 경우가 많아서 단순 정확도는 높게 나올 수 있다고 생각한다. 실제로 BCR을 보면 0.6에 육박하는 Accuracy에 비해 BCR이 좀 낮은 수치를 가지고 있다는 것을 보여주고 있다. 또한 ANN은 모든 변수를 사용하였을 때에 비해 GA를 통해 얻어낸 ACC와 BCR이 더 좋은 성능을 내고 있다는 것을 확인 할 수 있다. 하지만 둘 다, 정확도와 BCR이 0.5보다 낮다는 것은 매우 낮은 수치라고 생각을 하며 이 데이터셋은 ANN과 잘 어울리지 않은 것 같다는 것이 나의 의견이다. 추가적으로 가장 좋다고 판단되는 모델은 Decision Tree로 최적의 하이퍼 파라미터르 선정하여 돌려본 결과 ACC와 BCR에서 가장 높은 결과가 나왔다는 것을 확인 할 수 있었다. 결론적으로 내가 설정한 세팅들에 대해 가장 잘 맞는 모델은 의사 결정 나무였다는 결론을 내릴 수 있었다.