Homework #5 **20180047 김경수**

0. 이론적 배경

(1) KNN model(K-Nearest Neighbors)

이론상 질적 반응변수는 베이즈 분류기를 사용하는 것이 가장 좋지만, 실제 데이터셋에서는 주어진 X에 대한 Y의 조건부분포를 모르므로 베이즈 분류기를 계산할 수 없다. KNN model은 다음과 같은 분류기를 제안하여, 이에 따라 베이즈 규칙을 적용하여 확률이 가장 높은 클래스에 할당한다.

양의 정수 K와 검정 관측치 x에 대해, 먼저 x에 가장 가까운 K개의 점을 식별한 후, 클래스 j에 대한 조건부확률을 반응변수 값이 j인 점들의 비율로 추정한다.

K가 작을 경우 편향은 낮지만 분산이 높은 분류기에 해당하고, K가 클 경우 편향이 높지만 분산이 낮은 분류기에 해당한다. 따라서 올바른 수준의 K를 설정하는 것이 중요하고, 본 레포트에서 이용한 방법은 k-fold 교차검증이다.

(2) K-fold 교차검증

먼저 관측치셋을 임의로 K개의 fold로 분할한다. 첫 번째 fold는 검증셋으로 취급하고, 적합은 나머지 k-1개의 fold에 대해 수행한다. 그 다음 MSE(평균제곱오차)를 검증셋 fold에 대해 계산하고 이를 k번 반복한다(검증셋을 바꿔가며). K-fold CV 추정치는 이 값들을 평균하여 계산한다.

완벽한 검증 system을 존재하지 않는다. K-fold CV 역시 k값에 관련된 편향-분산 tradeoff가 존재한다. 일반적으로 k=5, k=10을 사용한다. 경험적으로 이 값들을 사용하면 지나치게 높은 편향 / 분산에 의한 문제없이 검정오차율 추정치를 얻을 수 있다는 것이 알려져 있다.

(3) Pre-processing

본 레포트에서 진행한 pre-processing은 다음과 같다.

먼저, Quantile Normalization 이다. 정규화는 많은 양의 데이터를 처리할 때, 데이터의 범위를 일치시키거나 분포를 유사하게 하기 위해 필요하다. Quantile 정규화에 대해 간단히 설명하면, D1, ..., DN 인 N 개의 data set이 있을 때, 각 data set의 p%에 위치하는 데이터의 값을 갖게 해 주는 작업이다.

다음으로, Min-Max Normalization 이다. 새로운 값 = (현재 값 - 최소) / (최대 - 최소)로 정규화 하며, 이는 각 data set들의 특성은 보존하고, scale을 맞추어 분석을 용이하게 한다.

세 번째, Undersampling이다. Stage1, stage2, stage3의 sample 개수는 다른데, 이를 min[stage i]로 맞추어 주는 작업이다. 이는 클래스 불균형을 없애는 작업인데, 데이터가 클래스에 불균형하게 분포되어 있을 경우 비정상 탐지(Anomaly detection)이 일어날 가능성이 크기 때문이다. Undersampling의 경우 클래스의 개수를 줄이는 것이므로, 유용한 정보가 버려질 수 있는 것이 단점이다.

마지막으로, ANOVA test를 통해 stage에 따른 가장 큰 변화를 보이는 gene을 select하여, 이에 대해 집중하여 방대한 계산량을 줄일 수 있다.

***Homework 5-1, 5-2 모두에 해당하는 동일코드이다.**

```
clin <- readRDS("clinical.rds")
gex <- readRDS("expression.rds")
```

#clinical과 expression.rds 데이터들을 위와 같이 implement함.

```
idx <- intersect(colnames(gex), clin$sample)
clin_p <- clin[clin$sample_id %in% idx, ]
stage <- as.vector(clin_p$stage)
stage_p <- rep(0,length(stage))
for (i in 1:length(stage)){
  if(stage[i]=='stage i'){stage_p[i]=1}
  else if(stage[i]=='stage ii'){stage_p[i]=2}
  else if(stage[i]=='stage iii'){stage_p[i]=3}
  else if(stage[i]=='stage iv'){stage_p[i]=4}
  else if(stage[i]=='stage v'){stage_p[i]=5}
}
gex_p <- gex[, colnames(gex) %in% idx]
gex_p <- na.omit(gex_p)
```

#Subtype이 NA인 sample을 제외하고, 두 데이터셋이 공통적으로 가지는 sample_id 들에 대해서만 데이터를 남김. 편의를 위해 stage_p 벡터를 정의하여 각 stage를 integer에 매칭시킴.

```
library(preprocessCore)
temp <- normalize.quantiles(gex_p)
colnames(temp) <- colnames(gex_p)
rownames(temp) <- rownames(gex_p)
gex_p <- temp
```

#Quantile Normalization 진행.

```
normalize <- function(x)
{
  return((x- min(x))/(max(x)-min(x)))
}
gex_p <- apply(gex_p, MARGIN=1, normalize)
```

#Min-Max Normalization진행

```
data <- data.frame(stage_p, gex_p)
```

Homework #5

1. Plot estimated accuracy with 5-fold CV, varying k of knn model and find optimal k.
2. Repeat 5-fold CV after removing class imbalance calibration step. Then, find the tendency of accuracy
according to k value. Discuss its reason in terms of kNN algorithm.

1. Procedure & R code

(1) Undersampling을 진행한다. Stage_1에 속하는 data의 개수가 89이고, 이것이 최소이기 때문에 이에 맞춰 undersampling을 진행한다.

R code :

```
sum(data$stage_p == 1);sum(data$stage_p == 2);sum(data$stage_p == 3);
set.seed(123)
stage2.sam <- sample(rownames(data)[data$stage_p == 2],
size=min(sum(data$stage_p==1)))
stage3.sam <- sample(rownames(data)[data$stage_p == 3],
size=min(sum(data$stage_p==1)))
data <- data[rownames(data) %in% c(rownames(data)[data$stage_p == 1], stage2.sam,
stage3.sam), ]
```

(2) ANOVA test를 통해 stage에 따른 가장 큰 변화를 보이는 gene을 select하고, 이에 대해 집중한다.

R code :

```
feature.size <- 2
aov.gene.pval <- c()
for(i in 2:(ncol(data))){
  aov.gene.pval <- c(aov.gene.pval, summary(aov(data$stage_p ~ data[,
i]))[[1]][["Pr(>F)"]][1])
}
aov.idx <- order(aov.gene.pval, decreasing=FALSE)[1:feature.size]
data.aov <- data[, c(1, 1+aov.idx)]
```

(3) 마지막으로, 5-fold CV를 진행할 것이므로 data를 4 : 1 비율로 나눈 후, 1~89까지의 k에 대해 accuracy를 측정한 후, 이를 plotting 한다.

R code :

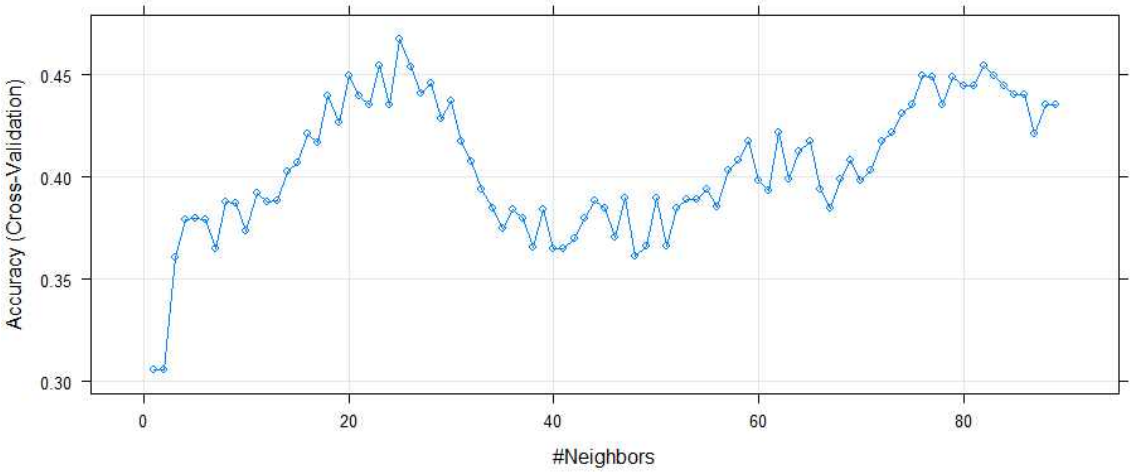
```
library(caret)
data.aov$stage_p <- as.factor(data.aov$stage_p)
partition <- createDataPartition(data.aov$stage_p, p=0.8, list=FALSE)
train <- data.aov[partition,]
test <- data.aov[-partition,]
k <- expand.grid(k = c(1:20))
set.seed(123)
```

```
ctrl <- trainControl(method="cv", number=5)
knn.cv <- train(stage_p~., train, method="knn", trControl=ctrl,
                preProcess=c("center", "scale"), tuneGrid=k)
knn.cv$results
plot(knn.cv)
```

(4) 새롭게 R 프로그램을 가동시킨 후, (1)의 undersampling 과정만을 제외하고 똑같이 (2)~(4)과정을 동일하게 진행한다.

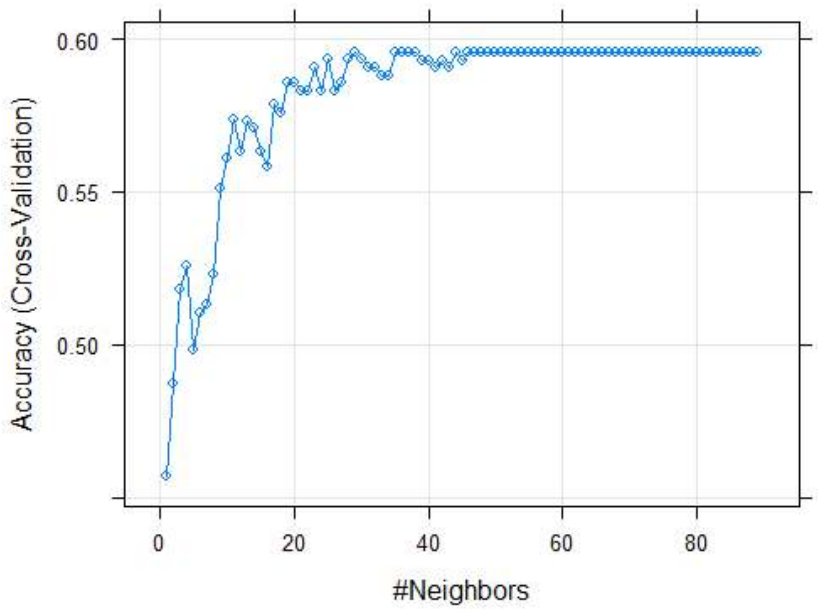
2. Results

(1) 5-fold CV를 이용한 knn model들의 accuracy의 plot은 아래와 같다.



이를 통해, accuracy가 가장 높은 K=25가 가장 optimal 하다고 할 수 있다.

(2) 클래스 불균형 작업을 하지 않은 accuracy plot은 아래와 같다.



이 경우, 역시 K=25일 때가 가장 optimal하다고 할 수 있다. Max 값은 아니지만, 그 옆에 있는 값들은 overfiting 된 것이다.

Accuracy의 tendency를 보면 sample의 수가 큰 경우(undersampling을 하지 않았을 때), 평균적으로 증가했음을 알 수 있다. Sample의 수가 크다면, training data의 수 역시 증가한다. 또한 많은 training data가 존재할수록, 우리는 더 정확한 추론을 할 수 있고,(input들의 밀집도가 높아지기 때문이다. 특정 범위 내에 기준으로 삼을 수 있는 data들이 많아 보다 정확하다. 극단적인 예로, filed에 하나의 training data가 있을 때와 100개의 training data가 있을 때를 생각해보자.) 그렇기에 error를 최소화 할 수 있고, accuracy가 증가한다.
또한 undersampling을 하여 클래스 불균형을 최소화하였더니, overfitting의 효과를 꽤 많이 배제한 것을 확인할 수 있다.