Assignment #2

knn 실습

20190552 손지영

사용 data

- CommomBank.csv: Common Bank의 예금계좌를 소유하고 있는 5,000명의 고객 정보 데이터

데이터 전처리

- 문제를 풀기 전 전처리를 수행한다.

```
# 사용할 패키지 추가
library(class)
library(caret)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(ggrepel)
library(ggforce)
```

```
# 데이터 불러오기
comm = read.csv('CommonBank.csv')

# 타겟 열 factor로 지정하고 순서 바꾸기
comm$PersonalLoan = factor(comm$PersonalLoan, levels=c(1,0), labels=c('Accept', 'Reject'))
str(comm)
```

```
## 'data.frame':    5000 obs. of  14 variables:
##  $ ID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ ZIP.Code          : int  91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ PersonalLoan      : Factor w/ 2 levels "Accept","Reject": 2 2 2 2 2 2 2 2 2 1 ...
##  $ SecuritiesAccount: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CDAccount         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

```
# accuracy을 평가지표로 사용할 때는 positive값이 무엇이든 결과가 동일하게 나오지만,
# precision을 쓸 때는 무엇이 positive인지 중요하다. 따라서, 목표에 맞게 순서를 바꿔준다.
```

1번문제

먼저 ID와 ZIP.code는 feature에서 제외한다. 그리고 z-score normalization을 활용하여 feature들의 scale을 일치시킨다. 첫 4,000명의 데이터를 training set으로, 나머지 1,000명의 데이터를 test set으로 사용하고, training set과 test set에서의 target variable의 분포를 비교해 보자.

```
# ID, ZIP.code열 제거
x = comm[, -c(1,5,10)]

# 타겟 열 추출
y = comm[, 10]
str(x)
```

```
## 'data.frame':    5000 obs. of  11 variables:
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : int  1 1 1 2 2 2 2 3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ SecuritiesAccount: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CDAccount         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

```
# z-score normalization을 위한 함수 생성
z_score = function(x) {
  (x-mean(x))/sd(x)
}

# 각 열에 대해 normalization 적용
x_scale = as.data.frame(lapply(x, z_score))
str(x_scale)
```

```
## 'data.frame':    5000 obs. of  11 variables:
##  $ Age               : num  -1.7742 -0.0295 -0.5529 -0.9019 -0.9019 ...
##  $ Experience        : num  -1.6659 -0.0963 -0.4451 -0.9683 -1.0555 ...
##  $ Income            : num  -0.538 -0.864 -1.364 0.57 -0.625 ...
##  $ Family            : num  1.397 0.526 -1.217 -1.217 1.397 ...
##  $ CCAvg             : num  -0.193 -0.251 -0.537 0.436 -0.537 ...
##  $ Education         : num  -1.049 -1.049 -1.049 0.142 0.142 ...
##  $ Mortgage          : num  -0.555 -0.555 -0.555 -0.555 -0.555 ...
##  $ SecuritiesAccount: num  2.929 2.929 -0.341 -0.341 -0.341 ...
##  $ CDAccount         : num  -0.254 -0.254 -0.254 -0.254 -0.254 ...
##  $ Online            : num  -1.22 -1.22 -1.22 -1.22 -1.22 ...
##  $ CreditCard        : num  -0.645 -0.645 -0.645 -0.645 1.549 ...
```

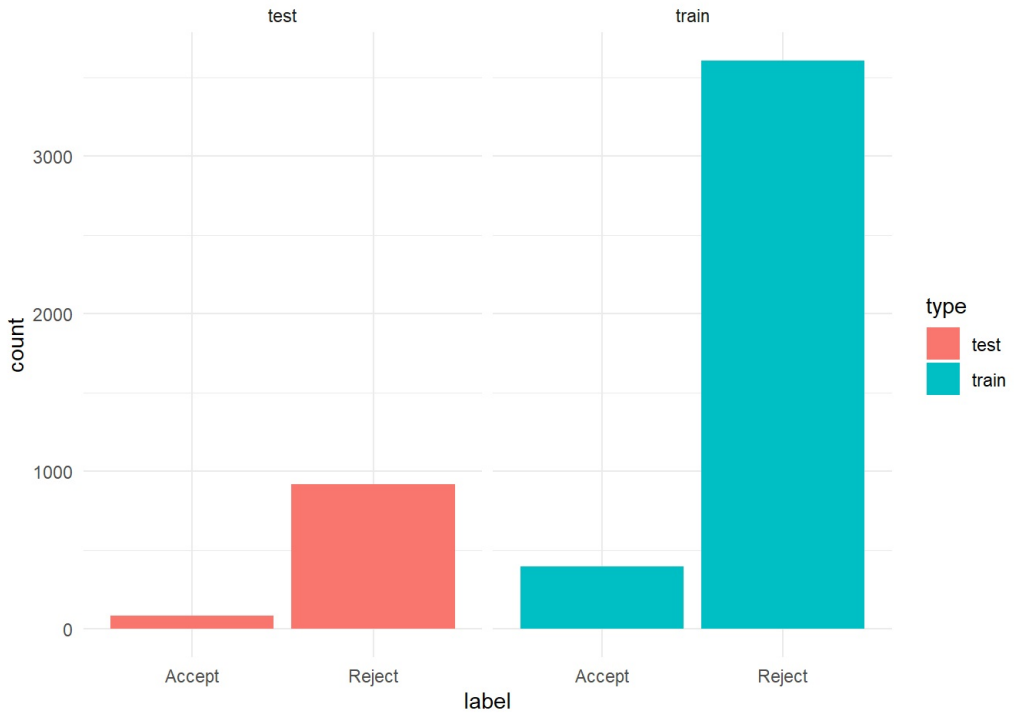
```
# 문제에 맞게 train set과 test set으로 나누기
x_train = x_scale[1:4000,]
x_test = x_scale[4001:5000,]
y_train = comm[1:4000,10]
y_test = comm[4001:5000,10]

# 나눈 데이터의 분포를 살펴보기 위해 데이터프레임 형태로 바꾸기
df_train = as.data.frame(y_train)
df_train[,'type'] = 'train'
df1 = rename(df_train, label=y_train)
df_test = as.data.frame(y_test)
df_test[,'type'] = 'test'    # 같은 컬럼명으로 지정
df2 = rename(df_test, label=y_test)

# 두 데이터프레임을 결합하기
df = rbind(df1, df2)
str(df)
```

```
## 'data.frame':    5000 obs. of  2 variables:
##  $ label: Factor w/ 2 levels "Accept","Reject": 2 2 2 2 2 2 2 2 2 1 ...
##  $ type : chr  "train" "train" "train" "train" ...
```

```
# 두 데이터를 같은 y축으로 비교
ggplot(df, aes(x=label, fill=type)) +
  geom_bar() +
  facet_wrap(~type) +
  theme_minimal()
```



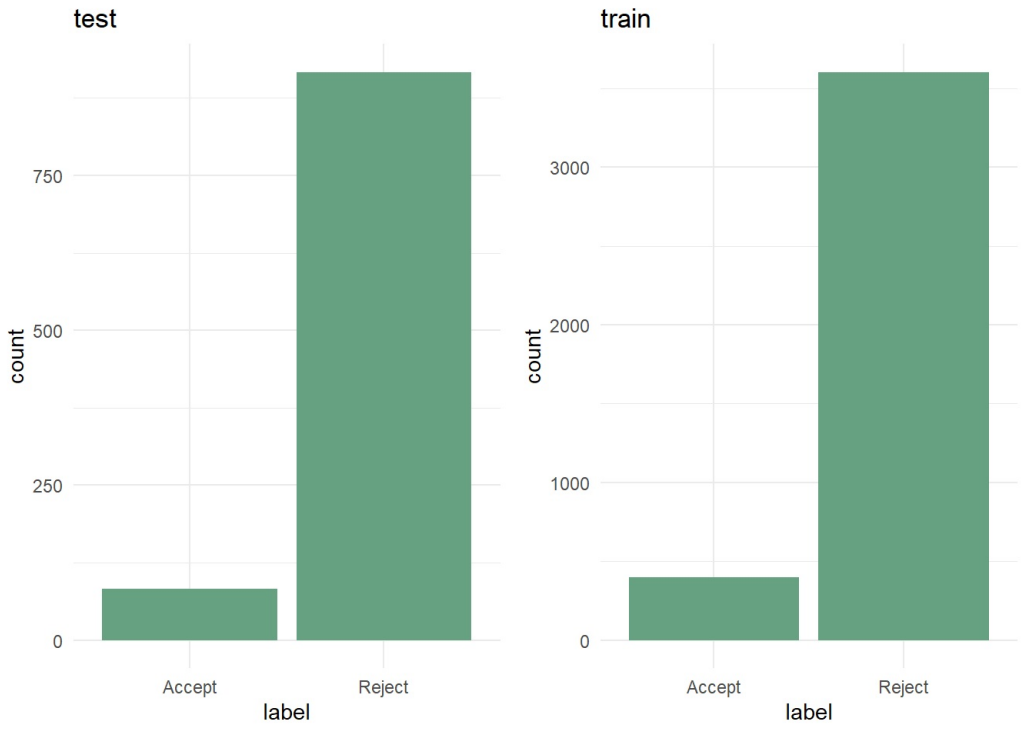
<그림 설명>

위의 그림을 통해 test set과 train set 모두 Accept한 고객에 비해 Reject한 고객이 많은 것을 확인할 수 있다. 하지만 같은 y축으로 설정이 되어있어 각각의 비율 차이를 한눈에 알아보기 어렵다. 따라서 최대 높이를 동일하게 설정하고 분포를 살펴보도록 하겠다.

```
# 최대 높이를 동일하게 설정하여 그리기
p1 = ggplot(df2, aes(x=label)) +
  geom_bar(fill='#66a182') +
  labs(title='test') +
  theme_minimal()


p2 = ggplot(df1, aes(x=label)) +
  geom_bar(fill='#66a182') +
  labs(title='train') +
  theme_minimal()

grid.arrange(p1, p2, ncol = 2)
```



<그림 설명>

위의 그림과 같이 최대 높이를 동일하게 설정하여 비교해보았다. 그림을 통해 두 데이터에서 Accept한 고객의 비율이 거의 동일하게 추출되었지만, train set에서의 Accept 고객 비율이 근소하게 높은 것을 그림을 통해 확인할 수 있다.

2번문제

7-NN을 적용하고, 결과를 분석해보자.

```
pred = knn(train=x_train, test=x_test, cl=y_train, k=7)
confusionMatrix(pred, y_test)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Accept Reject
##     Accept     49      4
##     Reject     34    913
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.917
##     P-Value [Acc > NIR] : 8.683e-09
##
##                   Kappa : 0.7013
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.5904
##             Specificity : 0.9956
##          Pos Pred Value : 0.9245
##          Neg Pred Value : 0.9641
##              Prevalence : 0.0830
##          Detection Rate : 0.0490
##    Detection Prevalence : 0.0530
##       Balanced Accuracy : 0.7930
##
##        'Positive' Class : Accept
##
```

<결과 해석>

해당 데이터에서 positive는 'Accept', negative는 'Reject'이다. 따라서 출력된 confusion matrix를 살펴보면, TP는 49, TN는 4, FP는 34, TN은 913인 것을 확인할 수 있다. 따라서 대표적인 분류 모델의 성능지표인 Accuracy((TP+TN)/(TP+FN+FP+TN))값을 구해보면 ((49+4)/(49+4+34+913))으로 0.962로 구할 수 있다. 계산 결과를 출력 결과와 비교해보면 동일한 것을 역시 확인할 수 있다. 또한 신뢰구간은 (0.9482, 0.973)으로 상당히 높은 정확도임을 확인할 수 있다. confusion matrix에서는 출력 결과로 Accuracy말고 Sensitivity, Specificity, Pos Pred Value와 같은 다양한 성능 지표를 제공한다. 이 중에서 중요하게 살펴봐야되는 지표는 Pos Pred Value, 즉 Precision값이다. Precision은 (TP/(TP+FP))로 모델이 예측한 값들 중에서 얼마나 잘 예측했는 지를 나타내는 척도이다. 해당 문제의 목표는 '고객들을 타겟팅하여 집중적으로 예산을 투입'하는 것이다. 대개 마케팅에서는 전체 고객 중에서 실제로 반응할 것 같은 고객의 비율(Accept)은 상대적으로 적기 때문에 모델이 실제 반응할 것 같은 고객을 찾는 것이 더 중요하다. 따라서 해당 문제에서는 Accuracy보다는 예측한 고객들 중에서 실제 반응한 고객들의 비율을 나타내는 precision이 더 중요하다. 현재 모델의 precision 값은 0.9245로 Accuracy값보다는 상대적으로 낮은 값이지만 그래도 높은 성능을 보이고 있다.

3번문제

Training set 중에서 마지막 800명의 데이터를 validation set으로 사용하여, 다양한 k 값에 대해 k-NN을 적용해 보고 예측 성능을 비교해 보자. k가 어떤 값을 가질때 모델의 성능이 가장 우수한가?

```
# 문제에 주어진대로 데이터 나누기
x_train2 = x_scale[1:3200,]
x_valid = x_scale[3201:4000,]
y_train2 = as.factor(comm[1:3200,10])
y_valid = as.factor(comm[3201:4000,10])
```

[Accuracy을 기준으로 진행]

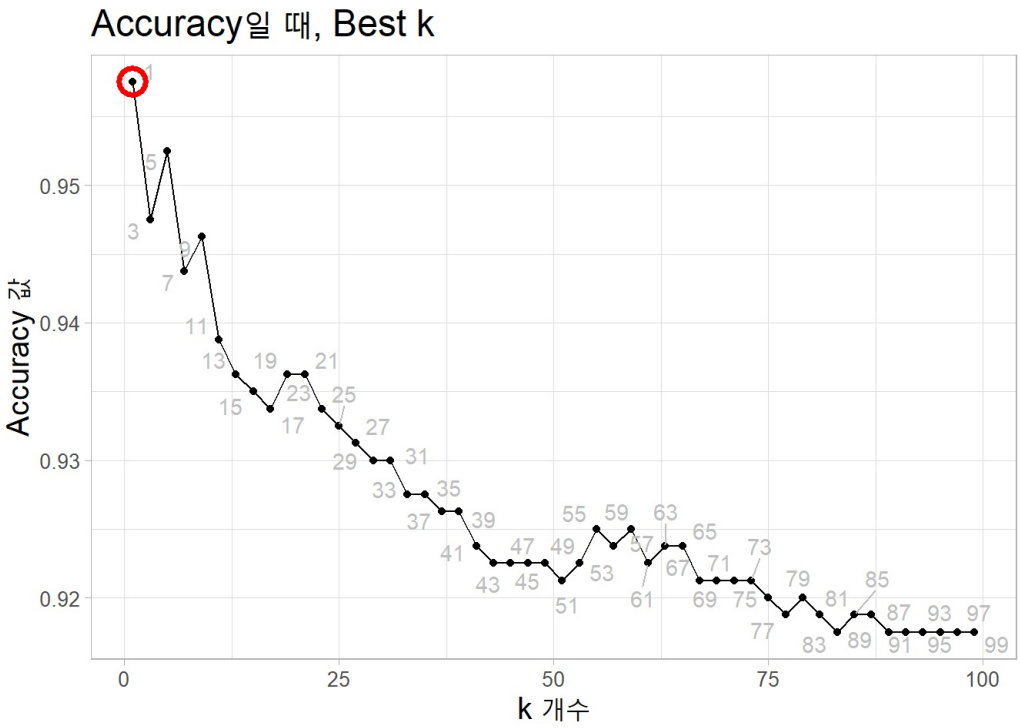
```
# k개수와 해당 accuracy값을 저장하는 반복문
kvalue = NULL
accuracy_k = NULL
for (kk in seq(1, 99, 2)) {
  knn_k = knn(x_train2, x_valid, cl=y_train2, k=kk)
  kvalue = c(kvalue, kk)
  cm = confusionMatrix(knn_k, y_valid, positive='Accept') # Accept가 지정되도록 설정해준다
  accuracy = cm$overall["Accuracy"]
  accuracy_k = c(accuracy_k, accuracy)
}

# 저장된 값을 데이터프레임으로 변환하여 사용
valid_k = data.frame(k=kvalue, accuracy=accuracy_k)

# 최대 accuracy값과 그에 해당하는 k값을 저장
max_row = valid_k[valid_k$accuracy==max(valid_k$accuracy), ]
max_k = max_row$k

ggplot(valid_k, aes(x=kvalue, y=accuracy_k)) +
  geom_line() +
  geom_point() +
  theme_light() +
  geom_text_repel(label=seq(1, 99,2), color='grey') +
  theme(text = element_text(size=15),
        axis.text = element_text(size=10)) +
  labs(title="Accuracy일 때, Best k",
       x ="k 개수", y = "Accuracy 값") +
  geom_point(aes(x = max_k, y = max(valid_k$accuracy)), size = 5, color = "red", shape=1, stroke=2)
```

```
## Warning: Use of `valid_k$accuracy` is discouraged.
## i Use `accuracy` instead.
```



[Precision을 기준으로 진행]

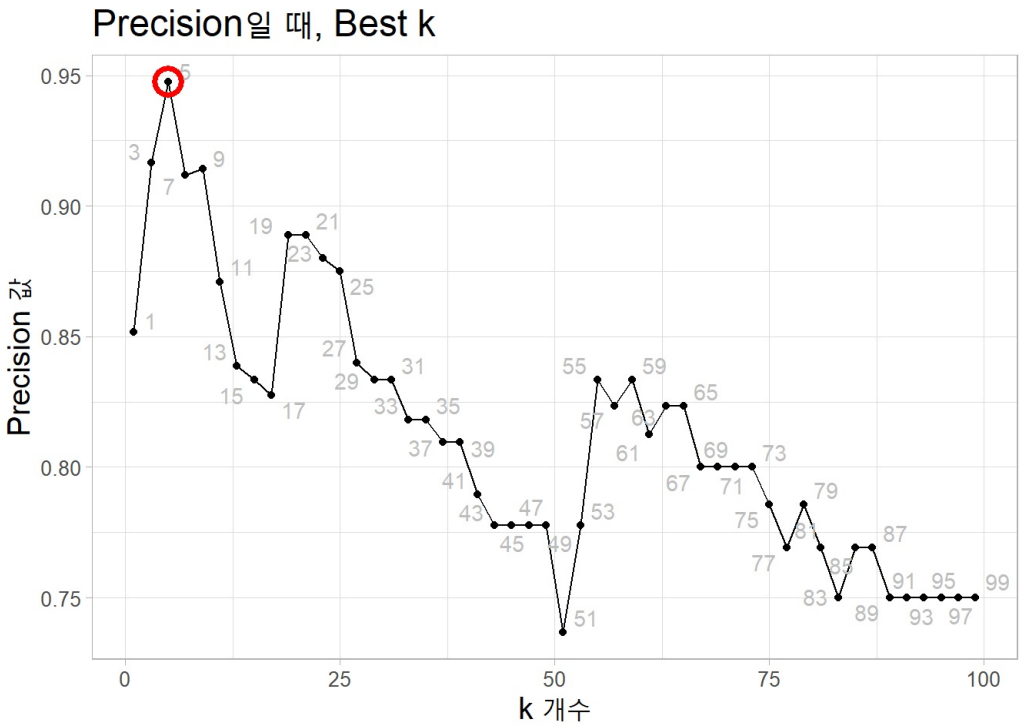
```
# k개수와 해당 precision값을 저장하는 반복문
kvalue = NULL
precision_k = NULL
for (kk in seq(1, 99,2)) {
  knn_k = knn(x_train2, x_valid, cl=y_train2, k=kk)
  kvalue = c(kvalue, kk)
  cm = confusionMatrix(knn_k, y_valid, positive='Accept') # Accept가 지정되도록 설정해준다
  precision = cm$byClass["Pos Pred Value"]
  precision_k = c(precision_k, precision)
}

# 저장된 값을 데이터프레임으로 변환하여 사용
valid_k = data.frame(k=kvalue, precision=precision_k)

# 최대 precision값과 그에 해당하는 k값을 저장
max_row = valid_k[valid_k$precision==max(valid_k$precision), ]
max_k = max_row$k

ggplot(valid_k, aes(x=kvalue, y=precision_k)) +
  geom_line() +
  geom_point() +
  theme_light() +
  geom_text_repel(label=seq(1, 99,2), color='grey') +
  theme(text = element_text(size=15),
        axis.text = element_text(size=10)) +
  labs(title="Precision일 때, Best k",
        x ="k 개수", y = "Precision 값") +
  geom_point(aes(x = max_k, y = max(valid_k$precision)), size = 5, color = "red", shape=1, stroke=2)
```

```
## Warning: Use of `valid_k$precision` is discouraged.
## ℹ Use `precision` instead.
```



<결과 해석>

두가지의 평가지표, Accuracy와 Precision을 기준으로 모델을 수행하였을 때 최적의 k값이 다르다는 것을 그래프를 통해 확인할 수 있다. 먼저 Accuracy를 기준으로 수행하였을 경우 최적의 k값은 1이며, Precision을 기준으로 하였을 때는 5로 구할 수 있었다. 따라서 마케팅을 목적으로 하는 해당 문제에서는 k=5가 적합할 것이다.

4번문제

Training set에 대해 5-fold cross validation을 5회 반복하여 best k 값을 찾아보자. Best k 값으로 만들어지는 최종 model에 test set을 적용하여 model의 성능을 report하자.

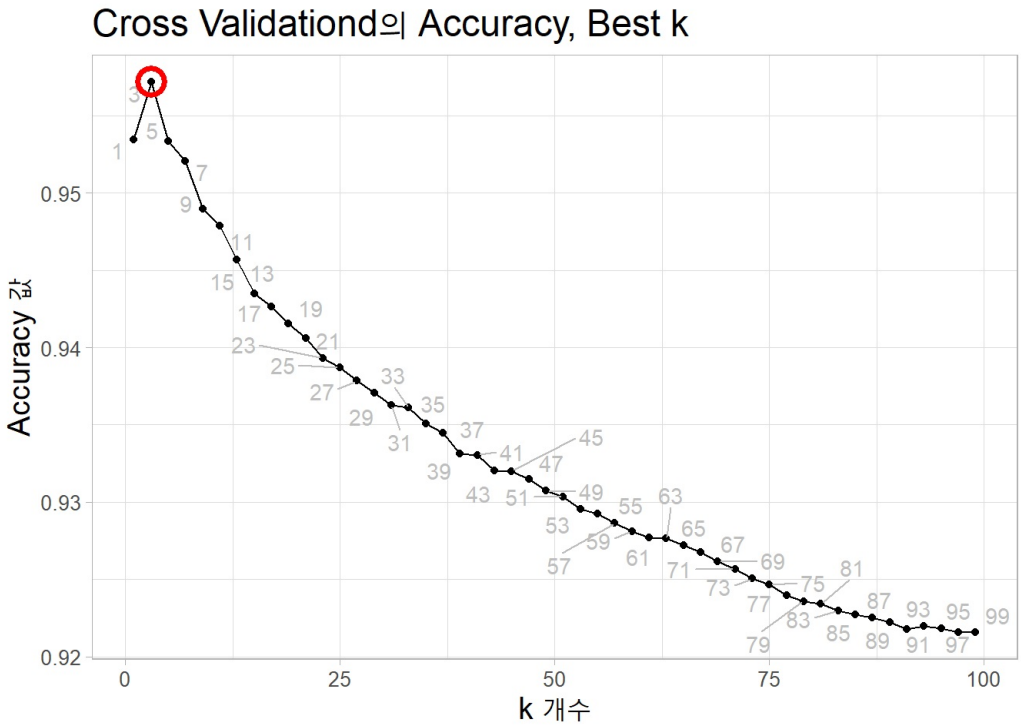
[Accuracy을 기준으로 진행]

```
# 문제에서 주어진대로 데이터 나누기
comm_n = comm[, -c(1,5)]
x_train = comm_n[1:4000,]
x_test = comm[4001:5000,]
y_train = comm[1:4000,10]
y_test = comm[4001:5000,10]

z_normalized = c('center', 'scale')
cv = trainControl(method='repeatedcv', number=5, repeats=5)
tune_grid = expand.grid(k=seq(1, 99, 2))
knn_fit = train(data=x_train, PersonalLoan~., method='knn', trControl=cv,
                preProcess=z_normalized, tuneGrid=tune_grid, metric='Accuracy')

# 최대 accuracy값과 그에 해당하는 k값을 저장
df = knn_fit$results
max_row = df[df$Accuracy==max(df$Accuracy), ]
max_k = max_row$k

ggplot(knn_fit) +
  geom_line() +
  geom_point() +
  theme_light() +
  geom_text_repel(label=seq(1, 99,2), color='grey') +
  theme(text = element_text(size=15),
        axis.text = element_text(size=10)) +
  labs(title="Cross Validationd의 Accuracy, Best k",
       x ="k 개수", y = "Accuracy 값") +
  geom_point(aes(x = max_k, y = max(df$Accuracy)), size = 5, color = "red", shape=1, stroke=2)
```



<결과 해석>

3번의 k=1의 결과와는 다르게 5-fold cross validation을 진행한 결과 k=3에서 최적임을 확인할 수 있다.

[Precision을 기준으로 진행]

기존의 trainControl 함수와 train 함수에서는 precision에 대한 평가지표를 제공하지 않는다. 따라서 precision에 대한 함수를 새로 생성하고 결과를 구했으나 그래프가 기존의 것들과는 다소 다른 양상을 띤다. 따라서 해당 결과가 옳은지 의심되었으나, 코드 상에서는 문제를 발견하지는 못하여 해당 코드를 기반으로 최적 k값을 구하였다.

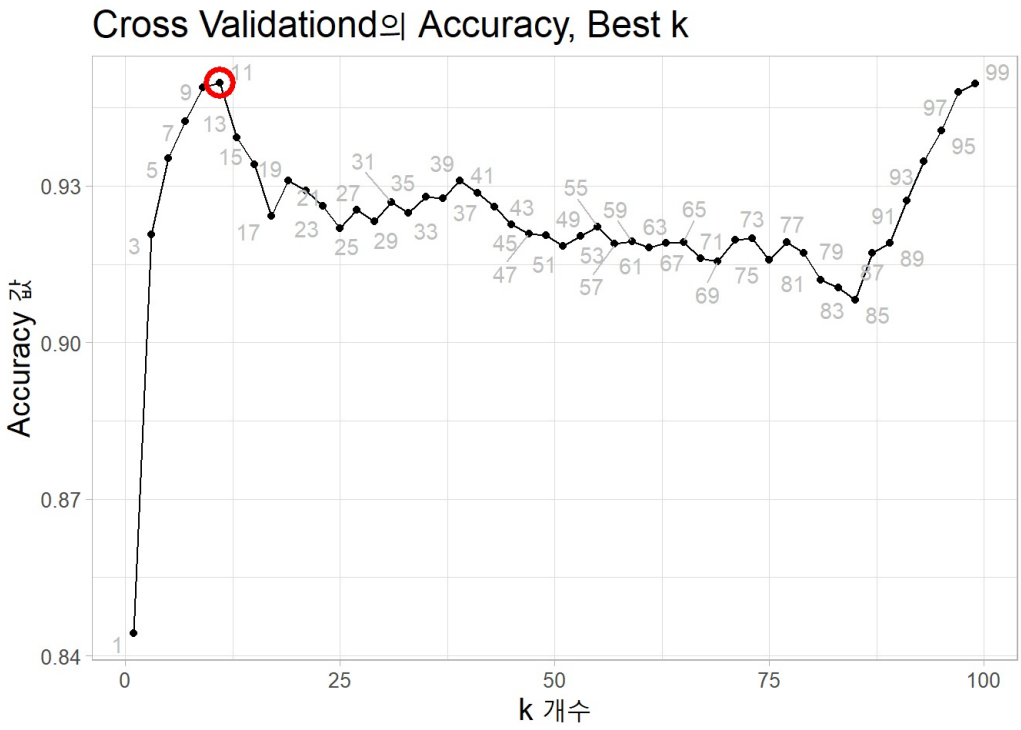
```
# precision으로 진행하기 위해서는 먼저 precision값을 구하는 함수를 정의해주어야한다
# 'precisionFunction'이라는 함수로 precision값을 구하고 저장하는 함수를 만든다
precisionFunction <- function(data, lev = NULL, model = NULL){
  cm <- confusionMatrix(data = data$pred, reference = data$obs)
  out <- cm$byClass['Pos Pred Value']
  names(out) <- 'Precision'
  out
}

# 위에서 생성한 함수를 summaryFunction이라는 함수에 넣어준다
cv <- trainControl(method = "repeatedcv", number = 5, repeats = 5,
                   summaryFunction = precisionFunction)

# metric='Precision'으로 지정하여 precision값을 기준으로 실행되게 한다
knn_fit <- train(data = x_train, PersonalLoan ~ ., method = 'knn',
                 trControl = cv, preProcess = z_normalized, tuneGrid = tune_grid,
                 metric='Precision')

# 최대 precision값과 그에 해당하는 k값을 저장
df = knn_fit$results
max_row = df[df$Precision==max(df$Precision), ]
max_k = max_row$k

ggplot(knn_fit) +
  geom_line() +
  geom_point() +
  theme_light() +
  geom_text_repel(label=seq(1, 99,2), color='grey') +
  theme(text = element_text(size=15),
        axis.text = element_text(size=10)) +
  labs(title="Cross Validationd의 Accuracy, Best k",
       x ="k 개수", y = "Accuracy 값") +
  geom_point(aes(x = max_k, y = max(df$Precision)), size = 5, color = "red", shape=1, stroke=2)
```



5번문제

3번과 4번에서 활용한 training 방식의 장단점을 비교해보자.

3번은 holdout 방식이며, 4번은 k-fold cross validation을 사용하였다. 먼저 holdout 방식은 전체 data set에서 test data를 분리하고 남은 train data의 일부를 validation set으로 분리하는 방법이다. 이 방식은 간단하고 빠르지만 train data의 손실이 있기 때문에 데이터가 적은 경우에 사용하기 힘들고, 검증을 한 번 밖에 진행할 수 없다는 단점이 있다. 다음으로 k-fold cross validation은 train data를 k개로 나눈 뒤 차례대로 하나씩을 validation set으로 활용하여 k번 검증을 진행하는 방식이다. 이 방식은 모든 데이터가 모델의 학습과 평가에 사용되며 holdout 방식보다 더욱 일반화된 모델 생성이 가능하다는 장점이 있다. 따라서 특정 data set에 대한 과적합을 방지할 수 있다. 하지만 holdout 방식에 비해 모델 훈련 및 평가에 소요되는 연산 비용이 늘어난다는 단점이 있다.