● **STEP1: 데이터 프레임 준비 및 전처리**

**(주어진 소스코드에 따라 진행+class(churn변수) 척도를 범주형으로 변경 추가)**

● **STEP2 : 두 개의 데이터 셋 구성(train/test)**

```
set.seed(123) #결과가 달라지지 않도록 함
ind <- sample(2, nrow(churn_z), replace = T, prob = c(0
.7,0.3))
ind #열벡터 형태#
churn_train <- churn_z[ind==1, ] #[행,열]
churn_test <- churn_z[ind==2, ] #행의 번호가 ind가 2인
것들을 추출해서 이를 test으로 할당

table(churn_z$churn==0)/nrow(churn_z)
table(churn_train$churn==0)/nrow(churn_train)
table(churn_test$churn ==0)/nrow(churn_test)
```

```
> table(churn_z$churn==0)/nrow(churn_z)

    FALSE      TRUE
0.2653699 0.7346301
> table(churn_train$churn==0)/nrow(churn_train)

    FALSE      TRUE
0.2668687 0.7331313
> table(churn_test$churn ==0)/nrow(churn_test)

    FALSE      TRUE
0.2618251 0.7381749
```

**->세 가지 데이터셋(churn_z, churn_train, churn_test)에서 0(No)의 측정값 비율이 0.733~0.738 정도로 비슷해서 다시 구분할 필요가 없음**

● **STEP3: train 데이터로 학습 후 최적의 k값 선정과 KNN 모델 수립**

```
library(caret)
grid1 <- expand.grid(k=3:30) #근접한 이웃개수 후보군
설정(k가 작아지면 이상치에 영향을 받고, 커지면 큰
딥단으로 분류되는 편향이 심해지니 적절하게 3~30으로 설정
)
control <- trainControl(method ="repeatedcv", number=10,
repeats=5) #학습방법 채택
set.seed(135)
knn.train <- train(churn~.,data=churn_train, method
="knn", trControl=control, tuneGrid=grid1)
```

```
Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was k = 15.
```

```
15   0.8670736   0.6290684
```

**->최적의 K값은 15로 도출되었고, A: 0.8670736 & K: 0.6290684의 값을 가짐**

```
> varImp(knn.train, scale=F)  #변수의 중요도 확인
ROC curve variable importance

                   Importance
total_contact          0.8019
duration               0.7381
total_service_issue    0.6843
long_charge            0.6571
monthly_charge         0.6456
noreferral             0.6376
no_dependents          0.6189
download               0.6033
extra_data_charge      0.5843
cltv                   0.5817
age                    0.5719
population             0.5341
```

**>독립변수들의 중요도 확인, 'total_contact' 변수의 중요도가 제일 높음**

● **STEP4: test 데이터를 이용한 KNN 모델 성능 평가**

```
pred.test <- predict(knn.train, newdata = churn_test)
pred.test
library(caret)
confusionMatrix(pred.test, churn_test$churn)
```

```
          Accuracy : 0.8696
            95% CI : (0.8544, 0.8837)
No Information Rate : 0.7382
P-Value [Acc > NIR] : < 2.2e-16

             Kappa : 0.6296
```

**A:0.8670736 & K:0.6290684/ A:0.8696 & K: 0.6296**

**(최적의 k일 때의 값들 vs matrix 일 때의 값들)**

**train모델보다 test모델의 값이 소폭이나마 증가되었기 때문에 현재 만든 KNN 모델이 train에만 너무 잘 부합하는 과적합의 문제가 발생하지 않고 test과 같이 다른 데이터 셋에도 잘 적용할 수 있다고 볼 수 있음(과대적합은 우려할 필요가 없음)**

**단, Kappa의 값이 0.6으로 다소 낮게 나와서 k의 범위를 3~10으로 좁게 재설정하여 STEP3~STEP4 과정 실행**

```
grid2 <- expand.grid(k=3:10) #근접한 이웃개수 후보군
설정(k의 범위를 좁게 수정)
set.seed(135)
knn.train2 <- train(churn~.,data=churn_train, method
="knn", trControl=control, tuneGrid=grid2)
```

```
Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was k = 9.
```

```
  9   0.8649737  0.6288988
```

**->최적의 K값은 9로 도출되었고, A: 0.8649737 & K: 0.6288988의 값을 가짐**

```
## 위에서 도출된 최적의 k값들을 기준으로 범위 재설정##
#범위를 3~10와 3~30으로 설정했을 때 각각 최적의 k가 9와
15가 나왔기에 범위를 9~15으로 설정함
grid3 <- expand.grid(k=9:15) #k의 값을 9~15으로 설정
set.seed(135)
knn.train3 <- train(churn~.,data=churn_train, method
="knn", trControl=control, tuneGrid=grid3)
```

```
Accuracy was used to select the optimal model using
 the largest value.
The final value used for the model was k = 15.
```

**-> 즉 최종적으로 선정된 최적의 K값은 15가 됨**

```
               Accuracy : 0.8696
                 95% CI : (0.8544, 0.8837)
    No Information Rate : 0.7382
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6296

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9605
            Specificity : 0.6131
         Pos Pred Value : 0.8750
         Neg Pred Value : 0.8463
             Prevalence : 0.7382
         Detection Rate : 0.7090
   Detection Prevalence : 0.8103
      Balanced Accuracy : 0.7868
```

->최적의 K가 15일 때의 KNN 모델의 성능들

● STEP5: KNN 모델 성능 개선

```
library(kknn)
set.seed(777)
kknn.train <- train.kknn(churn~., data = churn_train,
kmax = 25, distance = 2, kernel = c("rectangular",
"triangular", "Epanerchnikov", "biweight", "triweight",
"cosine", "inversion", "Gaussian", "rank", "optimal"))
```

```
Type of response variable: nominal
Minimal misclassification: 0.129697
Best kernel: rectangular
Best k: 15
```

-> 최적의 K값은 15, 방법은 rectangular, Accuaracy=1-0.129697(0.870303)

-> 기존 0.8670736보다 성능이 소폭 증가

```
kknn.pred.test <- predict(kknn.train, newdata =
churn_test)
confusionMatrix(kknn.pred.test, churn_test$churn)
```

```
  Accuracy : 0.871
```

->기존 0.8696보다 성능이 소폭 증가

● STEP6: 성능 개선된 KNN모델과 기존 KNN 모델을 활용하여 예측

```
churn_pred <- read_csv("churn_predict.csv")
churn_pred$id <- NULL #불필요한 변수 제거
churn_pred <- as.data.frame(scale(churn_pred))

pred.test2 <- predict(knn.train, newdata = churn_pred)
pred.test3<- predict(kknn.train, newdata = churn_pred)
```

```
> pred.test2
 [1] 0 1 0 0 1 0 0 0 0 0
Levels: 0 1
> pred.test3
 [1] 0 1 0 0 1 0 0 0 0 0
Levels: 0 1
```

-> 예측결과는 두 모델 다 0 1 0 0 1 0 0 0 0 0로 동일

-> churn_predict.csv의 새로운 열 개 사례의 종속변수 값은 0 1 0 0 1 0 0 0 0 0