KNN Model

bank <- read.csv("~/Desktop/MSA\_6440/customer\_churn/data/processed/BankChurners\_filtered.csv")  
source('/Users/JonathanVoth/Desktop/R Codes/myfunctions.R')  
  
# Creating dummy variables for Education  
bank$High\_School <- ifelse(bank$Education\_Level == 'High School', 1, 0)  
bank$College <- ifelse(bank$Education\_Level == 'College', 1, 0)  
bank$Graduate <- ifelse(bank$Education\_Level == 'Graduate', 1, 0)  
bank$Uneducated <- ifelse(bank$Education\_Level == 'Uneducated', 1, 0)  
bank$Post\_Graduate <- ifelse(bank$Education\_Level == 'Post-Graduate', 1, 0)  
bank$Doctorate <- ifelse(bank$Education\_Level == 'Doctorate', 1, 0)  
  
# Creating dummy variables for Marital Status  
bank$Married <- ifelse(bank$Marital\_Status == 'Married', 1, 0)  
bank$Single <- ifelse(bank$Marital\_Status == 'Single', 1, 0)  
bank$Divorced <- ifelse(bank$Marital\_Status == 'Divorced', 1, 0)  
  
# Creating dummy variables for Gender  
bank$Male <- ifelse(bank$Gender == 'M', 1, 0)  
bank$Female <- ifelse(bank$Gender == 'F', 1, 0)  
  
# Creating dummy variables for Income  
bank$Income\_5 <- ifelse(bank$Income\_Category == '$120K +', 1, 0)  
bank$Income\_4 <- ifelse(bank$Income\_Category == '$80K - $120K', 1, 0)  
bank$Income\_3 <- ifelse(bank$Income\_Category == '$60K - $80K', 1, 0)  
bank$Income\_2 <- ifelse(bank$Income\_Category == '$40K - $60K', 1, 0)  
bank$Income\_1 <- ifelse(bank$Income\_Category == 'Less than $40K', 1, 0)  
  
bank <- bank[,c(-1,-2,-5,-7:-10)]  
  
RNGkind (sample.kind = "Rounding")

## Warning in RNGkind(sample.kind = "Rounding"): non-uniform 'Rounding' sampler  
## used

set.seed(0)  
p2 <- partition.2(bank, 0.7) # 70:30 partition  
training.data <- p2$data.train  
test.data <- p2$data.test

## KNN Model

# Scaling the data  
training.scaled <- scale(training.data[,-1], center = TRUE, scale = TRUE)  
attrib <- attributes(training.scaled)  
test.scaled <- scale(test.data[,-1], center = attrib$`scaled:center`, scale = attrib$`scaled:scale`)  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

library(FNN)  
  
# Fitting 10-fold CV model  
set.seed(0)  
train\_control <- trainControl(method = "cv", number = 10)   
knn\_cv <- train(Attrition\_Flag ~ ., data = training.data, method = "knn", trControl = train\_control, preProcess = c("center","scale"), tuneGrid = data.frame(k = seq(1,7,1)), metric = "Kappa")  
  
# Final model  
print(knn\_cv)

## k-Nearest Neighbors   
##   
## 4957 samples  
## 30 predictor  
## 2 classes: 'Attrited Customer', 'Existing Customer'   
##   
## Pre-processing: centered (30), scaled (30)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4461, 4461, 4461, 4461, 4461, 4462, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.8470817 0.3635954  
## 2 0.8396151 0.3368632  
## 3 0.8678617 0.3699361  
## 4 0.8654415 0.3585564  
## 5 0.8674597 0.3313508  
## 6 0.8656452 0.3161207  
## 7 0.8680661 0.2924081  
##   
## Kappa was used to select the optimal model using the largest value.  
## The final value used for the model was k = 3.

knn\_cv$finalModel

## 3-nearest neighbor model  
## Training set outcome distribution:  
##   
## Attrited Customer Existing Customer   
## 776 4181

# Fit model on test data with k = 3  
knn.test <- knn(train = training.scaled, test = test.scaled, cl = training.data[,1], k = 3)  
confusionMatrix(as.factor(knn.test), as.factor(test.data[,1]), positive = "Existing Customer")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Attrited Customer Existing Customer  
## Attrited Customer 133 55  
## Existing Customer 204 1732  
##   
## Accuracy : 0.8781   
## 95% CI : (0.8634, 0.8917)   
## No Information Rate : 0.8413   
## P-Value [Acc > NIR] : 9.888e-07   
##   
## Kappa : 0.4434   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9692   
## Specificity : 0.3947   
## Pos Pred Value : 0.8946   
## Neg Pred Value : 0.7074   
## Prevalence : 0.8413   
## Detection Rate : 0.8154   
## Detection Prevalence : 0.9115   
## Balanced Accuracy : 0.6819   
##   
## 'Positive' Class : Existing Customer  
##

## KNN with Undersampling

set.seed(0)  
all\_exist <- training.data[which(training.data$Attrition\_Flag == 'Existing Customer'),]  
all\_attrit <- training.data[which(training.data$Attrition\_Flag == 'Attrited Customer'),]  
  
random <- sample(1:nrow(all\_exist), nrow(all\_attrit), replace = FALSE)  
train.data <- all\_exist[random,]  
  
training.under <- rbind(all\_attrit, train.data)  
table(training.under$Attrition\_Flag)

##   
## Attrited Customer Existing Customer   
## 776 776

# Scaling the data  
train.scaled.under <- scale(training.under[,-1], center = TRUE, scale = TRUE)  
attrib.under <- attributes(train.scaled.under)  
test.scaled.under <- scale(test.data[,-1], center = attrib.under$`scaled:center`, scale = attrib.under$`scaled:scale`)  
  
# Fitting 10-fold CV model  
set.seed(0)  
train\_control <- trainControl(method = "cv", number = 10)   
knn\_under <- train(Attrition\_Flag ~ ., data = training.under, method = "knn", trControl = train\_control, preProcess = c("center","scale"), tuneLength = 20, metric = "Kappa")  
# tuneGrid = data.frame(k = seq(33,45,1))  
  
# Final model  
print(knn\_under)

## k-Nearest Neighbors   
##   
## 1552 samples  
## 30 predictor  
## 2 classes: 'Attrited Customer', 'Existing Customer'   
##   
## Pre-processing: centered (30), scaled (30)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 1397, 1396, 1397, 1397, 1397, 1397, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 5 0.7538443 0.5077443  
## 7 0.7609370 0.5219096  
## 9 0.7622565 0.5245345  
## 11 0.7577027 0.5154061  
## 13 0.7544935 0.5090093  
## 15 0.7609120 0.5218486  
## 17 0.7680213 0.5360536  
## 19 0.7673552 0.5347396  
## 21 0.7692781 0.5385998  
## 23 0.7673552 0.5347441  
## 25 0.7686373 0.5372984  
## 27 0.7680295 0.5360462  
## 29 0.7647663 0.5295537  
## 31 0.7679922 0.5359911  
## 33 0.7718840 0.5437930  
## 35 0.7776946 0.5553970  
## 37 0.7828352 0.5656818  
## 39 0.7764043 0.5528075  
## 41 0.7738195 0.5476279  
## 43 0.7821610 0.5642996  
##   
## Kappa was used to select the optimal model using the largest value.  
## The final value used for the model was k = 37.

knn\_under$finalModel

## 37-nearest neighbor model  
## Training set outcome distribution:  
##   
## Attrited Customer Existing Customer   
## 776 776

# Fit model on test data with k = 37  
knn.test.under <- knn(train = train.scaled.under, test = test.scaled.under, cl = training.under[,1], k = 37)  
confusionMatrix(as.factor(knn.test.under), as.factor(test.data[,1]), positive = "Existing Customer")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Attrited Customer Existing Customer  
## Attrited Customer 267 337  
## Existing Customer 70 1450  
##   
## Accuracy : 0.8084   
## 95% CI : (0.791, 0.8249)   
## No Information Rate : 0.8413   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4569   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8114   
## Specificity : 0.7923   
## Pos Pred Value : 0.9539   
## Neg Pred Value : 0.4421   
## Prevalence : 0.8413   
## Detection Rate : 0.6827   
## Detection Prevalence : 0.7156   
## Balanced Accuracy : 0.8019   
##   
## 'Positive' Class : Existing Customer  
##

## KNN with Imortant Variables

train.important <- training.data[,c(-16:-31)]  
test.important <- test.data[,c(-16:-31)]  
  
# Scaling the data  
train.scaled.imp <- scale(train.important[,-1], center = TRUE, scale = TRUE)  
attrib.under.imp <- attributes(train.scaled.imp)  
test.scaled.imp <- scale(test.important[,-1], center = attrib.under.imp$`scaled:center`, scale = attrib.under.imp$`scaled:scale`)  
  
# Fitting 10-fold CV model  
set.seed(0)  
train\_control <- trainControl(method = "cv", number = 10)   
knn\_cv <- train(Attrition\_Flag ~ ., data = train.important, method = "knn", trControl = train\_control, preProcess = c("center","scale"), tuneGrid = data.frame(k = seq(1,7,1)), metric = "Kappa")  
  
# Final model  
print(knn\_cv)

## k-Nearest Neighbors   
##   
## 4957 samples  
## 14 predictor  
## 2 classes: 'Attrited Customer', 'Existing Customer'   
##   
## Pre-processing: centered (14), scaled (14)   
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 4461, 4461, 4461, 4461, 4461, 4462, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.8991377 0.5990960  
## 2 0.8967147 0.5917715  
## 3 0.9168870 0.6515683  
## 4 0.9112390 0.6301626  
## 5 0.9176914 0.6474263  
## 6 0.9166817 0.6383733  
## 7 0.9186995 0.6421482  
##   
## Kappa was used to select the optimal model using the largest value.  
## The final value used for the model was k = 3.

knn\_cv$finalModel

## 3-nearest neighbor model  
## Training set outcome distribution:  
##   
## Attrited Customer Existing Customer   
## 776 4181

# Fit model on test data with k = 3  
knn.test <- knn(train = train.scaled.imp, test = test.scaled.imp, cl = training.data[,1], k = 3)  
confusionMatrix(as.factor(knn.test), as.factor(test.data[,1]), positive = "Existing Customer")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Attrited Customer Existing Customer  
## Attrited Customer 215 61  
## Existing Customer 122 1726  
##   
## Accuracy : 0.9138   
## 95% CI : (0.9011, 0.9254)   
## No Information Rate : 0.8413   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6517   
##   
## Mcnemar's Test P-Value : 9.193e-06   
##   
## Sensitivity : 0.9659   
## Specificity : 0.6380   
## Pos Pred Value : 0.9340   
## Neg Pred Value : 0.7790   
## Prevalence : 0.8413   
## Detection Rate : 0.8126   
## Detection Prevalence : 0.8701   
## Balanced Accuracy : 0.8019   
##   
## 'Positive' Class : Existing Customer  
##