R Project

Grant Zhong

7/25/2019

This is my R Markdown file for the online shopper project.

```r
rm(list=ls())
setwd("~/Desktop/MSBA/Predictive Modeling/online_shopper_project")
```

```r
rm(list=ls())
```

Run kNN with the original data

```r
library(class)
library(kknn)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(naniar)
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2

##
## Attaching package: 'caret'

## The following object is masked from 'package:kknn':
##
##     contr.dummy
```

```r
library(dplyr)
library(ISLR)


data <- read.csv('online_shoppers_intention.csv') %>%
```

```r
naniar::replace_with_na_at(.vars = c("Administrative", "Administrative_Duration",
                                       "Informational", "Informational_Duration",
                                       "ProductRelated", "ProductRelated_Duration"),
                             condition = ~.x == -1) %>%

  transform(OperatingSystems=as.factor(OperatingSystems),
            Browser=as.factor(Browser),
            Region=as.factor(Region),
            TrafficType=as.factor(TrafficType))

set.seed(1)

data[is.na(data)]<-0
rand = sample(1:nrow(data),0.8*nrow(data))
norm <- function(x){
  (x-min(x))/(max(x)-min(x))
}

data_norm <- as.data.frame(lapply(data[,c(1,2,3,4,5,6,7,8,9,10,17)], norm))
summary(data_norm)
```

```
##  Administrative    Administrative_Duration Informational
##  Min.   :0.00000   Min.   :0.000000        Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.000000        1st Qu.:0.00000
##  Median :0.03704   Median :0.002152        Median :0.00000
##  Mean   :0.08575   Mean   :0.023778        Mean   :0.02098
##  3rd Qu.:0.14815   3rd Qu.:0.027438        3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.000000        Max.   :1.00000
##  Informational_Duration ProductRelated     ProductRelated_Duration
##  Min.   :0.00000        Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.00000        1st Qu.:0.009929   1st Qu.:0.002877
##  Median :0.00000        Median :0.025532   Median :0.009362
##  Mean   :0.01352        Mean   :0.045004   Mean   :0.018675
##  3rd Qu.:0.00000        3rd Qu.:0.053901   3rd Qu.:0.022887
##  Max.   :1.00000        Max.   :1.000000   Max.   :1.000000
##   BounceRates         ExitRates          PageValues         SpecialDay
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.07143   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.01544   Median :0.12522   Median :0.00000   Median :0.00000
##  Mean   :0.11064   Mean   :0.21477   Mean   :0.01628   Mean   :0.06143
##  3rd Qu.:0.08333   3rd Qu.:0.25000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##     Weekend
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.2326
##  3rd Qu.:0.0000
##  Max.   :1.0000
```

```r
train = data_norm[rand,]
test = data_norm[-rand,]
```

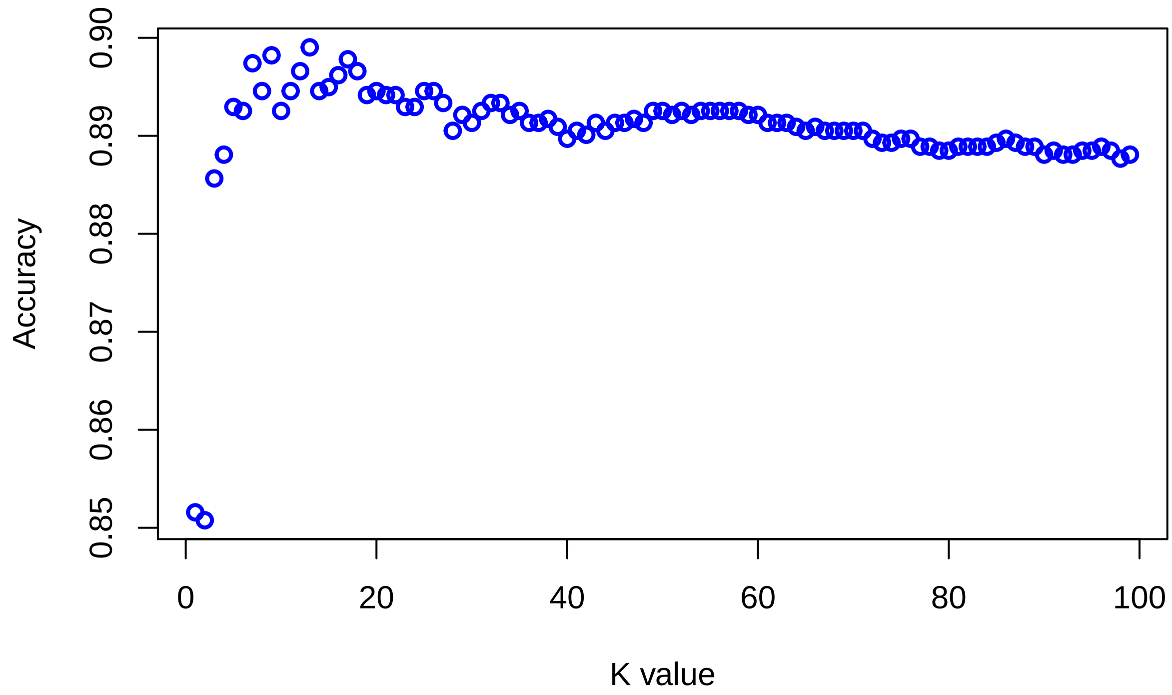
```r
train_revenue = data[rand,18]
test_revenue = data[-rand,18]

near <- knn(train,test,cl=train_revenue,k=25)
tbl = table(test_revenue,near)
accuracy = sum(diag(tbl))/sum(tbl)
135/(135+45)*100
```

```
## [1] 75
```

```r
cat('The accuracy when we use k=25 is',round(accuracy,4),'\n')
```

```
## The accuracy when we use k=25 is 0.8946
```

```r
overall_accuracy = NULL

#Find best k value to use for model to maximize accuracy (most accurate predictions)

for(i in 1:99){

  near = knn(train,test,cl=train_revenue,k=i)
  d = table(test_revenue,near)
  accuracy_i = sum(diag(d))/sum(d)

  overall_accuracy = c(overall_accuracy,accuracy_i)
}

plot(overall_accuracy,xlab='K value',ylab='Accuracy',col=4,lwd=2)
```




```
best = which.max(overall_accuracy)
cat('The best k value to use for best accuracy is',best,'.')
```

```
## The best k value to use for best accuracy is 13 .
```

```
near_best = knn(train,test,cl=train_revenue,k=13)
tbl_best= table(test_revenue,near_best)
accuracy_best = sum(diag(tbl_best))/sum(tbl_best)

cat('The accuracy when we use k=13 is', round(accuracy_best,4))
```

```
## The accuracy when we use k=13 is 0.899
```

```
confusionMatrix(tbl_best,positive='TRUE')
```

```
## Confusion Matrix and Statistics
##
##              near_best
## test_revenue FALSE TRUE
##        FALSE  2067   49
##        TRUE    200  150
##
##                Accuracy : 0.899
##                  95% CI : (0.8865, 0.9106)
```

```
##     No Information Rate : 0.9193
##     P-Value [Acc > NIR] : 0.9998
##
##                   Kappa : 0.4944
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.75377
##             Specificity : 0.91178
##          Pos Pred Value : 0.42857
##          Neg Pred Value : 0.97684
##              Prevalence : 0.08070
##          Detection Rate : 0.06083
##    Detection Prevalence : 0.14193
##       Balanced Accuracy : 0.83277
##
##        'Positive' Class : TRUE
##
```

```r
#Calculate precision, recall, and F scores for best model

precision = (150/(150+49))
cat('The precision of the kNN model is',precision,'\n')
```

```
## The precision of the kNN model is 0.7537688
```

```r
recall = 150/350
cat('The recall of the kNN model is',recall,'\n')
```

```
## The recall of the kNN model is 0.4285714
```

```r
f1_score = 2*precision*recall/(precision+recall)
cat('The F1 score of the model is',f1_score,'\n')
```

```
## The F1 score of the model is 0.5464481
```

Run kNN with SMOTE data

```r
library(class)
library(kknn)
library(dplyr)
library(naniar)
library(caret)
library(dplyr)
library(ISLR)
library(DMwR)
```

```
## Loading required package: grid
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

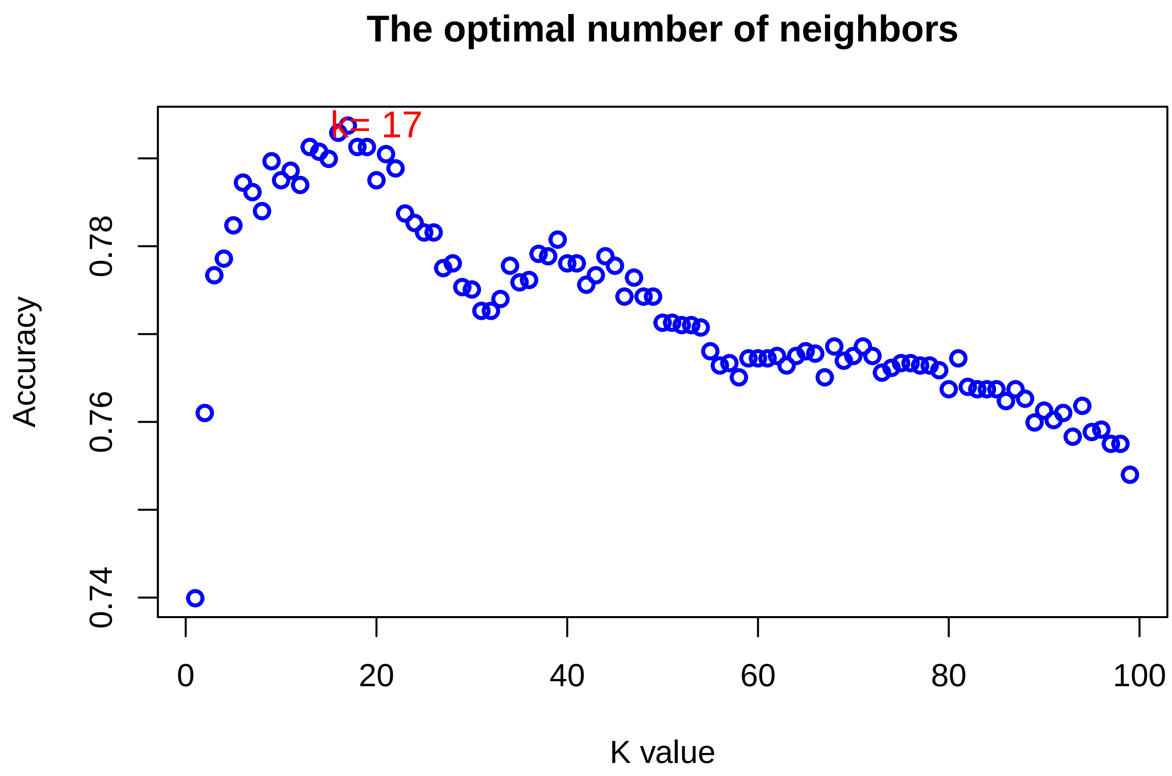
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
data <- read.csv('online_shoppers_intention.csv') %>%
naniar::replace_with_na_at(.vars = c("Administrative", "Administrative_Duration",
                                       "Informational", "Informational_Duration",
                                       "ProductRelated", "ProductRelated_Duration"),
                             condition = ~.x == -1) %>%

  transform(OperatingSystems=as.factor(OperatingSystems),
            Browser=as.factor(Browser),
            Region=as.factor(Region),
            TrafficType=as.factor(TrafficType))

data$missing_values <- apply(data, 1, function(x) any(is.na(x)))
data[is.na(data)]<-0
data <- subset(data, select=-19)

set.seed(1)
rand = sample(nrow(data),0.7*nrow(data))
train = data[rand,]
test = data[-rand,]


set.seed(1)
train$Revenue <- as.factor(train$Revenue)
smote_train_knn <- SMOTE(Revenue~.,data =train)

# Smote fit the data such that Weekend column was converted to numeric. Need to change it back to logic
# filter(smote_train, smote_train$Weekend >= 0.5)$Weekend = TRUE
# filter(smote_train, smote_train$Weekend < 0.5)$Weekend = FALSE

temp <- smote_train_knn$Weekend
temp[temp >= 0.5] = TRUE
temp[temp < 0.5] = FALSE
smote_train_knn$Weekend = sapply(temp, as.logical)
table(smote_train_knn$Revenue)
```

```
##
## FALSE  TRUE
##  5368  4026
```

```r
smote_train_knn$missing_values <- apply(smote_train_knn, 1, function(x) any(is.na(x)))
smote_train_knn[is.na(smote_train_knn)]<-0

test$missing_values <- apply(test, 1, function(x) any(is.na(x)))
test[is.na(test)]<-0

smote_train_knn <- subset(smote_train_knn, select=-c(Month,VisitorType,Weekend,missing_values))
test <- subset(test, select=-c(Month,VisitorType,Weekend,missing_values))

#Run kNN with the smote dataset
```

```r
near <- knn(smote_train_knn[,1:14],test[,1:14],cl=smote_train_knn$Revenue,k=13)
tbl = table(test$Revenue,near)
accuracy = sum(diag(tbl))/sum(tbl)

overall_accuracy = NULL

for(i in 1:99){

  near = knn(smote_train_knn[,1:14],test[,1:14],cl=smote_train_knn$Revenue,k=i)
  d = table(test$Revenue,near)
  accuracy_i = sum(diag(d))/sum(d)

  overall_accuracy = c(overall_accuracy,accuracy_i)
}

plot(overall_accuracy,xlab='K value',ylab='Accuracy',main = 'The optimal number of neighbors',col=4,lwd=
text(20,overall_accuracy[17]+0.0002,paste("k=",17),col=2,cex=1.2)
```




```r
best = which.max(overall_accuracy)
cat('The best k value to use for best accuracy is',best,'.')
```

```
## The best k value to use for best accuracy is 17 .
```

```r
near_best = knn(smote_train_knn[,1:14],test[,1:14],cl=smote_train_knn$Revenue,k=17)
tbl_best= table(test$Revenue,near_best)
accuracy_best = sum(diag(tbl_best))/sum(tbl_best)

cat('The accuracy when we use k=17 is', round(accuracy_best,4))
```

```
## The accuracy when we use k=17 is 0.7932
```

```r
confusionMatrix(tbl_best,positive='TRUE')
```

```
## Confusion Matrix and Statistics
##
##        near_best
##         FALSE TRUE
##   FALSE  2584  549
##   TRUE    216  350
##
##                Accuracy : 0.7932
##                  95% CI : (0.7798, 0.8061)
##     No Information Rate : 0.757
##     P-Value [Acc > NIR] : 9.58e-08
##
##                   Kappa : 0.3571
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.38932
##             Specificity : 0.92286
##          Pos Pred Value : 0.61837
##          Neg Pred Value : 0.82477
##              Prevalence : 0.24304
##          Detection Rate : 0.09462
##    Detection Prevalence : 0.15301
##       Balanced Accuracy : 0.65609
##
##        'Positive' Class : TRUE
##
```

```r
confusionMatrix(tbl,positive='TRUE')
```

```
## Confusion Matrix and Statistics
##
##        near
##         FALSE TRUE
##   FALSE  2556  577
##   TRUE    198  368
##
##                Accuracy : 0.7905
##                  95% CI : (0.777, 0.8035)
##     No Information Rate : 0.7445
##     P-Value [Acc > NIR] : 3.384e-11
```

```
##
##                   Kappa : 0.3657
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.38942
##             Specificity : 0.92810
##          Pos Pred Value : 0.65018
##          Neg Pred Value : 0.81583
##              Prevalence : 0.25547
##          Detection Rate : 0.09949
##    Detection Prevalence : 0.15301
##       Balanced Accuracy : 0.65876
##
##        'Positive' Class : TRUE
##
```

```r
#Calculate precision, recall, and F1 score for SMOTE model

precision = (350/(350+549))
cat('The precision of the kNN model is',precision,'\n')
```

```
## The precision of the kNN model is 0.3893215
```

```r
recall = 350/566
cat('The recall of the kNN model is',recall,'\n')
```

```
## The recall of the kNN model is 0.6183746
```

```r
f1_score = 2*precision*recall/(precision+recall)
cat('The F1 score of the model is',f1_score,'\n')
```

```
## The F1 score of the model is 0.4778157
```

Use 10-fold CV with SMOTE data

```r
library(caret)
trctrl <- trainControl(method='repeatedcv',number=10,repeats = 10)
knn_cv <- train(Revenue~.,data = smote_train_knn,method = 'knn', trControl=trctrl)

test_cv <- predict(knn_cv, newdata=test)
tbl_cv= table(test$Revenue,test_cv)
accuracy_cv = sum(diag(tbl_cv))/sum(tbl_cv)

cat('The accuracy for our 10-fold model is ', round(accuracy_cv,4))
```

```
## The accuracy for our 10-fold model is  0.7799
```

```r
confusionMatrix(tbl_cv,positive='TRUE')
```

```
## Confusion Matrix and Statistics
##
##          test_cv
##           FALSE TRUE
##   FALSE  2535  598
##   TRUE    216  350
##
##                Accuracy : 0.7799
##                  95% CI : (0.7662, 0.7932)
##     No Information Rate : 0.7437
##     P-Value [Acc > NIR] : 1.626e-07
##
##                   Kappa : 0.3349
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.36920
##             Specificity : 0.92148
##          Pos Pred Value : 0.61837
##          Neg Pred Value : 0.80913
##              Prevalence : 0.25629
##          Detection Rate : 0.09462
##    Detection Prevalence : 0.15301
##       Balanced Accuracy : 0.64534
##
##        'Positive' Class : TRUE
##
```

```r
#Calculate precision, recall, and F1 score for 10-fold model

precision_cv = 350/(350+598)
recall_cv = 350/566
F1_cv = 2*precision_cv*recall_cv/(recall_cv+precision_cv)

cat('The precision of the 10-fold model is',precision_cv,'\n')
```

```
## The precision of the 10-fold model is 0.3691983
```

```r
cat('The recall of the k-fold model is',recall_cv,'\n')
```

```
## The recall of the k-fold model is 0.6183746
```

```r
cat('The F1 score of the 10-fold model is',F1_cv,'\n')
```

```
## The F1 score of the 10-fold model is 0.4623514
```

kNN model with SMOTE training set and 10-fold CV kNN model with SMOTE training set had very similar results. kNN model with the original data had higher accuracy, precision, and lower recall (resulting in a higher F1 score) but our original training set had mostly FALSE classes with a baseline of 85% accuracy if predicted all FALSE.