```
##Data description

###
---------------------------------------------------------------------------
# This dataset contains information of cars purchased at the Auction.
# We will use this file to predict the quality of buying decisions and
visualize decision processes.

# VARIABLE DESCRIPTIONS:
#Auction: Auction provider at which the  vehicle was purchased
#Color: Vehicle Color
#IsBadBuy: Identifies if the kicked vehicle was an avoidable purchase
#MMRCurrentAuctionAveragePrice: Acquisition price for this vehicle in average
condition as of current day
#Size: The size category of the vehicle (Compact, SUV, etc.)
#TopThreeAmericanName:Identifies if the manufacturer is one of the top three
American manufacturers
#VehBCost: Acquisition cost paid for the vehicle at time of purchase
#VehicleAge: The Years elapsed since the manufacturer's year
#VehOdo: The vehicles odometer reading
#WarrantyCost: Warranty price (term=36month  and millage=36K)
#WheelType: The vehicle wheel type description (Alloy, Covers)
###
---------------------------------------------------------------------------


# 1. Import the datadet
carAuction <- read.csv(file = "carAuction.csv", stringsAsFactors = FALSE)

# 2. str() shows the structure of data
str(carAuction)

# 3. summary() shows the mean and the five-number statistics indicating the
spread of each column's values
summary(carAuction)

# 4. Change all categorical variables to factors
carAuction$Auction <- factor(carAuction$Auction)
carAuction$Color <- factor(carAuction$Color)
carAuction$IsBadBuy <- factor(carAuction$IsBadBuy)
carAuction$Size <- factor(carAuction$Size)
carAuction$TopThreeAmericanName <- factor(carAuction$TopThreeAmericanName)
carAuction$WheelType <- factor(carAuction$WheelType)
str(carAuction)
summary(carAuction)

# 5. Partition the dataset: 70% for training, 30% for testing
library(caret)
set.seed(1)
train_index <- createDataPartition(carAuction$IsBadBuy, p=0.7, list=FALSE)
datTrain <- carAuction[train_index,]
datTest <- carAuction[-train_index,]

# 6. Check the rows and porportion of target variable for both training and
testing datasets
nrow(datTrain)
```

```
nrow(datTest)
prop.table(table(datTrain$IsBadBuy))#check badnuy distribution
prop.table(table(datTest$IsBadBuy))

# 7. Build KNN model with caret package. Set k=1
library(rminer)
model <- knn3(IsBadBuy~., datTrain, k=1)#isbadbuy target variable.
#the dot before comma means we will use all the other variables to predict
#that car is bad buy. #then we specify training data set.
#k=1 specifies the number of neighbors we will use to predict that car is
#bad buy; in this case k =1 so we will only use the closest neighbor to
predict
#that the car is a badbuy.

# 8. Make predictions on both training and tessting sets
prediction_on_train <- predict(model, datTrain)
prediction_on_test <- predict(model, datTest)

# 9. Generate evaluation results on training and testing data
mmetric(datTrain$IsBadBuy,prediction_on_train, metric="CONF")
mmetric(datTest$IsBadBuy,prediction_on_test, metric="CONF")
mmetric(datTrain$IsBadBuy,prediction_on_train,metric=c("ACC","PRECISION","TPR","F1"))
mmetric(datTest$IsBadBuy,prediction_on_test,metric=c("ACC","PRECISION","TPR","F1"))

  # a. Why we have perfect evaluation results on the training data?
#It was perfect because we used the closest neighbor (k=1). The closes
neighbor
#in this case is that car itself, so we used that same car to preict whether
its
#bad buy. For each car the closest neighbor is the car itself, so we used each
#vehicles own bad buy value to predict itself.
  # b. Does the KNN model with k=1 generalize well on the testing set? why?
#No, it doesnt because if we look at the accuracy there is a big gap.

# 10. Build KNN model with k=5
model2 <- knn3(IsBadBuy~., datTrain, k=5)

# 11. Make predictions on both training and tessting sets
prediction_on_train2 <- predict(model2, datTrain)#change model and test names
prediction_on_test2 <- predict(model2, datTest)

# 12. Generate evaluation results on training and testing data
mmetric(datTrain$IsBadBuy,prediction_on_train2, metric="CONF")
mmetric(datTest$IsBadBuy,prediction_on_test2, metric="CONF")
mmetric(datTrain$IsBadBuy,prediction_on_train2,metric=c("ACC","PRECISION","TPR","F1"))
mmetric(datTest$IsBadBuy,prediction_on_test2,metric=c("ACC","PRECISION","TPR","F1"))
#first model had perfect performance on the traiing dataset, but its
#actual performance was worse than the performance of this model.

# 10. Build KNN model with k=10
model3 <- knn3(IsBadBuy~., datTrain, k=10)

# 11. Make predictions on both training and tessting sets
prediction_on_train3 <- predict(model3, datTrain)#change model and test names
prediction_on_test3 <- predict(model3, datTest)
```

```
# 12. Generate evaluation results on training and testing data
mmetric(datTrain$IsBadBuy,prediction_on_train3, metric="CONF")
mmetric(datTest$IsBadBuy,prediction_on_test3, metric="CONF")
mmetric(datTrain$IsBadBuy,prediction_on_train3,metric=c("ACC","PRECISION","TPR","F1"))
mmetric(datTest$IsBadBuy,prediction_on_test3,metric=c("ACC","PRECISION","TPR","F1"))

  # a. Which KNN model is the best for identifying bad buy cars (k=1, 5 or
10)? and why?
#when we increase k, in this case, the model becomes less overfit on traing,
#and the gap of accuracy on training and testing gets smaller.
#First model is the best for identifying bad buy cars. We compare
#the performance on the testing set. We look at confusion matrix and see
#which model predicted badbuy cars better. The overall performance becmomes
#better as we increase k number.On badbuy = No class performance also becomes
better
#better as we increase the number of k. But F measure and recall value decline
#with each increase in k.

# 13. Define functions for cross validation
# Load packages for cross validation
library(matrixStats)
library(knitr)

# cross validation function for training
cv_knn_train <- function(df, target, nFolds, seedVal, metrics_list, k)
{
  # create folds using the assigned values
  set.seed(seedVal)
  folds = createFolds(df[,target],nFolds)
  # The lapply loop
  cv_results <- lapply(folds, function(x)
  {
    # data preparation:
    test_target <- df[x,target]
    test_input <- df[x,-target]
    train_target <- df[-x,target]
    train_input <- df[-x,-target]
    pred_model <- knn3(train_target ~ .,data = train_input, k=k)
    pred_train <- predict(pred_model, train_input)
    return(mmetric(train_target,pred_train,metrics_list))
  })
  # convert a list to a data frame using as.data.frame and convert this data
frame to a matrix before using rowSds()
  cv_results_m <- as.matrix(as.data.frame(cv_results))
  cv_mean<- as.matrix(rowMeans(cv_results_m))
  cv_sd <- as.matrix(rowSds(cv_results_m))
  colnames(cv_mean) <- "Mean"
  colnames(cv_sd) <- "Sd"
  # Combine and show cv_results and Means and Sds
  cv_all <- cbind(cv_results_m, cv_mean, cv_sd)
  kable(t(cv_all),digits=3)
}#this cross val function generates training perforamcne

# cross validation function for testing
cv_knn_test <- function(df, target, nFolds, seedVal, metrics_list, k)
{#this cross val generate testing performance
```

```
  # create folds using the assigned values
  set.seed(seedVal)
  folds = createFolds(df[,target],nFolds)
  # The lapply loop
  cv_results <- lapply(folds, function(x)
  {
    # data preparation:
    test_target <- df[x,target]
    test_input <- df[x,-target]
    train_target <- df[-x,target]
    train_input <- df[-x,-target]
    pred_model <- knn3(train_target ~ .,data = train_input,k=k)
    pred <- predict(pred_model, test_input)
    return(mmetric(test_target,pred,metrics_list))
  })
  cv_results_m <- as.matrix(as.data.frame(cv_results))
  cv_mean<- as.matrix(rowMeans(cv_results_m))
  cv_sd <- as.matrix(rowSds(cv_results_m))
  colnames(cv_mean) <- "Mean"
  colnames(cv_sd) <- "Sd"
  kable(t(cbind(cv_mean,cv_sd)),digits=3)
}

# 14. Apply cross validation on carAuction data with nFolds = 3 and k = 5
df = carAuction#we define our frame here, in this case whole dataset, we dont
#need to split it into training adn testing.
target = 3#which column is the target variable
nFolds = 3 #we want to do 3fold cross validation
seedVal = 1 #we want to split the data in 3fold here, so if we use same seed
#then everyone will get the same evaluation results.
metrics_list = c("ACC","PRECISION","TPR","F1")#what metrics we want to see
k = 5#set k value; we want 5 nearest neighbors to make prediction.
# Evaluation results on training data
cv_knn_train(df, target, nFolds, seedVal, metrics_list, k)#this function will
#give training performance
# Evaluation results on testing data
cv_knn_test(df, target, nFolds, seedVal, metrics_list, k)#runing this function
#gives test performance.

# Apply cross validation on carAuction data with nFolds = 10 and k = 5, and
generate evaluation results on training and testing data.
df = carAuction
target = 3
nFolds = 3
seedVal = 1
metrics_list = c("ACC","PRECISION","TPR","F1")
k = 10
cv_knn_train(df, target, nFolds, seedVal, metrics_list, k)
cv_knn_test(df, target, nFolds, seedVal, metrics_list, k)
```