1. This customer will be classified as loan acceptance = 1. That is user will take a loan.
2. K=3 gives the highest accuracy of 0.962. Therefore, k value should be taken as 3.

K Accuracy

1 0.9610

2 0.9545

3 0.9620

4 0.9550

5 0.9580

6 0.9495

7 0.9545

8 0.9485

9 0.9540

10 0.9505

11 0.9520

12 0.9470

13 0.9510

14 0.9445

1. Below is the confusion matrix for the validation data using k=3:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Actual Validation data** | |
|  |  | **0** | **1** |
| **Predicted Validation data** | **0** | 1792 | 73 |
| **1** | 3 | 132 |

1. Using k = 3, the new customer will not take personal loan from bank.

**Below is the code:**

# get data from csv file

UniversalBank <- read.csv("/Users/praveshgarg/Documents/Predictive analytics/5 homework/UniversalBank(1).csv", header = TRUE)

#Remove columns ID and ZipCode

UniversalBank <- UniversalBank[,-c(1,5)]

#Convert Education categorical variable to dummy variables

UniversalBank$Education\_1 <- ifelse(UniversalBank$Education == 1, 1, 0)

UniversalBank$Education\_2 <- ifelse(UniversalBank$Education == 2, 1, 0)

UniversalBank$Education\_3 <- ifelse(UniversalBank$Education == 3, 1, 0)

UniversalBank[,"Education"]<-NULL

# divide data to training (60%) and validation (40%) sets

set.seed(1)

train.index <- sample(row.names(UniversalBank), 0.6\*dim(UniversalBank)[1])

valid.index <- setdiff(row.names(UniversalBank), train.index)

train.df <- UniversalBank[train.index, ]

valid.df <- UniversalBank[valid.index, ]

## new customer data

new.df <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education\_1 = 0, Education\_2 = 1, Education\_3 = 0, Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1)

## KNN process

#create normalize train and test variable with original data

train.norm <- train.df

valid.norm <- valid.df

# normalize numeric data

install.packages("caret")

library(caret)

norm.values <- preProcess(train.df[, c(1:6)], method=c("center", "scale"))

train.norm[, c(1:6)] <- predict(norm.values, train.df[, c(1:6)])

valid.norm[, c(1:6)] <- predict(norm.values, valid.df[, c(1:6)])

new.norm[, c(1:5,9)] <- predict(norm.values, new.df[, c(1:5,9)])

# use knn() to compute knn.

# knn() is available in library FNN (provides a list of the nearest neighbors)

# and library class (allows a numerical output variable).

install.packages("FNN")

library(FNN)

nn <- knn(train = train.norm[, c(1:6,8:14)], test = new.norm,

cl = train.norm[, 7], k = 1)

nn[1]

## find best k value

# initialize a data frame with two columns: k, and accuracy.

accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))

# compute knn for different k on validation.

for(i in 1:14) {

knn.pred <- knn(train.norm[, c(1:6,8:14)], valid.norm[, c(1:6,8:14)],

cl = train.norm[, 7], k = i)

# valid.norm.df = as.factor(valid.norm.df)

accuracy.df[i, 2] <- confusionMatrix(knn.pred,as.factor(valid.norm[, 7]))$overall[1]

}

accuracy.df

##Creating confusion matrix for validation data using k=3

knn.pred2 <- knn(train.norm[, c(1:6,8:14)], valid.norm[, c(1:6,8:14)],

cl = train.norm[, 7], k = 3)

table(knn.pred2, valid.norm[,7])

##Classifying new customer using k=3

nn2 <- knn(train = train.norm[, c(1:6,8:14)], test = new.norm,

cl = train.norm[, 7], k = 3)

nn2[1]