Assignment\_2

Stephen Gombos

2025-09-28

# Install packages if you haven’t already

# install.packages(“caret”)

# install.packages(“dplyr”)

# install.packages(“class”)

# install.packages(“gmodels”)

library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

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

library(class)  
library(gmodels)

Load the UniversalBank CSV into Rstudio

bank.df <- read.csv("/Users/stephengombos/Documents/KSU MBA PROGRAM/Fall 2025 FUNDAMENTALS OF MACHINE LEARNING (BA-64060-002)/CSV Files/UniversalBank.CSV")

# Drop ID and ZIP.Code columns

bank.df <- bank.df %>% select(-ID, -ZIP.Code)

# Convert categorical Education variable into dummy variables

# The fullRank=TRUE argument creates k-1 dummies to avoid multicollinearity

bank.df$Education <- as.factor(bank.df$Education)  
dummy\_model <- dummyVars(~ Education, data = bank.df, fullRank = FALSE)  
education\_dummies <- predict(dummy\_model, newdata = bank.df)  
bank.df <- cbind(bank.df, education\_dummies)  
bank.df <- bank.df %>% select(-Education)

# Ensure the target variable is a factor for classification

bank.df$Personal.Loan <- as.factor(bank.df$Personal.Loan)

# View the structure of the processed data

str(bank.df)

## 'data.frame': 5000 obs. of 14 variables:  
## $ Age : int 25 45 39 35 35 37 53 50 35 34 ...  
## $ Experience : int 1 19 15 9 8 13 27 24 10 9 ...  
## $ Income : int 49 34 11 100 45 29 72 22 81 180 ...  
## $ Family : int 4 3 1 1 4 4 2 1 3 1 ...  
## $ CCAvg : num 1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...  
## $ Mortgage : int 0 0 0 0 0 155 0 0 104 0 ...  
## $ Personal.Loan : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ Securities.Account: int 1 1 0 0 0 0 0 0 0 0 ...  
## $ CD.Account : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ Online : int 0 0 0 0 0 1 1 0 1 0 ...  
## $ CreditCard : int 0 0 0 0 1 0 0 1 0 0 ...  
## $ Education.1 : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ Education.2 : num 0 0 0 1 1 1 1 0 1 0 ...  
## $ Education.3 : num 0 0 0 0 0 0 0 1 0 1 ...

##Normalization of the Data

# Partition the data (60% training, 40% validation)

set.seed(0101) # for reproducibility  
train.index <- createDataPartition(bank.df$Personal.Loan, p = 0.6, list = FALSE)  
train.df <- bank.df[train.index, ]  
valid.df <- bank.df[-train.index, ]

# Separate predictors and the target variable (loan)

train.labels <- train.df$Personal.Loan  
valid.labels <- valid.df$Personal.Loan

# Normalize the data

#Select all predictors except target for normalization —

predictor.names <- setdiff(names(train.df), "Personal.Loan")

# Normalize train and validation sets using all predictors including dummies

norm.values <- preProcess(train.df[, predictor.names], method = c("center", "scale"))  
train.norm.df <- predict(norm.values, train.df[, predictor.names])  
valid.norm.df <- predict(norm.values, valid.df[, predictor.names])

# Define the new customer data

new.customer <- data.frame(  
 Age = 40,  
 Experience = 10,  
 Income = 84,  
 Family = 2,  
 CCAvg = 2,  
 Mortgage = 0,  
 Securities.Account = 0,  
 CD.Account = 0,  
 Online = 1,  
 CreditCard = 1,  
 Education.1 = 0,  
 Education.2 = 1,   
 Education.3 = 0  
)

# Normalize the new customer data using the training set’s model

new.customer.norm <- predict(norm.values, new.customer)

# Perform k-NN classification with k=1

knn.pred.k1 <- knn(train = train.norm.df, test = new.customer.norm, cl = train.labels, k = 1)

# Print the classification result

cat("How would this customer be classified (k=1)?\n")

## How would this customer be classified (k=1)?

cat("Prediction:", as.character(knn.pred.k1), "\n")

## Prediction: 0

# Find the best k value

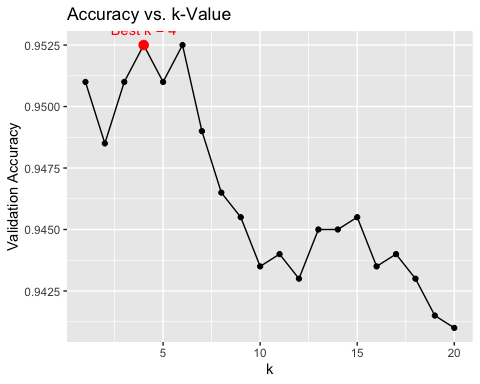
accuracy.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))  
  
for(i in 1:20) {  
 knn.pred <- knn(train.norm.df, valid.norm.df, cl = train.labels, k = i)  
 # In caret, the 'positive' class must be specified for metrics like Sensitivity  
 accuracy.df[i, 2] <- confusionMatrix(knn.pred, valid.labels, positive="1")$overall[1]  
}

# Find the k that maximizes accuracy

best.k <- accuracy.df$k[which.max(accuracy.df$accuracy)]

# Plot accuracy vs. k

ggplot(accuracy.df, aes(x = k, y = accuracy)) +  
 geom\_line() +  
 geom\_point() +  
 geom\_point(data = subset(accuracy.df, k == best.k), color = "red", size = 3) +  
 geom\_text(data = subset(accuracy.df, k == best.k),  
 aes(label = paste("Best k =", best.k)),  
 color = "red", vjust = -1, hjust = 0.5) +  
 labs(title = "Accuracy vs. k-Value",  
 x = "k",  
 y = "Validation Accuracy")



cat("\nWhat is a choice of k that balances between overfitting and ignoring predictor information?\n")

##   
## What is a choice of k that balances between overfitting and ignoring predictor information?

cat("The best value for k is:", best.k, "\n")

## The best value for k is: 4

# Generate predictions for the validation set using the best k

knn.pred.best <- knn(train.norm.df, valid.norm.df, cl = train.labels, k = best.k)

# Show the confusion matrix

cat("\nConfusion matrix for the validation data (k =", best.k, "):\n")

##   
## Confusion matrix for the validation data (k = 4 ):

confusionMatrix(knn.pred.best, valid.labels, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1786 69  
## 1 22 123  
##   
## Accuracy : 0.9545   
## 95% CI : (0.9444, 0.9632)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7057   
##   
## Mcnemar's Test P-Value : 1.42e-06   
##   
## Sensitivity : 0.6406   
## Specificity : 0.9878   
## Pos Pred Value : 0.8483   
## Neg Pred Value : 0.9628   
## Prevalence : 0.0960   
## Detection Rate : 0.0615   
## Detection Prevalence : 0.0725   
## Balanced Accuracy : 0.8142   
##   
## 'Positive' Class : 1   
##

#Classify the customer using the best k

knn.pred.best.k.customer <- knn(train = train.norm.df, test = new.customer.norm, cl = train.labels, k = best.k)  
  
cat("\nClassify the customer using the best k (k=", best.k, "):\n")

##   
## Classify the customer using the best k (k= 4 ):

cat("Prediction:", as.character(knn.pred.best.k.customer), "\n")

## Prediction: 0

# Repartition data: 50% training, 30% validation, 20% test

set.seed(0101)  
train.index.new <- createDataPartition(bank.df$Personal.Loan, p = 0.5, list = FALSE)  
train.df.new <- bank.df[train.index.new, ]  
temp.df <- bank.df[-train.index.new, ]  
  
valid.index.new <- createDataPartition(temp.df$Personal.Loan, p = 0.6, list = FALSE) # 60% of remaining 50% is 30%  
valid.df.new <- temp.df[valid.index.new, ]  
test.df.new <- temp.df[-valid.index.new, ] # Remaining 40% of 50% is 20%

# Separate labels

train.labels.new <- train.df.new$Personal.Loan  
valid.labels.new <- valid.df.new$Personal.Loan  
test.labels.new <- test.df.new$Personal.Loan

# Normalize the new partitions based on the new training set

norm.values.new <- preProcess(train.df.new[, -which(names(train.df.new) == "Personal.Loan")], method=c("center", "scale"))  
train.norm.new <- predict(norm.values.new, train.df.new[, -which(names(train.df.new) == "Personal.Loan")])  
valid.norm.new <- predict(norm.values.new, valid.df.new[, -which(names(valid.df.new) == "Personal.Loan")])  
test.norm.new <- predict(norm.values.new, test.df.new[, -which(names(test.df.new) == "Personal.Loan")])

# Get predictions for all three sets using best.k

train.pred.final <- knn(train.norm.new, train.norm.new, cl = train.labels.new, k = best.k)  
valid.pred.final <- knn(train.norm.new, valid.norm.new, cl = train.labels.new, k = best.k)  
test.pred.final <- knn(train.norm.new, test.norm.new, cl = train.labels.new, k = best.k)

# Generate and display confusion matrices

cat("\n--- Comparison of Confusion Matrices (k =", best.k, ") ---\n")

##   
## --- Comparison of Confusion Matrices (k = 4 ) ---

cat("\nTraining Set Performance:\n")

##   
## Training Set Performance:

print(confusionMatrix(train.pred.final, train.labels.new, positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2255 62  
## 1 5 178  
##   
## Accuracy : 0.9732   
## 95% CI : (0.9661, 0.9792)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8273   
##   
## Mcnemar's Test P-Value : 7.837e-12   
##   
## Sensitivity : 0.7417   
## Specificity : 0.9978   
## Pos Pred Value : 0.9727   
## Neg Pred Value : 0.9732   
## Prevalence : 0.0960   
## Detection Rate : 0.0712   
## Detection Prevalence : 0.0732   
## Balanced Accuracy : 0.8697   
##   
## 'Positive' Class : 1   
##

cat("\nValidation Set Performance:\n")

##   
## Validation Set Performance:

print(confusionMatrix(valid.pred.final, valid.labels.new, positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1343 59  
## 1 13 85  
##   
## Accuracy : 0.952   
## 95% CI : (0.9399, 0.9623)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 3.516e-12   
##   
## Kappa : 0.6774   
##   
## Mcnemar's Test P-Value : 1.137e-07   
##   
## Sensitivity : 0.59028   
## Specificity : 0.99041   
## Pos Pred Value : 0.86735   
## Neg Pred Value : 0.95792   
## Prevalence : 0.09600   
## Detection Rate : 0.05667   
## Detection Prevalence : 0.06533   
## Balanced Accuracy : 0.79035   
##   
## 'Positive' Class : 1   
##

cat("\nTest Set Performance:\n")

##   
## Test Set Performance:

print(confusionMatrix(test.pred.final, test.labels.new, positive = "1"))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 896 33  
## 1 8 63  
##   
## Accuracy : 0.959   
## 95% CI : (0.9448, 0.9704)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 3.732e-11   
##   
## Kappa : 0.7327   
##   
## Mcnemar's Test P-Value : 0.0001781   
##   
## Sensitivity : 0.6562   
## Specificity : 0.9912   
## Pos Pred Value : 0.8873   
## Neg Pred Value : 0.9645   
## Prevalence : 0.0960   
## Detection Rate : 0.0630   
## Detection Prevalence : 0.0710   
## Balanced Accuracy : 0.8237   
##   
## 'Positive' Class : 1   
##