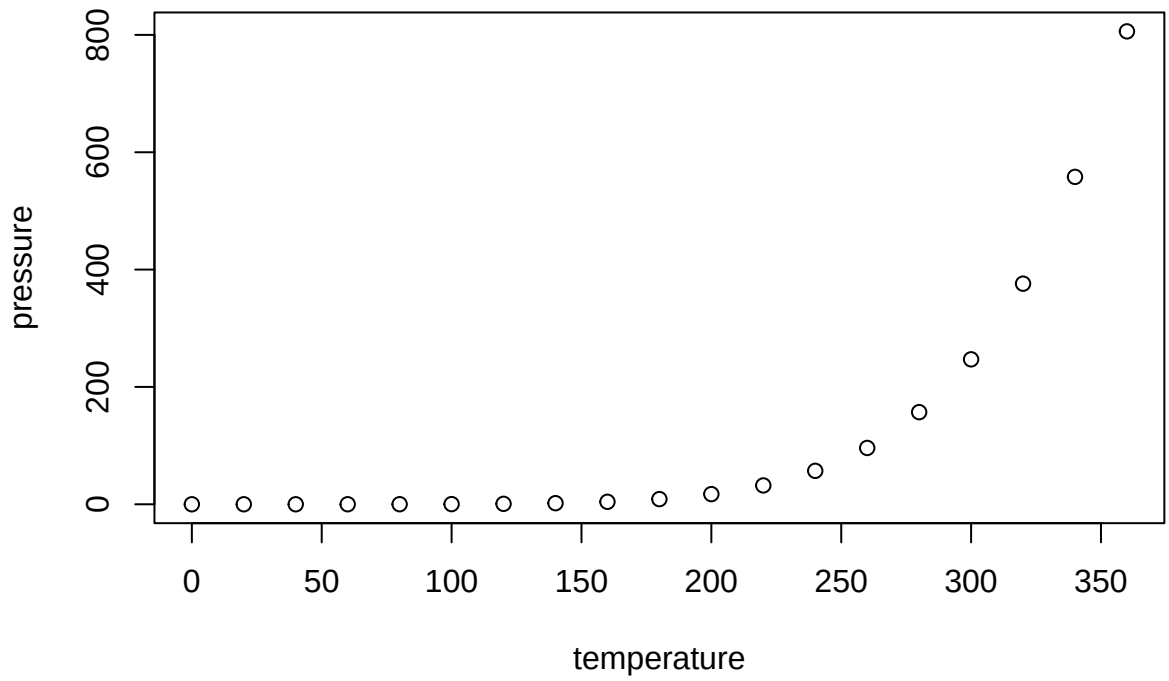
Assignment 2 - k-NN Classification for Universal Bank

Shahbaz

2025-09-26

Including Plots - You can also embed plots, for example:



Load required libraries

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(class)
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
```

Step 1 (Data Preparation/Loading)

```r
bd <- read.csv("/Users/shahbazshaikh/Downloads/UniversalBank.csv")
```

Removing ID and Zip Code Columns as Instructed

```r
bd <- bd[, !(names(bd) %in% c("ID", "ZIP.Code"))]
```

Converting Education to Factor and Creating Dummy Variables

```r
bd$Education <- as.factor(bd$Education)
dummy_edu <- model.matrix(~ Education - 1, data = bd)
bd <- cbind(bd, dummy_edu)
bd <- bd[, !(names(bd) %in% "Education")]
bd$Personal.Loan <- as.factor(bd$Personal.Loan)
str(bd)
```

```
## '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 ...
##  $ Education1        : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ Education2        : num  0 0 0 1 1 1 1 0 1 0 ...
##  $ Education3        : num  0 0 0 0 0 0 0 1 0 1 ...
```

Step 2 - First Partition (60% Training, 40% Validation)

```
set.seed(123)
train_index <- createDataPartition(bd$Personal.Loan, p = 0.6, list = FALSE)
train_set <- bd[train_index, ]
valid_set <- bd[-train_index, ]

cat("Training set size:", nrow(train_set), "\n")
```

```
## Training set size: 3000
```

```
cat("Validation set size:", nrow(valid_set))
```

```
## Validation set size: 2000
```

Preparing for K-NN: Normalization

```
train_labels <- train_set$Personal.Loan
valid_labels <- valid_set$Personal.Loan

train_pred <- train_set[, !(names(train_set) %in% "Personal.Loan")]
valid_pred <- valid_set[, !(names(valid_set) %in% "Personal.Loan")]

preproc_params <- preProcess(train_pred, method = c("center", "scale"))
train_norm <- predict(preproc_params, train_pred)
valid_norm <- predict(preproc_params, valid_pred)
```

Question 1: Classifying with K=1

```
new_customer <- data.frame(
  Age = 25, Experience = 2, Income = 100, Family = 2, CCAvg = 4,
  Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1,
  CreditCard = 1, Education1 = 0, Education2 = 1, Education3 = 0
)



new_customer_norm <- predict(preproc_params, new_customer)
```

Normalizing and Performing k-NN classification with k=1

```
knn_pred_k1 <- knn(train = train_norm, test = new_customer_norm,
                   cl = train_labels, k = 1)
cat("Question 1: Classification with  k=1:\n")
```

```
## Question 1: Classification with  k=1:
```

```
cat("The customer would be classified as:", as.character(knn_pred_k1),"\n")
```

```
## The customer would be classified as: 0
```

```
cat("This means the customer will", ifelse(knn_pred_k1 == 1, "ACCEPT", "NOT ACCEPT"), "the loan offer.\
```

```
## This means the customer will NOT ACCEPT the loan offer.
```

Question 2 - Finding Optimal K

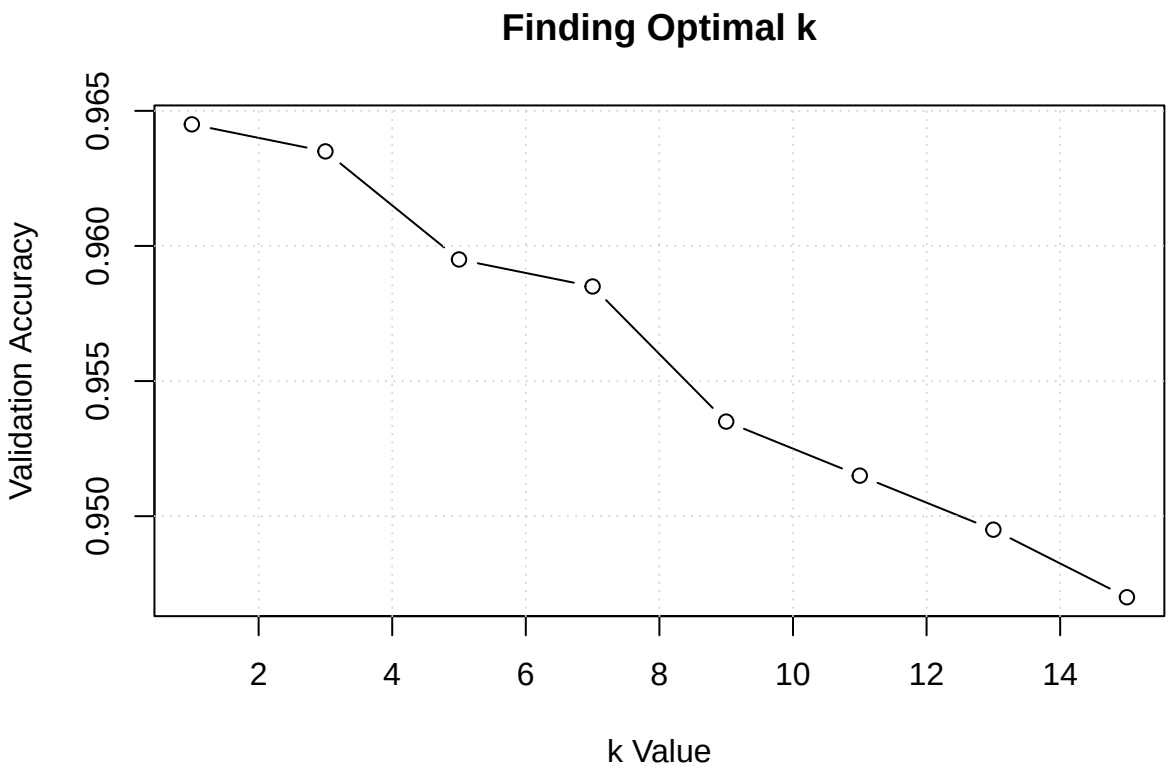
```
k_values <- seq(1, 15, 2)
accuracy_values <- numeric(length(k_values))

for (i in 1:length(k_values)) {
  knn_pred <- knn(train = train_norm, test = valid_norm,
                  cl = train_labels, k = k_values[i])
cm <- confusionMatrix(knn_pred, valid_labels)
accuracy_values[i] <- cm$overall["Accuracy"]
}

#Creating results dataframe
k_results <- data.frame(k = k_values, Accuracy = accuracy_values)
print(k_results)
```

```
##    k Accuracy
## 1  1   0.9645
## 2  3   0.9635
## 3  5   0.9595
## 4  7   0.9585
## 5  9   0.9535
## 6 11   0.9515
## 7 13   0.9495
## 8 15   0.9470
```

```
#Plotting k vs accuracy
plot(k_results$k, k_results$Accuracy, type = "b", xlab = "k Value",
     ylab = "Validation Accuracy", main = "Finding Optimal k")
grid()
```




```
#Finding best k
best_k <- k_results$k[which.max(k_results$Accuracy)]
cat("\nQuestion 2: Optimal k value is:", best_k, "\n")
```

```
##
## Question 2: Optimal k value is: 1
```

```
cat("This k balances between overfitting and ignoring predictor information.\n")
```

```
## This k balances between overfitting and ignoring predictor information.
```

Questions 3 - Validation Matrix and Classify with Best k

```
knn_pred_best <- knn(train = train_norm, test = valid_norm,
                     cl = train_labels, k = best_k)
cm_valid <- confusionMatrix(knn_pred_best, valid_labels)

cat("Question 3: Confusion Matrix for Validation Data (k =", best_k, ")\n")
```

```
## Question 3: Confusion Matrix for Validation Data (k = 1 )
```

```r
print(cm_valid$table)
```

```
##            Reference
## Prediction    0    1
##          0 1793   56
##          1   15  136
```

Question 4: Classify the customer with best k

```r
knn_customer_best <- knn(train = train_norm, test = new_customer_norm,
                         cl = train_labels, k = best_k)

cat("\nQuestion 4: Classification with optimal k =", best_k, ":\n")
```

```
##
## Question 4: Classification with optimal k = 1 :
```

```r
cat("The customer would be classified as:", as.character(knn_customer_best), "\n")
```

```
## The customer would be classified as: 0
```

```r
cat("This means the customer will", ifelse(knn_customer_best == 1, "ACCEPT", "NOT ACCEPT"), "the loan o
```

```
## This means the customer will NOT ACCEPT the loan offer.
```

Question 5: Three-Way Partition (50:30:20)

```r
set.seed(456) # Different seed for new partition
train_index2 <- createDataPartition(bd$Personal.Loan, p = 0.5, list = FALSE)
train_set2 <- bd[train_index2, ]
temp_set <- bd[-train_index2, ]

# Split the remaining 50% (temp_set) into 60% validation + 40% test
# This gives us 30% and 20% of the original data respectively
valid_index2 <- createDataPartition(temp_set$Personal.Loan, p = 0.6, list = FALSE) # 0.6 of 0.5 = 0.3
valid_set2 <- temp_set[valid_index2, ]
test_set2 <- temp_set[-valid_index2, ]

cat("New partition sizes:\n")
```

```
## New partition sizes:
```

```r
cat("Training set:", nrow(train_set2), "\n")
```

```
## Training set: 2500
```

```r
cat("Validation set:", nrow(valid_set2), "\n")
```

```
## Validation set: 1500
```

```r
cat("Test set:", nrow(test_set2), "\n")
```

```
## Test set: 1000
```

```r
train_labels2 <- train_set2$Personal.Loan
valid_labels2 <- valid_set2$Personal.Loan
test_labels2 <- test_set2$Personal.Loan

train_pred2 <- train_set2[, !(names(train_set2) %in% "Personal.Loan")]
valid_pred2 <- valid_set2[, !(names(valid_set2) %in% "Personal.Loan")]
test_pred2 <- test_set2[, !(names(test_set2) %in% "Personal.Loan")]

preproc_params2 <- preProcess(train_pred2, method = c("center", "scale"))
train_norm2 <- predict(preproc_params2, train_pred2)
valid_norm2 <- predict(preproc_params2, valid_pred2)
test_norm2 <- predict(preproc_params2, test_pred2)


accuracy_values2 <- numeric(length(k_values))
for (i in 1:length(k_values)) {
  knn_pred <- knn(train = train_norm2, test = valid_norm2,
                  cl = train_labels2, k = k_values[i])
  cm <- confusionMatrix(knn_pred, valid_labels2)
  accuracy_values2[i] <- cm$overall["Accuracy"]
}

best_k2 <- k_values[which.max(accuracy_values2)]
cat("Optimal k with new partition:", best_k2, "\n")
```

```
## Optimal k with new partition: 3
```

Generate confusion matrices for all three sets

```r
knn_train <- knn(train = train_norm2, test = train_norm2,
                 cl = train_labels2, k = best_k2)
knn_valid <- knn(train = train_norm2, test = valid_norm2,
                 cl = train_labels2, k = best_k2)
knn_test <- knn(train = train_norm2, test = test_norm2,
                cl = train_labels2, k = best_k2)

cm_train <- confusionMatrix(knn_train, train_labels2)
cm_valid2 <- confusionMatrix(knn_valid, valid_labels2)
cm_test <- confusionMatrix(knn_test, test_labels2)

cat("\nQuestion 5: Confusion Matrices Comparison\n")
```

```
##
## Question 5: Confusion Matrices Comparison
```

```
cat("Training Set Confusion Matrix:\n")
```

```
## Training Set Confusion Matrix:
```

```
print(cm_train$table)
```

```
##           Reference
## Prediction    0    1
##          0 2254   60
##          1    6  180
```

```
cat("\nValidation Set Confusion Matrix:\n")
```

```
##
## Validation Set Confusion Matrix:
```

```
print(cm_valid2$table)
```

```
##           Reference
## Prediction    0    1
##          0 1351   54
##          1    5   90
```

```
cat("\nTest Set Confusion Matrix:\n")
```

```
##
## Test Set Confusion Matrix:
```

```
print(cm_test$table)
```

```
##           Reference
## Prediction   0   1
##          0 902  38
##          1   2  58
```

```
#Calculate accuracies for comparison
accuracies <- data.frame(
  Set = c("Training", "Validation", "Test"),
  Accuracy = c(cm_train$overall["Accuracy"],
               cm_valid2$overall["Accuracy"],
               cm_test$overall["Accuracy"])
)
cat("\nAccuracy Comparison:\n")
```

```
##
## Accuracy Comparison:
```

```r
print(accuracies)
```

```
##          Set  Accuracy
## 1   Training 0.9736000
## 2 Validation 0.9606667
## 3       Test 0.9600000
```

Interpretation for Question 5

The differences in accuracy across the three sets are expected:

1. Training set has the highest accuracy because the model was built on this data.
2. Validation set accuracy is slightly lower as this data was used to tune the model (select k).
3. Test set accuracy provides the best estimate of real-world performance on completely new, unseen data.

The minor decrease in accuracy from validation to test set is normal and reflects the model's generalization error. The test set performance (0.96) is the most reliable indicator of how the model would perform in an actual marketing campaign.