BA 64060 - Assignment 2

Disney Maxwell

2025-09-28

Install packages

```
library(readr)
library(class)
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(ISLR)
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
```

Dataset: UniversalBank.csv - Data on 5000 customers

Import UniversalBank data into R

```
universal_bank_init<-read_csv("UniversalBank.csv")
```

```
## Rows: 5000 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): ID, Age, Experience, Income, ZIP Code, Family, CCAvg, Education, M...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Rename 2 column names
universal_bank_init <- universal_bank_init %>%
  rename(Securities.Account = `Securities Account`,
         CD.Account = 'CD Account')
```

Remove ID and ZIP code.

```r
universal_bank<-universal_bank_init[,-c(1,5)]
```

Transform Education categorical predictor into dummy variables

Transform Personal Loan to Categorical

```r
# Convert numeric to categorical using as.factor()
universal_bank$Education<-as.factor(universal_bank$Education)
universal_bank$'Personal Loan'<-as.factor(universal_bank$'Personal Loan')

# Dummy variables for Education
education_dmy<-dummyVars("~Education",data=universal_bank)
transformed_univbank_educ<-predict(education_dmy, newdata=universal_bank)
```

Final Data frame

```r
universal_bank<-universal_bank[,-6]
universal_bank_data<-cbind(universal_bank,transformed_univbank_educ)
```

First Split: Split Data into 60% for training and 40% for validation

```r
set.seed(123)
Train_Index1<-createDataPartition(universal_bank_data$'Personal Loan', p=0.6, list=FALSE)
Train_Data1<-universal_bank_data[Train_Index1,]
Validn_Data1<-universal_bank_data[-Train_Index1,]
```

Predictors and Labels

```r
Train_Predictors1<-Train_Data1[,c(1:6,8:14)]
Validn_Predictors1<-Validn_Data1[,c(1:6,8:14)]

Train_labels1<-Train_Data1[,7]
Validn_labels1<-Validn_Data1[,7]
```

Normalize Data

```r
set.seed(123)
# Use preProcess() to normalize
Norm_values <- preProcess(Train_Predictors1, method=c("range"))

Train_Predictors1_norm<-predict(Norm_values, Train_Predictors1)
Validn_Predictors1_norm<-predict(Norm_values, Validn_Predictors1)
```

Q1 Test case and train a knn model where k=1

```r
set.seed(123)
# Define Q1 Test case
q1_test_case <- 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
)
q1_test_case_norm<-predict(Norm_values,q1_test_case)

# knn prediction
q1_Predicted_Test_label<-knn(Train_Predictors1_norm, q1_test_case_norm, cl=Train_labels1, k=1)
print(paste0("The Customer would be classified as ", q1_Predicted_Test_label))
```

```
## [1] "The Customer would be classified as 1"
```

Q2 and Q3 - Choice of K - Hypertuning

Q2. The optimal value of k lies in the range of 1 to 20. It's in between overfitting to the predictor information and ignoring this information completely.

Q3. Confusion Matrix for Validation Data

```r
set.seed(123)

# Create an accuracy data frame
Accuracy_df<-data.frame(k = seq(1,20,1), Accuracy = rep(0,20))

# Compute knn for different values of k on validation set
for(i in 1:20) {
 Predicted_Label<-knn(Train_Predictors1_norm, Validn_Predictors1_norm, cl=Train_labels1, k=i)
 Accuracy_df[i,2]<-confusionMatrix(Predicted_Label, Validn_labels1)$overall[1]
}
Accuracy_df
```

```
##      k Accuracy
## 1    1   0.9620
## 2    2   0.9595
## 3    3   0.9615
## 4    4   0.9555
## 5    5   0.9525
## 6    6   0.9450
## 7    7   0.9435
## 8    8   0.9410
## 9    9   0.9390
## 10  10   0.9395
## 11  11   0.9365
## 12  12   0.9340
## 13  13   0.9320
## 14  14   0.9325
## 15  15   0.9305
## 16  16   0.9280
## 17  17   0.9280
## 18  18   0.9260
## 19  19   0.9240
## 20  20   0.9250
```

```r
best_k_value<-which.max(Accuracy_df$Accuracy)
print(paste0("The best value of k = ",best_k_value))
```

```
## [1] "The best value of k = 1"
```

```r
print(paste0("Highest Accuracy for the Validation set = ",max(Accuracy_df$Accuracy)))
```

```
## [1] "Highest Accuracy for the Validation set = 0.962"
```

Q4 Using the best k and k = 1

```r
set.seed(123)
# Define Q4 Test case
q4_test_case <- 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
)
q4_test_case_norm<-predict(Norm_values,q4_test_case)

# knn prediction
q4_Predicted_Test_label<-knn(Train_Predictors1_norm, q4_test_case_norm, cl=Train_labels1, k=1)
print(paste0("The Customer would be classified as ", q4_Predicted_Test_label))
```

```
## [1] "The Customer would be classified as 1"
```

Q5 Repartition data - Training:50%, Validation:30% and Test:20%

```
set.seed(123)
Temp_Index2<-createDataPartition(universal_bank_data$'Personal Loan', p=0.8, list=FALSE)
Temp_Data2<-universal_bank_data[Temp_Index2,]
Test_Data2<-universal_bank_data[-Temp_Index2,]

Train_Index2<-createDataPartition(Temp_Data2$'Personal Loan', p=0.5, list=FALSE)
Train_Data2<-Temp_Data2[Train_Index2,]
Validn_Data2<-Temp_Data2[-Train_Index2,]

# Predictors and Labels
Train_Predictors2<-Train_Data2[,c(1:6,8:14)]
Validn_Predictors2<-Validn_Data2[,c(1:6,8:14)]
Test_Predictors2<-Test_Data2[,c(1:6,8:14)]

Train_labels2<-Train_Data2[,7]
Validn_labels2<-Validn_Data2[,7]
Test_labels2<-Test_Data2[,7]

# Normalize Data
# Use preProcess() to normalize
Norm_values2 <- preProcess(Train_Predictors2, method=c("range"))

Train_Predictors2_norm<-predict(Norm_values2, Train_Predictors2)
Validn_Predictors2_norm<-predict(Norm_values2, Validn_Predictors2)
Test_Predictors2_norm<-predict(Norm_values2, Test_Predictors2)

# knn prediction with k from previous question
q5_Predicted_Test_label_ValidnSet<-knn(Train_Predictors2_norm, Validn_Predictors2_norm, cl=Train_labels2
q5_Predicted_Test_label_TestSet<-knn(Train_Predictors2_norm, Test_Predictors2_norm, cl=Train_labels2, k=
```

```
# Create an accuracy data frame for the Validation Set
Accuracy_V2_df<-data.frame(k = seq(1,20,1), Accuracy = rep(0,20))

# Compute knn for different values of k on validation set
for(i in 1:20) {
 Predicted_Label<-knn(Train_Predictors2_norm, Validn_Predictors2_norm, cl=Train_labels2, k=i)
 Accuracy_V2_df[i,2]<-confusionMatrix(Predicted_Label, Validn_labels2)$overall[1]
}
Accuracy_V2_df
```

```
##     k Accuracy
## 1   1   0.9565
## 2   2   0.9490
## 3   3   0.9465
## 4   4   0.9405
## 5   5   0.9380
## 6   6   0.9370
## 7   7   0.9320
## 8   8   0.9295
## 9   9   0.9290
```

```
## 10 10   0.9285
## 11 11   0.9265
## 12 12   0.9235
## 13 13   0.9225
## 14 14   0.9245
## 15 15   0.9225
## 16 16   0.9230
## 17 17   0.9205
## 18 18   0.9225
## 19 19   0.9195
## 20 20   0.9195
```

```r
best_k_value_Vldn<-which.max(Accuracy_V2_df$Accuracy)
print(paste0("The best value of k = ",best_k_value_Vldn))
```

```
## [1] "The best value of k = 1"
```

```r
print(paste0("Highest Accuracy for the Validation set = ",max(Accuracy_V2_df$Accuracy)))
```

```
## [1] "Highest Accuracy for the Validation set = 0.9565"
```

```r
# Create an accuracy data frame for the Test Set
Accuracy_T2_df<-data.frame(k = seq(1,20,1), Accuracy = rep(0,20))

# Compute knn for different values of k on Test set
for(i in 1:20) {
 Predicted_Label<-knn(Train_Predictors2_norm, Test_Predictors2_norm, cl=Train_labels2, k=i)
 Accuracy_T2_df[i,2]<-confusionMatrix(Predicted_Label, Test_labels2)$overall[1]
}
Accuracy_T2_df
```

```
##     k Accuracy
## 1   1    0.953
## 2   2    0.943
## 3   3    0.948
## 4   4    0.941
## 5   5    0.945
## 6   6    0.941
## 7   7    0.938
## 8   8    0.930
## 9   9    0.928
## 10 10    0.928
## 11 11    0.930
## 12 12    0.931
## 13 13    0.929
## 14 14    0.933
## 15 15    0.929
## 16 16    0.927
## 17 17    0.929
## 18 18    0.929
## 19 19    0.927
## 20 20    0.924
```

```r
best_k_value_Test<-which.max(Accuracy_T2_df$Accuracy)
print(paste0("The best value of k = ",best_k_value_Test))
```

```
## [1] "The best value of k = 1"
```

```r
print(paste0("Highest Accuracy for the Test set = ",max(Accuracy_T2_df$Accuracy)))
```

```
## [1] "Highest Accuracy for the Test set = 0.953"
```

On comparing the confusion matrices of the Validation and Test data sets, it is seen that in the Validation set, the Accuracy starts at a higher value for k=1 in comparison to the Test set.