Periandri_Anthony_Assignment 2

Anthony Periandri

2025-06-08

```r
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
```

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(class)
## Read csv file
df<- read.csv("UniversalBank.csv")
View(df)
## Remove ID and Zip Code from algortithm
df<- df %>% select(-ID,-ZIP.Code)
## Create dummy variables for education data
df<- df %>%
  mutate(Education_1 = ifelse(Education == 1,1,0),
         Education_2 = ifelse(Education == 2,1,0),
         Education_3 = ifelse(Education == 3,1,0)) %>%
  select(-Education)
## Normalise the  continuous variables to keep them numeric
normalize <- function(x) (x-min(x))/(max(x)-min(x))
num_vars <- c("Age", "Experience", "Income", "CCAvg", "Mortgage")
df[num_vars] <- lapply(df[num_vars], normalize)
norm_vals <- lapply(df[num_vars], function(x) c(min = min(x), max = max(x)))
new_normalization <- function(value, varname) {
  min_val <- norm_vals[[varname]]["min"]
  max_val <- norm_vals[[varname]]["max"]
```

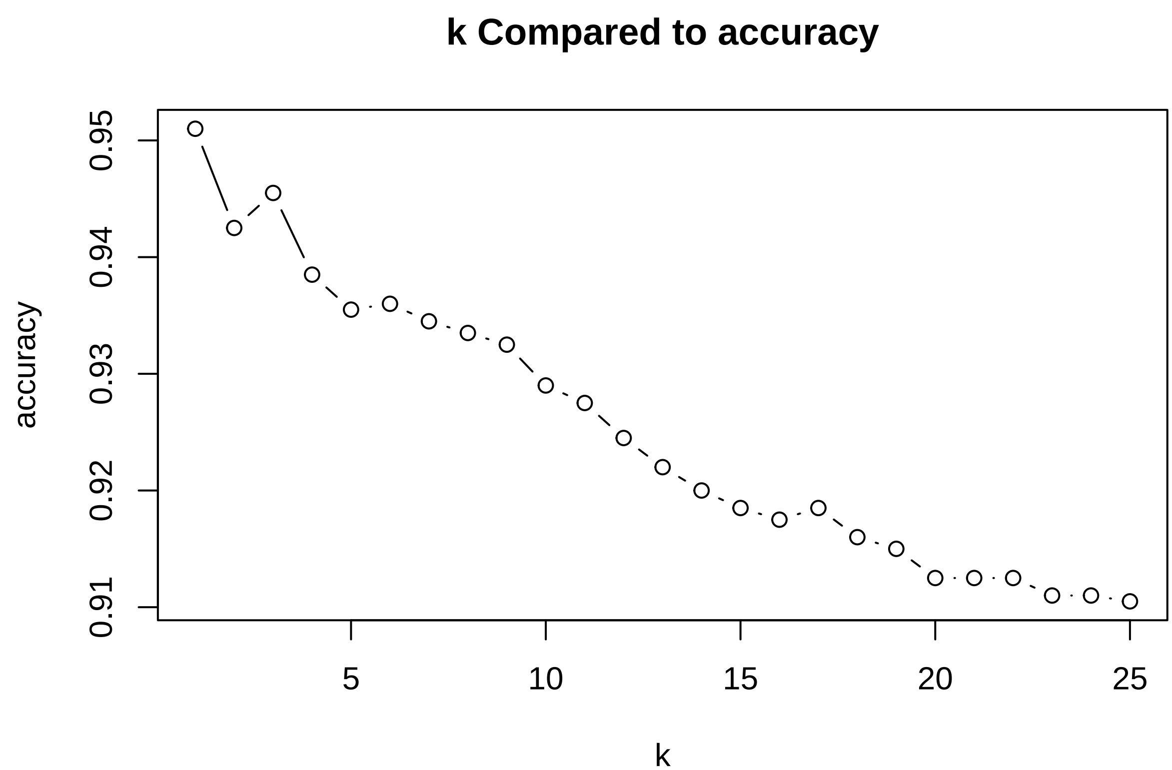
```r
  (value - min_val) / (max_val - min_val)
}
set.seed(572)
trainIndex <- createDataPartition(df$Personal.Loan, p = 0.6, list = FALSE)
train <- df[trainIndex,]
test <- df[-trainIndex,]
## Add new customer
new_customer <- data.frame(
  Age = new_normalization(40, "Age"),
  Experience = new_normalization(10, "Experience"),
  Income = new_normalization(84, "Income"),
  Family = 2,
  CCAvg = new_normalization(2, "CCAvg"),
  Mortgage = new_normalization(0, "Mortgage"),
  Securities.Account = 0,
  CD.Account = 0,
  CreditCard = 1,
  Online = 1,
  Education_1 = 0,
  Education_2 = 1,
  Education_3 = 0)
## Use Personal loan as predictor value
train_x <- train %>% select(-Personal.Loan)
train_y <- train$Personal.Loan
test_x <- test %>% select(-Personal.Loan)
## Integrate k=1
Predicted_Outcome <- knn(train = train_x, test = new_customer, cl = train_y, k = 1, prob = TRUE)
print(Predicted_Outcome)
```

```
## [1] 1
## attr(,"prob")
## [1] 1
## Levels: 0 1
```

```r
validate_x <- test %>% select(-Personal.Loan)
validate_y <- test$Personal.Loan
## Test accuracy
accuracy <- c()
for(k in 1:25) {
  pred_k <- knn(train = train_x, test = validate_x, cl = train_y, k = k)
  acc <- mean(pred_k == validate_y)
  accuracy[k] <- acc
}
best_k <- which.max(accuracy)
print(best_k)
```

```
## [1] 1
```

```r
plot(1:25, accuracy, type = "b", main = "k Compared to accuracy", xlab = "k", ylab = "accuracy")
```




```r
## Confusion Matrix
best_prediction <- knn(train_x, test_x, cl = train_y, k= best_k)
confusionMatrix(best_prediction, as.factor(test$Personal.Loan), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1789   76
##          1   22  113
##
##                Accuracy : 0.951
##                  95% CI : (0.9406, 0.96)
##     No Information Rate : 0.9055
##     P-Value [Acc > NIR] : 2.305e-14
##
##                   Kappa : 0.6717
##
##  Mcnemar's Test P-Value : 8.612e-08
##
##             Sensitivity : 0.5979
##             Specificity : 0.9879
##          Pos Pred Value : 0.8370
##          Neg Pred Value : 0.9592
##              Prevalence : 0.0945
##          Detection Rate : 0.0565
```

```
##    Detection Prevalence : 0.0675
##       Balanced Accuracy : 0.7929
##
##        'Positive' Class : 1
##
```

```r
## defining new customer using the best k
final_prediction <- knn(train = train_x, test = new_customer, cl = train_y, k = best_k)
print(final_prediction)
```

```
## [1] 1
## Levels: 0 1
```

```r
## Repartition data
set.seed(572)
trainIndex <- createDataPartition(df$Personal.Loan, p = 0.5, list = FALSE)
train <- df[trainIndex, ]
temporary <- df[-trainIndex, ]
Validindex <- createDataPartition(temporary$Personal.Loan, p = 0.6, list = FALSE)
validate <- temporary[Validindex, ]
test <- temporary[-Validindex, ]
train_x <- train %>% select(-Personal.Loan)
train_y <- train$Personal.Loan
validate_x <- validate %>% select(-Personal.Loan)
validate_y <- validate$Personal.Loan
test_x <- test %>% select(-Personal.Loan)
test_y <- test$Personal.Loan
## Evaluate sets of data
## Training Set
train_prediction <- knn(train = train_x, test = train_x, cl = train_y, k = best_k)
print(confusionMatrix(train_prediction, as.factor(train_y), positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2260    0
##          1    0  240
##
##                Accuracy : 1
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.000
##             Specificity : 1.000
##          Pos Pred Value : 1.000
##          Neg Pred Value : 1.000
##              Prevalence : 0.096
```

```
##          Detection Rate : 0.096
##    Detection Prevalence : 0.096
##       Balanced Accuracy : 1.000
##
##        'Positive' Class : 1
##
```

```r
## Validation Set
valid_prediction <- knn(train = train_x, test = validate_x, cl = train_y, k = best_k)
print(confusionMatrix(valid_prediction, as.factor(validate_y), positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1333   69
##          1   15   83
##
##                Accuracy : 0.944
##                  95% CI : (0.9311, 0.9551)
##     No Information Rate : 0.8987
##     P-Value [Acc > NIR] : 2.184e-10
##
##                   Kappa : 0.635
##
##  Mcnemar's Test P-Value : 7.348e-09
##
##             Sensitivity : 0.54605
##             Specificity : 0.98887
##          Pos Pred Value : 0.84694
##          Neg Pred Value : 0.95078
##              Prevalence : 0.10133
##          Detection Rate : 0.05533
##    Detection Prevalence : 0.06533
##       Balanced Accuracy : 0.76746
##
##        'Positive' Class : 1
##
```

```r
## Test Set
Test_prediction <- knn(train = train_x, test = test_x, cl = train_y, k = best_k)
print(confusionMatrix(Test_prediction, as.factor(test_y), positive = "1"))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 903  30
##          1   9  58
##
##                Accuracy : 0.961
##                  95% CI : (0.9471, 0.9721)
##     No Information Rate : 0.912
```

```
##     P-Value [Acc > NIR] : 9.666e-10
##
##                   Kappa : 0.7277
##
##  Mcnemar's Test P-Value : 0.001362
##
##             Sensitivity : 0.6591
##             Specificity : 0.9901
##          Pos Pred Value : 0.8657
##          Neg Pred Value : 0.9678
##              Prevalence : 0.0880
##          Detection Rate : 0.0580
##    Detection Prevalence : 0.0670
##       Balanced Accuracy : 0.8246
##
##        'Positive' Class : 1
##
```

```
## ANSWERS TO QUESTIONS
## 1. The customer would be likely to accept the loan as their predicted outcome is 1.
## 2. I tested K ranges from 1-25 and k=1 was the most accurate as can be seen in the graph.
## As k value goes up accuracy of the model goes down.
## 5. The training set had the highest accuracy percentage because the model was seeing the data.
## The Test set had the second highest accuracy and the validation set had the worst accuracy.
```