FML Assignment 2

2024-02-25

Loading the data using read.csv. There are 14 columns and 5000 observations.

```
dataset <- read.csv("./UniversalBank.csv")
dim(dataset)
```

```
## [1] 5000   14
```

Removing the columns ID and Zip code from the dataset.

```
dataset <- dataset[,-c(1,5)]
```

Data splitting.

Transformation of categorical variables with numerical values into factors is performed. The `dummyVars()` function is used to create dummy variables for the values in the Education column. This is then applied to the dataset using `predict()` function. The `set.seed()` function helps maintain uniformity in values even if the code is executed multiple times. The Dataset is split into `train_dataset` as 60% of the observations while the rest 40% are set as `valid_dataset` as mentioned in the question. Education contains three categories which on substituting with dummy variables is converted to binary output variables of 0 and 1.

```
dataset$Education <- as.factor(dataset$Education)

categories <- dummyVars(~., data = dataset) ## Converting education to dummy categories
dataset <- as.data.frame(predict(categories,dataset))


set.seed(1)
training_indices <- sample(row.names(dataset), 0.6*dim(dataset)[1])
validation_indices <- setdiff(row.names(dataset), training_indices)
train_dataset <- dataset[training_indices,]
valid_dataset <- dataset[validation_indices,]
t(t(names(train_dataset)))
```

```
##       [,1]
##  [1,] "Age"
##  [2,] "Experience"
##  [3,] "Income"
##  [4,] "Family"
##  [5,] "CCAvg"
##  [6,] "Education.1"
##  [7,] "Education.2"
##  [8,] "Education.3"
```

```
##  [9,] "Mortgage"
## [10,] "Personal.Loan"
## [11,] "Securities.Account"
## [12,] "CD.Account"
## [13,] "Online"
## [14,] "CreditCard"
```

Normalization of data is done using the z-score normalization method by passing the "center", "scale" into `method()` function. The function `preProcess()` performs normalization which is then applied to both training and validation dataset using the `predict()` function. The column with personal laon shouldn't be used for normalization and hence excluded.

```r
normalized.train <- train_dataset[,-10]
normalized.valid <- valid_dataset[,-10]

normalized.values <- preProcess(train_dataset[, -10], method=c("center", "scale"))
normalized.train <- predict(normalized.values, train_dataset[, -10])
normalized.valid <- predict(normalized.values, valid_dataset[, -10])
```

The following are the details provided about the first customer.

1. 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, and Credit Card = 1.

The details of the first customer are loaded as a dataframe into customer1. This is then normalised using the `normalized.values` model and applied using the `predict()` function to store the normalised vlaues of the first customer in `normalized_customer1`.

```r
customer1 <- 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
)

normalized_customer1 <- customer1
normalized_customer1 <- predict(normalized.values, normalized_customer1)
```

k-NN classification is performed and printed as `knn_predictor` using the `knn()` function in the `class` library. The k value is set to 1 as mentioned in the question.

The classification of the customer is "0". This means the customer would not accept the personal loan as printed by `knn_predictor`.

```r
knn_predictor <- class::knn(train = normalized.train,
                          test = normalized_customer1,
                          cl = train_dataset$Personal.Loan, k = 1)
knn_predictor
```

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

#Choice of k. The accuracy with respect to validation set is calculated using k-NN for each value of k from 1 to 15. The `plot()` function is used to strike a value for k such that it balances between overfitting and underfitting without the inclusion of noise in the data. The k value was found to be 3. Accuracy is plotted on the y-axis across the different values of k along the x-axis.

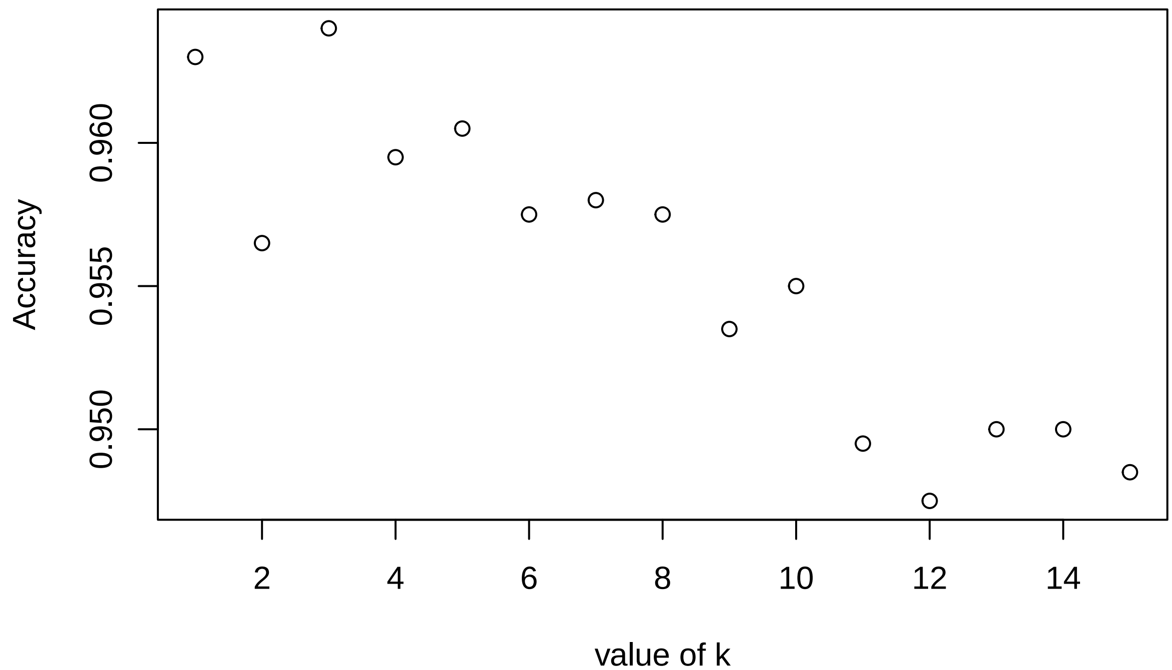
```r
accuracyK <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))
for(i in 1:15) {
  knn_predictor <- class::knn(train = normalized.train,
                            test = normalized.valid,
                            cl = train_dataset$Personal.Loan, k = i)
  accuracyK[i, 2] <- confusionMatrix(knn_predictor,
                                   as.factor(valid_dataset$Personal.Loan),positive = "1")$overall[1]
}

selectedK <- which(accuracyK[,2] == max(accuracyK[,2]))

print(paste("The selected k value is: ",selectedK))
```

```
## [1] "The selected k value is:  3"
```

```r
plot(accuracyK$k,accuracyK$overallaccuracy,xlab="value of k",ylab="Accuracy")
```



#Confusion matrix for validation data The given value of k for confusion matrix is 3. The same is performed using the `confusionMatrix()` function and is printed as `selectedk_matrix`. Here Personal Loan column is converted as factor.

```
knn_predictor <- class::knn(train = normalized.train,
                            test = normalized.valid,
                            cl = train_dataset$Personal.Loan, k = 3)

selectedk_matrix <- confusionMatrix(knn_predictor,
                                       as.factor(valid_dataset$Personal.Loan),positive = "1")
selectedk_matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    0    1
##          0 1786   63
##          1    9  142
##
##                Accuracy : 0.964
##                  95% CI : (0.9549, 0.9717)
##     No Information Rate : 0.8975
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7785
##
```

```
##  Mcnemar's Test P-Value : 4.208e-10
##
##             Sensitivity : 0.6927
##             Specificity : 0.9950
##          Pos Pred Value : 0.9404
##          Neg Pred Value : 0.9659
##              Prevalence : 0.1025
##          Detection Rate : 0.0710
##    Detection Prevalence : 0.0755
##       Balanced Accuracy : 0.8438
##
##        'Positive' Class : 1
##
```

The following details about the customer was provided: 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 and CreditCard = 1. This was stored as a data frame in `customer2`. This was then normalized using the `normalized.values` which has the `preProcess()` function and applied using the `predict()` function. This was set to `normalized_customer2`.

```
customer2 <- 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
)

normalized_customer2 <- customer2
normalized_customer2 <- predict(normalized.values,normalized_customer2)
```

k-NN is perfomred on the `normalized_customer2` using the previously identified k value of 3. The customer is classified as "0",that means no personal loan.

```
knn_predictor <- class::knn(train = normalized.train,
                       test = normalized_customer2,
                       cl = train_dataset$Personal.Loan, k = 3)
knn_predictor
```

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

#Re-partitioning data The dataset was re-partitioned into training, validation and test sets with 50%, 30% and 20% of the observations respectively and printed as `new_training_set`, `new_valid_set`, `new_test_set`.

```r
#Re partitioning the data.
new_training_index <- sample(row.names(dataset), 0.5*dim(dataset)[1])
new_valid_index <- sample(setdiff(row.names(dataset), new_training_index), 0.3*dim(dataset)[1])
new_test_index <- setdiff(setdiff(row.names(dataset), new_training_index), new_valid_index)
new_training_set <- dataset[new_training_index,]
new_valid_set <- dataset[new_valid_index,]
new_test_set <- dataset[new_test_index,]
```

Normalization of the data is performed using the `preProcess()` and `predict()` functions on all of the new training, validation and test datasets.

```r
new_train_norm <- new_training_set[,-10] # Note that Personal Income is the 10th variable
new_valid_norm <- new_valid_set[-10]
new_test_norm <- new_test_set[-10]

new_norm <- preProcess(new_training_set[, -10], method=c("center", "scale"))
new_train_norm <- predict(new_norm, new_training_set[, -10])
new_valid_norm <- predict(new_norm, new_valid_set[, -10])
new_test_norm<-  predict(new_norm, new_test_set[-10])
```

Using the `knn()`funtion and the `confusionMatrix()` functions led to the identification that model performs well on training data.

```r
knn_train_predictor <- class::knn(train = new_train_norm,
                         test = new_train_norm,
                         cl = new_training_set$Personal.Loan, k = 3)

training_matrix <- confusionMatrix(knn_train_predictor,as.factor(new_training_set$Personal.Loan),positi
training_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2254   50
##          1    6  190
##
##                Accuracy : 0.9776
##                  95% CI : (0.971, 0.983)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8594
##
##  Mcnemar's Test P-Value : 9.132e-09
##
##             Sensitivity : 0.7917
##             Specificity : 0.9973
##          Pos Pred Value : 0.9694
##          Neg Pred Value : 0.9783
##              Prevalence : 0.0960
##          Detection Rate : 0.0760
```

```
##    Detection Prevalence : 0.0784
##       Balanced Accuracy : 0.8945
##
##        'Positive' Class : 1
##
```

Using the `knn()`funtion and the `confusionMatrix()` functions on validation data.

```
knn_valid_predictor <- class::knn(train = new_train_norm,
                          test = new_valid_norm,
                          cl = new_training_set$Personal.Loan, k = 3)
validation_matrix <- confusionMatrix(knn_valid_predictor,as.factor(new_valid_set$Personal.Loan),positiv
validation_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1343   46
##          1   17   94
##
##                Accuracy : 0.958
##                  95% CI : (0.9466, 0.9676)
##     No Information Rate : 0.9067
##     P-Value [Acc > NIR] : 2.658e-14
##
##                   Kappa : 0.7264
##
##  Mcnemar's Test P-Value : 0.0004192
##
##             Sensitivity : 0.67143
##             Specificity : 0.98750
##          Pos Pred Value : 0.84685
##          Neg Pred Value : 0.96688
##              Prevalence : 0.09333
##          Detection Rate : 0.06267
##    Detection Prevalence : 0.07400
##       Balanced Accuracy : 0.82946
##
##        'Positive' Class : 1
##
```

Using the `knn()`and the `confusionMatrix()` functions on test data.

The accuracy of training set is greater than validation and testing data. Accuracy of test set is higher than validation set. This is described as overfitting. The inability of the model to generalise when new test set is provided leads to overfitting of the model.

```
knn_test_predictor <- class::knn(train = new_train_norm,
                          test = new_test_norm,
                          cl = new_training_set$Personal.Loan, k = 3)
testing_matrix <- confusionMatrix(knn_test_predictor,as.factor(new_test_set$Personal.Loan),positive = "
testing_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 896  34
##          1   4  66
##
##                Accuracy : 0.962
##                  95% CI : (0.9482, 0.973)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 1.383e-13
##
##                   Kappa : 0.7564
##
##  Mcnemar's Test P-Value : 2.546e-06
##
##             Sensitivity : 0.6600
##             Specificity : 0.9956
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.9634
##              Prevalence : 0.1000
##          Detection Rate : 0.0660
##    Detection Prevalence : 0.0700
##       Balanced Accuracy : 0.8278
##
##        'Positive' Class : 1
##
```