DSCI5340_HW3_Group13

Sai Vamshi Palakurthi, Kiran Kumar, Pavan Naramreddy, Sai Hari Vignesh

2024-04-07

##KNN Classification

```
pacman::p_load(caret, dplyr, data.table, ggplot2, FNN)
knitr::opts_chunk$set(echo = TRUE, fig.width=12, fig.height=6, fig.path = 'Figs/')
theme_set(theme_classic())
options(digits = 3)
```

loading data for partition

```
data.df <- read.csv("/Users/saivamshipalakurthi/Downloads/UniversalBank.csv", stringsAsFactors = TRUE)
data.df <- data.df[,-c(1, 5)]
data.df$Education <- as.factor(data.df$Education)

dummies_model <- dummyVars(~ . , data=data.df)
encoded_data <- predict(dummies_model, newdata=data.df)
data.df <- as.data.frame(encoded_data)
```

#Q1.Partition the data into training (75%) and validation (25%) sets

```
set.seed(42)
train_data.index <- sample(row.names(data.df), 0.7*dim(data.df)[1])
validation_data.index <- setdiff(row.names(data.df), train_data.index)
train_data.df <- data.df[train_data.index, ]
validation_data.df <- data.df[validation_data.index, ]
```

#Q2. Consider the following customer for classification: Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education_1 = 0, Education_2 = 1, Education_3 = 0, Mortgage = 0, Securities Account = 1, CD Account = 1, Online = 1, and Credit Card = 1.

```
newcustomer.df <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,
                             CCAvg = 2, Education_1 = 0, Education_2 = 1,
                             Education_3 = 0, Mortgage = 0, SecuritiesAccount = 1,
```

#Q3. Standardize all the data sets using mean and standard deviations Here we have created all the copies of the orginal datasets and we have used the copy datasets to standardize all the data sets using mean and std

```
# let us create a copy dataset of the original datasets
train_std.df <- train_data.df
validation_std.df <- validation_data.df
data_std.df <- data.df

set.seed(42)
norm_values <- preProcess(train_data.df[, c(1:5, 9:9)], method=c("center", "scale"))
train_std.df[, c(1:5, 9:9)] <- predict(norm_values, train_data.df[, c(1:5, 9:9)])
validation_std.df[, c(1:5, 9:9)] <- predict(norm_values, validation_data.df[, c(1:5, 9:9)])
data_std.df[, c(1:5, 9:9)] <- predict(norm_values, data.df[, c(1:5, 9:9)])
newcustomer_std.df <- predict(norm_values, newcustomer.df)
```

#Q4. Perform a k-NN classification with all predictors except ID and ZIP code using k = 1.

Let us now perform the K-NN classification with all the predictors except ID and ZIP code.

```
train_std.df$Personal.Loan <- factor(train_std.df$Personal.Loan)
validation_std.df$Personal.Loan <- factor(validation_std.df$Personal.Loan)

Knn <- knn(train = train_std.df[, c(1:9, 11:14)], test = newcustomer_std.df,
           cl = train_std.df[, 10], k = 1)
# let us find the Nearest-neighbor.
row.names(train_std.df)[attr(Knn, "nn.index")]
```

```
## [1] "2977"
```

#Note: The index 2977 is classified as 0.

How would this customer be classified? Here, the newcustomer will be classified as 0

#Q5. Now find the optimal value of k using the validation data set. What is the optimal k?

```
accuracy_data.df <- data.frame(k = seq(1, 20, 1), accuracy = rep(0, 20))
for(i in 1:20) {
  knn.pred <- knn(train_std.df[, c(1:9, 11:14)], validation_std.df[, c(1:9, 11:14)],
                  cl = train_std.df[, 10], k = i)
 accuracy_data.df[i, 2] <- confusionMatrix(knn.pred, validation_std.df[, 10])$overall[1]
}

print(accuracy_data.df)
```

```
##     k accuracy
## 1   1    0.960
## 2   2    0.959
## 3   3    0.968
## 4   4    0.956
## 5   5    0.962
## 6   6    0.955
## 7   7    0.961
## 8   8    0.957
## 9   9    0.961
## 10 10    0.953
## 11 11    0.957
```

```
## 12 12    0.953
## 13 13    0.955
## 14 14    0.951
## 15 15    0.952
## 16 16    0.945
## 17 17    0.950
## 18 18    0.947
## 19 19    0.949
## 20 20    0.946
```

#Q6.Using ConfusionMatrix() function from the caret package, print the confusion matrix for the validation data that results from using the optimal k.

```
knn <- knn(train = train_std.df[, c(1:9, 11:14)],
           test = validation_std.df[, c(1:9, 11:14)],
          cl = train_std.df[, 10], k = 3)

confusionMatrix(knn, validation_std.df[, 10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1356   48
##          1    0   96
##
##                Accuracy : 0.968
##                  95% CI : (0.958, 0.976)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.783
##
##  Mcnemar's Test P-Value : 1.17e-11
##
##             Sensitivity : 1.000
##             Specificity : 0.667
##          Pos Pred Value : 0.966
##          Neg Pred Value : 1.000
##              Prevalence : 0.904
##          Detection Rate : 0.904
##    Detection Prevalence : 0.936
##       Balanced Accuracy : 0.833
##
##        'Positive' Class : 0
##
```

Note: accuracy - 96.8%. Specificity is 0.667 for which there is 48 We can find false positives from the confusion matrix

#Q7. Classify the customer specified in Question 2 using the best k.

```
Knn <- knn(train = train_std.df[, c(1:9, 11:14)], test = newcustomer_std.df,
          cl = train_std.df[, 10], k = 3)
row.names(train_std.df)[attr(Knn, "nn.index")]
```

```
## [1] "2977" "1323" "1888"
```

Here it is predicted that all the three customer have not taken the loan so, the new customer also will not taken the loan.

#Q8.Now repartition the data into three parts: training, validation, and test sets (50%, 30%, and 20%).

```
#let us again split the data
set.seed(42)
newtrain_data.index <- sample(row.names(data.df), 0.50 * dim(data.df)[1])
newvalidation_data.index <-
  sample(setdiff(row.names(data.df), train_data.index), 0.30 * dim(data.df)[1])
newtest_data.index <- setdiff(row.names(data.df), c(newtrain_data.index, newvalidation_data.index))

newtrain_data.df <- data.df[newtrain_data.index, ]
newvalidation_data.df <- data.df[newvalidation_data.index, ]
newtest_data.df <- data.df[newtest_data.index, ]
```

```
#let us Standardize all the data sets and create copy datasets.
#
newtrain_data_std.df <- newtrain_data.df
newvalidation_data_std.df <- newvalidation_data.df
newtest_data_std.df <- newtest_data.df
#let us do the preProcess
set.seed(42)
newnorm.values <-
preProcess(newtrain_data_std.df[, c(1:5, 9:9)], method=c("center", "scale"))
newtrain_data_std.df[, c(1:5, 9:9)] <-
predict(newnorm.values, newtrain_data_std.df[, c(1:5, 9:9)])
newvalidation_data_std.df[, c(1:5, 9:9)] <-
predict(newnorm.values, newvalidation_data_std.df[, c(1:5, 9:9)])
newtest_data_std.df[, c(1:5, 9:9)] <-
predict(newnorm.values, newtest_data.df[, c(1:5, 9:9)])
```

#Q9. Apply the k-NN method with the optimal k chosen above.

```
newtrain_data.df[, 10] <- factor(newtrain_data.df[, 10])
newvalidation_data.df[, 10] <- factor(newvalidation_data.df[, 10])
newtest_data.df[, 10] <- factor(newtest_data.df[, 10])


knn_train <- knn(train = newtrain_data.df[, c(1:9, 11:13)],
                 test = newtrain_data.df[, c(1:9, 11:13)],
          cl = newtrain_data.df[, 10], k = 3)


knn_validation <- knn(train = newtrain_data.df[, c(1:9, 11:13)],
                      test = newvalidation_data.df[, c(1:9, 11:13)],
```

```
          cl = newtrain_data.df[, 10], k = 3)


knn_test <- knn(train = newtrain_data.df[, c(1:9, 11:13)],
               test = newtest_data.df[, c(1:9, 11:13)],
          cl = newtrain_data.df[, 10], k = 3)
```

#Q10. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
confusionMatrix(knn_train, newtrain_data.df[, 10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2209   93
##          1   47  151
##
##                Accuracy : 0.944
##                  95% CI : (0.934, 0.953)
##     No Information Rate : 0.902
##     P-Value [Acc > NIR] : 2.92e-14
##
##                   Kappa : 0.653
##
##  Mcnemar's Test P-Value : 0.000143
##
##             Sensitivity : 0.979
##             Specificity : 0.619
##          Pos Pred Value : 0.960
##          Neg Pred Value : 0.763
##              Prevalence : 0.902
##          Detection Rate : 0.884
##    Detection Prevalence : 0.921
##       Balanced Accuracy : 0.799
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(knn_validation, newvalidation_data.df[, 10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1318   93
##          1   38   51
##
##                Accuracy : 0.913
##                  95% CI : (0.897, 0.926)
##     No Information Rate : 0.904
```

```
##     P-Value [Acc > NIR] : 0.136
##
##                   Kappa : 0.393
##
##  Mcnemar's Test P-Value : 2.38e-06
##
##             Sensitivity : 0.972
##             Specificity : 0.354
##          Pos Pred Value : 0.934
##          Neg Pred Value : 0.573
##              Prevalence : 0.904
##          Detection Rate : 0.879
##    Detection Prevalence : 0.941
##       Balanced Accuracy : 0.663
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(knn_test, newtest_data.df[, 10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 876  63
##          1  32  29
##
##                Accuracy : 0.905
##                  95% CI : (0.885, 0.922)
##     No Information Rate : 0.908
##     P-Value [Acc > NIR] : 0.65379
##
##                   Kappa : 0.33
##
##  Mcnemar's Test P-Value : 0.00208
##
##             Sensitivity : 0.965
##             Specificity : 0.315
##          Pos Pred Value : 0.933
##          Neg Pred Value : 0.475
##              Prevalence : 0.908
##          Detection Rate : 0.876
##    Detection Prevalence : 0.939
##       Balanced Accuracy : 0.640
##
##        'Positive' Class : 0
##
```

A majority class (class 0) is displayed in all three sets, with accuracy varying from 0.905 (test) to 0.944 (training). Moving from training to validation and test sets appears to marginally reduce accuracy, which may indicate overfitting of the training set. Sensitivity which is between 0.96 and 0.98, it indicates that the model detetcs positive class instances with a high degree of cosistency across all sets. The training score which is 0.619 which is really low, suggesting that the model as trouble telling about the ngative

class instances apart from the majority class. Verification which is at 0.354, indicates that overfitting may exacerbate negative class. The test score which is 0.315, indicates that the model has trouble with the negative class on the test set's unseen data. Coming to the Kappa values, the possible values are -1 and 1, where -1 is for perfect disagreement and 1 is for perfect agreement, the training data has the value of 0.653, which is about the moderate agreement, and the validation set which is about 0.393 and the lowest agreement forms the test set which is about 0.33, which indicates that the model has difficulty applying effectively to new data.

Causes of the Disparities:

Overfitting: When compared to validation and test sets, the training set confusion matrix exhibits better accuracy and specificity. This implies that the model may be learning details from training data and underperforming on data that hasn't been seen yet (validation and test sets). Limited training data or complex models may cause this. Imbalanced Classes: The model may emphasize categorizing everything as the majority class (class 0) to achieve higher overall accuracy if the data has an imbalanced class distribution (more instances of class 0). This may result in subpar identification of the minority class (class 1).