

mchris26\_2

Mark Christian

10/4/2020

```
library(caret)
```

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

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

```
library(FNN)
library(gmodels)
library(dummies)
```

```
## dummies-1.5.6 provided by Decision Patterns
```

```
BankData <- read.csv("UniversalBank.csv")
```

Excluding ID and Zip Code

```
BankData1<-BankData[,c(-1,-5)]
str(BankData1)
```

```
## 'data.frame': 5000 obs. of 12 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 ...
## $ Education : int 1 1 1 2 2 2 2 3 2 3 ...
## $ Mortgage : int 0 0 0 0 0 155 0 0 104 0 ...
## $ Personal.Loan : int 0 0 0 0 0 0 0 0 0 1 ...
## $ 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 ...
```

Transforming catagorical predictors with more than 2 catagories into dumy variables

```
dummymodel <- dummyVars(~Education,data=BankData1)
head(predict(dummymodel,BankData1))
```

```
## Education
## 1      1
## 2      1
## 3      1
## 4      2
## 5      2
## 6      2
```

```
UBank<- dummy.data.frame(BankData1, names = c("Education"), sep= ".")
```

```
## Warning in model.matrix.default(~x - 1, model.frame(~x - 1), contrasts = FALSE):
## non-list contrasts argument ignored
```

Here I normalized the data

```
norm_model<-preProcess(UBank, method = c('range'))
UBank_normalized<-predict(norm_model,UBank)
UBank_Predictors<-UBank_normalized[, -10]
UBank_labels<-UBank_normalized[,10]
```

Data partition p=0.6 for 60%

```
set.seed(15)
inTrain = createDataPartition(UBank_normalized$Personal.Loan,p=0.6, list=FALSE)
Train_Data = UBank_normalized[inTrain,]
Val_Data = UBank_normalized[-inTrain,]
dim(Train_Data)
```

```
## [1] 3000  14
```

```
summary(Train_Data)
```

```
##      Age      Experience      Income      Family
## Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.2727   1st Qu.:0.2826   1st Qu.:0.1435   1st Qu.:0.0000
## Median :0.5000   Median :0.5000   Median :0.2546   Median :0.3333
## Mean   :0.5068   Mean   :0.5015   Mean   :0.3076   Mean   :0.4660
## 3rd Qu.:0.7273   3rd Qu.:0.7174   3rd Qu.:0.4213   3rd Qu.:0.6667
## Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##      CCAvg      Education.1      Education.2      Education.3
## Min.   :0.0000   Min.   :0.000   Min.   :0.0000   Min.   :0.0000
## 1st Qu.:0.0670   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
## Median :0.1500   Median :0.000   Median :0.0000   Median :0.0000
## Mean   :0.1948   Mean   :0.426   Mean   :0.2727   Mean   :0.3013
## 3rd Qu.:0.2600   3rd Qu.:1.000   3rd Qu.:1.0000   3rd Qu.:1.0000
## Max.   :1.0000   Max.   :1.000   Max.   :1.0000   Max.   :1.0000
##      Mortgage      Personal.Loan      Securities.Account      CD.Account
## Min.   :0.000000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
## 1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
## Median :0.00000   Median :0.00000   Median :0.0000   Median :0.00000
## Mean   :0.09196   Mean   :0.09967   Mean   :0.1033   Mean   :0.06333
```

```
## 3rd Qu.:0.16220 3rd Qu.:0.00000 3rd Qu.:0.0000 3rd Qu.:0.00000
## Max. :1.00000 Max. :1.00000 Max. :1.0000 Max. :1.00000
## Online CreditCard
## Min. :0.0000 Min. :0.000
## 1st Qu.:0.0000 1st Qu.:0.000
## Median :1.0000 Median :0.000
## Mean :0.5957 Mean :0.296
## 3rd Qu.:1.0000 3rd Qu.:1.000
## Max. :1.0000 Max. :1.000
```

```
summary(Val_Data)
```

```
## Age Experience Income Family
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000
## 1st Qu.:0.2727 1st Qu.:0.2826 1st Qu.:0.1389 1st Qu.:0.0000
## Median :0.5227 Median :0.5000 Median :0.2593 Median :0.3333
## Mean :0.5090 Mean :0.5034 Mean :0.2999 Mean :0.4647
## 3rd Qu.:0.7273 3rd Qu.:0.7174 3rd Qu.:0.4167 3rd Qu.:0.6667
## Max. :1.0000 Max. :1.0000 Max. :0.9120 Max. :1.0000
## CCAvg Education.1 Education.2 Education.3
## Min. :0.0000 Min. :0.000 Min. :0.0000 Min. :0.0000
## 1st Qu.:0.0700 1st Qu.:0.000 1st Qu.:0.0000 1st Qu.:0.0000
## Median :0.1500 Median :0.000 Median :0.0000 Median :0.0000
## Mean :0.1923 Mean :0.409 Mean :0.2925 Mean :0.2985
## 3rd Qu.:0.2500 3rd Qu.:1.000 3rd Qu.:1.0000 3rd Qu.:1.0000
## Max. :1.0000 Max. :1.000 Max. :1.0000 Max. :1.0000
## Mortgage Personal.Loan Securities.Account CD.Account
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.000
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.000 1st Qu.:0.000
## Median :0.0000 Median :0.0000 Median :0.000 Median :0.000
## Mean :0.0845 Mean :0.0905 Mean :0.106 Mean :0.056
## 3rd Qu.:0.1528 3rd Qu.:0.0000 3rd Qu.:0.000 3rd Qu.:0.000
## Max. :0.9638 Max. :1.0000 Max. :1.000 Max. :1.000
## Online CreditCard
## Min. :0.0000 Min. :0.000
## 1st Qu.:0.0000 1st Qu.:0.000
## Median :1.0000 Median :0.000
## Mean :0.5985 Mean :0.291
## 3rd Qu.:1.0000 3rd Qu.:1.000
## Max. :1.0000 Max. :1.000
```

```
Train_Predictors<-Train_Data[, -10]
Val_Predictors<-Val_Data[, -10]
Train_labels <-Train_Data[, 10]
Val_labels <-Val_Data[, 10]

Train_labels=as.factor(Train_labels)
Val_labels=as.factor(Val_labels)
UBank_labels<-as.factor(UBank_labels)
```

Knn where K=1

```
knn.pred <- knn(Train_Predictors,Val_Predictors,cl=Train_labels,k=1,prob = TRUE)
```

```
Q1 <- data.frame(40, 10, 84, 2, 2, 0, 1, 0, 0, 0, 0, 1, 1)
knn.pred1 <- knn(Train_Predictors, Q1, cl=Train_labels, k=1, prob = 0.5)
knn.pred1
```

```
## [1] 1
## attr(,"prob")
## [1] 1
## attr(,"nn.index")
##      [,1]
## [1,] 567
## attr(,"nn.dist")
##      [,1]
## [1,] 92.34856
## Levels: 1
```

Discuss the choice of k that balances between overfitting and ignoring the predictor information?

```
#install.packages("e1071")
library(caret)
accuracy.df <- data.frame(k = seq(1, 14, 1), accuracy = rep(0, 14))
for(i in 1:14) {
  knn <- knn(Train_Predictors, Val_Predictors, cl = Train_labels, k = i)
  accuracy.df[i, 2] <- confusionMatrix(knn, Val_labels)$overall[1]
}
accuracy.df
```

```
##      k accuracy
## 1     1   0.9600
## 2     2   0.9555
## 3     3   0.9615
## 4     4   0.9490
## 5     5   0.9550
## 6     6   0.9470
## 7     7   0.9485
## 8     8   0.9430
## 9     9   0.9460
## 10    10  0.9380
## 11    11  0.9410
## 12    12  0.9370
## 13    13  0.9390
## 14    14  0.9350
```

```
which.max( (accuracy.df$accuracy) )
```

```
## [1] 3
```

I have the optimal k=3

Show the confusion matrix for the validation data that results from using the best k

```
knn.pred3 <- knn(Train_Predictors,Val_Predictors,cl=Train_labels,k=3,prob = TRUE)
confusionMatrix(knn.pred3,Val_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##           0 1807   65
##           1   12  116
##
##           Accuracy : 0.9615
##           95% CI : (0.9521, 0.9695)
##       No Information Rate : 0.9095
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.7306
##
##  Mcnemar's Test P-Value : 3.105e-09
##
##           Sensitivity : 0.9934
##           Specificity : 0.6409
##           Pos Pred Value : 0.9653
##           Neg Pred Value : 0.9062
##           Prevalence : 0.9095
##           Detection Rate : 0.9035
##       Detection Prevalence : 0.9360
##           Balanced Accuracy : 0.8171
##
##           'Positive' Class : 0
##
```

: 50%, 30% and 20%

```
set.seed(15)
Bank_Partition = createDataPartition(UBank_normalized$Personal,p=0.5, list=FALSE)
TrainingData = UBank_normalized[Bank_Partition,]
TestValidData = UBank_normalized[-Bank_Partition,]
```

```
Test_Index = createDataPartition(TestValidData$Personal.Loan, p=0.6, list=FALSE)
ValidationData = TestValidData[Test_Index,]

Test_Data = TestValidData[-Test_Index,]
```

used p=0.6 to get 50:30:20 ratio testvaliddata to test and train

```
Training_Predictors<-TrainingData[,-10]
Test_Predictors<-Test_Data[,-10]
Validation_Predictors<-ValidationData[,-10]
Training_labels <-TrainingData[,10]
Test_labels <-Test_Data[,10]
Validation_labels <-ValidationData[,10]
```

```

Training_labels=as.factor(Training_labels)
Test_labels<-as.factor(Test_labels)
Validation_labels=as.factor(Validation_labels)

```

Apply the k-NN method with the k chosen above

```

knn.pred5 <- knn(Training_Predictors, Test_Predictors , cl=Training_labels, k=3, prob = TRUE)
confusionMatrix(knn.pred5,Test_labels)

```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 906  35
##              1   6  53
##
##              Accuracy : 0.959
##              95% CI : (0.9448, 0.9704)
##              No Information Rate : 0.912
##              P-Value [Acc > NIR] : 5.257e-09
##
##              Kappa : 0.6999
##
##  Mcnemar's Test P-Value : 1.226e-05
##
##              Sensitivity : 0.9934
##              Specificity : 0.6023
##              Pos Pred Value : 0.9628
##              Neg Pred Value : 0.8983
##              Prevalence : 0.9120
##              Detection Rate : 0.9060
##              Detection Prevalence : 0.9410
##              Balanced Accuracy : 0.7978
##
##              'Positive' Class : 0
##

```

```

knn.pred6 <- knn(Validation_Predictors, Test_Predictors, cl=Validation_labels, k=3, prob = TRUE)
confusionMatrix(knn.pred6,Test_labels)

```

```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0    1
##              0 910  47
##              1   2  41
##
##              Accuracy : 0.951
##              95% CI : (0.9357, 0.9635)
##              No Information Rate : 0.912
##              P-Value [Acc > NIR] : 1.785e-06
##

```

```

##                Kappa : 0.603
##
## Mcnemar's Test P-Value : 3.263e-10
##
##          Sensitivity : 0.9978
##          Specificity : 0.4659
##          Pos Pred Value : 0.9509
##          Neg Pred Value : 0.9535
##          Prevalence : 0.9120
##          Detection Rate : 0.9100
##          Detection Prevalence : 0.9570
##          Balanced Accuracy : 0.7319
##
##          'Positive' Class : 0
##

```

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

Difference:

Training Set:0.96 Validation set:0.95

Training set is more accuratne because it had more data compared to valdaton set and hence the results was more accurate.