k\_NN Classification - Universal Bank

##Loading Data and packages

getwd()

## [1] "/Users/sampathnikhilkumar/Desktop"

data1 <- data.frame(read.csv("UniversalBank.csv"))  
str(data1)

## 'data.frame': 5000 obs. of 14 variables:  
## $ ID : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ 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 ...  
## $ ZIP.Code : int 91107 90089 94720 94112 91330 92121 91711 93943 90089 93023 ...  
## $ 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 ...

library("ISLR")  
library("caret")

## Loading required package: ggplot2

## Loading required package: lattice

library("class")  
library("ggplot2")  
library("gmodels")

##Data Cleaning

data1 <- data1[,c(-1,-5)]  
head(data1, n=5)

## Age Experience Income Family CCAvg Education Mortgage Personal.Loan  
## 1 25 1 49 4 1.6 1 0 0  
## 2 45 19 34 3 1.5 1 0 0  
## 3 39 15 11 1 1.0 1 0 0  
## 4 35 9 100 1 2.7 2 0 0  
## 5 35 8 45 4 1.0 2 0 0  
## Securities.Account CD.Account Online CreditCard  
## 1 1 0 0 0  
## 2 1 0 0 0  
## 3 0 0 0 0  
## 4 0 0 0 0  
## 5 0 0 0 1

test.na <- is.na.data.frame("data1")  
  
##Converting data types of attributes  
data1$Education <- as.character(data1$Education)  
is.character(data1$Education)

## [1] TRUE

data1$Personal.Loan <- as.factor(data1$Personal.Loan)  
is.factor(data1$Personal.Loan)

## [1] TRUE

##Dummying Variables  
DummyVariables <- dummyVars(~Education, data1)  
head(predict(DummyVariables, data1))

## Education1 Education2 Education3  
## 1 1 0 0  
## 2 1 0 0  
## 3 1 0 0  
## 4 0 1 0  
## 5 0 1 0  
## 6 0 1 0

data2 <- predict(DummyVariables,data1)  
  
##Combining Data  
data3 <- data1[,-6]  
data4 <- cbind(data3,data2)  
colnames(data4)

## [1] "Age" "Experience" "Income"   
## [4] "Family" "CCAvg" "Mortgage"   
## [7] "Personal.Loan" "Securities.Account" "CD.Account"   
## [10] "Online" "CreditCard" "Education1"   
## [13] "Education2" "Education3"

##Data Partition and Normalization

set.seed(123)  
Data\_Part\_Train <- createDataPartition(data4$Personal.Loan, p=0.6, list=F)  
Train\_Data <- data4[Data\_Part\_Train,]  
Validation\_Data <- data4[-Data\_Part\_Train,]  
  
#Normalizing the training dataset  
Model\_Z\_Normalized <- preProcess(Train\_Data[,-c(7,12:14)], method=c("center","scale"))  
  
Normalized\_Data\_Train <- predict(Model\_Z\_Normalized, Train\_Data)  
  
Normalized\_Data\_Validation <- predict(Model\_Z\_Normalized, Validation\_Data)  
  
#summary(Normalized\_Data\_Train)  
#summary(Normalized\_Data\_Validation)

##Inserting a test set and normalizing it

test\_data <- cbind.data.frame(Age = 40,Experience = 10, Income = 84, Family = 2, CCAvg = 2, Mortgage = 0, Securities.Account = 0, CD.Account = 0, Online = 1, CreditCard = 1, Education1 = 0, Education2 = 1, Education3 = 0)  
  
Test\_Normalized <- predict(Model\_Z\_Normalized, test\_data)

#1. Running the knn model on the test dataset with k=1

Train\_Predictors <- Normalized\_Data\_Train[,-7]  
Validation\_Predictors <- Normalized\_Data\_Validation[,-7]  
  
Train\_Labels <- Normalized\_Data\_Train[,7]  
Validate\_Lables <- Normalized\_Data\_Validation[,7]  
  
Predicted\_K <- knn(Train\_Predictors, Test\_Normalized, cl=Train\_Labels, k=1)  
  
head(Predicted\_K)

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

When k=1 the customer is classified as 0 which indicates that the loan is not accepted. Since factor 1 is classified as loan acceptance and 0 is not accepted.

#2. Choice of k that balances between overfitting and ignoring the predictor information

set.seed(455)  
search\_grid <- expand.grid(k=c(1:30))  
#trtcontrol <- trainControl(method="repeatedcv")  
model <- train(Personal.Loan~Age+Experience+Income+Family+CCAvg+Mortgage+Securities.Account+CD.Account+Online+CreditCard+Education1+Education2+Education3, data=Normalized\_Data\_Train, method="knn", tuneGrid = search\_grid)  
model

## k-Nearest Neighbors   
##   
## 3000 samples  
## 13 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 3000, 3000, 3000, 3000, 3000, 3000, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.9486329 0.6739443  
## 2 0.9405480 0.6118584  
## 3 0.9403636 0.5980966  
## 4 0.9392894 0.5855648  
## 5 0.9407990 0.5836956  
## 6 0.9408178 0.5747129  
## 7 0.9404153 0.5630616  
## 8 0.9398896 0.5554862  
## 9 0.9396189 0.5509399  
## 10 0.9380957 0.5331053  
## 11 0.9380233 0.5288976  
## 12 0.9369616 0.5198382  
## 13 0.9365720 0.5134500  
## 14 0.9360384 0.5068074  
## 15 0.9355341 0.4985038  
## 16 0.9356776 0.4971309  
## 17 0.9345174 0.4854181  
## 18 0.9341181 0.4822415  
## 19 0.9335817 0.4755456  
## 20 0.9338016 0.4741894  
## 21 0.9335631 0.4724520  
## 22 0.9323913 0.4612421  
## 23 0.9319614 0.4563577  
## 24 0.9317099 0.4537991  
## 25 0.9311729 0.4471843  
## 26 0.9314622 0.4503100  
## 27 0.9309886 0.4446960  
## 28 0.9304798 0.4385601  
## 29 0.9303716 0.4362160  
## 30 0.9297296 0.4283101  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 1.

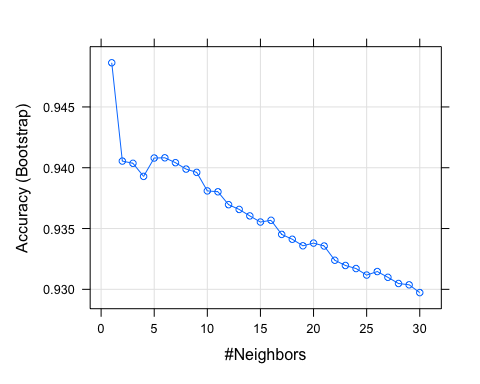
best\_k <- model$bestTune[[1]]  
best\_k

## [1] 1

The k value which balances between over fitting and ignoring the predictor information is k = 1.

#Plotting the model

plot(model)



#3. Confusion matrix being deployed over the validation data

pred\_training <- predict(model,Normalized\_Data\_Validation[,-7])  
confusionMatrix(pred\_training, Validate\_Lables)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1789 54  
## 1 19 138  
##   
## Accuracy : 0.9635   
## 95% CI : (0.9543, 0.9713)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7711   
##   
## Mcnemar's Test P-Value : 6.909e-05   
##   
## Sensitivity : 0.9895   
## Specificity : 0.7188   
## Pos Pred Value : 0.9707   
## Neg Pred Value : 0.8790   
## Prevalence : 0.9040   
## Detection Rate : 0.8945   
## Detection Prevalence : 0.9215   
## Balanced Accuracy : 0.8541   
##   
## 'Positive' Class : 0   
##

Miscalculations = 73, Accuracy = 0.9635, Sensitivity = 0.9895

#4. Running the test data with best k choosen above

test\_best\_k <- knn(Train\_Predictors, Test\_Normalized, cl=Train\_Labels, k=best\_k)  
head(test\_best\_k)

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

With the best k being choosen, the customer is classified as 0 which indicates that the loan is not accepted.

#5. Repartitioning the data into training(50%), validation(30%) and test(20%) and running the entire model with best k

set.seed(422)  
data\_part <- createDataPartition(data4$Personal.Loan, p=0.5, list = F)  
n\_train\_data <- data4[data\_part,]  
nd\_test\_data <- data4[-data\_part,]  
  
data\_part\_v <- createDataPartition(nd\_test\_data$Personal.Loan,p=0.6, list =F)  
n\_validate\_data <- nd\_test\_data[data\_part\_v,]  
n\_test\_data <- nd\_test\_data[-data\_part\_v,]  
  
#Normalization  
norm\_m <- preProcess(n\_train\_data[,-c(7,12:14)],method=c("center","scale"))  
  
train\_z <- predict(norm\_m, n\_train\_data)  
validate\_z <- predict(norm\_m, n\_validate\_data)  
test\_z <- predict(norm\_m, n\_test\_data)  
  
#Defining the predictors and labels  
n\_train\_predictor <- train\_z[,-7]  
n\_validate\_predictor <- validate\_z[,-7]  
n\_test\_predictor <- test\_z[,-7]  
  
n\_train\_labels <- train\_z[,7]  
n\_validate\_labels <- validate\_z[,7]  
n\_test\_labels <- test\_z[,7]  
  
#running the knn model over train dataset  
n\_model <- knn(n\_train\_predictor,n\_train\_predictor,cl=n\_train\_labels,k=best\_k)  
head(n\_model)

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

#running the knn model over validation dataset  
n\_model1 <- knn(n\_train\_predictor,n\_validate\_predictor,cl=n\_train\_labels,k=best\_k)  
head(n\_model1)

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

#running the knn model over test dataset  
n\_model2 <- knn(n\_train\_predictor,n\_test\_predictor,cl=n\_train\_labels,k=best\_k)  
head(n\_model2)

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

#Using CrossTable to compare the Test vs Training and Validation

confusionMatrix(n\_model,n\_train\_labels)

## 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 : 0.904   
## Detection Rate : 0.904   
## Detection Prevalence : 0.904   
## Balanced Accuracy : 1.000   
##   
## 'Positive' Class : 0   
##

#Train\_Data - Miscalculations = 0 Accuracy = 1 Sensitivity = 1 #(This is because both the train and test datasets are same, model has already seen the data and hence it cannot predict anything wrong, which results in 100% Accuracy and 0 Miscalulations).

confusionMatrix(n\_model1,n\_validate\_labels)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1334 55  
## 1 22 89  
##   
## Accuracy : 0.9487   
## 95% CI : (0.9363, 0.9593)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 1.261e-10   
##   
## Kappa : 0.6705   
##   
## Mcnemar's Test P-Value : 0.0002656   
##   
## Sensitivity : 0.9838   
## Specificity : 0.6181   
## Pos Pred Value : 0.9604   
## Neg Pred Value : 0.8018   
## Prevalence : 0.9040   
## Detection Rate : 0.8893   
## Detection Prevalence : 0.9260   
## Balanced Accuracy : 0.8009   
##   
## 'Positive' Class : 0   
##

#Validation Data - Miscalculations = 22 + 55 = 77 Accuracy = 0.9487 Sensitivity = 0.9838

confusionMatrix(n\_model2,n\_test\_labels)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 891 26  
## 1 13 70  
##   
## Accuracy : 0.961   
## 95% CI : (0.9471, 0.9721)  
## No Information Rate : 0.904   
## P-Value [Acc > NIR] : 5.695e-12   
##   
## Kappa : 0.7608   
##   
## Mcnemar's Test P-Value : 0.05466   
##   
## Sensitivity : 0.9856   
## Specificity : 0.7292   
## Pos Pred Value : 0.9716   
## Neg Pred Value : 0.8434   
## Prevalence : 0.9040   
## Detection Rate : 0.8910   
## Detection Prevalence : 0.9170   
## Balanced Accuracy : 0.8574   
##   
## 'Positive' Class : 0   
##

#Test\_Data - Miscalculations = 39 Accuracy = 0.961 Sensitivity = 0.9856

#Interpretation: When comparing the test with that of training and validation, we shall exclude train from this consideration because a model will mostly result in 100% accuracy when it has the seen data.

Miscalculations: Validation - 77, Test - 39

Accuracy: Validation - 0.9487, Test - 0.961

Sensitivty: Validation - 0.9838, Test - 0.9856

We see that the Test data has fewer misalculations, greater accuracy and sensitivity when compared to that of the validation data, by this we can say that the model works well on the unseen data.