Assignment_2

```
UniversalBank <- read.csv("C:/Users/Dell/Desktop/UniversalBank.csv")
View(UniversalBank)

summary(UniversalBank)
```

```
##        ID            Age           Experience       Income          ZIP.Code
##  Min.   :   1   Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   : 9307
##  1st Qu.:1251   1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:91911
##  Median :2500   Median :45.00   Median :20.0   Median : 64.00   Median :93437
##  Mean   :2500   Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :93153
##  3rd Qu.:3750   3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:94608
##  Max.   :5000   Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :96651
##      Family          CCAvg          Education        Mortgage
##  Min.   :1.000   Min.   : 0.000   Min.   :1.000   Min.   :  0.0
##  1st Qu.:1.000   1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0
##  Median :2.000   Median : 1.500   Median :2.000   Median :  0.0
##  Mean   :2.396   Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.:3.000   3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :4.000   Max.   :10.000   Max.   :3.000   Max.   :635.0
##  Personal.Loan   Securities.Account   CD.Account         Online
##  Min.   :0.000   Min.   :0.0000     Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.000   Median :0.0000     Median :0.0000   Median :1.0000
##  Mean   :0.096   Mean   :0.1044     Mean   :0.0604   Mean   :0.5968
##  3rd Qu.:0.000   3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :1.000   Max.   :1.0000     Max.   :1.0000   Max.   :1.0000
##    CreditCard
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.294
##  3rd Qu.:1.000
##  Max.   :1.000
```

NULL Variables

```
UniversalBank$ID <- NULL
UniversalBank$ZIP.Code <- NULL
UniversalBank$Personal.Loan = as.factor(UniversalBank$Personal.Loan)
summary(UniversalBank)
```

```
##       Age          Experience       Income          Family
```

```
##  Min.   :23.00   Min.   :-3.0   Min.   :  8.00   Min.   :1.000
##  1st Qu.:35.00   1st Qu.:10.0   1st Qu.: 39.00   1st Qu.:1.000
##  Median :45.00   Median :20.0   Median : 64.00   Median :2.000
##  Mean   :45.34   Mean   :20.1   Mean   : 73.77   Mean   :2.396
##  3rd Qu.:55.00   3rd Qu.:30.0   3rd Qu.: 98.00   3rd Qu.:3.000
##  Max.   :67.00   Max.   :43.0   Max.   :224.00   Max.   :4.000
##      CCAvg          Education        Mortgage     Personal.Loan
##  Min.   : 0.000   Min.   :1.000   Min.   :  0.0   0:4520
##  1st Qu.: 0.700   1st Qu.:1.000   1st Qu.:  0.0   1: 480
##  Median : 1.500   Median :2.000   Median :  0.0
##  Mean   : 1.938   Mean   :1.881   Mean   : 56.5
##  3rd Qu.: 2.500   3rd Qu.:3.000   3rd Qu.:101.0
##  Max.   :10.000   Max.   :3.000   Max.   :635.0
##  Securities.Account   CD.Account         Online         CreditCard
##  Min.   :0.0000     Min.   :0.0000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000     Median :0.0000   Median :1.0000   Median :0.000
##  Mean   :0.1044     Mean   :0.0604   Mean   :0.5968   Mean   :0.294
##  3rd Qu.:0.0000     3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:1.000
##  Max.   :1.0000     Max.   :1.0000   Max.   :1.0000   Max.   :1.000
```

Normalizing the data

```
library(caret)
```

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

```
## Warning in register(): Can't find generic 'scale_type' in package ggplot2 to
## register S3 method.
```

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

```
library(class)


Norm_model <- preProcess(UniversalBank[,-8], method = c("center", "scale"))
UniversalBank_norm=predict(Norm_model,UniversalBank[,-8])
summary(UniversalBank_norm)
```

```
##       Age              Experience           Income            Family
##  Min.   :-1.94871   Min.   :-2.014710   Min.   :-1.4288   Min.   :-1.2167
##  1st Qu.:-0.90188   1st Qu.:-0.881116   1st Qu.:-0.7554   1st Qu.:-1.2167
##  Median :-0.02952   Median :-0.009121   Median :-0.2123   Median :-0.3454
##  Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.84284   3rd Qu.: 0.862874   3rd Qu.: 0.5263   3rd Qu.: 0.5259
##  Max.   : 1.88967   Max.   : 1.996468   Max.   : 3.2634   Max.   : 1.3973
##      CCAvg           Education          Mortgage       Securities.Account
##  Min.   :-1.1089   Min.   :-1.0490   Min.   :-0.5555   Min.   :-0.3414
##  1st Qu.:-0.7083   1st Qu.:-1.0490   1st Qu.:-0.5555   1st Qu.:-0.3414
```

```
##  Median :-0.2506   Median : 0.1417   Median :-0.5555   Median :-0.3414
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.3216   3rd Qu.: 1.3324   3rd Qu.: 0.4375   3rd Qu.:-0.3414
##  Max.   : 4.6131   Max.   : 1.3324   Max.   : 5.6875   Max.   : 2.9286
##    CD.Account          Online         CreditCard
##  Min.   :-0.2535   Min.   :-1.2165   Min.   :-0.6452
##  1st Qu.:-0.2535   1st Qu.:-1.2165   1st Qu.:-0.6452
##  Median :-0.2535   Median : 0.8219   Median :-0.6452
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.:-0.2535   3rd Qu.: 0.8219   3rd Qu.: 1.5495
##  Max.   : 3.9438   Max.   : 0.8219   Max.   : 1.5495
```

adding back the target variable

```
UniversalBank_norm$Personal.Loan=UniversalBank$Personal.Loan
```

dividing the data into train and validation

```
Train_Index = createDataPartition(UniversalBank$Personal.Loan,p=0.6, list=FALSE)
Train.df=UniversalBank_norm[Train_Index,]
Validation.df=UniversalBank_norm[-Train_Index,]
```

Task 1

Use the train set and knn method with k=1 to predict if a new customer will accept a loan offer

```
To_Predict=data.frame(Age=40, Experience=10,Income=84,Family=2,
                      CCAvg=2,Education=1, Mortgage=0,
                      Securities.Account=0,CD.Account=0,Online=1,CreditCard=1 )
print(To_Predict)
```

```
##   Age Experience Income Family CCAvg Education Mortgage Securities.Account
## 1  40         10     84      2     2         1        0                   0
##   CD.Account Online CreditCard
## 1          0      1          1
```

#applying the normalization

```
To_Predict_norm <- predict(Norm_model,To_Predict)
print(To_Predict_norm)
```

```
##          Age Experience    Income     Family     CCAvg Education   Mortgage
## 1 -0.4657003 -0.8811162 0.2221371 -0.3453975 0.0355115 -1.048973 -0.5554684
##   Securities.Account CD.Account    Online CreditCard
## 1         -0.3413892 -0.2535149 0.8218687   1.549477
```

#Knn Prediction

```
Prediction <-knn(train=Train.df[,1:7,9:12],
             test=To_Predict_norm[,1:7,9:12],
             cl=Train.df$Personal.Loan,
             k=1)
print(Prediction)
```

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

- As output is zero new customer will not accept a loan offer

Task 2

crossvalidation for overfitting

```
set.seed(123)
fitControl <- trainControl(method = "repeatedcv",number = 3,repeats = 2)

searchGrid = expand.grid(k = 1:10)

Knn.model=train(Personal.Loan~.,
                data=Train.df,
                method='knn',
                tuneGrid=searchGrid,
                trControl = fitControl)

Knn.model
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 2 times)
## Summary of sample sizes: 2000, 2000, 2000, 2000, 2000, 2000, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9493333  0.6757433
##    2  0.9461667  0.6629767
```

```
##    3  0.9561667  0.7072465
##    4  0.9521667  0.6714834
##    5  0.9518333  0.6631060
##    6  0.9518333  0.6647418
##    7  0.9495000  0.6356070
##    8  0.9491667  0.6319718
##    9  0.9476667  0.6155330
##   10  0.9455000  0.5942243
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

#Task 3

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

```
To_Predict=data.frame(Age=40, Experience=10,Income=84,Family=2,
                      CCAvg=2,Education=1, Mortgage=0,
                      Securities.Account=0,CD.Account=0,Online=1,CreditCard=1 )

predictions<-predict(Knn.model,Validation.df)
confusionMatrix(predictions,Validation.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1800   73
##          1    8  119
##
##                Accuracy : 0.9595
##                  95% CI : (0.9499, 0.9677)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7251
##
##  Mcnemar's Test P-Value : 1.151e-12
##
##             Sensitivity : 0.9956
##             Specificity : 0.6198
##          Pos Pred Value : 0.9610
##          Neg Pred Value : 0.9370
##              Prevalence : 0.9040
##          Detection Rate : 0.9000
##    Detection Prevalence : 0.9365
##       Balanced Accuracy : 0.8077
##
##        'Positive' Class : 0
##
```

considering some attributes using the best k.

```
new.df=data.frame(Age=40, Experience=10,Income=84,Family=2,
                      CCAvg=2,Education=1, Mortgage=0,
                      Securities.Account=0,CD.Account=0,Online=1,CreditCard=1 )



Prediction <-knn(train=Train.df[,1:7,9:12],
             test=new.df[,1:7,9:12],
             cl=Train.df$Personal.Loan,
             k=3,prob = TRUE)

print(Prediction)
```

```
## [1] 1
## attr(,"prob")
## [1] 0.6666667
## Levels: 0 1
```

Question 5

Dividing the data into training, validation, and test sets (50% : 30% : 20%)

```
Train_Index = createDataPartition(UniversalBank$Personal.Loan,p=0.8, list=FALSE)

Train0.8.df = UniversalBank_norm[Train_Index,(-8)]
Test.df=UniversalBank_norm[-Train_Index,(-8)]

validation_Index=createDataPartition(UniversalBank$Personal.Loan,p=0.3, list=FALSE)
validation.df=Train0.8.df[!is.na(validation_Index),(-8)]


Train.df=Train0.8.df[-validation_Index,(-8)]
```

```
To_Predict=data.frame(Age=40, Experience=10,Income=84,Family=2,
                      CCAvg=2,Education=1, Mortgage=0,
                      Securities.Account=0,CD.Account=0,Online=1,CreditCard=1 )
print(To_Predict)
```

```
##   Age Experience Income Family CCAvg Education Mortgage Securities.Account
## 1  40         10     84      2     2         1        0                   0
##   CD.Account Online CreditCard
## 1          0      1          1
```

```
To_Predict_norm <- predict(Norm_model,To_Predict)
print(To_Predict_norm)
```

```
##          Age Experience    Income     Family     CCAvg Education   Mortgage
## 1 -0.4657003 -0.8811162 0.2221371 -0.3453975 0.0355115 -1.048973 -0.5554684
##   Securities.Account CD.Account    Online CreditCard
## 1         -0.3413892 -0.2535149 0.8218687   1.549477
```

```
Prediction <-knn(train=Train.df[,1:7,9:11],
             test=To_Predict_norm[,1:7,9:11],
             cl=Train.df$Personal.Loan,
             k=3)
print(Prediction)
```

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

```
set.seed(123)
fitControl <- trainControl(method = "repeatedcv",number = 3,repeats = 2)

searchGrid = expand.grid(k = 1:10)

Trainknn=train(Personal.Loan~.,
               data=Train.df,
               method='knn',
               tuneGrid=searchGrid,
               trControl = fitControl)
Trainknn
```

```
## k-Nearest Neighbors
##
## 2796 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 2 times)
## Summary of sample sizes: 1864, 1864, 1864, 1864, 1864, 1864, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9560086  0.7287245
##    2  0.9472461  0.6766033
##    3  0.9540415  0.6988456
##    4  0.9508226  0.6733502
##    5  0.9499285  0.6567114
##    6  0.9483190  0.6429133
##    7  0.9459943  0.6173217
##    8  0.9451001  0.6083861
##    9  0.9436695  0.5902673
##   10  0.9409871  0.5646453
##
```

```
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 1.
```

```
validationknn=train(Personal.Loan~.,
                data=validation.df,
                method='knn',
                tuneGrid=searchGrid,
                trControl = fitControl)
validationknn
```

```
## k-Nearest Neighbors
##
## 4000 samples
##    9 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 2 times)
## Summary of sample sizes: 2666, 2667, 2667, 2666, 2667, 2667, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9546254  0.7056514
##    2  0.9526246  0.6928529
##    3  0.9567506  0.7012011
##    4  0.9546255  0.6846627
##    5  0.9538758  0.6690585
##    6  0.9521254  0.6547172
##    7  0.9518756  0.6474152
##    8  0.9511254  0.6395906
##    9  0.9493757  0.6208902
##   10  0.9475003  0.6037468
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
Testknn=train(Personal.Loan~.,
                data=Test.df,
                method='knn',
                tuneGrid=searchGrid,
                trControl = fitControl)
Testknn
```

```
## k-Nearest Neighbors
##
## 1000 samples
##   10 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 2 times)
## Summary of sample sizes: 667, 667, 666, 666, 667, 667, ...
## Resampling results across tuning parameters:
```

```
##
##   k   Accuracy   Kappa
##    1  0.9430014  0.6217901
##    2  0.9355014  0.5558319
##    3  0.9430014  0.5701336
##    4  0.9389944  0.5358721
##    5  0.9415014  0.5539398
##    6  0.9410009  0.5466892
##    7  0.9319994  0.4553176
##    8  0.9295014  0.4193924
##    9  0.9285019  0.4043175
##   10  0.9289994  0.4167793
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
To_Predict=data.frame(Age=40, Experience=10,Income=84,Family=2,
                      CCAvg=2,Education=1, Mortgage=0,
                      Securities.Account=0,CD.Account=0,Online=1,CreditCard=1 )

predictions<-predict(Trainknn,Train.df)
confusionMatrix(predictions,Train.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2514    0
##          1    0  282
##
##                Accuracy : 1
##                  95% CI : (0.9987, 1)
##     No Information Rate : 0.8991
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0000
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 1.0000
##              Prevalence : 0.8991
##          Detection Rate : 0.8991
##    Detection Prevalence : 0.8991
##       Balanced Accuracy : 1.0000
##
##        'Positive' Class : 0
##
```

```
predictions<-predict(validationknn,validation.df)
confusionMatrix(predictions,validation.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3613   88
##          1    3  296
##
##                Accuracy : 0.9772
##                  95% CI : (0.9721, 0.9816)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8545
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9992
##             Specificity : 0.7708
##          Pos Pred Value : 0.9762
##          Neg Pred Value : 0.9900
##              Prevalence : 0.9040
##          Detection Rate : 0.9032
##    Detection Prevalence : 0.9253
##       Balanced Accuracy : 0.8850
##
##        'Positive' Class : 0
##
```

```
predictions<-predict(Testknn,Test.df)
confusionMatrix(predictions,Test.df$Personal.Loan)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 902  23
##          1   2  73
##
##                Accuracy : 0.975
##                  95% CI : (0.9633, 0.9838)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8404
##
##  Mcnemar's Test P-Value : 6.334e-05
##
##             Sensitivity : 0.9978
##             Specificity : 0.7604
##          Pos Pred Value : 0.9751
##          Neg Pred Value : 0.9733
##              Prevalence : 0.9040
##          Detection Rate : 0.9020
##    Detection Prevalence : 0.9250
```

```
##       Balanced Accuracy : 0.8791
##
##        'Positive' Class : 0
##
```