Assignment 02

Task 1

Loading the UniversalBank.csv and display the summary of dataset

```
data <- read.csv("UniversalBank.csv")
summary(data)
```

```
##        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
```

Now observe the attribute of dataframe

```
str(data)
```

```
## '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 ...
```

Task 1

```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(caret)

# Load the dataset
df <- read.csv("UniversalBank.csv", header = TRUE)

# Create a data frame for the new customer
new_customer <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2,
                          CCAvg = 2, Education_1 = 0, Mortgage = 0, Securities.Account = 0,
                          CD.Account = 0, Online = 1, CreditCard = 1)

# Transform categorical predictors into dummy variables
df$Education <- as.factor(df$Education)
df$Personal.Loan <- as.factor(df$Personal.Loan)
df$Securities.Account <- as.factor(df$Securities.Account)
df$CD.Account <- as.factor(df$CD.Account)
df$Online <- as.factor(df$Online)
df$CreditCard <- as.factor(df$CreditCard)

# Partition the dataset into 60% training and 40% validation sets
set.seed(123)
trainIndex <- createDataPartition(df$Personal.Loan, p = 0.6, list = FALSE)
train_set <- df[trainIndex, ]
valid_set <- df[-trainIndex, ]

#remove column ID and ZipCode

library(class)
# Apply k-NN classification with k = 1
knn_pred <- knn(train_set[, -c(1, 5,10)], valid_set[, -c(1, 5,10)], train_set$Personal.Loan, k = 1)

# Predict the new class of new customer
```

```r
new_cust_pred <- knn(train_set[, -c(1, 5,10)], new_customer, train_set$Personal.Loan, k = 1)
new_cust_pred
```

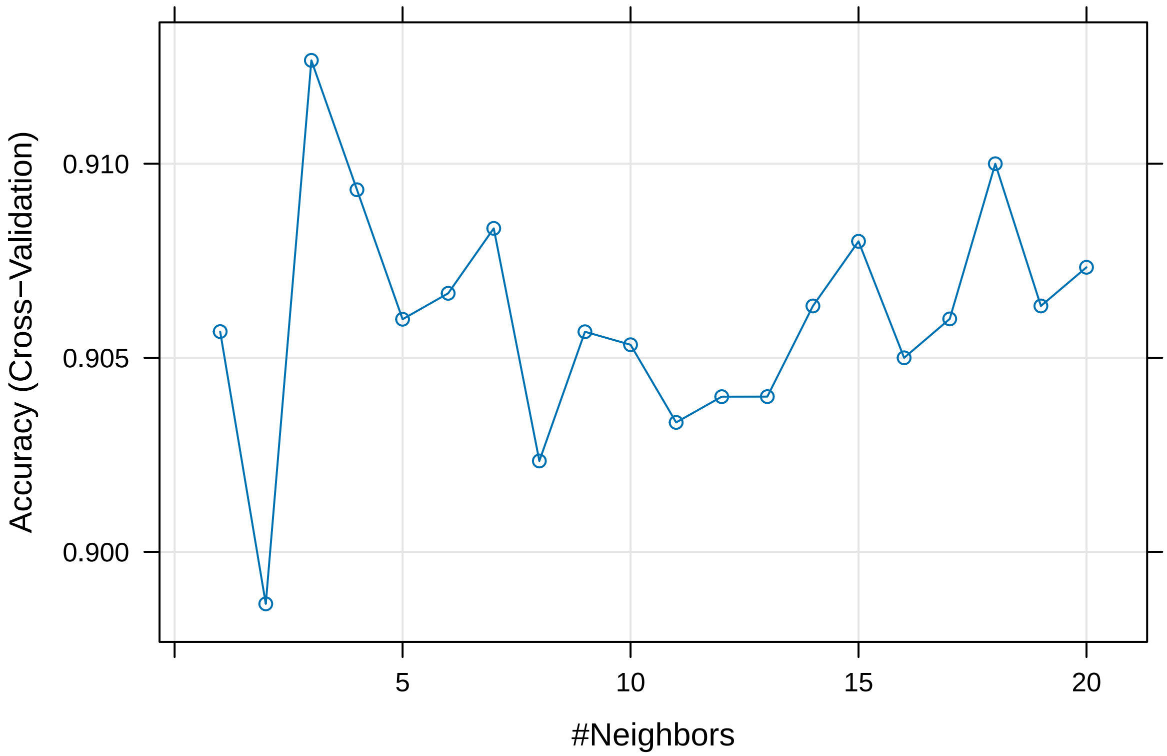
```
## [1] 0
## Levels: 0 1
```

task 2 What is the choice of the k that balances between the overfitting and ignoring the predictor information

```r
# Apply k fold cross-validation to compute the best value of k
set.seed(123)
n_fold <- trainControl(method = "cv", number = 10)
n_seq <- seq(1, 20, by = 1)
model <- train(Personal.Loan ~ ., data = train_set[, -c(1, 5)], method = "knn",
                   trControl = n_fold, tuneGrid = data.frame(k = n_seq))
model
```

```
## k-Nearest Neighbors
##
## 3000 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2700, 2700, 2700, 2701, 2700, 2700, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9056742  0.4196198
##    2  0.8986609  0.3944403
##    3  0.9126609  0.4182487
##    4  0.9093276  0.4045227
##    5  0.9059920  0.3405011
##    6  0.9066598  0.3471799
##    7  0.9083332  0.3358333
##    8  0.9023409  0.2889865
##    9  0.9056698  0.2990069
##   10  0.9053354  0.3018046
##   11  0.9033342  0.2722690
##   12  0.9039976  0.2683909
##   13  0.9039987  0.2519058
##   14  0.9063343  0.2793607
##   15  0.9079998  0.2869012
##   16  0.9049987  0.2757670
##   17  0.9060009  0.2770585
##   18  0.9099976  0.2952039
##   19  0.9063343  0.2579730
##   20  0.9073309  0.2789819
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```r
# Plot different values of k
plot(model)
```



Task 3 Show the confusion matrix for validation data that result form using the best k

```r
pred <- knn(train = train_set[,-10],test = valid_set[,-10], cl = train_set[,10], k=3)
confusionMatrix(pred, valid_set[,10])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1780  180
##          1   28   12
##
##                Accuracy : 0.896
##                  95% CI : (0.8818, 0.909)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.8938
##
##                   Kappa : 0.0728
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9845
##             Specificity : 0.0625
```

```
##          Pos Pred Value : 0.9082
##          Neg Pred Value : 0.3000
##              Prevalence : 0.9040
##          Detection Rate : 0.8900
##    Detection Prevalence : 0.9800
##       Balanced Accuracy : 0.5235
##
##        'Positive' Class : 0
##
```

Task 4

Consider the following customer: 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. Classify the customer using the best k.

```
new_customer2 <- data.frame(Age = 40, Experience = 10, Income = 84, Family = 2, CCAvg = 2, Education = (
knn_build <- knn(train = train_set[,-c(1,5,10)],test = new_customer2, cl = train_set[,10], k=3)
knn_build
```

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

Task 5

Repartition the data, this time into training, validation, and test sets (50% : 30% : 20%). Apply the k-NN method with the k chosen above. Compare the confusion matrix of the test set with that of the training and validation sets. Comment on the differences and their reason.

```
## Repartition the data into training, validation, and test sets (50% : 30% : 20%)
set.seed(123) # Setting seed to reproduce results
index_train <- createDataPartition(df$Personal.Loan, p = 0.5, list = FALSE)
train_set <- df[index_train,]
index_val_test <- createDataPartition(df[-index_train, ]$Personal.Loan, p = 0.6, list = FALSE)
val_test_set <- df[-index_train, ]
val_set <- val_test_set[index_val_test, ]
test_set <- val_test_set[-index_val_test, ]

# Set up the train control with 10-fold cross-validation
trControl <- trainControl(method="cv", number=10)

set.seed(123)
valid_pred <- knn(train_set[,-c(1,5,10)], val_set[,-c(1,5,10)], cl = train_set[,10], k = 3)
test_pred <- knn(train_set[,-c(1,5,10)], test_set[,-c(1,5,10)], cl = train_set[,10], k = 3)
train_pred <- knn(train_set[,-c(1,5,10)], train_set[,-c(1,5,10)], cl = train_set[,10], k = 3)



# Create confusion matrices
confusion_train <- confusionMatrix(train_pred, train_set$Personal.Loan)
confusion_val <- confusionMatrix(valid_pred, val_set$Personal.Loan)
confusion_test <- confusionMatrix(test_pred, test_set$Personal.Loan)
```

```r
# Display the confusion matrices
confusion_train
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2223   91
##          1   37  149
##
##                Accuracy : 0.9488
##                  95% CI : (0.9394, 0.9571)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.672
##
##  Mcnemar's Test P-Value : 2.805e-06
##
##             Sensitivity : 0.9836
##             Specificity : 0.6208
##          Pos Pred Value : 0.9607
##          Neg Pred Value : 0.8011
##              Prevalence : 0.9040
##          Detection Rate : 0.8892
##    Detection Prevalence : 0.9256
##       Balanced Accuracy : 0.8022
##
##        'Positive' Class : 0
##
```

```r
confusion_val
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1311   87
##          1   45   57
##
##                Accuracy : 0.912
##                  95% CI : (0.8965, 0.9259)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.1566455
##
##                   Kappa : 0.417
##
##  Mcnemar's Test P-Value : 0.0003589
##
##             Sensitivity : 0.9668
##             Specificity : 0.3958
##          Pos Pred Value : 0.9378
```

```
##          Neg Pred Value : 0.5588
##              Prevalence : 0.9040
##          Detection Rate : 0.8740
##    Detection Prevalence : 0.9320
##       Balanced Accuracy : 0.6813
##
##        'Positive' Class : 0
##
```

```
confusion_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 876  59
##          1  28  37
##
##                Accuracy : 0.913
##                  95% CI : (0.8938, 0.9297)
##     No Information Rate : 0.904
##     P-Value [Acc > NIR] : 0.181360
##
##                   Kappa : 0.4142
##
##  Mcnemar's Test P-Value : 0.001298
##
##             Sensitivity : 0.9690
##             Specificity : 0.3854
##          Pos Pred Value : 0.9369
##          Neg Pred Value : 0.5692
##              Prevalence : 0.9040
##          Detection Rate : 0.8760
##    Detection Prevalence : 0.9350
##       Balanced Accuracy : 0.6772
##
##        'Positive' Class : 0
##
```

Observation The confusion matrices reveal good model accuracy but highlight issues with specificity, indicating potential difficulties in correctly identifying negative cases. This, combined with a drop in performance on validation and test data compared to training data, suggests possible overfitting and a need for model adjustments or additional strategies to address class imbalance.