Learning outcomes:

After solving these exercises, you should be able to understand the following:

Learning Outcomes:

- KNN classification
  - Impact of standardizing and not standardizing
  - Play with neighbors to determine the number of nearest neighbors
  - Condensing points/ Border points
  - To get the number of nearest neighbors
- KNN regression
  - Not removing the target variable
  - Remove the target variable from the data
  - with test data and w/o test data
- Collaborative Filtering

KNN-Classification:

Identifying the loan takers and non-loan takers.

Steps to follow to execute the problem:

1. Load data 'UniversalBank.csv' into R
2. Consider "Personal.Loan " as target attribute
3. Understand the summary of data
4. Check for the missing values, if so, impute them
5. Remove the columns that may not be used for analysis (ID and Zip Code) bankdata2 = subset(bankdata, select=-c(ID,ZIP.Code))
6. Convert categorical attributes into numeric
   a. Convert "education" as factor variable
   b. Modify into dummy variable, add the dummy attributes to data and drop the original one

```
bankdata2$Education=as.factor(as.character(bankdata2$Education))
Education=dummy(bankdata2$Education)
bankdata3=subset(bankdata2,select=-c(Education))
bankdata4=cbind(bankdata3,Education)
```

7. Now let us check KNN results with and without standardizing the data

Without standardizing the data

- Split this data set into train and test and observe the distribution of personal loan in train and test data
- Separate out independent attributes and target variable into two data frames

```
bankdata_trainwithoutclass=subset(bankdata_train,select=-c(Personal.Loan))
bankdata_testwithoutclass=subset(bankdata_test,select=-c(Personal.Loan))
```

- Run the model on test data using different k values and check the accuracy values and come up with optimal k value

```
pred=knn(bankdata_trainwithoutclass,bankdata_testwithoutclass,
bankdata_train$Personal.Loan,k=1)
a=table(pred,bankdata_test$Personal.Loan)
a = sum(diag(a))/nrow(bankdata_testwithoutclass)
accu
```

b. **Standardizing the data**

- Standardize the independent attributes data using 'Range' method and then merge thetarget variable with this standardize data library(vegan)
  `bankdata5=decostand(bankdata4,"range")`
- Split this data set into train and test and observe the distribution of personal loan in train and test data
- Separate out independent attributes and target variable in two data frames

```
bankdata_trainwithoutclass=subset(bankdata_train,select=-c(Personal.Loan))
bankdata_testwithoutclass = subset(bankdata_test,select=-c(Personal.Loan))
```

- Run the model on test data using different k values and check the accuracy values and come up with optimal k value

```
pred=knn(bankdata_trainwithoutclass,bankdata_testwithoutclass,
bankdata_train$Personal.Loan,k=1)
a=table(pred,bankdata_test$Personal.Loan)

a
accu= sum(diag(a))/nrow(bankdata_testwithoutclass)
accu
```

8. Condensing to reduce the complexity of the model

```
keep=condense(bankdata_trainwithoutclass, bankdata_train$Personal.Loan)
keep
```

9. Take condensed data and run the model compare the accuracies with whole data and condensed data

```
pred=knn(bankdata_trainwithoutclass[keep,],bankdata_testwithoutclass,
bankdata_train$Personal.Loan[keep],k=5)
a <- table(pred,bankdata_test$Personal.Loan)
a
accu=sum(diag(a))/nrow(bankdata_testwithoutclass)
accu
```

10. Now we can find the indices of the records that are considered for prediction in the model for a specific record of test data using FNN library.
    - First install library FNN and do the following

```
# run the model using FNN library
library(FNN)
pred=FNN::knn(bankdata_trainwithoutclass[keep,],bankdata_t
estwithoutclass, bankdata_train$Personal.Loan[keep],k=5)
a<- table(pred,bankdata_test$Personal.Loan)
a
accu=sum(diag(a))/nrow(bankdata_test)
accu
indices=knnx.index(bankdata_trainwithoutclass[keep,],
bankdata_testwithoutclass, k=5)

# If you want the indices of the 5 nearest neighbors for the row 20 of test dataset
print(indices[20, ])
```

KNN-Regression:

Steps to follow to execute the problem:

- Install the packages FNN, Metrics
  install.packages("FNN") #"Fast Nearest Neighbours" for knn regression
  install.packages("Metrics") #to calculate error metrics for regression
- Let us generate the data for regression
  #set.seed() set.seed(12345) #to get same random numbers generated every time
  #Create a dataframe of 100 rows and 25 columns
  data <- data.frame(matrix(data = runif(2500, 24,65), nrow = 100, ncol = 25))

3. Target attribute is "x25"

4. Split this data set into train and test

5. Separate out independent attributes and target variable in two data frame

```
## Excluding Target Variable
testData <- data[sample(81:100),1:24]
trainData <- data[1:80,1:24]
 train.tgt <- data[1:80,25]
test.tgt <- data[sample(81:100),25]
```

6. Let us run KNN model now & compute rmse for different k values and come up with optimal k value

```
# Run the model
pred <- knn.reg(train = trainData, test = testData, y = train.tgt, k = 1 )
actual <- test.tgt
pred <- data.frame(pred$pred)
result2 <- rmse(actual = actual, predicted = pred)
```

Assignment:

1. **Classification:** Read the dataset "dataforAssignement.csv" and consider "class" as target attribute for classification.
   - Apply the necessary pre-processing steps.
   - Prediction: Predict whether a person's income exceeds $50K/yr based on the data given
2. **Regression:** Read the dataset "CustomerData.csv" and "Revenue" is the target attribute.
   - Apply the necessary pre-processing steps.
   - Prediction: Predict the revenue generated by the customer.

Collaborative Filtering:

Based on a given user-item ratings matrix, recommend a rating for a given user for given item, using user-based and item-based collaborative filtering using "recommenderlab" package in R

Simulated data: 5X10 user-ratings matrix
R Packages to use: recommenderlab for collaborative filtering

1. Simulate a sparse user-item ratings matrix with 5 users, 10 items, and randomly assign users' ratings (in the range of 0 to 5) for each item
2. Convert the user-item ratings matrix into the "realRatingMatrix" format
3. Mean-normalize the real ratings matrix
4. Provide normalized ratings matrix as input to Recommender() function from recommenderlab, and say method = "UBCF" for building a user-based collaborative filtering model
5. Using this model, predict the rating for a given item for a given user
6. Repeat steps 4 and 5, specifying method = "IBCF", for building an item-based collaborative filtering model
7. Compare and analyze the results