KENTUCKY STATE UNIVERSITY, COMPUTER SCIENCE

Data Mining
HW4

Anil Varma B

October 25, 2019

Download dataset, first normalize it and then partition it into two subset: training (70%) and test (30%). Apply KNN with different k to the dataset and determine the best k in terms of error rate.

1. Reading the data set

```
#Read CSV file
risk <-read.csv(file.choose(),head=TRUE)
#Finding the Data set summary
summary(risk)
#Reading the type of sample data
head(risk)
#Dimensions of the dataset
dim(risk)
```

```
> risk <-read.csv(file.choose(),head=TRUE)
> summary(risk)
 mortgage      loans            age        marital_status      income
 n: 71    Min.   :0.000   Min.   :17.00   married: 78     Min.   :15301
 y:175    1st Qu.:1.000   1st Qu.:32.00   other  : 57     1st Qu.:26882
          Median :1.000   Median :41.00   single :111     Median :37662
          Mean   :1.309   Mean   :40.64                   Mean   :38790
          3rd Qu.:2.000   3rd Qu.:50.00                   3rd Qu.:49398
          Max.   :3.000   Max.   :66.00                   Max.   :78399
        risk
 bad loss :123
 good risk:123
```

```
> head(risk)
  mortgage loans age marital_status   income     risk
1        y     3  34          other 28060.70 bad loss
2        n     2  37          other 28009.34 bad loss
3        n     2  29          other 27614.60 bad loss
4        y     2  33          other 27287.18 bad loss
5        y     2  39          other 26954.06 bad loss
6        n     2  28          other 26271.86 bad loss
> dim(risk)
[1] 246   6
>
```

Figure 0.1: Reading the data set

As we can see from the sample data, we have the predictor variable column Risk, the rest of the columns are separated into several classes to predict the classification of a new sample point.

2. Converting the categorical columns to numerical

We are converting the categorical values to numerical for analysis using the dummies method.

```
library(dummies)
risk_df<-dummy.data.frame(as.data.frame(risk), sep = "_")
head(risk_df)
```

```
> head(risk_df)
  mortgage_n mortgage_y loans age marital_status_married marital_status_other marital_status_single   income
1          0          1     3  34                      0                    1                     0 28060.70
2          1          0     2  37                      0                    1                     0 28009.34
3          1          0     2  29                      0                    1                     0 27614.60
4          0          1     2  33                      0                    1                     0 27287.18
5          0          1     2  39                      0                    1                     0 26954.06
6          1          0     2  28                      0                    1                     0 26271.86
  risk_bad loss risk_good risk
1             1              0
2             1              0
3             1              0
4             1              0
5             1              0
6             1              0
> 
```

Figure 0.2: Using Dummies

```
#Dropping the dummy columns which are not required for the train and test data set.
risk_df<-risk_df[-c(1)]
risk_df<-risk_df[-c(5)]
risk_df<-risk_df[-c(7:8)]
head(risk_df)
```

```
> head(risk_df)
  mortgage_y loans age marital_status_married marital_status_single   income
1          1     3  34                      0                     0 28060.70
2          0     2  37                      0                     0 28009.34
3          0     2  29                      0                     0 27614.60
4          1     2  33                      0                     0 27287.18
5          1     2  39                      0                     0 26954.06
6          0     2  28                      0                     0 26271.86
>
```

Figure 0.3: Final Data frame

3. Normalizing the data

Here we are normalizing only the columns age,income and loans as the other variable columns are already normalized.

```
#Normalize function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

#Normalize the required columns
risk_df$age<-normalize(risk_df$age)
risk_df$income<-normalize(risk_df$income)
risk_df$loans<-normalize(risk_df$loans)
summary(risk_df)
```

```
> summary(risk_df)
   mortgage_y         loans             age          marital_status_married marital_status_single     income      
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000         Min.   :0.0000         Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.:0.3333   1st Qu.:0.3061   1st Qu.:0.0000         1st Qu.:0.0000         1st Qu.:0.1835  
 Median :1.0000   Median :0.3333   Median :0.4898   Median :0.0000         Median :0.0000         Median :0.3544  
 Mean   :0.7114   Mean   :0.4363   Mean   :0.4825   Mean   :0.3171         Mean   :0.4512         Mean   :0.3723  
 3rd Qu.:1.0000   3rd Qu.:0.6667   3rd Qu.:0.6735   3rd Qu.:1.0000         3rd Qu.:1.0000         3rd Qu.:0.5404  
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000         Max.   :1.0000         Max.   :1.0000  
>
```

Figure 0.4: Normalized dataset summary

4. Partitioning the data set

Partitioning the dataset into two subset: 70% for the training set and 30% for the testing set.

```
set.seed(123)
ran<- sample(1:nrow(risk_df), 0.7 * nrow(risk_df))

##extract training set
risk_train <- risk_df[ran,]
##extract testing set
risk_test <- risk_df[-ran,]

#The category variable is the risk column taken from the original data set
risk_target_category <- risk[ran,6]
risk_test_category <- risk[-ran,6]
```

5. Applying KNN with different k to the dataset.

We are applying KNN with different k value starting from value 13 till 5, to determine the best value of K we are creating the confusion matrix for each KNN and determining the accuracy of prediction for each case.

```
> ##load the package class
> library(class)
> #Run KNN function for different K values
> pr <- knn(risk_train,risk_test,cl=risk_target_category,k=13)
> ##create confusion matrix
> table(pr,risk_test_category)
            risk_test_category
pr            bad loss good risk
  bad loss          27         5
  good risk          9        33
> #Function to determine Accuracy of prediction
> accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
> tab<-table(pr,risk_test_category)
> accuracy(tab)
[1] 81.08108
```

Figure 0.5: Applying KNN part 1

```
> #Run KNN function for different K values
> pr_1 <- knn(risk_train,risk_test,cl=risk_target_category,k=12)
> table(pr_1,risk_test_category)
           risk_test_category
pr_1         bad loss good risk
  bad loss         27         5
  good risk         9        33
> tab1<-table(pr_1,risk_test_category)
> accuracy(tab1)
[1] 81.08108
> pr_2 <- knn(risk_train,risk_test,cl=risk_target_category,k=11)
> table(pr_2,risk_test_category)
           risk_test_category
pr_2         bad loss good risk
  bad loss         27         5
  good risk         9        33
> tab2<-table(pr_2,risk_test_category)
> accuracy(tab2)
[1] 81.08108
> pr_3 <- knn(risk_train,risk_test,cl=risk_target_category,k=10)
> table(pr_3,risk_test_category)
           risk_test_category
pr_3         bad loss good risk
  bad loss         27         5
  good risk         9        33
> tab3<-table(pr_3,risk_test_category)
> accuracy(tab3)
[1] 81.08108
> pr_4 <- knn(risk_train,risk_test,cl=risk_target_category,k=9)
> table(pr_4,risk_test_category)
           risk_test_category
pr_4         bad loss good risk
  bad loss         28         4
  good risk         8        34
> tab4<-table(pr_4,risk_test_category)
> accuracy(tab4)
[1] 83.78378
```

Figure 0.6: Applying KNN part 2

```
> pr_5 <- knn(risk_train,risk_test,cl=risk_target_category,k=8)
> table(pr_5,risk_test_category)
             risk_test_category
pr_5          bad loss good risk
  bad loss          28         4
  good risk          8        34
> tab5<-table(pr_5,risk_test_category)
> accuracy(tab5)
[1] 83.78378
> pr_6 <- knn(risk_train,risk_test,cl=risk_target_category,k=7)
> table(pr_6,risk_test_category)
             risk_test_category
pr_6          bad loss good risk
  bad loss          29         6
  good risk          7        32
> tab6<-table(pr_6,risk_test_category)
> accuracy(tab6)
[1] 82.43243
> pr_7 <- knn(risk_train,risk_test,cl=risk_target_category,k=6)
> table(pr_7,risk_test_category)
             risk_test_category
pr_7          bad loss good risk
  bad loss          31         5
  good risk          5        33
> tab7<-table(pr_7,risk_test_category)
> accuracy(tab7)
[1] 86.48649
> pr_8 <- knn(risk_train,risk_test,cl=risk_target_category,k=5)
> table(pr_8,risk_test_category)
             risk_test_category
pr_8          bad loss good risk
  bad loss          29         7
  good risk          7        31
> tab8<-table(pr_8,risk_test_category)
> accuracy(tab8)
[1] 81.08108
```

Figure 0.7: Applying KNN part 3

From the above observation, the best K value is 6 as it has the highest accuracy.

6. Validating K-value with silhouette Curve

We plot the silhouette curve to know the best suggested k value and get the k value 6 which validates the above KNN analysis.

```
#Plotting silhouette Curve to get the best value of k.
library(factoextra)
fviz_nbclust(risk_df, cluster::pam, method = "silhouette", k.max = 15,
 print.summary = TRUE) + theme_minimal() + ggtitle("the silhouette Method")
```

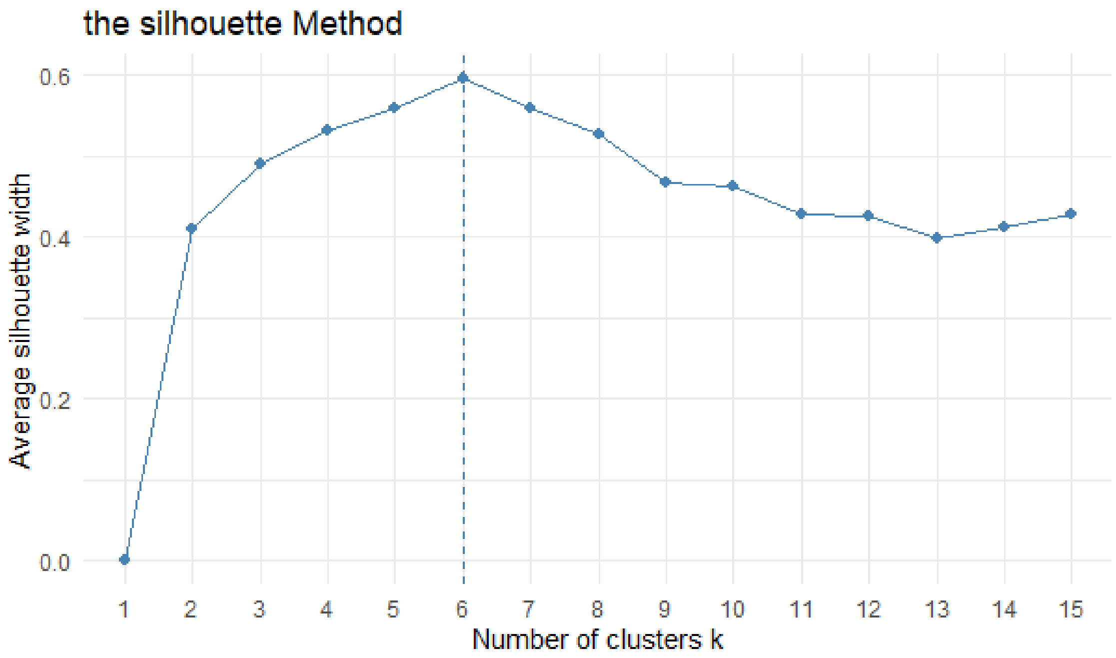

Figure 0.8: Normalized dataset summary