EXPERIMENT-4

AIM: Implementation of KNN on the German Credit Data set.

DESCRIPTION: To minimize loss from the bank's perspective, the bank needs a decision rule regarding who to give approval of the loan and who not to. An applicant's demographic and socio-economic profiles are considered by loan managers before a decision is taken regarding his/her loan application. The German Credit Data contains data on 20 variables and the classification whether an applicant is considered a Good or a Bad credit risk for 1000 loan applicants. A predictive model developed on this data is expected to provide bank manager guidance for making a decision whether to approve a loan to a prospective applicant based on his/her profiles

DESCRIPTION:
The K-NN working can be explained on the bias of the below algorithm:
o Step-1: Select the number K of the neighbours
o Step-2: Calculate the Euclidean distance of K number of neighbours
o Step-3: Take the K nearest neighbours as per the calculated Euclidean distance.
o Step-4: Among these k neighbours, count the number of the data points in each category.
o Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.
o Step-6: Our model is ready.2

Steps Involved in performing KNN algorithm:
1. Data Collection.
2. Preparing and exploring the data.
   - Understanding data structure.
   - Feature selection (if required)
   - Data normalization.
   - Creating Training and Test data set.
3. Training a model on data.
4. Evaluate the model performance.
5. Improve the performance of model.

1.DATA COLLECTION:
CODE:
```
gc <- read.csv("C:/Users/Hp/OneDrive/Desktop/ML-LAB/german_credit.csv")
gc.bkup <- gc
head (gc)
```
OUTPUT:
```
> gc <- read.csv("C:/Users/Hp/OneDrive/Desktop/ML-LAB/german_credit.csv")
> gc.bkup <- gc
> head (gc)
  Creditability Account.Balance Duration.of.Credit..month. Payment.Status.of.Previous.Credit Purpose
1             1               1                         18                                 4       2
2             1               1                          9                                 4       0
3             1               2                         12                                 2       9
4             1               1                         12                                 4       0
5             1               1                         12                                 4       0
6             1               1                         10                                 4       0
  Credit.Amount Value.Savings.Stocks Length.of.current.employment Instalment.per.cent Sex...Marital.Status
1          1049                    1                            2                   4                    2
2          2799                    1                            3                   2                    3
3           841                    2                            4                   2                    2
4          2122                    1                            3                   3                    3
5          2171                    1                            3                   4                    3
6          2241                    1                            2                   1                    3
  Guarantors Duration.in.Current.address Most.valuable.available.asset Age..years. Concurrent.Credits
1          1                           4                             2          21                  3
2          1                           2                             1          36                  3
3          1                           4                             1          23                  3
4          1                           2                             1          39                  3
5          1                           4                             2          38                  1
6          1                           3                             1          48                  3
  Type.of.apartment No.of.Credits.at.this.Bank Occupation No.of.dependents Telephone Foreign.Worker
1                 1                          1          3                1         1              1
2                 1                          2          3                2         1              1
3                 1                          1          2                1         1              1
4                 1                          2          2                2         1              2
5                 2                          2          2                1         1              2
6                 1                          2          2                2         1              2
```

2.PREPARING AND EXPLORING THE DATA:

CODE:

```
str(gc)
gc.subset <-
gc[c('Creditability','Age..years.','Sex...Marital.Status','Occupation','Account.Balance','Credit.Amount','Length.of.curre
nt.employment','Purpose')]
head(gc.subset)
normalize <- function(x) {  return ((x - min(x)) / (max(x) - min(x))) }
gc.subset.n<- as.data.frame(lapply(gc.subset[,2:8], normalize))
head(gc.subset.n)
set.seed(123)
dat.d <- sample(1:nrow(gc.subset.n),size=nrow(gc.subset.n)*0.7,replace = FALSE)
train.gc <- gc.subset[dat.d,]
test.gc <- gc.subset[-dat.d,]
train.gc_labels <- gc.subset[dat.d,1]
test.gc_labels <- gc.subset[-dat.d,1]
```

OUTPUT:

```
> str(gc)
'data.frame':	1000 obs. of  21 variables:
 $ Creditability                    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Account.Balance                  : int  1 1 2 1 1 1 1 1 4 2 ...
 $ Duration.of.Credit..month.       : int  18 9 12 12 12 10 8 6 18 24 ...
 $ Payment.Status.of.Previous.Credit: int  4 4 2 4 4 4 4 4 4 2 ...
 $ Purpose                          : int  2 0 9 0 0 0 0 0 3 3 ...
 $ Credit.Amount                    : int  1049 2799 841 2122 2171 2241 3398 1361 1098 3758 ...
 $ Value.Savings.Stocks             : int  1 1 2 1 1 1 1 1 1 3 ...
 $ Length.of.current.employment     : int  2 3 4 3 3 2 4 2 1 1 ...
 $ Instalment.per.cent              : int  4 2 2 3 4 1 1 2 4 1 ...
 $ Sex...Marital.Status             : int  2 3 2 3 3 3 3 3 2 2 ...
 $ Guarantors                       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Duration.in.Current.address      : int  4 2 4 2 4 3 4 4 4 4 ...
 $ Most.valuable.available.asset    : int  2 1 1 1 2 1 1 1 3 4 ...
 $ Age..years.                      : int  21 36 23 39 38 48 39 40 65 23 ...
 $ Concurrent.Credits               : int  3 3 3 3 1 3 3 3 3 3 ...
 $ Type.of.apartment                : int  1 1 1 1 2 1 2 2 2 1 ...
 $ No.of.Credits.at.this.Bank       : int  1 2 1 2 2 2 2 1 2 1 ...
 $ Occupation                       : int  3 3 2 2 2 2 2 2 1 1 ...
 $ No.of.dependents                 : int  1 2 1 2 1 2 1 2 1 1 ...
 $ Telephone                        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Foreign.Worker                   : int  1 1 1 2 2 2 2 2 1 1 ...
> gc.subset <- gc[c('Creditability','Age..years.','Sex...Marital.Status','Occupation','Account.Balance','Credit.Amou
t','Length.of.current.employment','Purpose')]
> head(gc.subset)
  Creditability Age..years. Sex...Marital.Status Occupation Account.Balance Credit.Amount
1             1          21                    2          3               1          1049
2             1          36                    3          3               1          2799
3             1          23                    2          2               2           841
4             1          39                    3          2               1          2122
5             1          38                    3          2               1          2171
6             1          48                    3          2               1          2241
  Length.of.current.employment Purpose
1                            2       2
2                            3       0
3                            4       9
4                            3       0
5                            3       0
6                            2       0
> normalize <- function(x) {  return ((x - min(x)) / (max(x) - min(x))) }
> gc.subset.n<- as.data.frame(lapply(gc.subset[,2:8], normalize))
> head(gc.subset.n)
  Age..years. Sex...Marital.Status Occupation Account.Balance Credit.Amount Length.of.current.employment Purpose
1  0.03571429            0.3333333  0.6666667       0.0000000    0.04396390                         0.25     0.2
2  0.30357143            0.6666667  0.6666667       0.0000000    0.14025531                         0.50     0.0
3  0.07142857            0.3333333  0.3333333       0.3333333    0.03251898                         0.75     0.9
4  0.35714286            0.6666667  0.3333333       0.0000000    0.10300429                         0.50     0.0
5  0.33928571            0.6666667  0.3333333       0.0000000    0.10570045                         0.50     0.0
6  0.51785714            0.6666667  0.3333333       0.0000000    0.10955211                         0.25     0.0
> set.seed(123)
> dat.d <- sample(1:nrow(gc.subset.n),size=nrow(gc.subset.n)*0.7,replace = FALSE)
> train.gc <- gc.subset[dat.d,]
> test.gc <- gc.subset[-dat.d,]
> train.gc_labels <- gc.subset[dat.d,1]
> test.gc_labels <- gc.subset[-dat.d,1]
```

3.TRAINING A MODEL ON DATA:

CODE:

```
library(class)
NROW(train.gc_labels)
knn.26 <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=26)
knn.27 <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=27)
```

OUTPUT:

```
> library(class)
> NROW(train.gc_labels)
[1] 700
> knn.26 <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=26)
> knn.27 <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=27)
```

4.EVALUATE THE MODEL PERFORMANCE:

CODE:

```
ACC.26 <- 100 * sum(test.gc_labels == knn.26)/NROW(test.gc_labels)
ACC.27 <- 100 * sum(test.gc_labels == knn.27)/NROW(test.gc_labels)
ACC.26
ACC.27
table(knn.26 ,test.gc_labels)
table(knn.27 ,test.gc_labels)
install.packages('caret')
library(caret)
test.gc_labels=as.factor(test.gc_labels)
confusionMatrix(knn.26 ,test.gc_labels)
```

OUPUT:

```
> ACC.26 <- 100 * sum(test.gc_labels == knn.26)/NROW(test.gc_labels)
> ACC.27 <- 100 * sum(test.gc_labels == knn.27)/NROW(test.gc_labels)
> ACC.26
[1] 69
> ACC.27
[1] 69
> table(knn.26 ,test.gc_labels)
      test.gc_labels
knn.26   0   1
     0   8   6
     1  87 199
> table(knn.27 ,test.gc_labels)
      test.gc_labels
knn.27   0   1
     0   8   6
     1  87 199
> install.packages('caret')
Error in install.packages : Updating loaded packages
> library(caret)
> test.gc_labels=as.factor(test.gc_labels)
> confusionMatrix(knn.26 ,test.gc_labels)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   8   6
         1  87 199

               Accuracy : 0.69
                 95% CI : (0.6343, 0.7419)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 0.4291

                  Kappa : 0.0712

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.08421
            Specificity : 0.97073
         Pos Pred Value : 0.57143
         Neg Pred Value : 0.69580
             Prevalence : 0.31667
         Detection Rate : 0.02667
   Detection Prevalence : 0.04667
      Balanced Accuracy : 0.52747

       'Positive' Class : 0
```

CODE:

confusionMatrix(knn.27 ,test.gc_labels)

OUTPUT:

```
> confusionMatrix(knn.27 ,test.gc_labels)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   8   6
         1  87 199

               Accuracy : 0.69
                 95% CI : (0.6343, 0.7419)
    No Information Rate : 0.6833
    P-Value [Acc > NIR] : 0.4291

                  Kappa : 0.0712

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.08421
            Specificity : 0.97073
         Pos Pred Value : 0.57143
         Neg Pred Value : 0.69580
             Prevalence : 0.31667
         Detection Rate : 0.02667
   Detection Prevalence : 0.04667
      Balanced Accuracy : 0.52747

       'Positive' Class : 0
```

5.IMPROVE THE PERFORMANCE OF MODEL:

CODE:

```
i=1
k.optm=1
for (i in 1:28){
 knn.mod <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=i)
 k.optm[i] <- 100 * sum(test.gc_labels == knn.mod)/NROW(test.gc_labels)
 k=i
 cat(k,'=',k.optm[i],'\n')
 }
```

OUTPUT:

```
> i=1
> k.optm=1
> for (i in 1:28){
+     knn.mod <- knn(train=train.gc, test=test.gc, cl=train.gc_labels, k=i)
+     k.optm[i] <- 100 * sum(test.gc_labels == knn.mod)/NROW(test.gc_labels)
+     k=i
+     cat(k,'=',k.optm[i],'\n')
+     }
1 = 57.33333
2 = 59.66667
3 = 59.33333
4 = 60.33333
5 = 62.66667
6 = 63.33333
7 = 64.66667
8 = 67.66667
9 = 67
10 = 66.66667
11 = 66.66667
12 = 67
13 = 68
14 = 67.33333
15 = 67.33333
16 = 67.66667
17 = 69
18 = 68.33333
19 = 69
20 = 68.33333
21 = 68.66667
22 = 69
23 = 68.66667
24 = 68.33333
25 = 68.66667
26 = 69
27 = 69
28 = 69
> plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```

CODE:

```
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```

OUTPUT: