UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

In collaboration, with

University of Westminster, UK

BSc. (Hons) in Information Systems and Business Management

Business Intelligence

6BUIS001W

Coursework 2

Dr. Sachintha Pitigala

Student Name: D. M. V. Oshadhi

Student ID: 2015078 (w15831756)

Date of submission: 17th December 2018

TABLE OF CONTENTS

LIST OF FIGURES

1 Objective 1 – Partitioning Clustering

1.1 Complete Code

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Objective 01 latest.R*

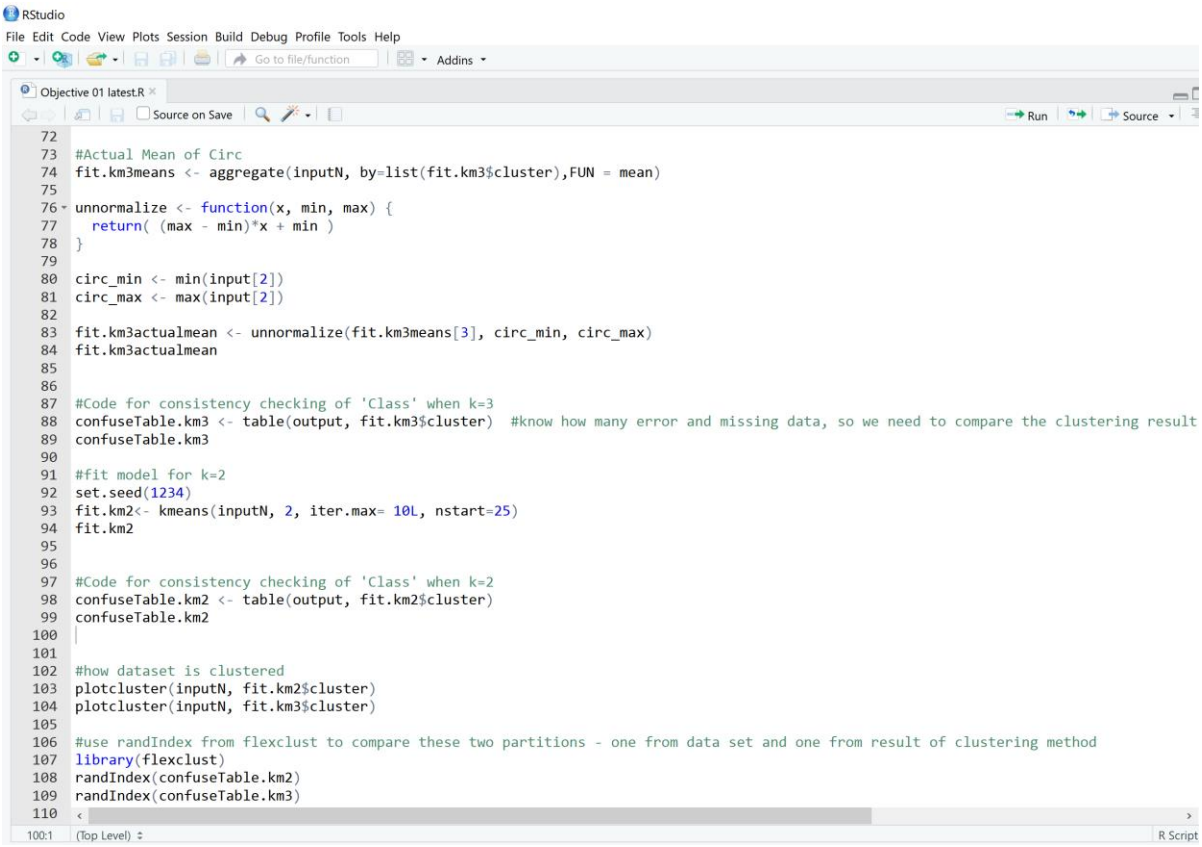
```r
library(NbClust)
library(fpc)

# Loading the original vehicles dataset
original_vehicles <- read.csv("vehicles.csv")
View(original_vehicles)

#Pre processing
#Remove first column (record no)and last column
input <- original_vehicles[,2:19]
str(input) # Check if 20th and 1st columns are removed

#Normalization
normalize<-function(x){return((x-min(x))/(max(x)-min(x)))}
inputN <- as.data.frame(lapply(input,normalize))
head(inputN) #Normalized Data
summary(input)

#Saving last column
output<-original_vehicles$Class

#Determine the best no. of clusters k for k means using NbClust (Method 1)
set.seed(1234)
clusterNo=NbClust(inputN,distance = "euclidean",
                  min.nc = 2, max.nc = 15,
                  method = "kmeans", index = "all")

#Plotting the no. of clusters in a bar chart
barplot(table(clusterNo$Best.n[1,]),
        xlab="Numer of Clusters",
        ylab="Number of Criteria",
        main="Optimal Number of Clusters Chosen by Criteria") #we can find the best number of clusters is 3 using the method 1
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Objective 01 latest.R*

```r
#Find the Optimal K using Elbow method (Method 2)

#The Within Sum of Squares and the number of clusters to find the location of a bend or a knee in the plot which is considered as an indicator of the
set.seed(1234)
k = 1:15
wss = sapply(k, function(k) {kmeans(inputN, centers=k)$tot.withinss})
#lineplot to plot the within sum of squares
plot(k, wss, type="b", pch = 19,
     xlab= "Number of Cluster (k value)",
     ylab="Within sum of squares",
     main="Elbow graph")

abline(v=3,lty = 2, col="blue", lwd=2)


#Both two methods suggest k=3 is best choice for us. Next best aternative is k=2.

#K- means code for best 2 chosen clusters as given in the question

#fit model for k=3
#set.seed(1234)
fit.km3<- kmeans(inputN, 3, iter.max=10L, nstart=25)
fit.km3

#Actual means of each atrributes  in wining cluster k=3

#Actual Mean of Comp
fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)

unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}

comp_min <- min(input[1])
comp_max <- max(input[1])

fit.km3actualmean <- unnormalize(fit.km3means[2], comp_min, comp_max)
fit.km3actualmean
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Objective 01 latest.R
Source on Save
Run
Source
#Actual Mean of Circ
fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)
unnormalize <- function(x, min, max) {
return((max - min)*x + min)
}
circ_min <- min(input[2])
circ_max <- max(input[2])
fit.km3actualmean <- unnormalize(fit.km3means[3], circ_min, circ_max)
fit.km3actualmean
#Code for consistency checking of 'Class' when k=3
confuseTable.km3 <- table(output, fit.km3$cluster) #know how many error and missing data, so we need to compare the clustering result
confuseTable.km3
#fit model for k=2
set.seed(1234)
fit.km2<- kmeans(inputN, 2, iter.max= 10L, nstart=25)
fit.km2
#Code for consistency checking of 'Class' when k=2
confuseTable.km2 <- table(output, fit.km2$cluster)
confuseTable.km2
#how dataset is clustered
plotcluster(inputN, fit.km2$cluster)
plotcluster(inputN, fit.km3$cluster)
#use randIndex from flexclust to compare these two partitions - one from data set and one from result of clustering method
library(flexclust)
randIndex(confuseTable.km2)
randIndex(confuseTable.km3)
100:1
(Top Level)
R Script

1.2 Cluster Identification

1) Libraries should be loaded
2) Load the Vehicles dataset after setting the working directory to the location of the CSV file.

RStudio Source Editor

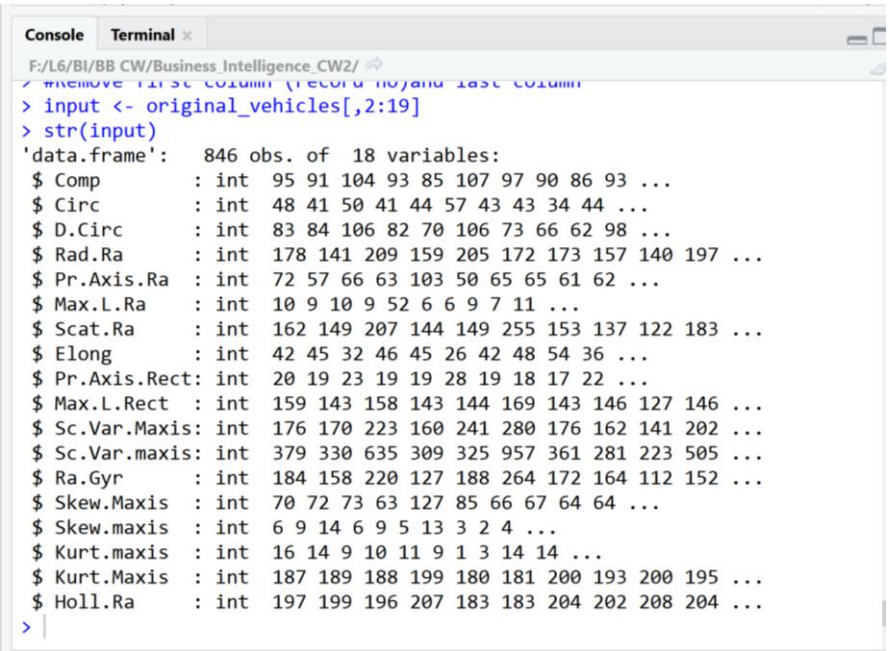
original_vehicles

Filter

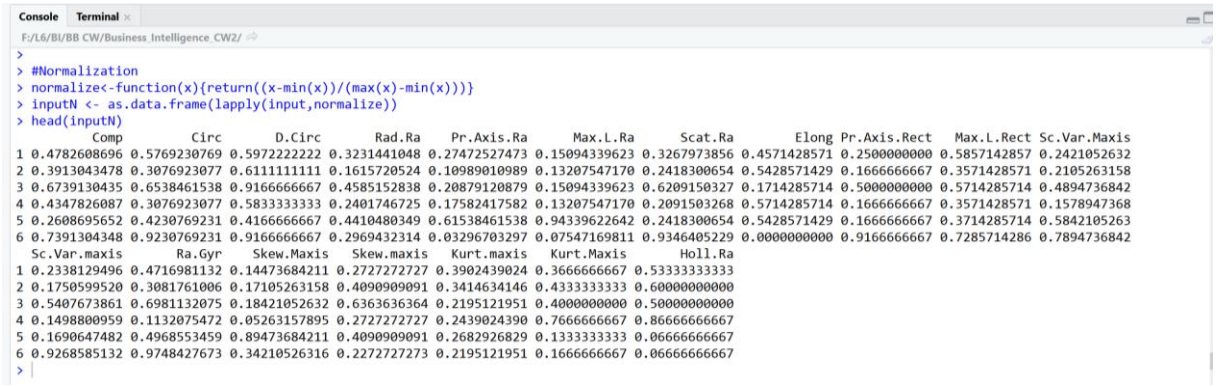
	ï..Samples	Comp	Circ	D.Circ	Rad.Ra	Pr.Axis.Ra	Max.L.Ra	Scat.Ra	Elong	Pr.Axis.Rect	Max.L.Rect	Sc.Var.Maxis	Sc.Var.maxis	Ra.Gyr	Skew.Maxis	Skew.maxis	Kurt.maxis	Kurt.Maxis	Holl.Ra	Class
1	1	95	48	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	van
2	2	91	41	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	van
3	3	104	50	106	209	66	10	207	32	23	158	223	635	220	73	14	9	188	196	saab
4	4	93	41	82	159	63	9	144	46	19	143	160	309	127	63	6	10	199	207	van
5	5	85	44	70	205	103	52	149	45	19	144	241	325	188	127	9	11	180	183	bus
6	6	107	57	106	172	50	6	255	26	28	169	280	957	264	85	5	9	181	183	bus
7	7	97	43	73	173	65	6	153	42	19	143	176	361	172	66	13	1	200	204	bus
8	8	90	43	66	157	65	9	137	48	18	146	162	281	164	67	3	3	193	202	van
9	9	86	34	62	140	61	7	122	54	17	127	141	223	112	64	2	14	200	208	van
10	10	93	44	98	197	62	11	183	36	22	146	202	505	152	64	4	14	195	204	saab
11	11	86	36	70	143	61	9	133	50	18	130	153	266	127	66	2	10	194	202	van
12	12	90	34	66	136	55	6	123	54	17	118	148	224	118	65	5	26	196	202	saab
13	13	88	46	74	171	68	6	152	43	19	148	180	349	192	71	5	11	189	195	bus
14	14	89	42	85	144	58	10	152	44	19	144	173	345	161	72	8	13	187	197	van
15	15	94	49	79	203	71	5	174	37	21	154	196	465	206	71	6	2	197	199	bus
16	16	96	55	103	201	65	9	204	32	23	166	227	624	246	74	6	2	186	194	opel
17	17	89	36	51	109	52	6	118	57	17	129	137	206	125	80	2	14	181	185	van
18	18	99	41	77	197	69	6	177	36	21	139	202	485	151	72	4	10	198	199	bus
19	19	104	54	100	186	61	10	216	31	24	173	225	686	220	74	5	11	185	195	saab
20	20	101	56	100	215	69	10	208	32	24	169	227	651	223	74	6	5	186	193	opel
21	21	84	47	75	153	64	6	154	43	19	145	175	354	184	75	0	3	185	192	bus
22	22	84	37	53	121	59	5	123	55	17	125	141	221	133	82	7	1	179	183	van
23	23	94	43	64	173	69	7	150	43	19	142	169	344	177	68	9	1	199	206	bus
24	24	87	39	70	148	61	7	143	46	18	136	164	307	141	69	1	2	192	199	bus
25	25	99	53	105	219	66	11	204	32	23	165	221	623	224	68	0	6	191	201	saab
26	26	85	45	80	154	64	9	147	45	19	148	169	324	174	71	1	4	188	199	van
27	27	83	36	54	119	57	6	128	53	18	125	143	238	139	82	6	3	179	183	saab
28	28	107	54	98	203	65	11	218	31	25	167	229	696	216	72	1	28	187	199	saab
29	29	102	45	85	193	64	6	192	33	22	146	217	570	163	76	6	7	195	193	bus
30	30	80	38	63	129	55	7	146	46	19	130	168	314	158	83	9	20	180	185	saab
31	31	89	43	85	160	64	11	155	43	19	151	173	356	174	72	5	9	185	196	van
32	32	88	42	77	151	58	8	140	47	18	142	165	293	158	64	10	11	198	205	saab
33	33	93	35	66	154	59	6	142	46	18	128	162	304	120	64	5	13	197	202	opel

3) Pre-processing needs to be carried out by;

   a. Removing the **First** (Contains Sample no.) and **Last** columns (Last column "Class", as this the column that will be checked against. Hence, these columnsneeds to be removed. The 20th Column will be used for consistency checking)



b. Normalizing the Data



4) Save the last column to a new data frame to check consistency at a later stage

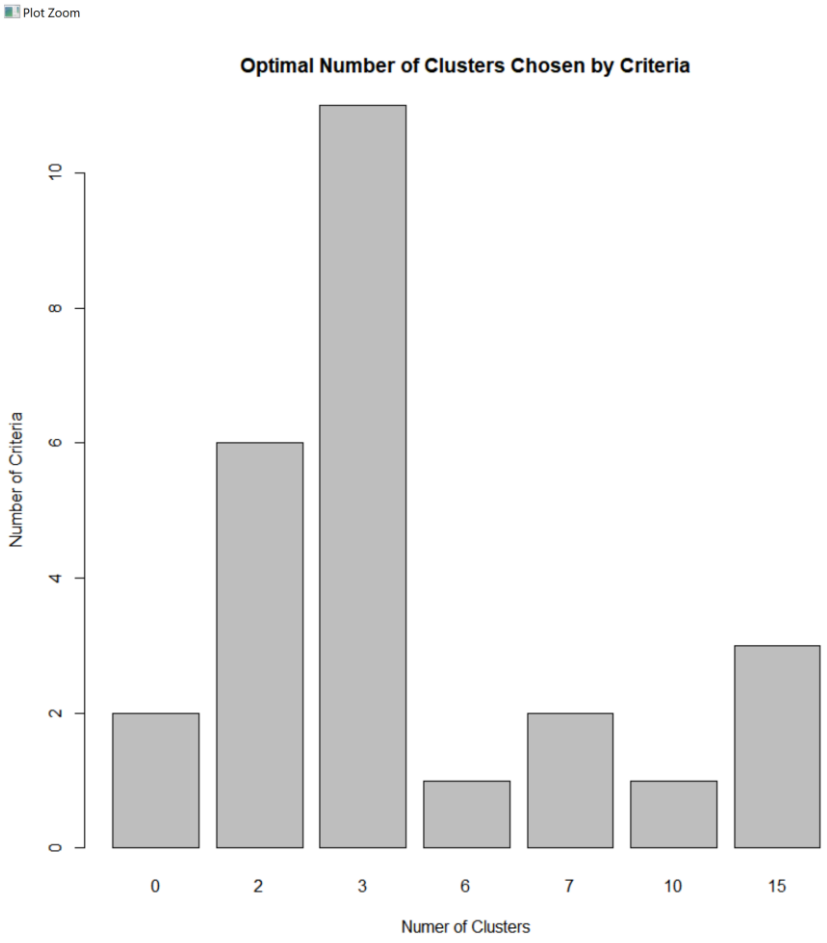
```
#Saving last column
output<-original_vehicles$Class
```

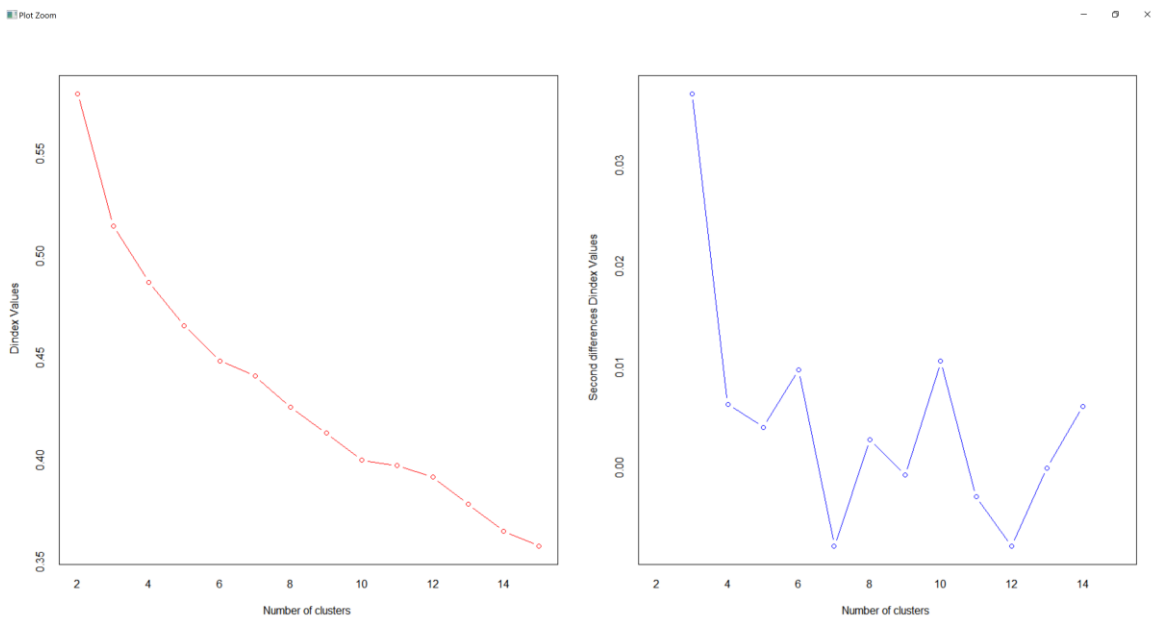
5) Determine the best no. of clusters (K Value)

Two Methods have being used here;

a. Method 1 – Using NB Clust

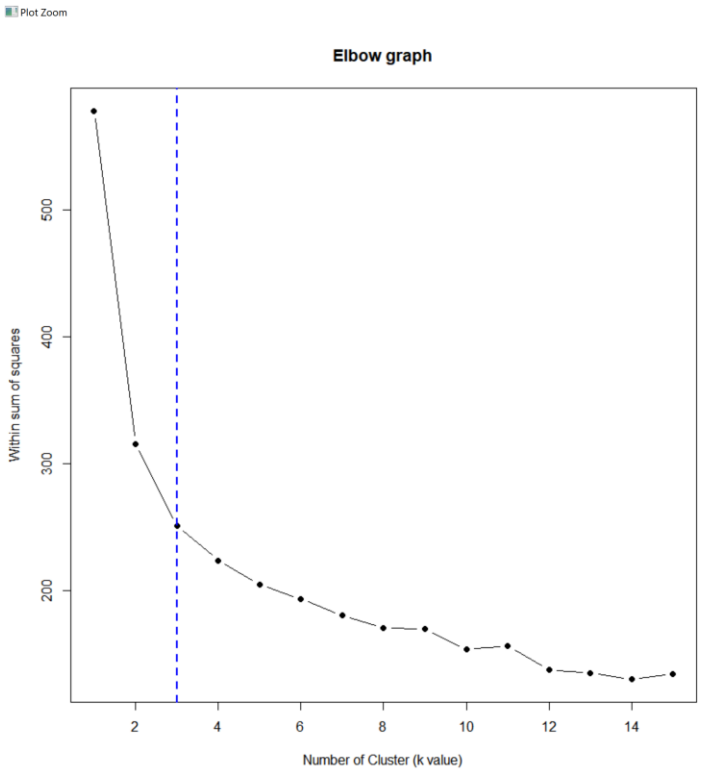
```
>
> #Determine the best no. of clusters k for k means using NbClust (Method 1)
> set.seed(1234)
> clusterNo=NbClust(inputN,distance = "euclidean",
+                   min.nc = 2, max.nc = 15,
+                   method = "kmeans", index = "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
               In the plot of Hubert index, we seek a significant knee that corresponds to a
               significant increase of the value of the measure i.e the significant peak in Hubert
               index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
               In the plot of D index, we seek a significant knee (the significant peak in Dindex
               second differences plot) that corresponds to a significant increase of the value of
               the measure.

*******************************************************************
* Among all indices:
* 6 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 3 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
> #Plotting the no. of clusters in a bar chart
> barplot(table(clusterNo$Best.n[1,]),
+         xlab="Numer of Clusters",
+         ylab="Number of Criteria",
+         main="Optimal Number of Clusters Chosen by Criteria") #we can find the best number of clusters is 3 using the method 1
>
```





b. Method 2 – Using Elbow Method

```
>
> #Plotting the no. of clusters in a bar chart
> barplot(table(clusterNo$Best.n[1,]),
+         xlab="Numer of Clusters",
+         ylab="Number of Criteria",
+         main="Optimal Number of Clusters Chosen by Criteria") #we can find the best number of clusters is 3 using the method 1
>
> #Find the Optimal K using Elbow method (Method 2)
>
> #The Within Sum of Squares and the number of clusters to find the location of a bend or a knee in the plot which is considered as an indicator of th
opriate number of clusters
> set.seed(1234)
> k = 1:15
> wss = sapply(k, function(k) {kmeans(inputN, centers=k)$tot.withinss})
> #lineplot to plot the within sum of squares
> plot(k, wss, type="b", pch = 19,
+      xlab= "Number of Cluster (k value)",
+      ylab="Within sum of squares",
+      main="Elbow graph")
>
> abline(v=3,lty = 2, col="blue", lwd=2)
>
```



6) Result for ideal number of clusters according to the majority rule

Therefore, both two methods suggest **k=3** is the ideal no. of clusters/best no. of clusters.

1.3 K-Means with the best two clusters

```
>
> #Determine the best no. of clusters k for k means using NbClust (Method 1)
> set.seed(1234)
> clusterNo=NbClust(inputN,distance = "euclidean",
+                   min.nc = 2, max.nc = 15,
+                   method = "kmeans", index = "all")
*** : The Hubert index is a graphical method of determining the number of clusters.
               In the plot of Hubert index, we seek a significant knee that corresponds to a
               significant increase of the value of the measure i.e the significant peak in Hubert
               index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
               In the plot of D index, we seek a significant knee (the significant peak in Dindex
               second differences plot) that corresponds to a significant increase of the value of
               the measure.

*******************************************************************
* Among all indices:
* 6 proposed 2 as the best number of clusters
* 11 proposed 3 as the best number of clusters
* 1 proposed 6 as the best number of clusters
* 2 proposed 7 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 3 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  3


*******************************************************************
```

1) According to majority rule,

   11 have proposed 3 as the best no. of cluster. The second-best no. of clusters is 2. Therefore, K – Means has being done k=3 and k=2.

2) K means code for the best 2 chosen clusters

```
#K- means code for best 2 chosen clusters as given in the question

#fit model for k=3
set.seed(1234)
fit.km3<- kmeans(inputN, 3, iter.max=10L, nstart=25)
fit.km3
```

```
#fit model for k=2
set.seed(1234)
fit.km2<- kmeans(inputN, 2, iter.max= 10L, nstart=25)
fit.km2
```

Once executed K-Means for Cluster 3 and K-Means for Cluster 2 can be identified.

K-Means for Cluster 3

```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
>
> #K- means code for best 2 chosen clusters as given in the question
>
> #fit model for k=3
> set.seed(1234)
> fit.km3<- kmeans(inputN, 3, iter.max=10L, nstart=25)
> fit.km3
K-means clustering with 3 clusters of sizes 271, 290, 285

Cluster means:
          Comp         Circ       D.Circ       Rad.Ra   Pr.Axis.Ra     Max.L.Ra      Scat.Ra        Elong Pr.Axis.Rect   Max.L.Rect
1 0.6528156586 0.7348850412 0.8457359574 0.4289466475 0.1752159280 0.1478103460 0.6486988399 0.1593041645 0.5744157442 0.6550869794
2 0.4137181409 0.3103448276 0.5169061303 0.2867640416 0.1831754452 0.1170461939 0.2708586883 0.5003940887 0.1928160920 0.3098029557
3 0.2926773455 0.3396761134 0.4050682261 0.1421282464 0.1263157895 0.1081761006 0.2103199174 0.6059147870 0.1438596491 0.3339849624
  Sc.Var.Maxis Sc.Var.maxis       Ra.Gyr   Skew.Maxis   Skew.maxis   Kurt.maxis   Kurt.Maxis      Holl.Ra
1 0.5095358322 0.5790570496 0.6353361647 0.1734317343 0.3274069104 0.3601836018 0.4318573186 0.5301353014
2 0.2396551724 0.2085090548 0.2799392756 0.1097549909 0.2816614420 0.3247266611 0.6248275862 0.6901149425
3 0.1875530933 0.1480752240 0.3376586119 0.2492151431 0.2625199362 0.2392811297 0.2332163743 0.2415204678

Clustering vector:
  [1] 2 2 1 2 3 1 2 2 2 2 2 2 2 2 1 3 2 1 1 3 3 2 2 1 2 3 1 1 3 3 2 2 1 2 2 3 1 1 3 1 2 2 3 1 3 3 2 3 3
 [67] 3 1 2 1 1 1 2 3 2 1 2 3 1 3 3 1 2 3 2 1 2 3 2 3 1 2 1 3 3 1 3 3 1 3 3 2 2 3 1 1 1 3 3 1 2 3 3 3 2 3 3 3 3 3 3 2 1 1 2
[133] 2 3 1 2 3 2 3 2 2 3 1 3 2 1 2 2 2 2 1 2 2 1 2 1 2 3 2 2 3 1 2 3 1 1 2 1 3 3 1 1 2 1 3 2 2 2 2 3 1 3 2 3 1 2 2 2 1 2 1 2 3 1 2 3 1 3
[199] 3 3 3 2 1 1 2 2 2 3 3 1 2 3 2 1 3 3 3 1 3 2 1 3 1 3 3 2 1 2 1 3 3 3 3 1 2 3 2 3 1 3 2 2 2 3 1 3 3 2 2 1 3 3 1 3 2 3 1 2 2 1 1 3 2 2 2
[265] 1 3 3 2 2 3 3 2 2 2 1 2 3 3 1 2 2 3 3 1 3 2 3 3 1 3 3 3 3 3 2 1 2 1 3 2 2 1 2 2 2 3 3 1 1 1 1 3 3 1 3 3 3 2 3 1 1 3 1 2 3 1 3 2 2
[331] 2 1 1 3 1 1 3 1 2 3 2 3 3 1 1 1 1 2 2 2 1 3 2 3 1 2 2 1 2 1 1 1 2 3 3 1 3 3 3 3 2 2 2 2 3 1 1 3 3 1 3 1 3 1 2 3 2 3 1 1 3 2 2 3 1 3
[397] 1 3 2 2 1 2 1 2 1 2 3 3 3 2 2 3 3 2 3 1 2 2 3 2 3 1 2 3 2 2 1 2 1 2 1 1 3 3 1 2 3 3 1 1 1 3 2 1 1 3 1 1 1 2 2 2 2 2 1 3 3 2 1 2 2 1
[463] 2 3 1 3 3 1 1 2 3 1 1 1 3 1 1 3 2 3 1 1 2 2 3 3 1 2 3 1 1 2 3 1 1 2 1 3 3 1 1 1 3 3 1 1 1 2 2 1 3 2 1 2 3 3 2 1 3 2 2 3 2 2 1 2 1 1
[529] 2 3 2 1 1 3 3 2 1 2 1 1 2 2 2 2 3 3 3 2 2 1 3 3 2 3 1 2 1 3 3 1 1 2 1 2 2 2 1 3 3 2 1 2 2 3 1 1 1 1 2 3 3 3 1 1 1 2 1 3 3 1 3 3 3 2
[595] 3 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 3 1 3 3 2 3 2 2 3 3 1 1 3 2 1 2 1 3 3 1 2 3 1 3 1 3 3 2 3 2 1 1 3 1 2 2 3 2 3 1 2 1 3 2 2 2 3 3 3 2 1
[661] 2 1 3 3 2 3 2 1 2 3 1 2 1 2 2 1 3 1 3 2 3 2 3 1 2 3 2 1 3 1 2 3 1 3 2 3 3 2 3 3 1 1 2 2 1 2 2 3 2 1 1 1 1 2 1 2 2 1 1 2 1 2 1 2 3
[727] 1 2 3 1 1 1 2 1 3 3 1 1 1 2 1 2 2 1 2 3 2 3 2 1 2 3 2 2 2 3 1 3 3 3 1 1 3 1 1 3 2 2 1 2 3 1 1 3 2 2 1 1 1 3 1 2 1 1 3 3 1 3 1 3 3 2
[793] 1 1 2 3 2 1 1 2 2 3 2 2 1 3 3 1 3 3 1 3 2 3 3 3 3 1 1 2 3 1 2 1 1 3 2 1 3 3 2 2 1 3 3 1 3 2 2 2 2 2 2 1 2 3

Within cluster sum of squares by cluster:
[1] 88.63415701 88.74050775 73.56042414
 (between_SS / total_SS =  56.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

K-Means for Cluster 2

```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
>
> #fit model for k=2
> set.seed(1234)
> fit.km2<- kmeans(inputN, 2, iter.max= 10L, nstart=25)
> fit.km2
K-means clustering with 2 clusters of sizes 554, 292

Cluster means:
          Comp         Circ       D.Circ       Rad.Ra   Pr.Axis.Ra     Max.L.Ra      Scat.Ra        Elong Pr.Axis.Rect   Max.L.Rect
1 0.3488855753 0.3184532074 0.4528429603 0.2079201677 0.1533700956 0.1124582794 0.2330525471 0.5624548736 0.1606498195 0.3173543063
2 0.6404854080 0.7175974710 0.8344748858 0.4271400371 0.1768402830 0.1456448695 0.6341659952 0.1690802348 0.5602168950 0.6395303327
  Sc.Var.Maxis Sc.Var.maxis       Ra.Gyr   Skew.Maxis   Skew.maxis   Kurt.maxis   Kurt.Maxis      Holl.Ra
1 0.2080657420 0.1713026691 0.3040324229 0.1814791944 0.2728093206 0.2804878049 0.4230445247 0.4592057762
2 0.4992069214 0.5640131730 0.6204014819 0.1688896900 0.3222291407 0.3581690611 0.4463470320 0.5418949772

Clustering vector:
  [1] 1 1 2 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 1 1 2 2 1 1 1 1 2 1 1 1 2 2 1 2 1 1 2 1 2 1 2 1 2 1 1 1 2 1
 [67] 1 2 1 2 2 2 1 1 1 2 1 1 2 1 1 2 1 1 1 2 1 1 1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 1 2 2 1
[133] 2 1 2 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 2 2 1 2 1 2 1 1 1 1 1 2 1 1 2 2 1 2 1 1 2 2 1 2 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 2 1 1 1 2 1 1 2 1
[199] 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 2 1 2 1 1 1 2 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2 1 1 1 1 2 2 1 1 1 1
[265] 2 1 1 1 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 1 2 2 2 2 2 1 1 2 1 1 1 2 1 2 2 2 1 2 1 1 1 2 1 1 1
[331] 1 2 2 1 2 2 1 2 1 1 1 1 1 2 2 2 2 1 1 1 2 1 1 1 2 1 1 2 1 2 1 2 2 2 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 2 1 2 2 1 1 1 1 2 1
[397] 2 1 1 1 2 1 2 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 2 1 2 2 1 1 2 1 1 1 2 2 1 1 2 2 2 1 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2
[463] 1 1 2 1 1 2 2 1 1 2 2 2 1 2 2 1 2 1 2 1 2 2 1 1 1 1 2 1 1 2 2 1 1 2 2 1 2 1 1 2 2 1 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 2 2
[529] 1 1 2 2 2 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 2 1 1 2 2 1 2 1 1 2 2 1 1 1 2 1 1 1 2 1 1 1 1 1 2 2 2 1 1 1 1 2 2 2 1 2 1 1 1 2 1 1 1
[595] 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 2 1 1 1 2 1 2 1 2 2 2 1 1 1 1 1 2 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1 2
[661] 1 2 1 1 1 1 2 2 1 1 2 1 2 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 2 1 2 1 1 2 1 1 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 2 2 2 1 2 1 1 2 2 1 2 1 2 1 2 1 1
[727] 2 1 1 2 2 2 1 2 1 1 2 2 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 2 1 2 2 1 1 1 1 2 2 2 1 1 2 1 2 1 2 2 1 1 1 1 2 1 2 1 1 1
[793] 2 2 1 1 1 2 2 1 1 1 1 2 2 1 1 2 1 1 2 1 1 1 1 1 1 1 2 2 2 1 2 1 2 2 1 2 2 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 2 1 1

Within cluster sum of squares by cluster:
[1] 214.3676102 101.2180633
 (between_SS / total_SS =  45.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
>
```

Based on K-Means the *Sum of Squares by cluster* can be identified. For K-Means with 3 clusters, when the equation (between_SS/ total_SS) is run, the sum of squares are 56.6% and for K-Means with 2 clusters) is 45.4%.

The results are displayed below.

K=3 – Sum of Squares

```
Within cluster sum of squares by cluster:
[1] 88.63415701 88.74050775 73.56042414
 (between_SS / total_SS =  56.6 %)
```

K=2 – Sum of Squares

```
Within cluster sum of squares by cluster:
[1] 214.3676102 101.2180633
 (between_SS / total_SS =  45.4 %)
```

1.4 Means of Attributes of the Wining Cluster

The mean attributes for the winning cluster which is 3 can then be identified from the sum of squares.

Note: randindex was not calculated as there was a difference. The code for randindex is included in Complete Code(1.1) which gave the answer as k=2.

Wining Test – Means of each Attribute of each group (Normalized)

```
K-means clustering with 3 clusters of sizes 271, 290, 285

Cluster means:
          Comp         Circ       D.Circ       Rad.Ra   Pr.Axis.Ra     Max.L.Ra      Scat.Ra        Elong Pr.Axis.Rect   Max.L.Rect
1 0.6528156586 0.7348850412 0.8457359574 0.4289466475 0.1752159280 0.1478103460 0.6486988399 0.1593041645 0.5744157442 0.6550869794
2 0.4137181409 0.3103448276 0.5169061303 0.2867640416 0.1831754452 0.1170461939 0.2708586883 0.5003940887 0.1928160920 0.3098029557
3 0.2926773455 0.3396761134 0.4050682261 0.1421282464 0.1263157895 0.1081761006 0.2103199174 0.6059147870 0.1438596491 0.3339849624
  Sc.Var.Maxis Sc.Var.maxis       Ra.Gyr   Skew.Maxis   Skew.maxis   Kurt.maxis   Kurt.Maxis      Holl.Ra
1 0.5095358322 0.5790570496 0.6353361647 0.1734317343 0.3274069104 0.3601836018 0.4318573186 0.5301353014
2 0.2396551724 0.2085090548 0.2799392756 0.1097549909 0.2816614420 0.3247266611 0.6248275862 0.6901149425
3 0.1875530933 0.1480752240 0.3376586119 0.2492151431 0.2625199362 0.2392811297 0.2332163743 0.2415204678
```

Wining Test – Means of each Attribute of each group (Actual)

- Actual Cluster Mean of Compactness Attribute

```
#Actual means of each atrributes  in wining cluster k=3

#Actual Mean of Comp
fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)

unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}

comp_min <- min(input[1])
comp_max <- max(input[1])

fit.km3actualmean <- unnormalize(fit.km3means[2], comp_min, comp_max)
fit.km3actualmean
```

```
> #Actual Mean of Comp
> fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)
>
> unnormalize <- function(x, min, max) {
+   return( (max - min)*x + min )
+ }
>
> comp_min <- min(input[1])
> comp_max <- max(input[1])
>
> fit.km3actualmean <- unnormalize(fit.km3means[2], comp_min, comp_max)
> fit.km3actualmean
         Comp
1 103.02952030
2  92.03103448
3  86.46315789
>
```

- Actual Cluster Means of Circularity Attribute

```
#Actual Mean of Circ
fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)

unnormalize <- function(x, min, max) {
  return( (max - min)*x + min )
}

circ_min <- min(input[2])
circ_max <- max(input[2])

fit.km3actualmean <- unnormalize(fit.km3means[3], circ_min, circ_max)
fit.km3actualmean
```

```
> #Actual Mean of Circ
> fit.km3means <- aggregate(inputN, by=list(fit.km3$cluster),FUN = mean)
>
> unnormalize <- function(x, min, max) {
+   return( (max - min)*x + min )
+ }
>
> circ_min <- min(input[2])
> circ_max <- max(input[2])
>
> fit.km3actualmean <- unnormalize(fit.km3means[3], circ_min, circ_max)
> fit.km3actualmean
       Circ
1 52.10701107
2 41.06896552
3 41.83157895
>
```

Note: Actual Cluster means for the rest of the attributes have not being included here, but the above code could be adjusted to calculate the rest.

1.5 Check consistency of your results against Class column

- Saving the Last Column

```
#Saving last column
output<-original_vehicles$Class
```

- Code and Output for Consistency checking of K=3

```
#Code for consistency checking of 'Class' when k=3
confuseTable.km3 <- table(output, fit.km3$cluster)
confuseTable.km3
```

```
> #Code for consistency checking of 'Class' when k=3
> confuseTable.km3 <- table(output, fit.km3$cluster)  #kn
 the class
> confuseTable.km3

output   1   2   3
  bus   50  69  99
  opel 113  58  41
  saab 106  66  45
  van    2  97 100
>
```

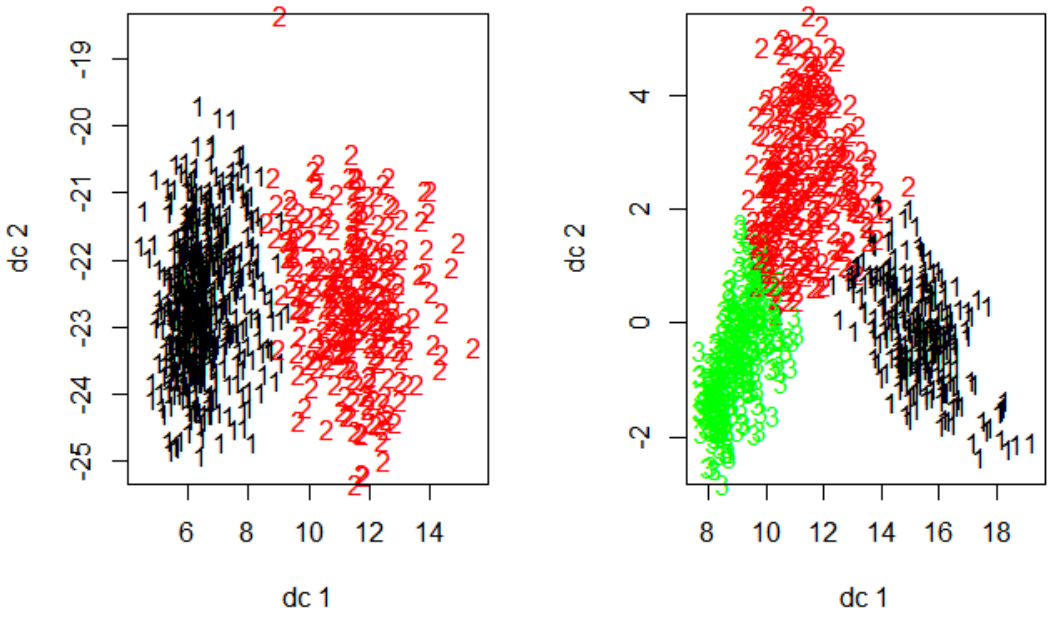
- Code and Output for Consistency checking of K=2

```
#Code for consistency checking of 'Class' when k=2
confuseTable.km2 <- table(output, fit.km2$cluster)
confuseTable.km2
```

```
>
> #Code for consistency checking of 'Class' when k=2
> confuseTable.km2 <- table(output, fit.km2$cluster)
> confuseTable.km2

output   1   2
  bus  161  57
  opel  95 117
  saab 101 116
  van  197   2
>
```

- To Check how the datasets are clustered

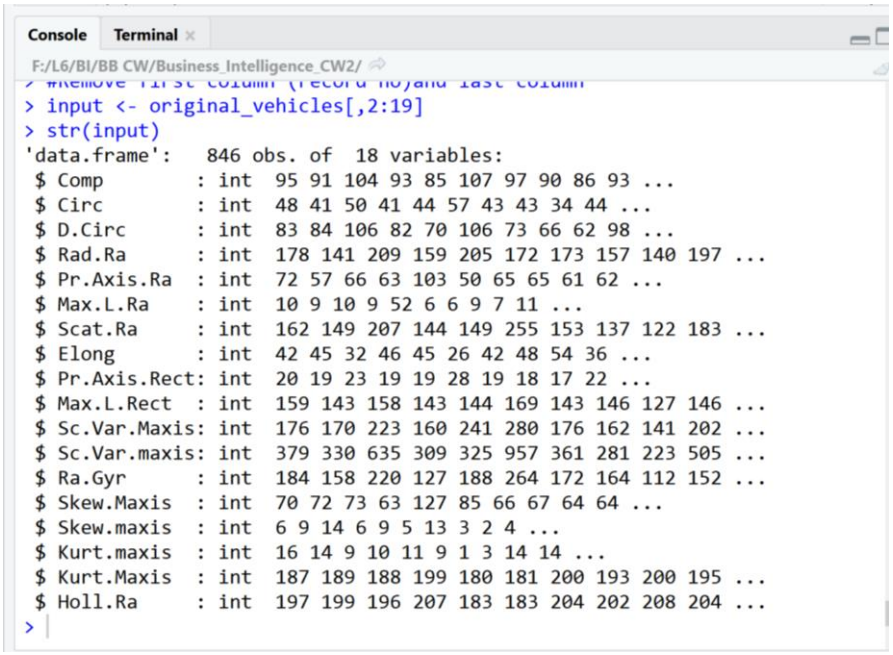
```
#how dataset is clustered
plotcluster(inputN, fit.km2$cluster)
plotcluster(inputN, fit.km3$cluster)
```



1.6 Check for any pre-processing tasks

1) Pre-processing needs to be carried out by;

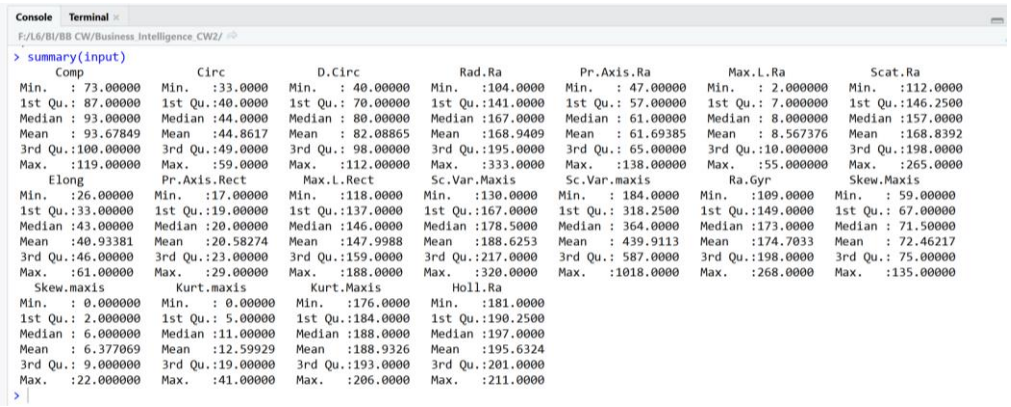
a. Removing the **First** (Contains Sample no.) and **Last** columns (Last column "Class", as this the column that will be checked against. Hence, these columns needs to be removed. The 20th Column will be used for consistency checking)



```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
> input <- original_vehicles[,2:19]
> str(input)
'data.frame':   846 obs. of  18 variables:
 $ Comp         : int  95 91 104 93 85 107 97 90 86 93 ...
 $ Circ         : int  48 41 50 41 44 57 43 43 34 44 ...
 $ D.Circ       : int  83 84 106 82 70 106 73 66 62 98 ...
 $ Rad.Ra       : int  178 141 209 159 205 172 173 157 140 197 ...
 $ Pr.Axis.Ra   : int  72 57 66 63 103 50 65 65 61 62 ...
 $ Max.L.Ra     : int  10 9 10 9 52 6 6 9 7 11 ...
 $ Scat.Ra      : int  162 149 207 144 149 255 153 137 122 183 ...
 $ Elong        : int  42 45 32 46 45 26 42 48 54 36 ...
 $ Pr.Axis.Rect: int  20 19 23 19 19 28 19 18 17 22 ...
 $ Max.L.Rect   : int  159 143 158 143 144 169 143 146 127 146 ...
 $ Sc.Var.Maxis: int  176 170 223 160 241 280 176 162 141 202 ...
 $ Sc.Var.maxis: int  379 330 635 309 325 957 361 281 223 505 ...
 $ Ra.Gyr       : int  184 158 220 127 188 264 172 164 112 152 ...
 $ Skew.Maxis   : int  70 72 73 63 127 85 66 67 64 64 ...
 $ Skew.maxis   : int  6 9 14 6 9 5 13 3 2 4 ...
 $ Kurt.maxis   : int  16 14 9 10 11 9 1 3 14 14 ...
 $ Kurt.Maxis   : int  187 189 188 199 180 181 200 193 200 195 ...
 $ Holl.Ra      : int  197 199 196 207 183 183 204 202 208 204 ...
>
```

b. Check Summary of Data to check whether Normalization or Scaling is ideal

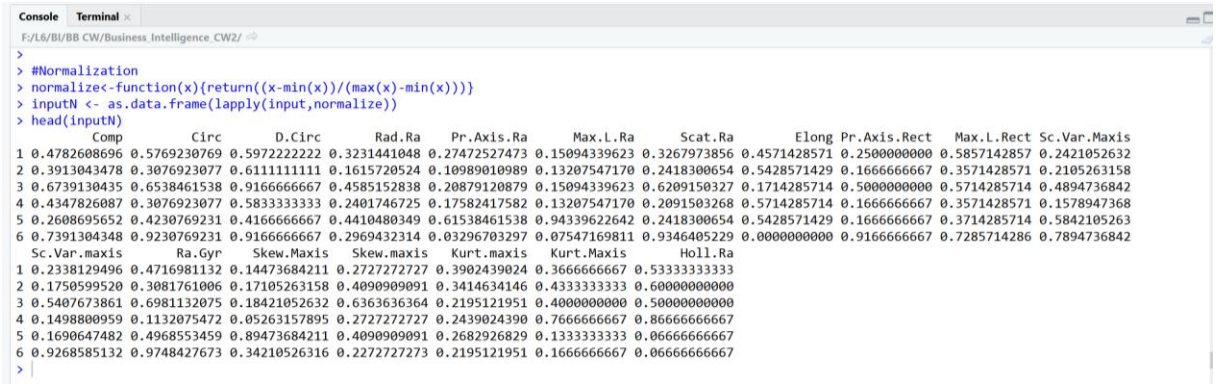
```
summary(input)
```



```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
> summary(input)
      Comp            Circ           D.Circ          Rad.Ra         Pr.Axis.Ra      Max.L.Ra        Scat.Ra
 Min.   : 73.00000   Min.   :33.0000   Min.   : 40.00000   Min.   :104.0000   Min.   : 47.00000   Min.   : 2.000000   Min.   :112.0000
 1st Qu.: 87.00000   1st Qu.:40.0000   1st Qu.: 70.00000   1st Qu.:141.0000   1st Qu.: 57.00000   1st Qu.: 7.000000   1st Qu.:146.2500
 Median : 93.00000   Median :44.0000   Median : 80.00000   Median :167.0000   Median : 61.00000   Median : 8.000000   Median :157.0000
 Mean   : 93.67849   Mean   :44.8617   Mean   : 82.08865   Mean   :168.9409   Mean   : 61.69385   Mean   : 8.567376   Mean   :168.8392
 3rd Qu.:100.00000   3rd Qu.:49.0000   3rd Qu.: 98.00000   3rd Qu.:195.0000   3rd Qu.: 65.00000   3rd Qu.:10.000000   3rd Qu.:198.0000
 Max.   :119.00000   Max.   :59.0000   Max.   :112.00000   Max.   :333.0000   Max.   :138.00000   Max.   :55.000000   Max.   :265.0000
     Elong          Pr.Axis.Rect     Max.L.Rect      Sc.Var.Maxis     Sc.Var.maxis       Ra.Gyr          Skew.Maxis
 Min.   :26.00000   Min.   :17.00000   Min.   :118.0000   Min.   :130.0000   Min.   : 184.0000   Min.   :109.0000   Min.   : 59.00000
 1st Qu.:33.00000   1st Qu.:19.00000   1st Qu.:137.0000   1st Qu.:167.0000   1st Qu.: 318.2500   1st Qu.:149.0000   1st Qu.: 67.00000
 Median :43.00000   Median :20.00000   Median :146.0000   Median :178.5000   Median : 364.0000   Median :173.0000   Median : 71.50000
 Mean   :40.93381   Mean   :20.58274   Mean   :147.9988   Mean   :188.6253   Mean   : 439.9113   Mean   :174.7033   Mean   : 72.46217
 3rd Qu.:46.00000   3rd Qu.:23.00000   3rd Qu.:159.0000   3rd Qu.:217.0000   3rd Qu.: 587.0000   3rd Qu.:198.0000   3rd Qu.: 75.00000
 Max.   :61.00000   Max.   :29.00000   Max.   :188.0000   Max.   :320.0000   Max.   :1018.0000   Max.   :268.0000   Max.   :135.00000
   Skew.maxis       Kurt.maxis       Kurt.Maxis       Holl.Ra
 Min.   : 0.000000   Min.   : 0.00000   Min.   :176.0000   Min.   :181.0000
 1st Qu.: 2.000000   1st Qu.: 5.00000   1st Qu.:184.0000   1st Qu.:190.2500
 Median : 6.000000   Median :11.00000   Median :188.0000   Median :197.0000
 Mean   : 6.377069   Mean   :12.59929   Mean   :188.9326   Mean   :195.6324
 3rd Qu.: 9.000000   3rd Qu.:19.00000   3rd Qu.:193.0000   3rd Qu.:201.0000
 Max.   :22.000000   Max.   :41.00000   Max.   :206.0000   Max.   :211.0000
>
```

It is noticed that the data is not uniformly distributed and there are wide ranges between minimum and maximum values. Therefore, Normalization is carried out as a pre-processing task.

c. Normalizing the Data



```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
>
> #Normalization
> normalize<-function(x){return((x-min(x))/(max(x)-min(x)))}
> inputN <- as.data.frame(lapply(input,normalize))
> head(inputN)
       Comp       Circ     D.Circ     Rad.Ra  Pr.Axis.Ra    Max.L.Ra    Scat.Ra      Elong Pr.Axis.Rect Max.L.Rect Sc.Var.Maxis
1 0.4782608696 0.5769230769 0.5972222222 0.3231441048 0.27472527473 0.15094339623 0.3267973856 0.4571428571 0.2500000000 0.5857142857 0.2421052632
2 0.3913043478 0.3076923077 0.6111111111 0.1615720524 0.10989010989 0.13207547170 0.2418300654 0.5428571429 0.1666666667 0.3571428571 0.2105263158
3 0.6739130435 0.6538461538 0.9166666667 0.4585152838 0.20879120879 0.15094339623 0.6209150327 0.1714285714 0.5000000000 0.5714285714 0.4894736842
4 0.4347826087 0.3076923077 0.5833333333 0.2401746725 0.17582417582 0.13207547170 0.2091503268 0.5714285714 0.1666666667 0.3571428571 0.1578947368
5 0.2608695652 0.4230769231 0.4166666667 0.4410480349 0.61538461538 0.94339622642 0.2418300654 0.5428571429 0.1666666667 0.3714285714 0.5842105263
6 0.7391304348 0.9230769231 0.9166666667 0.2969432314 0.03296703297 0.07547169811 0.9346405229 0.0000000000 0.9166666667 0.7285714286 0.7894736842
  Sc.Var.maxis       Ra.Gyr    Skew.Maxis   Skew.maxis   Kurt.maxis   Kurt.Maxis       Holl.Ra
1 0.2338129496 0.4716981132 0.14473684211 0.2727272727 0.3902439024 0.3666666667 0.53333333333
2 0.1750599520 0.3081761006 0.17105263158 0.4090909091 0.3414634146 0.4333333333 0.60000000000
3 0.5407673861 0.6981132075 0.18421052632 0.6363636364 0.2195121951 0.4000000000 0.50000000000
4 0.1498800959 0.1132075472 0.05263157895 0.2727272727 0.2439024390 0.7666666667 0.86666666667
5 0.1690647482 0.4968553459 0.89473684211 0.4090909091 0.2682926829 0.1333333333 0.06666666667
6 0.9268585132 0.9748427673 0.34210526316 0.2272727273 0.2195121951 0.1666666667 0.06666666667
>
```

2. Objective 2 – MLP

2.1. Selection of Variables

The financial market is a nonlinear dynamic system which is constantly developing and evolving. Thus, determining the behaviour of this global system is a complex task due to vast number of variables and uncertainties. The Exchange currency market which is part of this global financial market is paramount to be predicted for day to day activities such as when making decisions for importing/exporting. (Johansson and Nafar, 2017) In order to avoid exchange risk, either Fundamental Analysis or Technical analysis needs to be carried out. In this scenario, a technical analysis will be conducted as the given values are statistics received from trading activity.

In the given question, the daily ***Exchange Rates*** of GBP/EUR along with the ***Date*** has being listed down from January 2010 until December 2011. Hence, previous results will be used to forecast the future exchange rate. This could be termed as Time Forecasting because an act of predicting the future takes place where the past is first understood.

Therefore, to ensure higher accuracy, it is necessary to select and prepare the data for optimal results. When it comes to time forecasting, it's important to ensure that the right inputs are selected (Tran, Muttil and Perera, 2015). The exchange rates at most times changes daily according to, researches previously conducted. According to, Frank et. al.(2001), The standard neural network method of performing time series prediction is to induce the function *f* using any feedforward function approximating neural network architecture, such as, a standard MLP, using a set of N-tupules as inputs and a single output. This method, often called as sliding window method is represented in the following diagram.

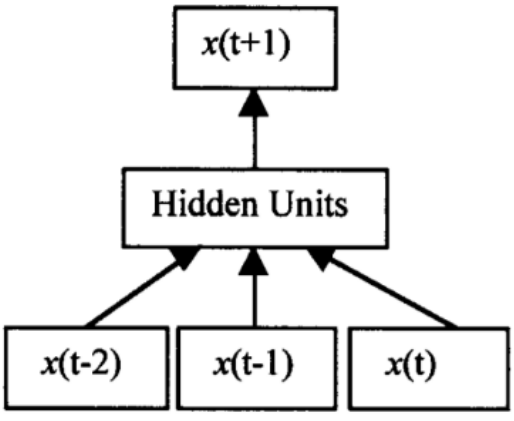


Figure 1 The standard method of performing time series prediction using a sliding window three-time steps

Furthermore, researchers suggest that neural network models outperform ARIMA models. One such research which proved is the forecasting of US monthly live cattle with wheat cash

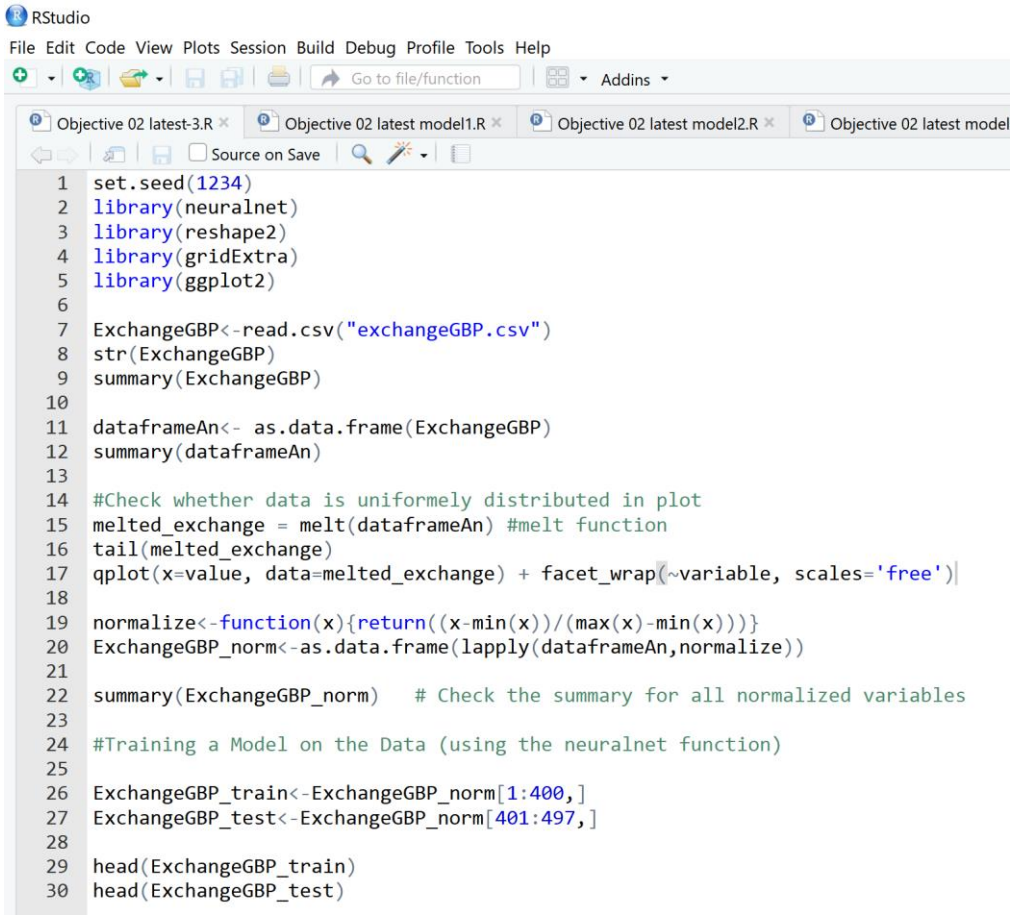
prices. (Nowrouz Kohzadi *et al.*, 1996) This is mainly due to reason such as the dataset being non-linear.

Therefore, as Exchange rates are nonlinear as well, Sliding Window method will be used for the univariate time series analysis. (Jason Brownlee, 2016) Hence, it was decided to use the input attributes as "Day1", Day2", "Day3" with different window sizes (3-time steps initially) to predict the "Day4" exchange rate initially according to the illustrated diagram in Figure1.

Selection of Varying Input Considerations

Inputs	Output (x(t+1)	Model
Day1, Day2, Day3	Day4	Model1, Model2, Model3
Day1, Day2	Day3	Model4, Model5, Model6

2.2. Complete Codes



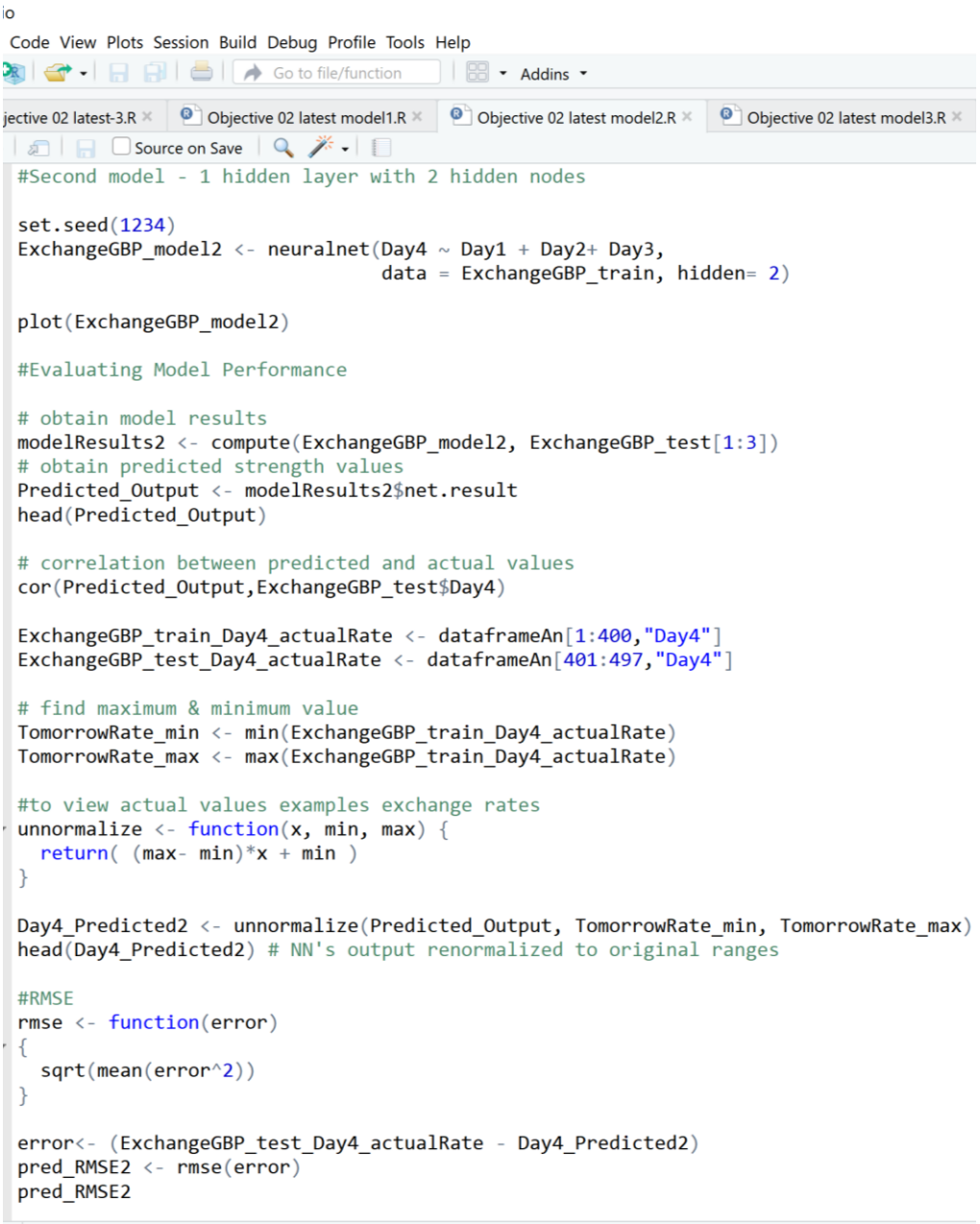
Codes for Models

Model 1

```
#First model - 1 input node for 3 days past data followed by a single hidden node
#and a single output node that predicts the th Day's Exchange Rate

set.seed(1234)
ExchangeGBP_model1 <- neuralnet(Day4 ~ Day1 + Day2+ Day3, data = ExchangeGBP_train)
plot(ExchangeGBP_model1)

#Some bias terms are indicated by nodes labelled with the number 1. They are numeric constants that
#allow the value at the indicated nodes to be shifted upward or downward, much like the intercept in a linear equation.

#Evaluating Model Performance
# obtain model results
modelResults1 <- compute(ExchangeGBP_model1, ExchangeGBP_test[1:3])
# obtain predicted strength values
Predicted_Output <- modelResults1$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day4)

ExchangeGBP_train_Day4_actualRate <- dataframeAn[1:400,"Day4"]
ExchangeGBP_test_Day4_actualRate <- dataframeAn[401:497,"Day4"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day4_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day4_actualRate)
#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day4_Predicted1 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day4_Predicted1) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted1)
pred_RMSE1 <- rmse(error)
pred_RMSE1
```

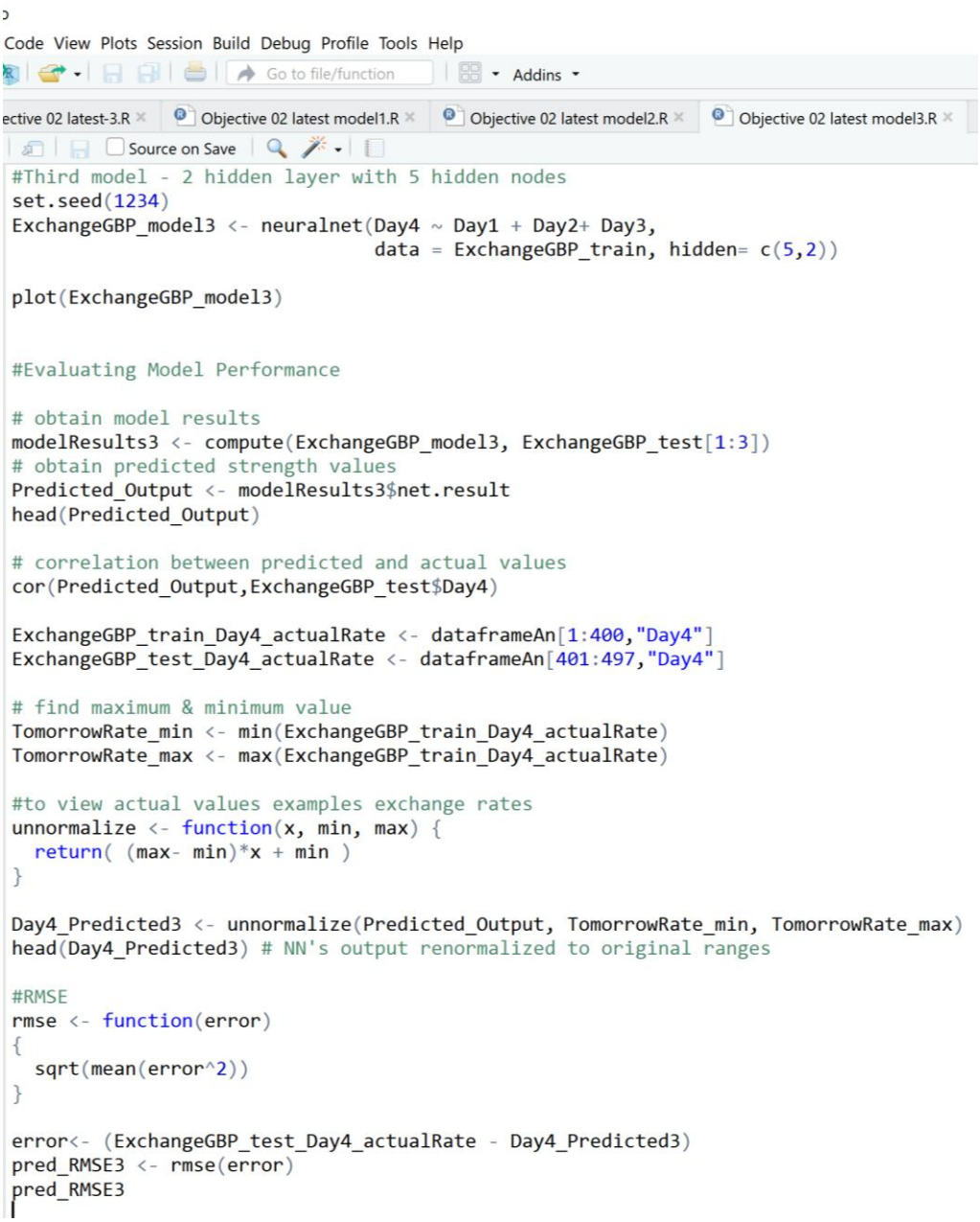
```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result1 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1)
colnames(final_result1) <- c("Expected Output", "Neural Net Output")
final_result1
```

Model 2




```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted2 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result2 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted2)
colnames(final_result2) <- c("Expected Output", "Neural Net Output")
final_result2
```

Model3




```
#Third model - 2 hidden layer with 5 hidden nodes
set.seed(1234)
ExchangeGBP_model3 <- neuralnet(Day4 ~ Day1 + Day2+ Day3,
                                data = ExchangeGBP_train, hidden= c(5,2))

plot(ExchangeGBP_model3)


#Evaluating Model Performance

# obtain model results
modelResults3 <- compute(ExchangeGBP_model3, ExchangeGBP_test[1:3])
# obtain predicted strength values
Predicted_Output <- modelResults3$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day4)

ExchangeGBP_train_Day4_actualRate <- dataframeAn[1:400,"Day4"]
ExchangeGBP_test_Day4_actualRate <- dataframeAn[401:497,"Day4"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day4_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day4_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day4_Predicted3 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day4_Predicted3) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted3)
pred_RMSE3 <- rmse(error)
pred_RMSE3
```

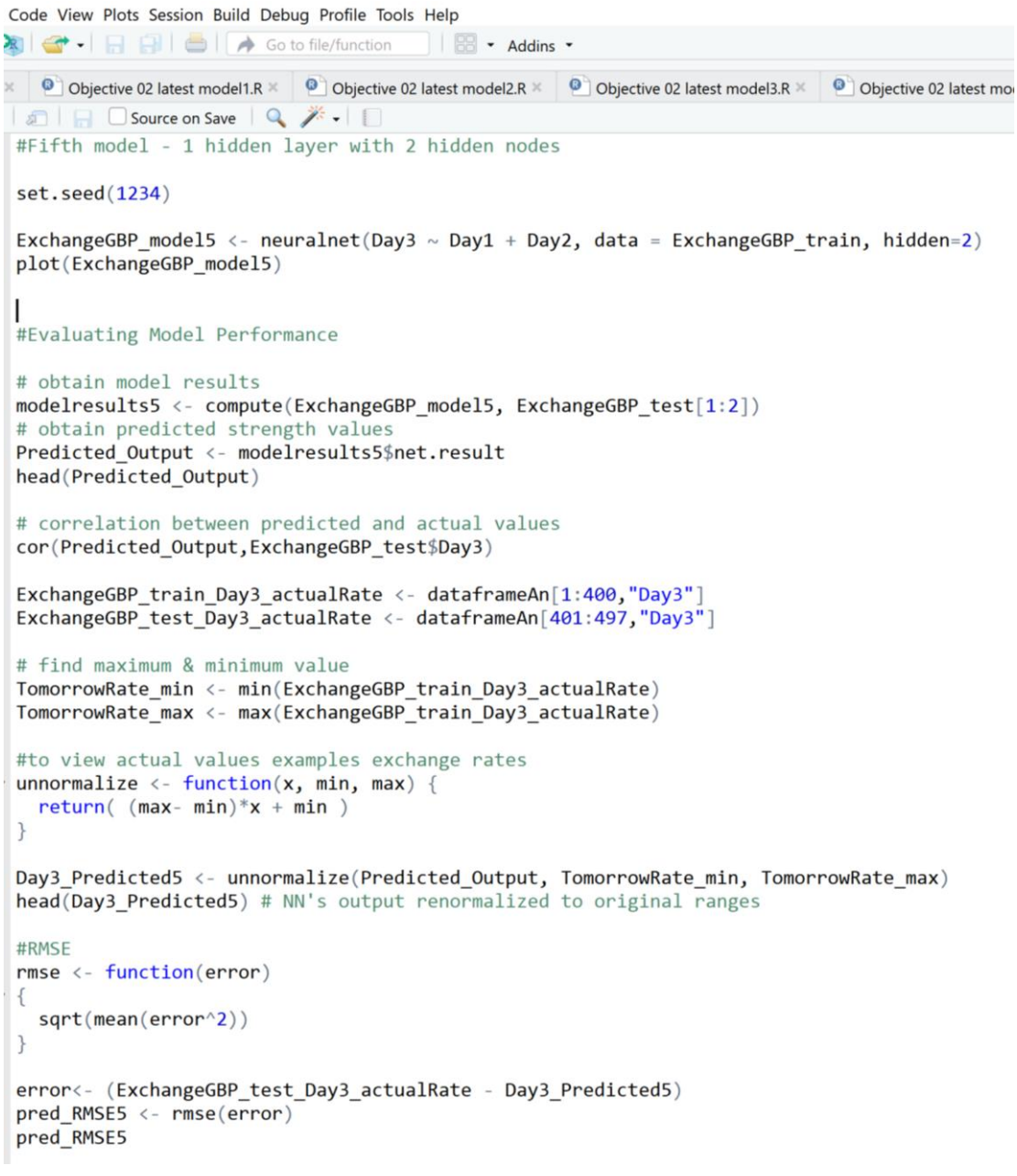
```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted3 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result3 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1)
colnames(final_result3) <- c("Expected Output", "Neural Net Output")
final_result3
```

Model 4

```
#Fourth model - 1 input node for 3 days past data followed by a single hidden node
#and a single output node that predicts the  Exchange Rate
set.seed(1234)
ExchangeGBP_model4 <- neuralnet(Day3 ~ Day1 + Day2, data = ExchangeGBP_train)
plot(ExchangeGBP_model4)

#Some bias terms are indicated by nodes labelled with the number 1. They are numeric constants that
#allow the value at the indicated nodes to be shifted upward or downward, much like the intercept in a linear equation.

#Evaluating Model Performance
# obtain model results
modelResults4 <- compute(ExchangeGBP_model4, ExchangeGBP_test[1:2])
# obtain predicted strength values
Predicted_Output <- modelResults4$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day3)

ExchangeGBP_train_Day3_actualRate <- dataframeAn[1:400,"Day3"]
ExchangeGBP_test_Day3_actualRate <- dataframeAn[401:497,"Day3"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day3_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day3_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day3_Predicted4 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day3_Predicted4) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted4)
pred_RMSE4 <- rmse(error)
pred_RMSE4
```

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted4 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result4 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted4)
colnames(final_result4) <- c("Expected Output", "Neural Net Output")
final_result4
```

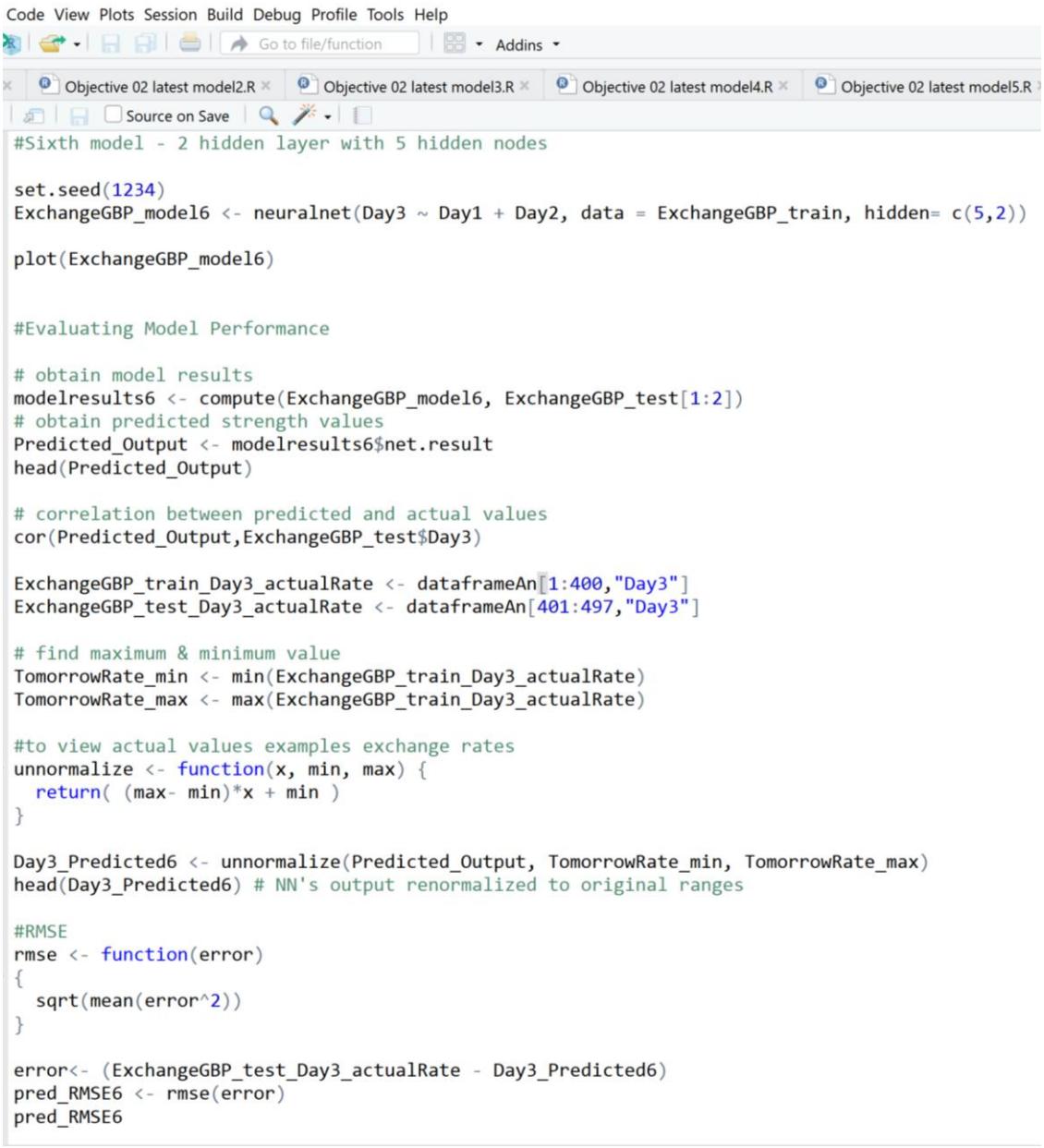
Model 5




```r
#Fifth model - 1 hidden layer with 2 hidden nodes

set.seed(1234)

ExchangeGBP_model5 <- neuralnet(Day3 ~ Day1 + Day2, data = ExchangeGBP_train, hidden=2)
plot(ExchangeGBP_model5)

#Evaluating Model Performance

# obtain model results
modelresults5 <- compute(ExchangeGBP_model5, ExchangeGBP_test[1:2])
# obtain predicted strength values
Predicted_Output <- modelresults5$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day3)

ExchangeGBP_train_Day3_actualRate <- dataframeAn[1:400,"Day3"]
ExchangeGBP_test_Day3_actualRate <- dataframeAn[401:497,"Day3"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day3_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day3_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day3_Predicted5 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day3_Predicted5) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted5)
pred_RMSE5 <- rmse(error)
pred_RMSE5
```

```r
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted5 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result5 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted5)
colnames(final_result5) <- c("Expected Output", "Neural Net Output")
final_result5
```

Model 6



```
#Sixth model - 2 hidden layer with 5 hidden nodes

set.seed(1234)
ExchangeGBP_model6 <- neuralnet(Day3 ~ Day1 + Day2, data = ExchangeGBP_train, hidden= c(5,2))

plot(ExchangeGBP_model6)


#Evaluating Model Performance

# obtain model results
modelresults6 <- compute(ExchangeGBP_model6, ExchangeGBP_test[1:2])
# obtain predicted strength values
Predicted_Output <- modelresults6$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day3)

ExchangeGBP_train_Day3_actualRate <- dataframeAn[1:400,"Day3"]
ExchangeGBP_test_Day3_actualRate <- dataframeAn[401:497,"Day3"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day3_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day3_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day3_Predicted6 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day3_Predicted6) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted6)
pred_RMSE6 <- rmse(error)
pred_RMSE6
```

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted6 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')

#desired test output and the predicted NN output.

final_result6 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted6)
colnames(final_result6) <- c("Expected Output", "Neural Net Output")
final_result6
```

2.3. Data Pre-processing

Since the original data does not fit the original assumptions, data was rearranged by removing the **Date** column and **GBP/EUR** was copied to next column while removing the 1st variable to fill the Day2 Column. Day2 Column was used to fill Day3 Column and Day3 Column was used to fill the Day4 Column.

Once the Data is transformed and columns with null values were removed, there were 497 records in the ExchangeGBP dataset. The Day 4 and Day3 Columns will be used respectively in the 2 types of models to Predict the tomorrow's exchange rate.

RStudio Source Editor

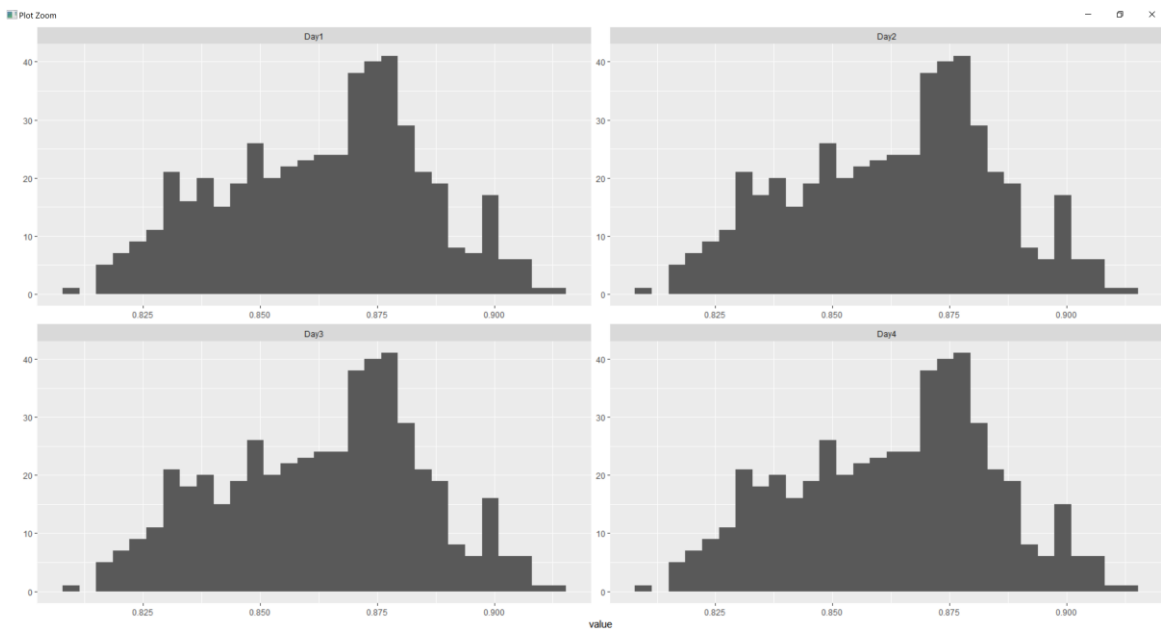
ExchangeGBP

Filter

	Day1	Day2	Day3	Day4
1	0.89513	0.89966	0.89934	0.89963
2	0.89966	0.89934	0.89963	0.89771
3	0.89934	0.89963	0.89771	0.90022
4	0.89963	0.89771	0.90022	0.89754
5	0.89771	0.90022	0.89754	0.88977
6	0.90022	0.89754	0.88977	0.88718
7	0.89754	0.88977	0.88718	0.88536
8	0.88977	0.88718	0.88536	0.88135
9	0.88718	0.88536	0.88135	0.87170
10	0.88536	0.88135	0.87170	0.86539
11	0.88135	0.87170	0.86539	0.87012
12	0.87170	0.86539	0.87012	0.87802
13	0.86539	0.87012	0.87802	0.87130

- Determine between Data Normalization and Scaling

```
dataframeAn<- as.data.frame(ExchangeGBP)
summary(dataframeAn)

#Check whether data is uniformely distributed in plot
melted_exchange = melt(dataframeAn) #melt function
tail(melted_exchange)
qplot(x=value, data=melted_exchange) + facet_wrap(~variable, scales='free')
```



According to the, q plots data is not normally distributed nor uniformly distributed as there is no perfect bell curve to be seen. Hence, the Data was normalized and not scaled.

- Conduct Data Normalization and check if data is normalized

```
normalize<-function(x){return((x-min(x))/(max(x)-min(x)))}
ExchangeGBP_norm<-as.data.frame(lapply(dataframeAn,normalize))

summary(ExchangeGBP_norm)    # Check the summary for all normalized variables
```

```
>
> normalize<-function(x){return((x-min(x))/(max(x)-min(x)))}
> ExchangeGBP_norm<-as.data.frame(lapply(dataframeAn,normalize))
>
> summary(ExchangeGBP_norm)
      Day1              Day2              Day3              Day4
 Min.   :0.0000000   Min.   :0.0000000   Min.   :0.0000000   Min.   :0.0000000
 1st Qu.:0.3777072   1st Qu.:0.3744345   1st Qu.:0.3743382   1st Qu.:0.3727019
 Median :0.5599191   Median :0.5577053   Median :0.5546251   Median :0.5544326
 Mean   :0.5320999   Mean   :0.5309377   Mean   :0.5296852   Mean   :0.5285720
 3rd Qu.:0.6813938   3rd Qu.:0.6805275   3rd Qu.:0.6805275   3rd Qu.:0.6799499
 Max.   :1.0000000   Max.   :1.0000000   Max.   :1.0000000   Max.   :1.0000000
>
```

- Partition the Data for training using NeuraNet function. Question specifies to use the first 400 of them have to be used as training data, while the remaining ones as testing set.

```
#Training a Model on the Data (using the neuralnet function)

ExchangeGBP_train<-ExchangeGBP_norm[1:400,]
ExchangeGBP_test<-ExchangeGBP_norm[401:497,]

head(ExchangeGBP_train)
head(ExchangeGBP_test)
```

```
Console  Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
> head(ExchangeGBP_train)
         Day1         Day2         Day3         Day4
1 0.8382904996 0.8818943113 0.8788141303 0.8816055443
2 0.8818943113 0.8788141303 0.8816055443 0.8631244586
3 0.8788141303 0.8816055443 0.8631244586 0.8872846280
4 0.8816055443 0.8631244586 0.8872846280 0.8614881124
5 0.8631244586 0.8872846280 0.8614881124 0.7866974685
6 0.8872846280 0.8614881124 0.7866974685 0.7617672538
> head(ExchangeGBP_test)
            Day1         Day2         Day3         Day4
401 0.5507748580 0.6038117239 0.6874578882 0.6713831938
402 0.6038117239 0.6874578882 0.6713831938 0.6616613726
403 0.6874578882 0.6713831938 0.6616613726 0.6390412937
404 0.6713831938 0.6616613726 0.6390412937 0.7119068245
405 0.6616613726 0.6390412937 0.7119068245 0.6738858408
406 0.6390412937 0.7119068245 0.6738858408 0.5975551064
>
```

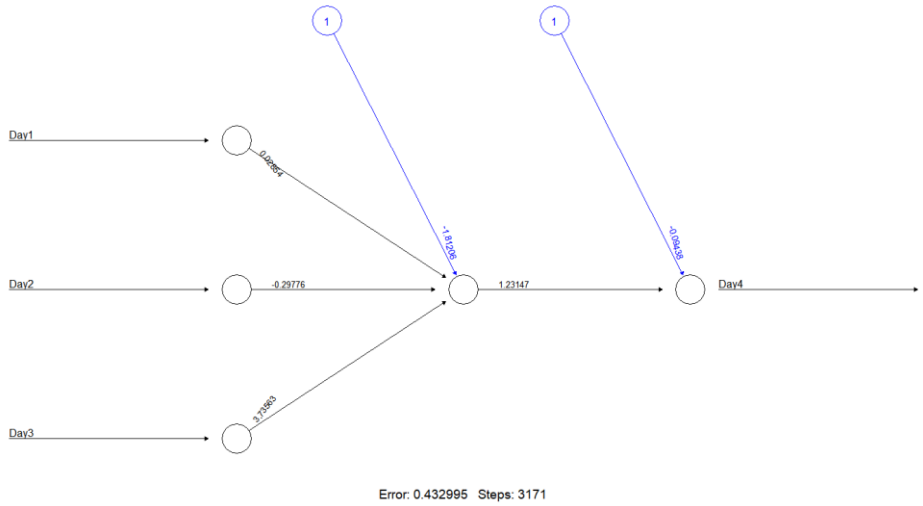
2.4. Experiment with various K values for different input options

The Implementation of MLP's were carried out using various structures (layers/nodes) / input parameters and the results for the code were run.

First Model Results

```r
#First model - 1 input node for 3 days past data followed by a single hidden node
#and a single output node that predicts the th Day's Exchange Rate

set.seed(1234)
ExchangeGBP_model1 <- neuralnet(Day4 ~ Day1 + Day2+ Day3, data = ExchangeGBP_train)
plot(ExchangeGBP_model1)

#Some bias terms are indicated by nodes labelled with the number 1. They are numeric constants that
#allow the value at the indicated nodes to be shifted upward or downward, much like the intercept in a linear equation.

#Evaluating Model Performance
# obtain model results
modelResults1 <- compute(ExchangeGBP_model1, ExchangeGBP_test[1:3])
# obtain predicted strength values
Predicted_Output <- modelResults1$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day4)

ExchangeGBP_train_Day4_actualRate <- dataframeAn[1:400,"Day4"]
ExchangeGBP_test_Day4_actualRate <- dataframeAn[401:497,"Day4"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day4_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day4_actualRate)
#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day4_Predicted1 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day4_Predicted1) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted1)
pred_RMSE1 <- rmse(error)
pred_RMSE1
```

ExchangeGBP Neural Network Model

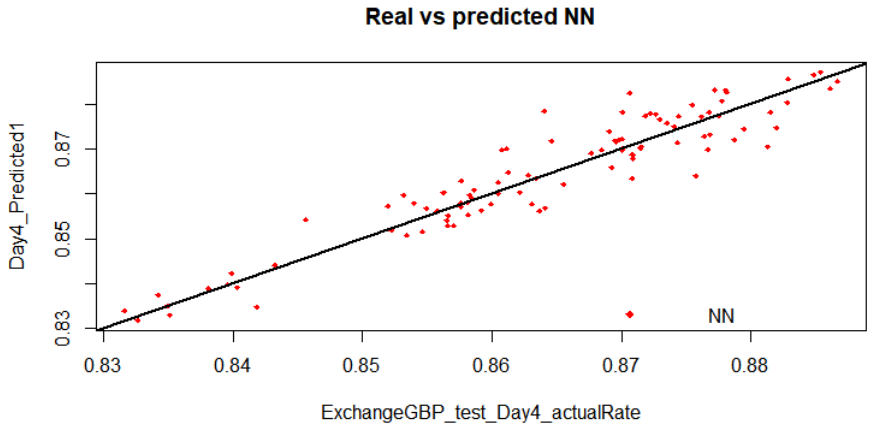


Model1 RMSE

```
>
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted1)
> pred_RMSE1 <- rmse(error)
> pred_RMSE1
[1] 0.004554154516
>
```

Performance of Network

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```



Desired Test Output and Predicted NN output

Console Terminal

F:/L6/BI/BB CW/Business_Intelligence_CW2/

```
> #desired test output and the predicted NN output.
>
> final_result1 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1)
> colnames(final_result1) <- c("Expected Output", "Neural Net Output")
> final_result1
    Expected Output Neural Net Output
401         0.87779      0.8806029912
402         0.87678      0.8781266430
403         0.87443      0.8772491340
404         0.88200      0.8747468113
405         0.87805      0.8830367714
406         0.87012      0.8782188443
407         0.86848      0.8698385231
408         0.87087      0.8686591647
409         0.87432      0.8714454632
410         0.87409      0.8750671395
411         0.87947      0.8745290516
412         0.88285      0.8804071930
413         0.88613      0.8834753991
414         0.88488      0.8865380493
415         0.88671      0.8850620591
416         0.88541      0.8869979629
417         0.88287      0.8855613340
418         0.87722      0.8831093742
419         0.87753      0.8773365961
420         0.88154      0.8781384909
421         0.87065      0.8823103027
422         0.86115      0.8700851755
423         0.86218      0.8602338272
424         0.86555      0.8621929952
425         0.86926      0.8658868000
426         0.87672      0.8698214712
427         0.87225      0.8778059813
428         0.87005      0.8722508939
429         0.87151      0.8702372555
430         0.87874      0.8720427115
431         0.87549      0.8798168416
432         0.87354      0.8756902841
433         0.86908      0.8738820074
434         0.86770      0.8690093981
435         0.87088      0.8678253272
436         0.86961      0.8715177775
437         0.86076      0.8697844136
```

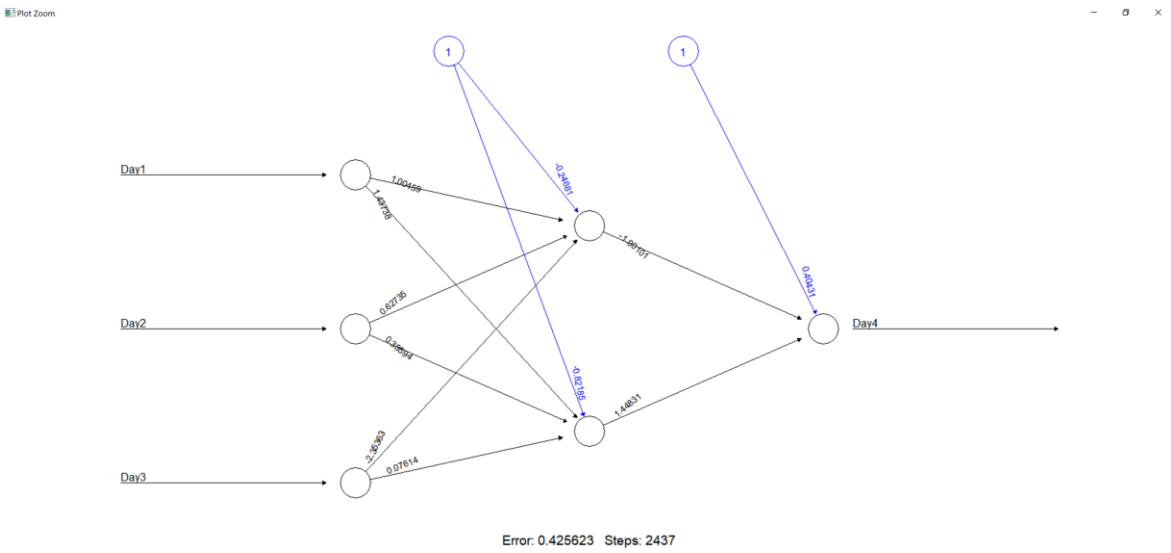
Second Model Results

atest-3.R | Objective 02 latest model1.R | Objective 02 latest model2.R* | Objective 02 latest model3.R | Objective 02 late

Source on Save

```
#Second model - 1 hidden layer with 2 hidden nodes

set.seed(1234)
ExchangeGBP_model2 <- neuralnet(Day4 ~ Day1 + Day2+ Day3,
                                data = ExchangeGBP_train, hidden= 2)

plot(ExchangeGBP_model2)

#Evaluating Model Performance

# obtain model results
modelResults2 <- compute(ExchangeGBP_model2, ExchangeGBP_test[1:3])
# obtain predicted strength values
Predicted_Output <- modelResults2$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day4)

ExchangeGBP_train_Day4_actualRate <- dataframeAn[1:400,"Day4"]
ExchangeGBP_test_Day4_actualRate <- dataframeAn[401:497,"Day4"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day4_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day4_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day4_Predicted2 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day4_Predicted2) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted2)
pred_RMSE2 <- rmse(error)
pred_RMSE2
```

ExchangeGBP Neural Network Model

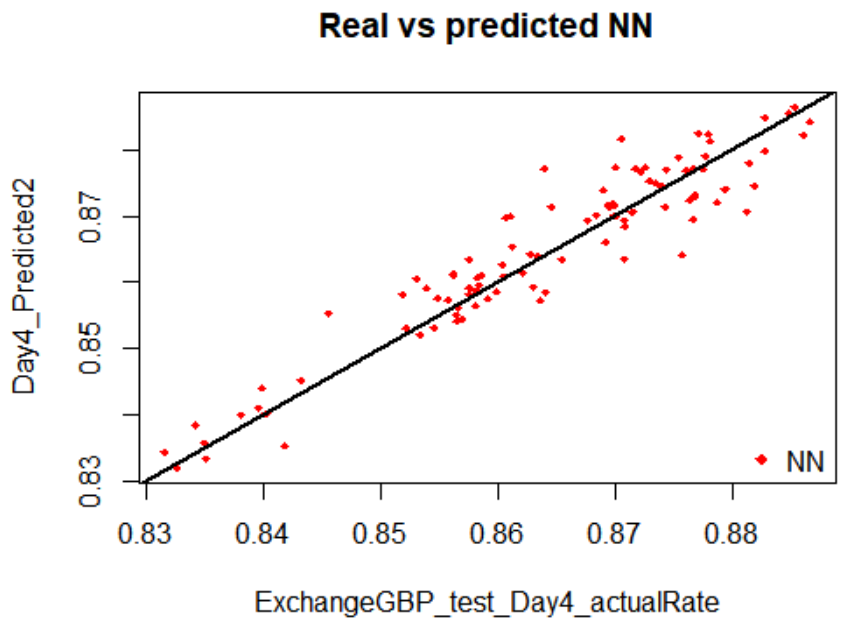


Model2 RMSE

```
>
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted2)
> pred_RMSE2 <- rmse(error)
> pred_RMSE2
[1] 0.004397518917
>
```

Performance of Network

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted2 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```



Desired Test Output and Predicted NN output

Console Terminal

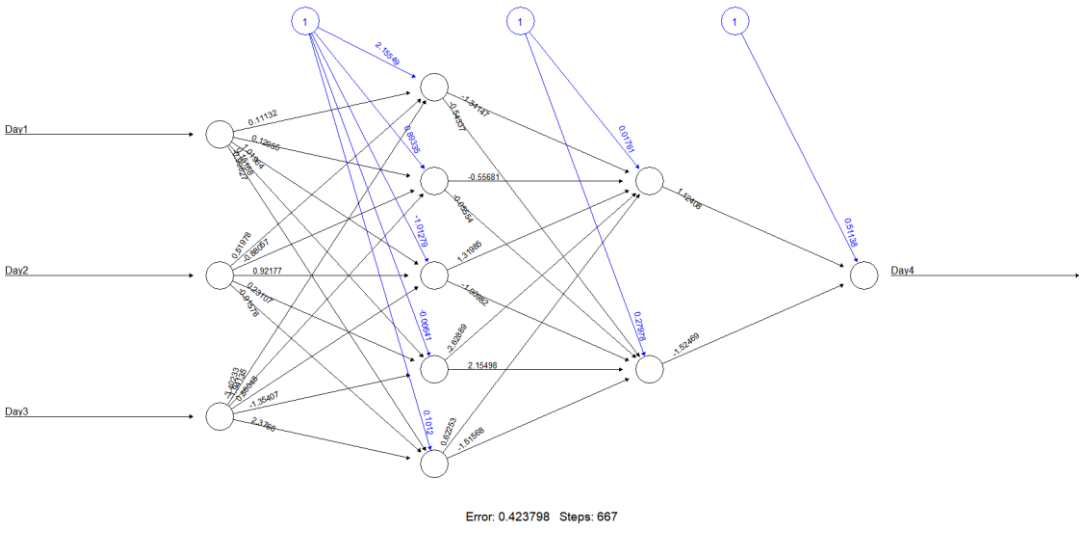
F:/L6/BI/BB CW/Business_Intelligence_CW2/

```
> #desired test output and the predicted NN output.
>
> final_result2 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted2)
> colnames(final_result2) <- c("Expected Output", "Neural Net Output")
> final_result2
    Expected Output Neural Net Output
401         0.87779      0.8790572389
402         0.87678      0.8770762799
403         0.87443      0.8769749016
404         0.88200      0.8745876297
405         0.87805      0.8823520074
406         0.87012      0.8773400313
407         0.86848      0.8700960049
408         0.87087      0.8693021225
409         0.87432      0.8714436321
410         0.87409      0.8744597158
411         0.87947      0.8740285816
412         0.88285      0.8797033089
413         0.88613      0.8823197278
414         0.88488      0.8855894333
415         0.88671      0.8842554081
416         0.88541      0.8864420132
417         0.88287      0.8848450903
418         0.87722      0.8825844039
419         0.87753      0.8770616473
420         0.88154      0.8780365465
421         0.87065      0.8815822230
422         0.86115      0.8699466063
423         0.86218      0.8614113094
424         0.86555      0.8634074109
425         0.86926      0.8660126117
426         0.87672      0.8694377805
427         0.87225      0.8766654150
428         0.87005      0.8717214818
429         0.87151      0.8705810447
430         0.87874      0.8720713974
431         0.87549      0.8788562535
432         0.87354      0.8749370595
433         0.86908      0.8739036693
434         0.86770      0.8693042998
435         0.87088      0.8683447279
436         0.86961      0.8714617053
437         0.86076      0.8696441044
```

Third Model Results

```
#Third model - 2 hidden layer with 5 hidden nodes
set.seed(1234)
ExchangeGBP_model3 <- neuralnet(Day4 ~ Day1 + Day2+ Day3,
                                data = ExchangeGBP_train, hidden= c(5,2))

plot(ExchangeGBP_model3)


#Evaluating Model Performance

# obtain model results
modelResults3 <- compute(ExchangeGBP_model3, ExchangeGBP_test[1:3])
# obtain predicted strength values
Predicted_Output <- modelResults3$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day4)

ExchangeGBP_train_Day4_actualRate <- dataframeAn[1:400,"Day4"]
ExchangeGBP_test_Day4_actualRate <- dataframeAn[401:497,"Day4"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day4_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day4_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day4_Predicted3 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day4_Predicted3) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted3)
pred_RMSE3 <- rmse(error)
pred_RMSE3
```

ExchangeGBP Neural Network Model

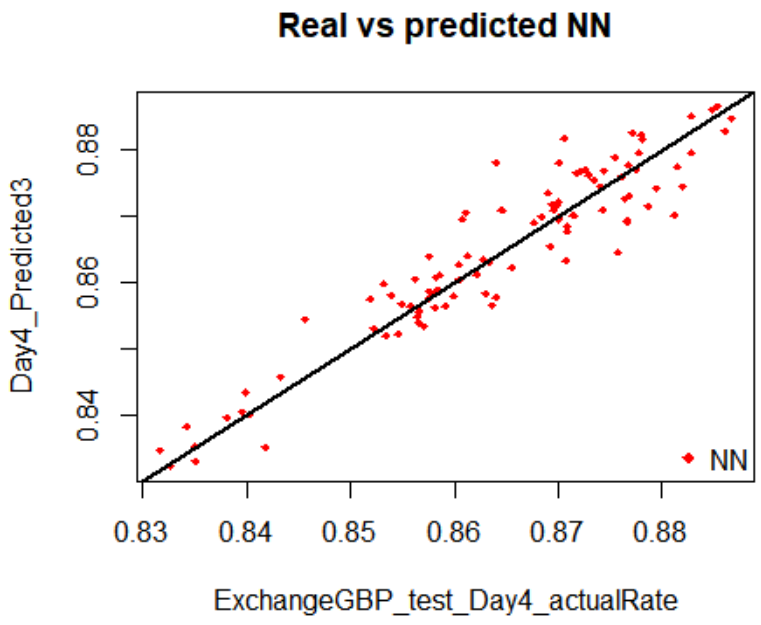


Model3 RMSE

```
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day4_actualRate - Day4_Predicted3)
> pred_RMSE3 <- rmse(error)
> pred_RMSE3
[1] 0.004467412655
>
```

Performance of Network

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day4_actualRate, Day4_Predicted3 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```



Desired Test Output and Predicted NN output

Source

Console Terminal

F:/L6/BI/BB CW/Business_Intelligence_CW2/

```
>
> #desired test output and the predicted NN output.
>
> final_result3 <- cbind(ExchangeGBP_test_Day4_actualRate, Day4_Predicted1)
> colnames(final_result3) <- c("Expected Output", "Neural Net Output")
> final_result3
    Expected Output Neural Net Output
401         0.87779      0.8806029912
402         0.87678      0.8781266430
403         0.87443      0.8772491340
404         0.88200      0.8747468113
405         0.87805      0.8830367714
406         0.87012      0.8782188443
407         0.86848      0.8698385231
408         0.87087      0.8686591647
409         0.87432      0.8714454632
410         0.87409      0.8750671395
411         0.87947      0.8745290516
412         0.88285      0.8804071930
413         0.88613      0.8834753991
414         0.88488      0.8865380493
415         0.88671      0.8850620591
416         0.88541      0.8869979629
417         0.88287      0.8855613340
418         0.87722      0.8831093742
419         0.87753      0.8773365961
420         0.88154      0.8781384909
421         0.87065      0.8823103027
422         0.86115      0.8700851755
423         0.86218      0.8602338272
424         0.86555      0.8621929952
425         0.86926      0.8658868000
426         0.87672      0.8698214712
427         0.87225      0.8778059813
428         0.87005      0.8722508939
429         0.87151      0.8702372555
430         0.87874      0.8720427115
431         0.87549      0.8798168416
432         0.87354      0.8756902841
433         0.86908      0.8738820074
434         0.86770      0.8690093981
435         0.87088      0.8678253272
436         0.86961      0.8715177775
437         0.86076      0.8697844136
```

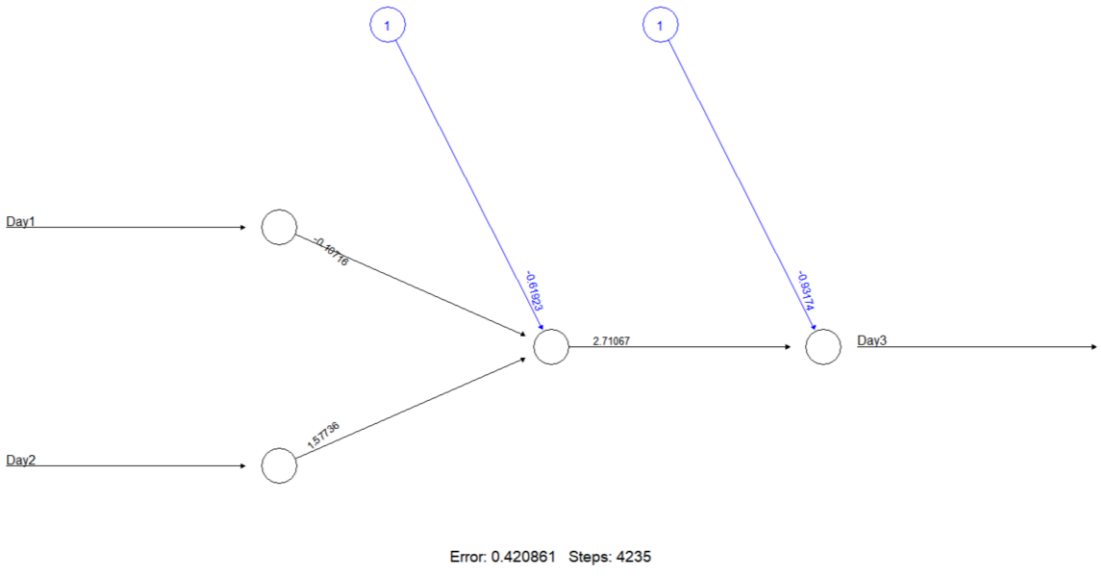
Fourth Model Results

Objective 02 latest model1.R | Objective 02 latest model2.R | Objective 02 latest model3.R | Objective 02 latest model4.R | Objective 02 latest model5.R

Source on Save | Run

```
#Fourth model - 1 input node for 3 days past data followed by a single hidden node
#and a single output node that predicts the  Exchange Rate
set.seed(1234)
ExchangeGBP_model4 <- neuralnet(Day3 ~ Day1 + Day2, data = ExchangeGBP_train)
plot(ExchangeGBP_model4)

#Some bias terms are indicated by nodes labelled with the number 1. They are numeric constants that
#allow the value at the indicated nodes to be shifted upward or downward, much like the intercept in a linear equation.

#Evaluating Model Performance
# obtain model results
modelResults4 <- compute(ExchangeGBP_model4, ExchangeGBP_test[1:2])
# obtain predicted strength values
Predicted_Output <- modelResults4$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day3)

ExchangeGBP_train_Day3_actualRate <- dataframeAn[1:400,"Day3"]
ExchangeGBP_test_Day3_actualRate <- dataframeAn[401:497,"Day3"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day3_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day3_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day3_Predicted4 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day3_Predicted4) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted4)
pred_RMSE4 <- rmse(error)
pred_RMSE4
```

ExchangeGBP Neural Network Model

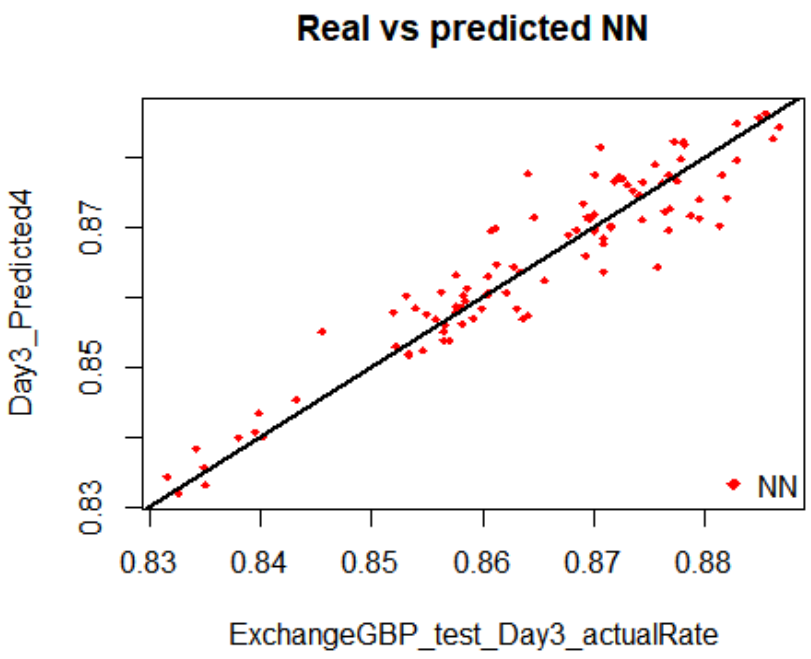


Model4 RMSE

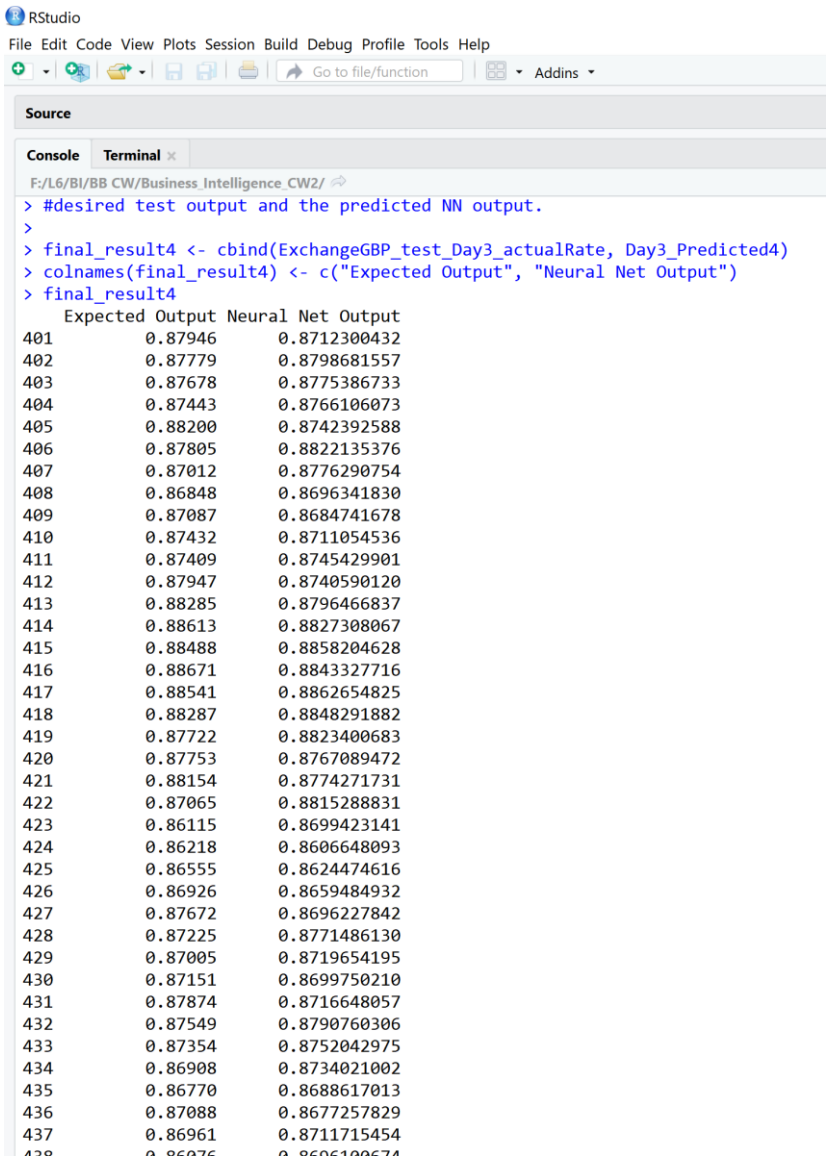
```
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted4)
> pred_RMSE4 <- rmse(error)
> pred_RMSE4
[1] 0.004481170106
```

Performance of Network

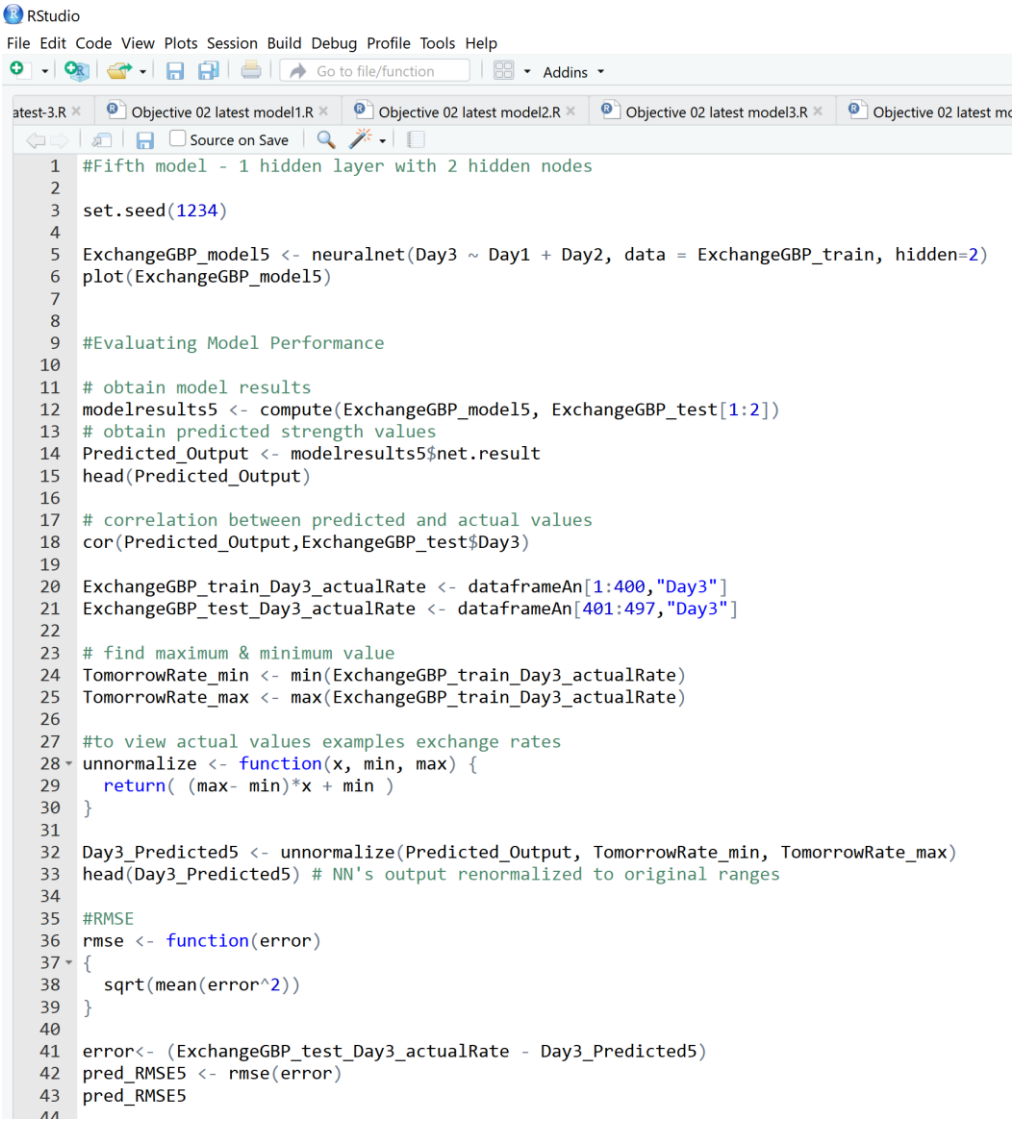
```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted4 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```



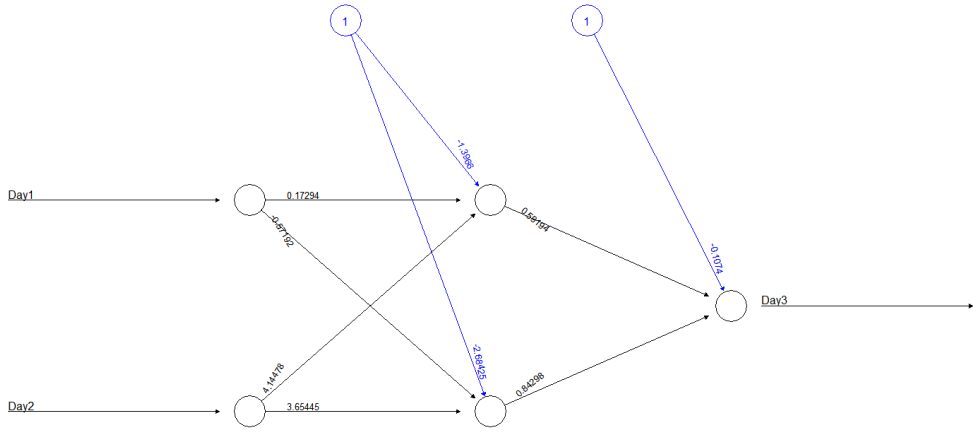
Desired Test Output and Predicted NN output


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
F:/L6/BI/BB CW/Business_Intelligence_CW2/
> #desired test output and the predicted NN output.
>
> final_result4 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted4)
> colnames(final_result4) <- c("Expected Output", "Neural Net Output")
> final_result4
Expected Output Neural Net Output
401 0.87946 0.8712300432
402 0.87779 0.8798681557
403 0.87678 0.8775386733
404 0.87443 0.8766106073
405 0.88200 0.8742392588
406 0.87805 0.8822135376
407 0.87012 0.8776290754
408 0.86848 0.8696341830
409 0.87087 0.8684741678
410 0.87432 0.8711054536
411 0.87409 0.8745429901
412 0.87947 0.8740590120
413 0.88285 0.8796466837
414 0.88613 0.8827308067
415 0.88488 0.8858204628
416 0.88671 0.8843327716
417 0.88541 0.8862654825
418 0.88287 0.8848291882
419 0.87722 0.8823400683
420 0.87753 0.8767089472
421 0.88154 0.8774271731
422 0.87065 0.8815288831
423 0.86115 0.8699423141
424 0.86218 0.8606648093
425 0.86555 0.8624474616
426 0.86926 0.8659484932
427 0.87672 0.8696227842
428 0.87225 0.8771486130
429 0.87005 0.8719654195
430 0.87151 0.8699750210
431 0.87874 0.8716648057
432 0.87549 0.8790760306
433 0.87354 0.8752042975
434 0.86908 0.8734021002
435 0.86770 0.8688617013
436 0.87088 0.8677257829
437 0.86961 0.8711715454

Fifth Model Results



ExchangeGBP Neural Network Model

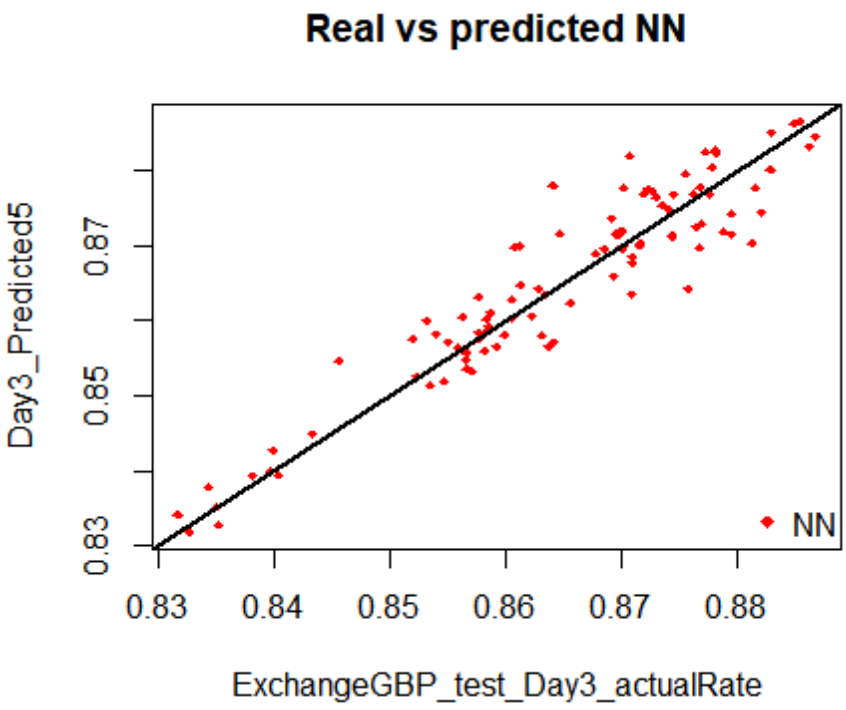


Model5 RMSE

```
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted5)
> pred_RMSE5 <- rmse(error)
> pred_RMSE5
[1] 0.004509078835
```

Performance of Network

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted5 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```


Real vs predicted NN
Day3_Predicted5
0.83
0.85
0.87
0.83
0.84
0.85
0.86
0.87
0.88
ExchangeGBP_test_Day3_actualRate
NN

Desired Test Output and Predicted NN output

Source

Console Terminal

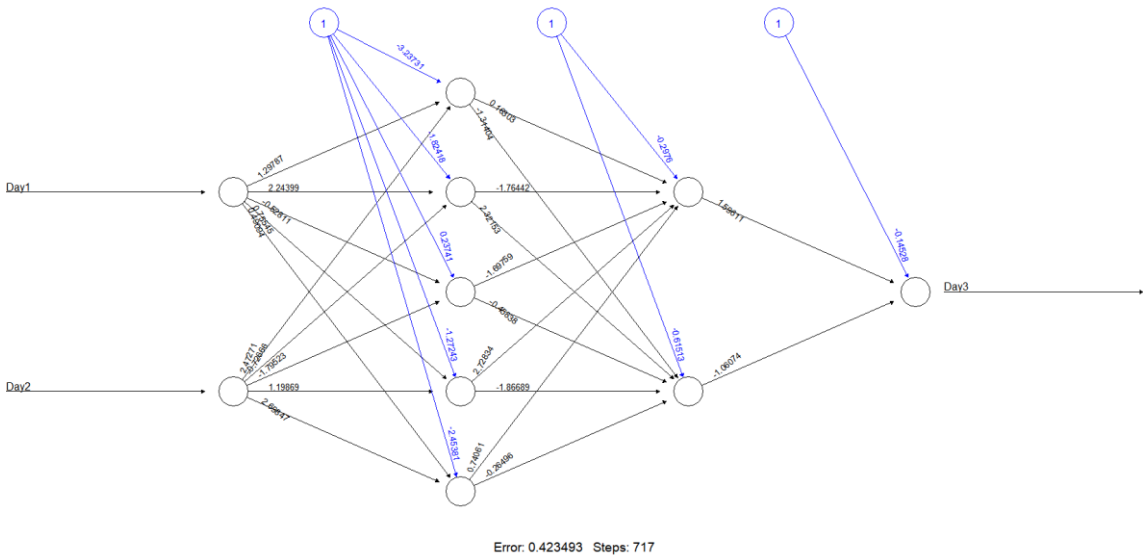
F:/L6/BI/BB CW/Business_Intelligence_CW2/

```
> #desired test output and the predicted NN output.
>
> final_result5 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted5)
> colnames(final_result5) <- c("Expected Output", "Neural Net Output")
> final_result5
    Expected Output Neural Net Output
401         0.87946      0.8714473725
402         0.87779      0.8803839313
403         0.87678      0.8777618513
404         0.87443      0.8768408059
405         0.88200      0.8744148426
406         0.87805      0.8827306828
407         0.87012      0.8777985638
408         0.86848      0.8696758213
409         0.87087      0.8685426836
410         0.87432      0.8712831238
411         0.87409      0.8748267455
412         0.87947      0.8742685471
413         0.88285      0.8800697274
414         0.88613      0.8831186789
415         0.88488      0.8861956814
416         0.88671      0.8845582280
417         0.88541      0.8865823690
418         0.88287      0.8850474291
419         0.87722      0.8825393332
420         0.87753      0.8768372215
421         0.88154      0.8776957498
422         0.87065      0.8819316646
423         0.86115      0.8699587850
424         0.86218      0.8605504634
425         0.86555      0.8623186526
426         0.86926      0.8659538204
427         0.87672      0.8697665236
428         0.87225      0.8775752667
429         0.87005      0.8720767170
430         0.87151      0.8700756591
431         0.87874      0.8718467880
432         0.87549      0.8795384778
433         0.87354      0.8753742798
434         0.86908      0.8735721099
435         0.86770      0.8689194914
436         0.87088      0.8677755084
437         0.86961      0.8713602186
```

Sixth Model Results

```
#Sixth model - 2 hidden layer with 5 hidden nodes

set.seed(1234)
ExchangeGBP_model6 <- neuralnet(Day3 ~ Day1 + Day2, data = ExchangeGBP_train, hidden= c(5,2))

plot(ExchangeGBP_model6)


#Evaluating Model Performance

# obtain model results
modelresults6 <- compute(ExchangeGBP_model6, ExchangeGBP_test[1:2])
# obtain predicted strength values
Predicted_Output <- modelresults6$net.result
head(Predicted_Output)

# correlation between predicted and actual values
cor(Predicted_Output,ExchangeGBP_test$Day3)

ExchangeGBP_train_Day3_actualRate <- dataframeAn[1:400,"Day3"]
ExchangeGBP_test_Day3_actualRate <- dataframeAn[401:497,"Day3"]

# find maximum & minimum value
TomorrowRate_min <- min(ExchangeGBP_train_Day3_actualRate)
TomorrowRate_max <- max(ExchangeGBP_train_Day3_actualRate)

#to view actual values examples exchange rates
unnormalize <- function(x, min, max) {
  return( (max- min)*x + min )
}

Day3_Predicted6 <- unnormalize(Predicted_Output, TomorrowRate_min, TomorrowRate_max)
head(Day3_Predicted6) # NN's output renormalized to original ranges

#RMSE
rmse <- function(error)
{
  sqrt(mean(error^2))
}

error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted6)
pred_RMSE6 <- rmse(error)
pred_RMSE6
```

ExchangeGBP Neural Network Model

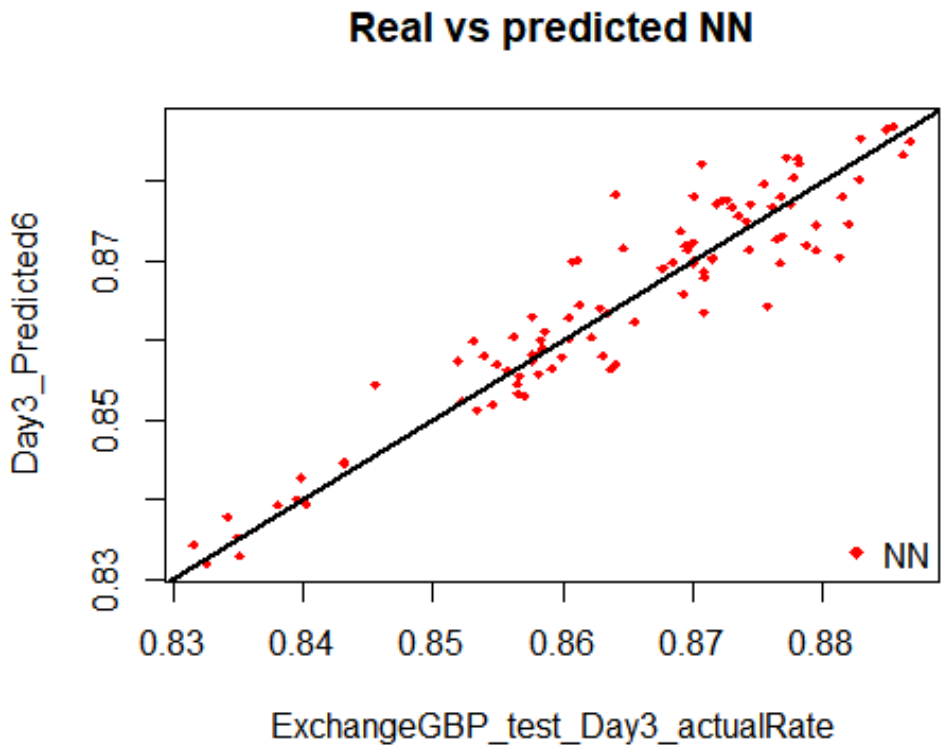


Model6 RMSE

```
> #RMSE
> rmse <- function(error)
+ {
+   sqrt(mean(error^2))
+ }
>
> error<- (ExchangeGBP_test_Day3_actualRate - Day3_Predicted6)
> pred_RMSE6 <- rmse(error)
> pred_RMSE6
[1] 0.004530374513
```

Performance of Network

```
#Performance Graphically
par(mfrow=c(1,1))
plot(ExchangeGBP_test_Day3_actualRate, Day3_Predicted6 ,col='red',main='Real vs predicted NN',pch=18,cex=0.7)
abline(0,1,lwd=2)
legend('bottomright',legend='NN', pch=18,col='red', bty='n')
```



Desired Test Output and Predicted NN output

Source

Console Terminal

F:/L6/BI/BB CW/Business_Intelligence_CW2/

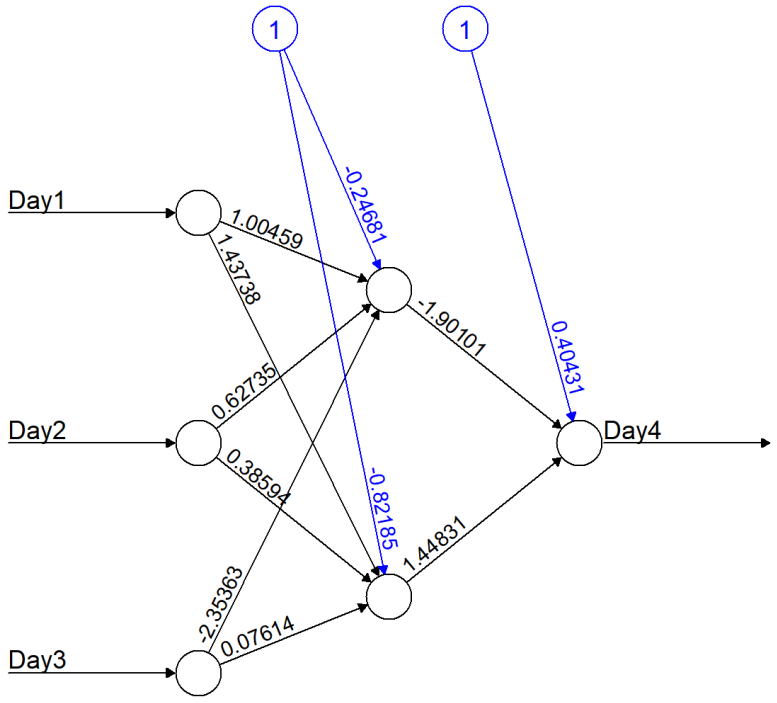
```
> final_result6 <- cbind(ExchangeGBP_test_Day3_actualRate, Day3_Predicted6)
> colnames(final_result6) <- c("Expected Output", "Neural Net Output")
> final_result6
    Expected Output Neural Net Output
401         0.87946      0.8713968713
402         0.87779      0.8803932508
403         0.87678      0.8780530841
404         0.87443      0.8771063373
405         0.88200      0.8746417308
406         0.87805      0.8828367880
407         0.87012      0.8781034671
408         0.86848      0.8697621597
409         0.87087      0.8686144972
410         0.87432      0.8713424207
411         0.87409      0.8749413014
412         0.87947      0.8744641657
413         0.88285      0.8802273299
414         0.88613      0.8833822389
415         0.88488      0.8864641024
416         0.88671      0.8848993537
417         0.88541      0.8868747247
418         0.88287      0.8853856579
419         0.87722      0.8828892885
420         0.87753      0.8771212220
421         0.88154      0.8779567992
422         0.87065      0.8821656320
423         0.86115      0.8699904138
424         0.86218      0.8603686269
425         0.86555      0.8622165104
426         0.86926      0.8658571221
427         0.87672      0.8697452688
428         0.87225      0.8775864548
429         0.87005      0.8722545047
430         0.87151      0.8701920427
431         0.87874      0.8719457861
432         0.87549      0.8796049153
433         0.87354      0.8756291320
434         0.86908      0.8737749107
435         0.86770      0.8690144874
436         0.87088      0.8678253717
437         0.86961      0.8713967970
438         0.86076      0.8698074246
439         0.85830      0.8600264330
440         0.86309      0.8579723011
```

Selection of Best Neural Network

Once the code was executed the RMSE values will be displayed as outputs. The results for every attempts are displayed in Table.

Attempt	Inputs			Hidden Layers and Nodes	RMSE	Correlation
	Day1	Day2	Day3			
1	Yes	Yes	Yes	No	0.004554154516	0.9430029478
2	Yes	Yes	Yes	2	0.004397518917	0.945180366
3	Yes	Yes	Yes	c (5,2)	0.004467412655	0.9433085362
4	Yes	Yes	Yes	No	0.004481170106	0.941072228
5	Yes	Yes	Yes	2	0.004509078835	0.9410000878
6	Yes	Yes	Yes	c (5,2)	0.004530374513	0.9408200816

From table a conclusion can be drawn that the second attempt has the lowest RMSE value. Once the RMSE value is calculated the exchange model can be plotted.



Error: 0.425623 Steps: 2437

2.5. Prediction Output Vs. Desired Output

After attempt 2 was identified as the best attempt. Once identified the desired output along with the prediction output for the code could be executed.

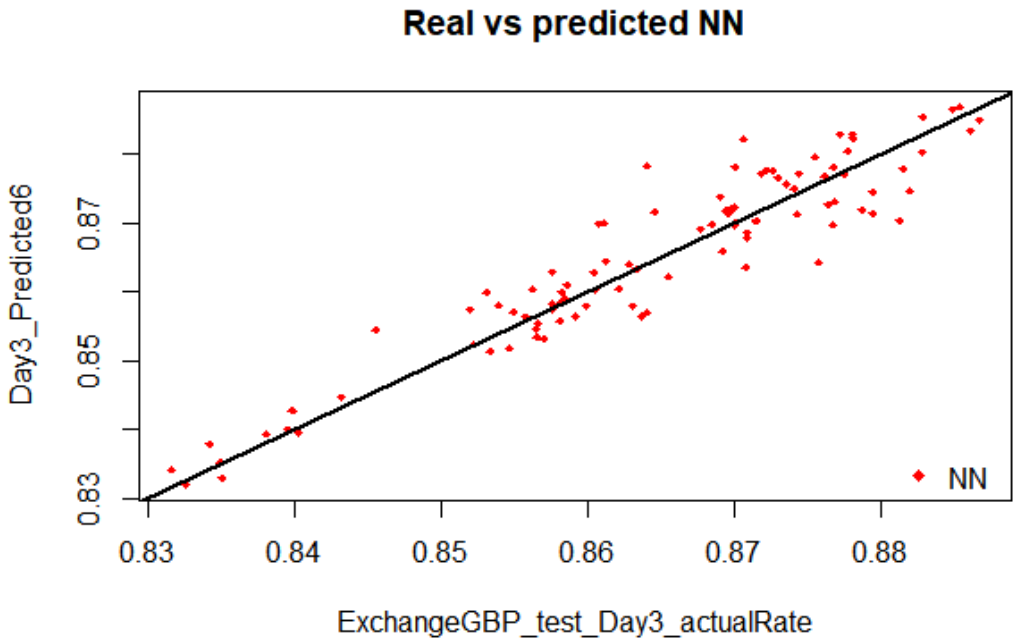
RStudio Source Editor

final_result2

Filter

	Expected Output	Neural Net Output
401	0.87779	0.8790572389
402	0.87678	0.8770762799
403	0.87443	0.8769749016
404	0.88200	0.8745876297
405	0.87805	0.8823520074
406	0.87012	0.8773400313
407	0.86848	0.8700960049
408	0.87087	0.8693021225
409	0.87432	0.8714436321
410	0.87409	0.8744597158
411	0.87947	0.8740285816
412	0.88285	0.8797033089
413	0.88613	0.8823197278
414	0.88488	0.8855894333
415	0.88671	0.8842554081
416	0.88541	0.8864420132
417	0.88287	0.8848450903
418	0.87722	0.8825844039
419	0.87753	0.8770616473
420	0.88154	0.8780365465
421	0.87065	0.8815822230
422	0.86115	0.8699466063
423	0.86218	0.8614113094
424	0.86555	0.8634074109
425	0.86926	0.8660126117
426	0.87672	0.8694377805
427	0.87225	0.8766654150
428	0.87005	0.8717214818
429	0.87151	0.8705810447
430	0.87874	0.8720713974
431	0.87549	0.8788562535
432	0.87354	0.8749370595

Showing 1 to 34 of 97 entries


Real vs predicted NN
Day3_Predicted6
0.83
0.85
0.87
0.83
0.84
0.85
0.86
0.87
0.88
ExchangeGBP_test_Day3_actualRate
NN

3. Objective 3 – KNN

3.1. Consideration of Alternative Input Options

The Vehicles dataset contains both numeric and symbolic attributes. Numeric attributes would be used instead of symbolic attributes during this scenario. The Class attribute will be considered as an output, as it's an factor. In order, to check different alternative options different tests are proposed as follows;

Test 1 – 3 attributes - Comp, Circ, D.Circ,

Test 2– 5 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra

Test 3– 8 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra, Max.L.Ra,Scat.Ra, Elong,

Test 4– 10 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra, Max.L.Ra,Scat.Ra, Elong, Pr.Axis.Rect, Max.L.Rect,

Test 5– 12 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra, Max.L.Ra,Scat.Ra, Elong, Pr.Axis.Rect, Max.L.Rect, Sc.Var.Maxis, Sc.Var.maxis,

Test 6– 15 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra, Max.L.Ra,Scat.Ra, Elong, Pr.Axis.Rect, Max.L.Rect, Sc.Var.Maxis, Sc.Var.maxis, Ra.Gyr, Skew.Maxis, Skew.maxis,

Test 7– 18 attributes - Comp, Circ, D.Circ, Rad. Ra, Pr. Axis.Ra, Max.L.Ra,Scat.Ra, Elong, Pr.Axis.Rect, Max.L.Rect, Sc.Var.Maxis, Sc.Var.maxis, Ra.Gyr, Skew.Maxis, Skew.maxis, Kurt.Maxis, Kurt.Maxis, Holl.Ra

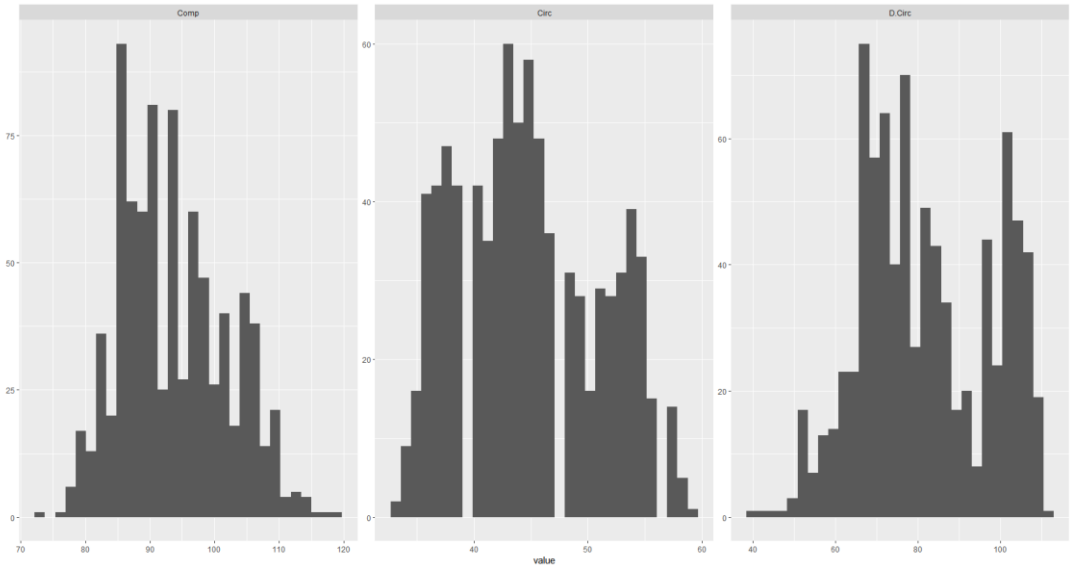
3.2. Pre-processing

- First Column which is called as the "Sample" Column needs to be removed from the Vehicles Data set.
- Data needs to be shuffled after selection of input options for the current test to be conducted.

```
#Shuffling data
random_class <- Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed
str(random_class)
```

- Needs to check if data is uniformly distributed in order to choose between Normalization and Scaling

```
#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')
```



- Conduct Normalization as Data is not uniformly distributed and also a bell shape is not visible. Once the normalize code is executed the data scatter would be stopped and a high performing cluster will be created

```
#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}
vehiclesN <- as.data.frame(lapply(random_class[1:3], normalize))
summary(vehiclesN)
vehiclesNnew <- cbind(vehiclesN,random_class[4])
```

Filter

	Comp	Circ	D.Circ
1	0.3478260870	0.34615384615	0.5555555556
2	0.4130434783	0.46153846154	0.5000000000
3	0.3478260870	0.19230769231	0.4722222222
4	0.2826086957	0.11538461538	0.5277777778
5	0.4565217391	0.46153846154	0.6250000000
6	0.6086956522	0.61538461538	0.8750000000
7	0.3695652174	0.38461538462	0.3611111111
8	0.2826086957	0.38461538462	0.3611111111
9	0.2608695652	0.38461538462	0.3611111111
10	0.6739130435	0.76923076923	0.9444444444
11	0.2391304348	0.19230769231	0.3611111111

- Partition Data according to the criteria in the question

```
#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]
```

- Check for any Missing Values as it will affect the accuracy of the results generated

```
#Check for NULL values in normalized data
anyNA(vehiclesNnew)
summary(vehiclesNnew)
```

```
> #Check for NULL values in normalized data
> anyNA(vehiclesNnew)
[1] FALSE
```

3.3. Test1

Objective 03 latest Test2 norm.R | Objective 03 latest Test1 norm.R | vehiclesNnew | Objective 03 lates

Source on Save

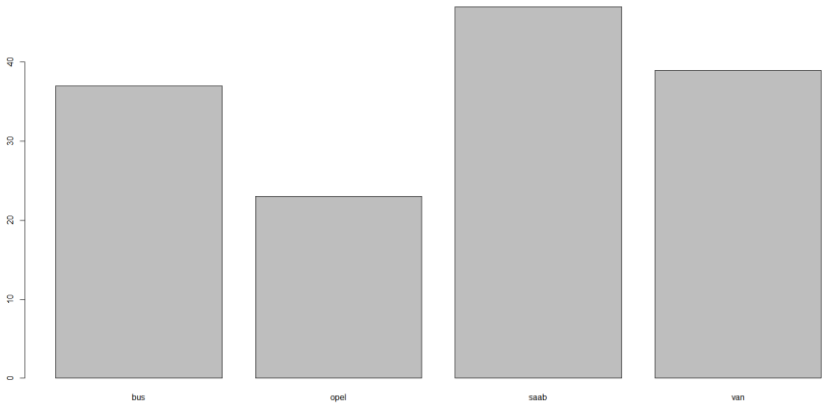
```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,19)]
head(Newvehicles)
str(Newvehicles)

#Shuffling data
random_class <- Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed
str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')

#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}
vehiclesN <- as.data.frame(lapply(random_class[1:3], normalize))
summary(vehiclesN)
vehiclesNnew <- cbind(vehiclesN,random_class[4])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]
```

```
#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)
summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Using selected K values
set.seed(1234)
#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-4]
testset_without_class<-testset[-4]

#Labels
trainset_labels <- vehiclesNnew[1:700, 4]
testset_labels <- vehiclesNnew[701:846, 4]

#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)

#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)
```

```
# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

(Top Level)

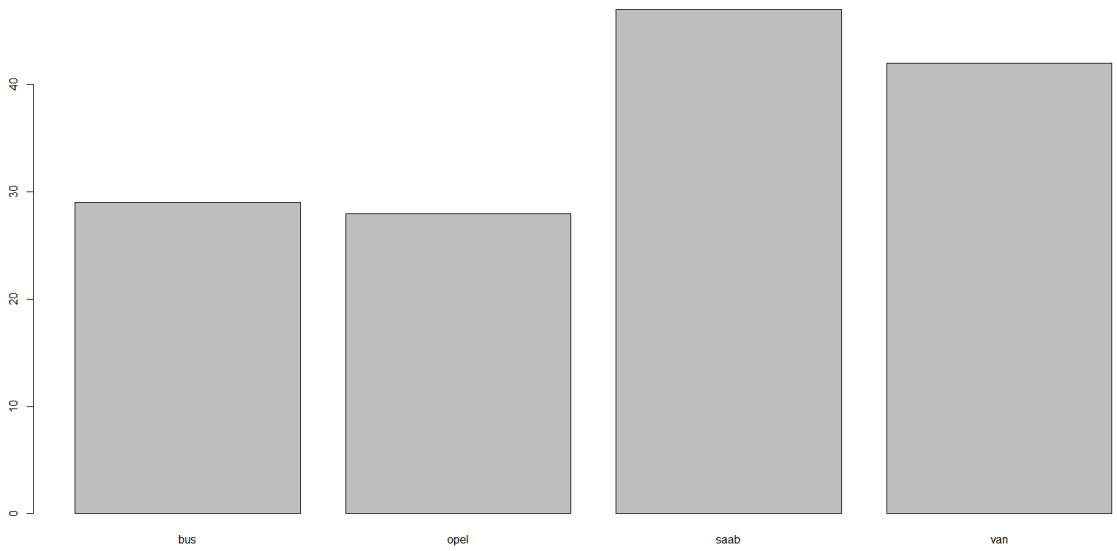
K=3




```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics
```

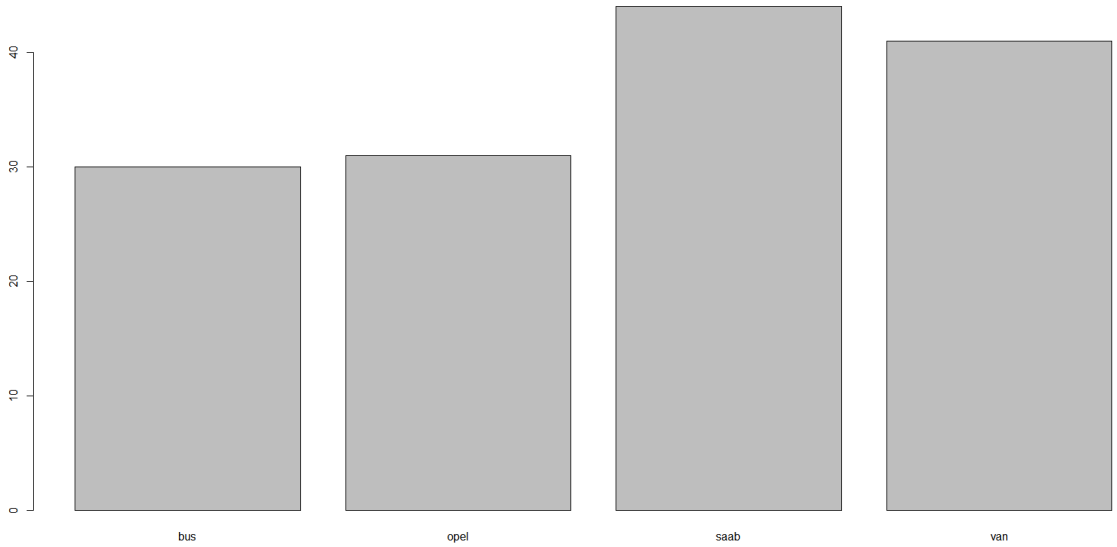
```
          Reference
Prediction bus opel saab van
      bus   26    3    2   6
      opel   3    7   10   3
      saab   1   20   22   4
      van    5    5    6  23

Overall Statistics

               Accuracy : 0.5342466
                 95% CI : (0.4499323, 0.617143)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : 0.00000000002988979

                  Kappa : 0.3768516
 Mcnemar's Test P-Value : 0.5884194

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.7428571  0.20000000   0.5500000  0.6388889
Specificity           0.9009009  0.85585586   0.7641509  0.8545455
Pos Pred Value        0.7027027  0.30434783   0.4680851  0.5897436
Neg Pred Value        0.9174312  0.77235772   0.8181818  0.8785047
Prevalence            0.2397260  0.23972603   0.2739726  0.2465753
Detection Rate        0.1780822  0.04794521   0.1506849  0.1575342
Detection Prevalence  0.2534247  0.15753425   0.3219178  0.2671233
Balanced Accuracy     0.8218790  0.52792793   0.6570755  0.7467172
```

K=5



```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   23    1    1   4
      opel   5   13    8   2
      saab   2   16   25   4
      van    5    5    6  26

Overall Statistics

               Accuracy : 0.5958904
                 95% CI : (0.5115909, 0.6762166)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : 0.000000000000000004038981

                  Kappa : 0.4592253
 Mcnemar's Test P-Value : 0.2800998

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.6571429   0.3714286   0.6250000
Specificity           0.9459459   0.8648649   0.7924528
Pos Pred Value        0.7931034   0.4642857   0.5319149
Neg Pred Value        0.8974359   0.8135593   0.8484848
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.1575342   0.0890411   0.1712329
Detection Prevalence  0.1986301   0.1917808   0.3219178
Balanced Accuracy     0.8015444   0.6181467   0.7087264
                     Class: van
Sensitivity           0.7222222
Specificity           0.8545455
Pos Pred Value        0.6190476
Neg Pred Value        0.9038462
Prevalence            0.2465753
Detection Rate        0.1780822
Detection Prevalence  0.2876712
Balanced Accuracy     0.7883838
```

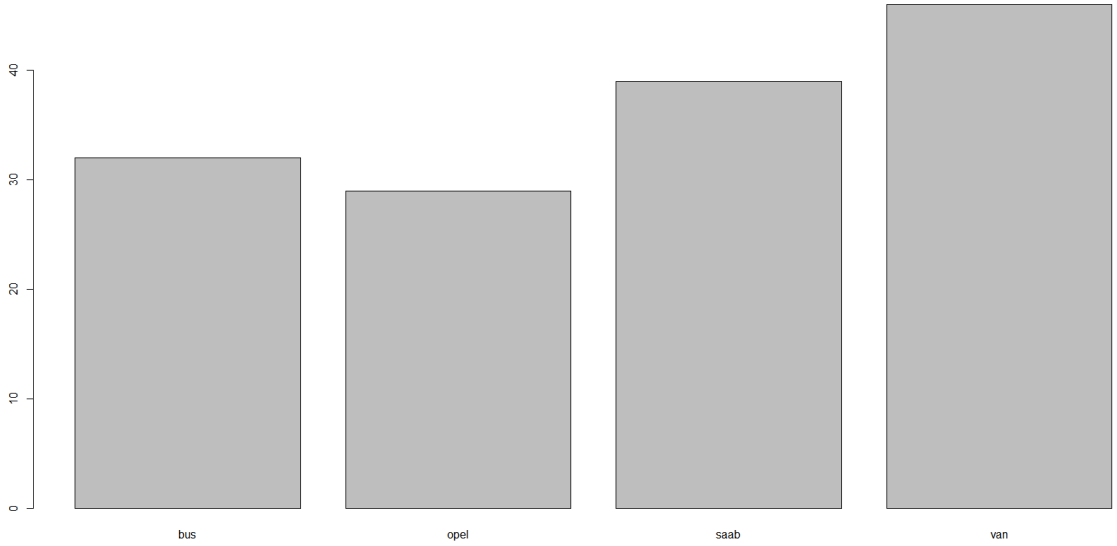
K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   25    1    1   3
      opel   5   12    9   5
      saab   0   17   24   3
      van    5    5    6  25

Overall Statistics

               Accuracy : 0.5890411
                 95% CI : (0.5046783, 0.669715)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : 0.00000000000000001566832

                  Kappa : 0.4506115
 Mcnemar's Test P-Value : 0.2666265

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.7142857  0.34285714   0.6000000
Specificity           0.9549550  0.82882883   0.8113208
Pos Pred Value        0.8333333  0.38709677   0.5454545
Neg Pred Value        0.9137931  0.80000000   0.8431373
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.1712329  0.08219178   0.1643836
Detection Prevalence  0.2054795  0.21232877   0.3013699
Balanced Accuracy     0.8346203  0.58584299   0.7056604
                     Class: van
Sensitivity           0.6944444
Specificity           0.8545455
Pos Pred Value        0.6097561
Neg Pred Value        0.8952381
Prevalence            0.2465753
```

```
Detection Rate        0.1712329
Detection Prevalence  0.2808219
Balanced Accuracy     0.7744949
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   26    1    1   4
      opel   3   13    9   4
      saab   0   14   22   3
      van    6    7    8  25

Overall Statistics

               Accuracy : 0.5890411
                 95% CI : (0.5046783, 0.669715)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : 0.00000000000000001566832

                  Kappa : 0.4512997
 Mcnemar's Test P-Value : 0.3616539

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.7428571   0.3714286   0.5500000
Specificity           0.9459459   0.8558559   0.8396226
Pos Pred Value        0.8125000   0.4482759   0.5641026
```

```
Neg Pred Value         0.9210526   0.8119658   0.8317757
Prevalence             0.2397260   0.2397260   0.2739726
Detection Rate         0.1780822   0.0890411   0.1506849
Detection Prevalence   0.2191781   0.1986301   0.2671233
Balanced Accuracy      0.8444015   0.6136422   0.6948113
                     Class: van
Sensitivity            0.6944444
Specificity            0.8090909
Pos Pred Value         0.5434783
Neg Pred Value         0.8900000
Prevalence             0.2465753
Detection Rate         0.1712329
Detection Prevalence   0.3150685
Balanced Accuracy      0.7517677
```

3.4. Test2

ctive 03 latest Test2 norm.R

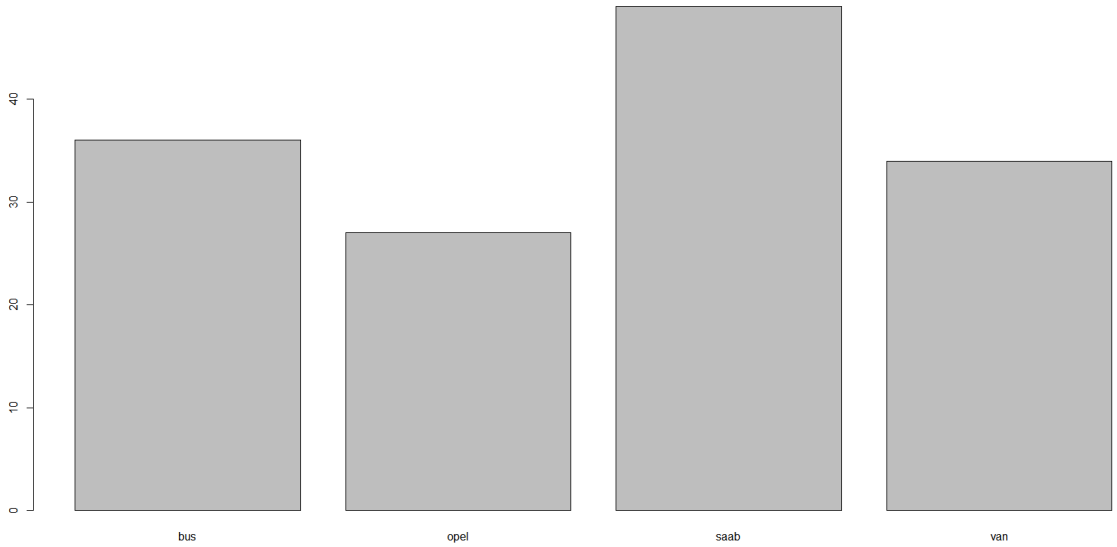
Source on Save

```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,19)]
head(Newvehicles)
str(Newvehicles)


#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')


#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:5], normalize))

summary(vehiclesN)

vehiclesNnew <- cbind(vehiclesN,random_class[6])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]
```

```
#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)
summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier
#Plotting accuracy vs K vlue graph
plot(fit.knn)
#Using selected K values
set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-6]
testset_without_class<-testset[-6]

#Labels
trainset_labels <- vehiclesNnew[1:700, 6]
testset_labels <- vehiclesNnew[701:846, 6]


#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)
```

```
#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

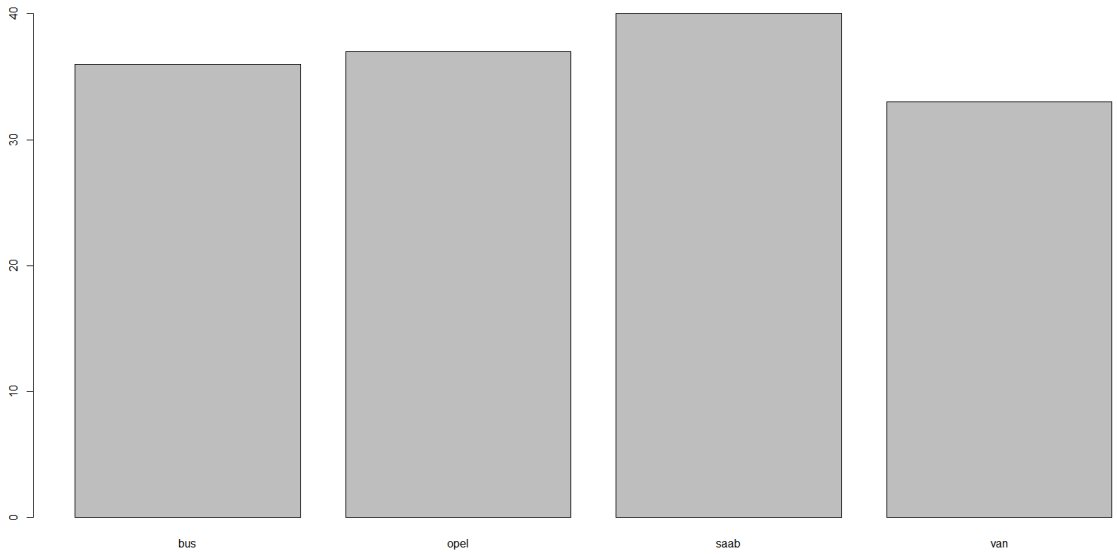
K=3




```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   29    4    0   3
      opel   2   11   10   4
      saab   3   16   27   3
      van    1    4    3  26

Overall Statistics

               Accuracy : 0.6369863
                 95% CI : (0.5534027, 0.714888)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.0000000000000002

                  Kappa : 0.5141583
 Mcnemar's Test P-Value : 0.41747

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.8285714  0.31428571   0.6750000
Specificity           0.9369369  0.85585586   0.7924528
Pos Pred Value        0.8055556  0.40740741   0.5510204
Neg Pred Value        0.9454545  0.79831933   0.8659794
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.1986301  0.07534247   0.1849315
Detection Prevalence  0.2465753  0.18493151   0.3356164
Balanced Accuracy     0.8827542  0.58507079   0.7337264
                     Class: van
Sensitivity           0.7222222
Specificity           0.9272727
Pos Pred Value        0.7647059
Neg Pred Value        0.9107143
```

```
Prevalence            0.2465753
Detection Rate        0.1780822
Detection Prevalence  0.2328767
Balanced Accuracy     0.8247475
```

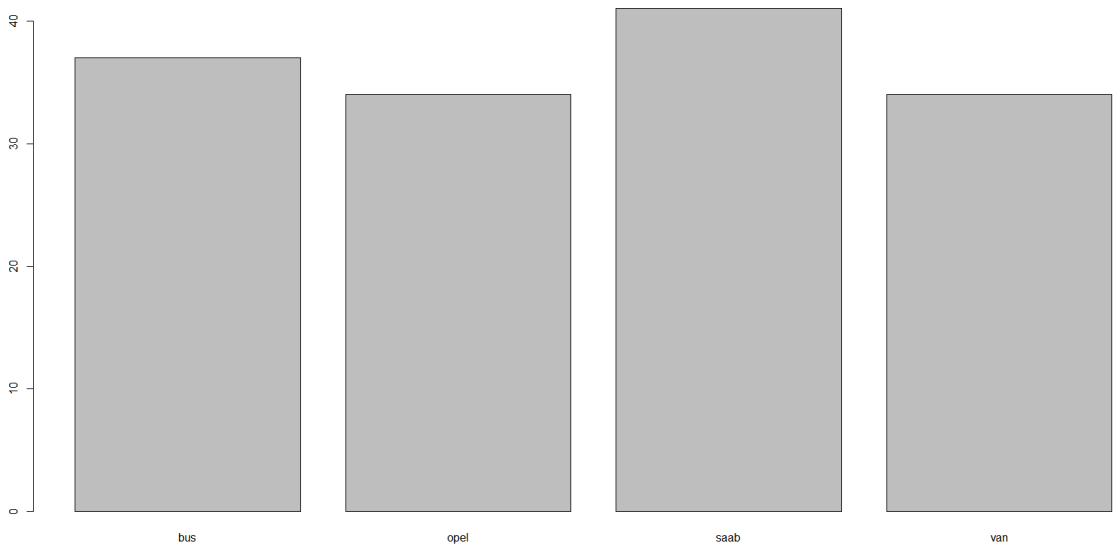
K=5




```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   28    3    1   4
      opel   4   13   15   5
      saab   2   12   23   3
      van    1    7    1  24

Overall Statistics

               Accuracy : 0.6027397
                 95% CI : (0.5185193, 0.6827024)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.4698554
 Mcnemar's Test P-Value : 0.6844093

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.8000000   0.3714286   0.5750000
Specificity           0.9279279   0.7837838   0.8396226
Pos Pred Value        0.7777778   0.3513514   0.5750000
Neg Pred Value        0.9363636   0.7981651   0.8396226
```

```
Prevalence             0.2397260   0.2397260   0.2739726
Detection Rate         0.1917808   0.0890411   0.1575342
Detection Prevalence   0.2465753   0.2534247   0.2739726
Balanced Accuracy      0.8639640   0.5776062   0.7073113
                     Class: van
Sensitivity            0.6666667
Specificity            0.9181818
Pos Pred Value         0.7272727
Neg Pred Value         0.8938053
Prevalence             0.2465753
Detection Rate         0.1643836
Detection Prevalence   0.2260274
Balanced Accuracy      0.7924242
```

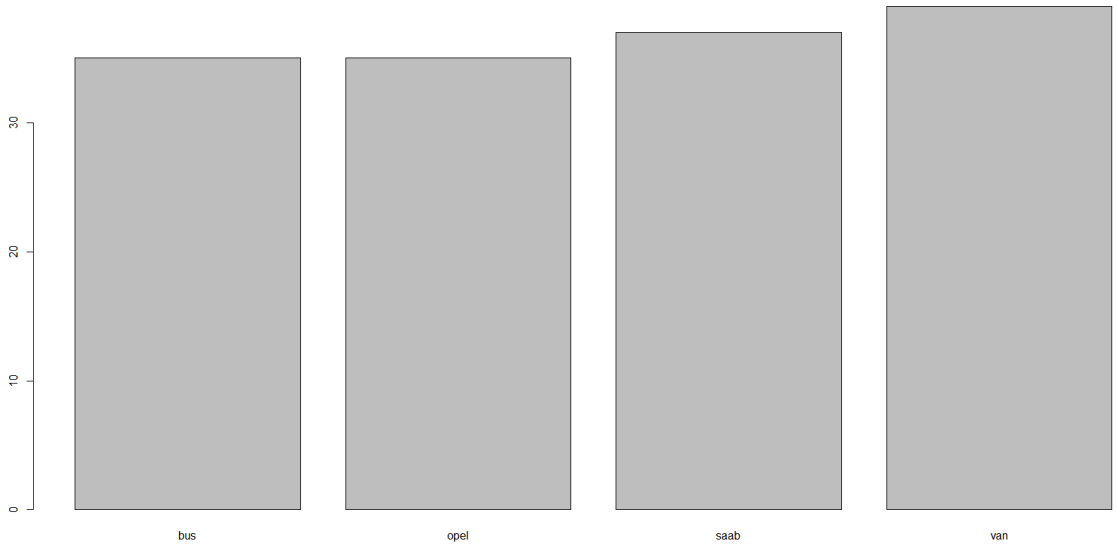
K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   28    2    2   5
      opel   4   17   10   3
      saab   2   11   25   3
      van    1    5    3  25

Overall Statistics

               Accuracy : 0.6506849
                 95% CI : (0.5674717, 0.7276452)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5336632
```

```
 Mcnemar's Test P-Value : 0.6927829

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.8000000   0.4857143   0.6250000
Specificity           0.9189189   0.8468468   0.8490566
Pos Pred Value        0.7567568   0.5000000   0.6097561
Neg Pred Value        0.9357798   0.8392857   0.8571429
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.1917808   0.1164384   0.1712329
Detection Prevalence  0.2534247   0.2328767   0.2808219
Balanced Accuracy     0.8594595   0.6662806   0.7370283
                     Class: van
Sensitivity           0.6944444
Specificity           0.9181818
Pos Pred Value        0.7352941
Neg Pred Value        0.9017857
Prevalence            0.2465753
Detection Rate        0.1712329
Detection Prevalence  0.2328767
Balanced Accuracy     0.8063131
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   28    2    1   4
      opel   4   16   15   0
      saab   2   12   21   2
      van    1    5    3  30
```

```
Overall Statistics

               Accuracy : 0.6506849
                 95% CI : (0.5674717, 0.7276452)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5341009
 Mcnemar's Test P-Value : 0.2146853

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.8000000   0.4571429   0.5250000
Specificity           0.9369369   0.8288288   0.8490566
Pos Pred Value        0.8000000   0.4571429   0.5675676
Neg Pred Value        0.9369369   0.8288288   0.8256881
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.1917808   0.1095890   0.1438356
Detection Prevalence  0.2397260   0.2397260   0.2534247
Balanced Accuracy     0.8684685   0.6429858   0.6870283
                     Class: van
Sensitivity           0.8333333
Specificity           0.9181818
Pos Pred Value        0.7692308
Neg Pred Value        0.9439252
Prevalence            0.2465753
Detection Rate        0.2054795
Detection Prevalence  0.2671233
Balanced Accuracy     0.8757576
```

3.5. Test3

o Source Editor

:tive 03 latest Test3 norm.R ×

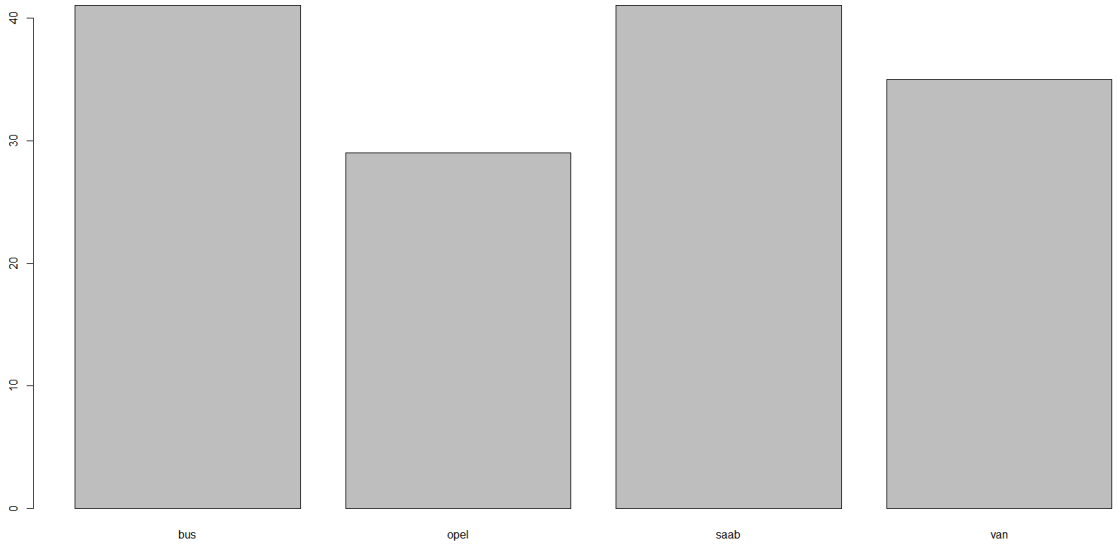
Source on Save

```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,6,7,8,19)]
head(Newvehicles)
str(Newvehicles)


#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')


#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:8], normalize))

summary(vehiclesN)

vehiclesNnew <- cbind(vehiclesN,random_class[9])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]
```

```
#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)

summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Using selected K values

set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-9]
testset_without_class<-testset[-9]

#Labels
trainset_labels <- vehiclesNnew[1:700, 9]
testset_labels <- vehiclesNnew[701:846, 9]

#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)

#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
```

```
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

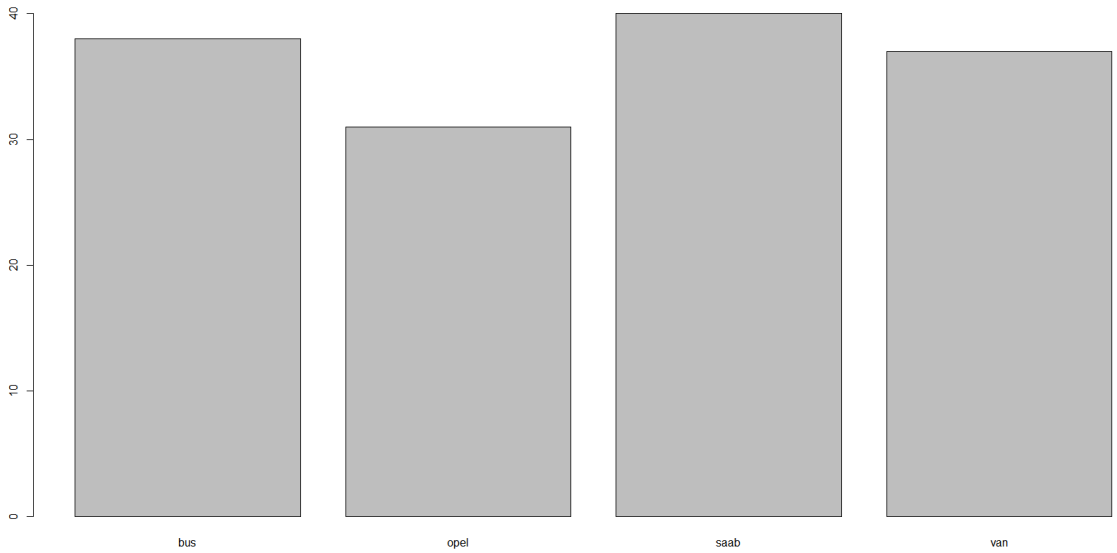
K=3




```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   35    2    2   2
      opel   0   13   14   2
      saab   0   17   23   1
      van    0    3    1  31

Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.597645
 Mcnemar's Test P-Value : 0.3705584

Statistics by Class:
```

```
                     Class: bus Class: opel Class: saab
Sensitivity           1.0000000   0.3714286   0.5750000
Specificity           0.9459459   0.8558559   0.8301887
Pos Pred Value        0.8536585   0.4482759   0.5609756
Neg Pred Value        1.0000000   0.8119658   0.8380952
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2397260   0.0890411   0.1575342
Detection Prevalence  0.2808219   0.1986301   0.2808219
Balanced Accuracy     0.9729730   0.6136422   0.7025943
                     Class: van
Sensitivity           0.8611111
Specificity           0.9636364
Pos Pred Value        0.8857143
Neg Pred Value        0.9549550
Prevalence            0.2465753
Detection Rate        0.2123288
Detection Prevalence  0.2397260
Balanced Accuracy     0.9123737
```

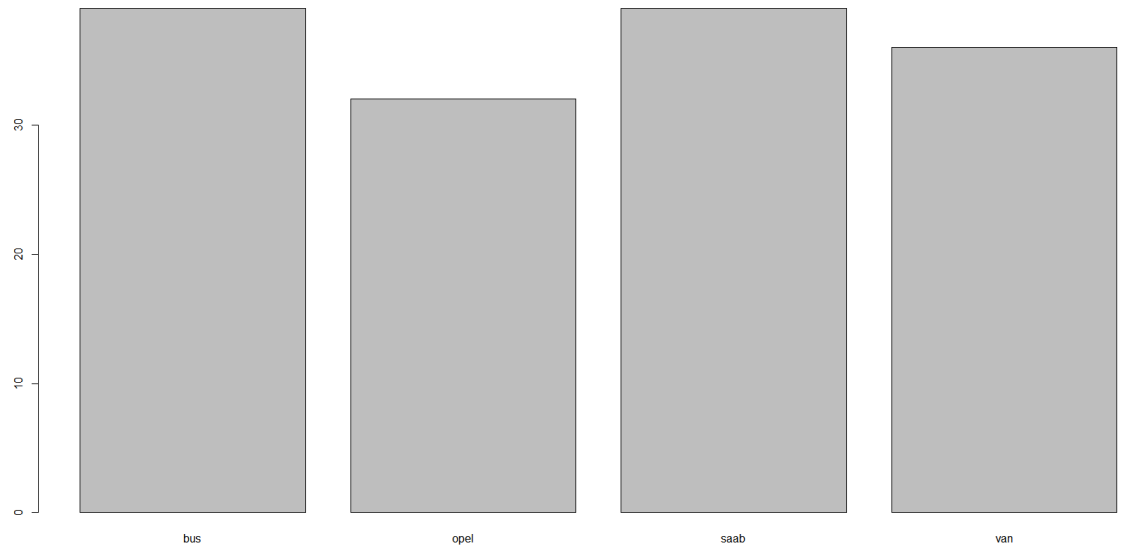
K=5




```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   34    0    2   2
      opel   0   15   12   4
      saab   0   16   24   0
      van    1    4    2  30

Overall Statistics

               Accuracy : 0.7054795
```

```
                  95% CI : (0.6244538, 0.7779581)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022204

                   Kappa : 0.6068633
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9714286   0.4285714   0.6000000
Specificity           0.9639640   0.8558559   0.8490566
Pos Pred Value        0.8947368   0.4838710   0.6000000
Neg Pred Value        0.9907407   0.8260870   0.8490566
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2328767   0.1027397   0.1643836
Detection Prevalence  0.2602740   0.2123288   0.2739726
Balanced Accuracy     0.9676963   0.6422136   0.7245283
                     Class: van
Sensitivity           0.8333333
Specificity           0.9363636
Pos Pred Value        0.8108108
Neg Pred Value        0.9449541
Prevalence            0.2465753
Detection Rate        0.2054795
Detection Prevalence  0.2534247
Balanced Accuracy     0.8848485
```

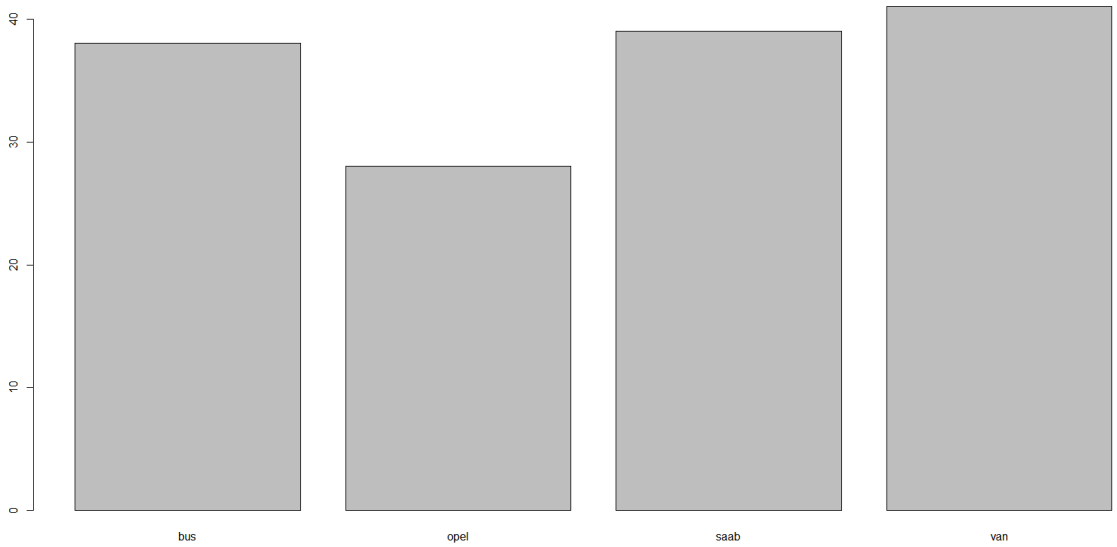
K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
```

```
Prediction bus opel saab van
      bus   34    1    2   2
      opel   1   15   13   3
      saab   0   15   23   1
      van    0    4    2  30

Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5978717
 Mcnemar's Test P-Value : 0.5935148

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9714286   0.4285714   0.5750000
Specificity           0.9549550   0.8468468   0.8490566
Pos Pred Value        0.8717949   0.4687500   0.5897436
Neg Pred Value        0.9906542   0.8245614   0.8411215
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2328767   0.1027397   0.1575342
Detection Prevalence  0.2671233   0.2191781   0.2671233
Balanced Accuracy     0.9631918   0.6377091   0.7120283
                     Class: van
Sensitivity           0.8333333
Specificity           0.9454545
Pos Pred Value        0.8333333
Neg Pred Value        0.9454545
Prevalence            0.2465753
Detection Rate        0.2054795
Detection Prevalence  0.2465753
Balanced Accuracy     0.8893939
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    2   2
      opel   1   14   11   2
      saab   1   13   25   0
      van    0    7    2  32

Overall Statistics

               Accuracy : 0.7123288
                 95% CI : (0.6316608, 0.7841615)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.6160301
 Mcnemar's Test P-Value : 0.2959207

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571  0.40000000   0.6250000
Specificity           0.9549550  0.87387387   0.8679245
Pos Pred Value        0.8684211  0.50000000   0.6410256
Neg Pred Value        0.9814815  0.82203390   0.8598131
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.2260274  0.09589041   0.1712329
Detection Prevalence  0.2602740  0.19178082   0.2671233
Balanced Accuracy     0.9489060  0.63693694   0.7464623
                     Class: van
Sensitivity           0.8888889
Specificity           0.9181818
Pos Pred Value        0.7804878
Neg Pred Value        0.9619048
Prevalence            0.2465753
```

```
Detection Rate         0.2191781
Detection Prevalence   0.2808219
Balanced Accuracy      0.9035354
```

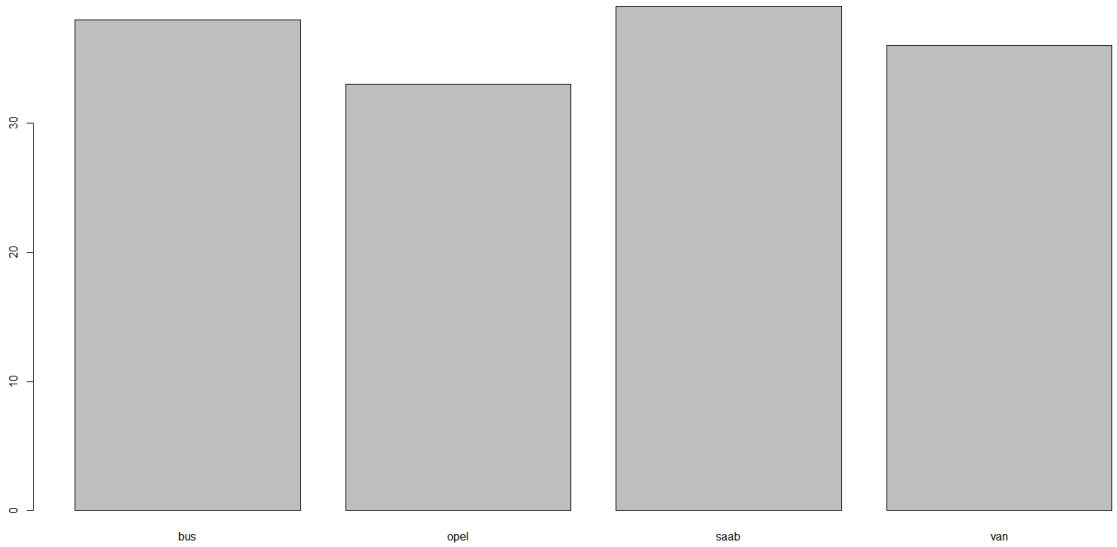
3.6. Test4

```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,6,7,8,9,10,19)]
head(Newvehicles)
str(Newvehicles)


#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')


#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:10], normalize))

summary(vehiclesN)


vehiclesNnew <- cbind(vehiclesN,random_class[11])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]
```

```
#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)

summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Using selected K values

set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-11]
testset_without_class<-testset[-11]

#Labels
trainset_labels <- vehiclesNnew[1:700, 11]
testset_labels <- vehiclesNnew[701:846, 11]



#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)
```

```
#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

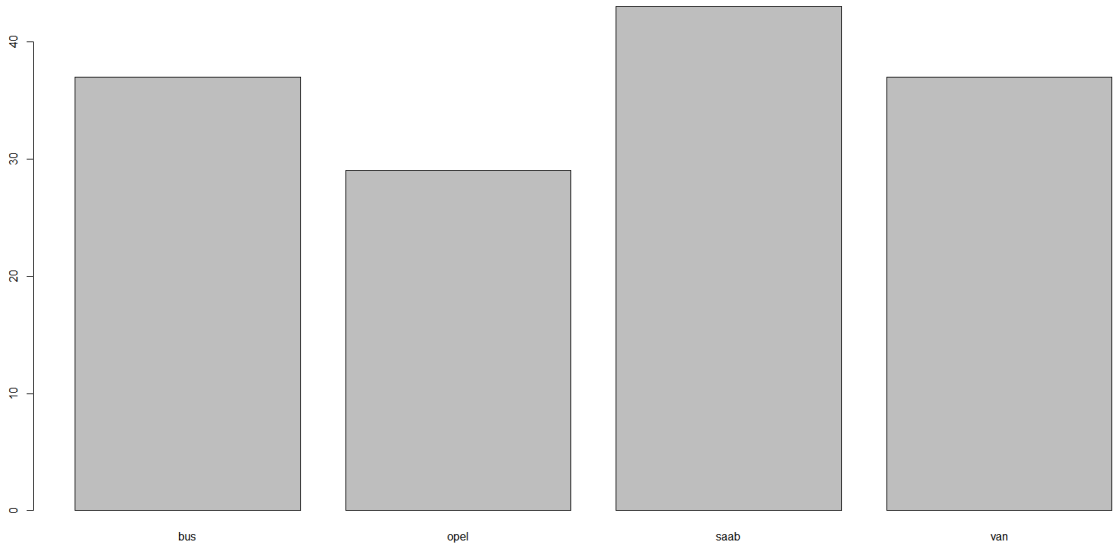
K=3




```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    2   2
      opel   0   14   16   3
      saab   1   17   21   0
      van    1    3    1  31

Overall Statistics

               Accuracy : 0.6780822
                 95% CI : (0.5958173, 0.7529494)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5704538
 Mcnemar's Test P-Value : 0.8458051

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571  0.40000000   0.5250000
Specificity           0.9549550  0.82882883   0.8301887
Pos Pred Value        0.8684211  0.42424242   0.5384615
Neg Pred Value        0.9814815  0.81415929   0.8224299
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.2260274  0.09589041   0.1438356
Detection Prevalence  0.2602740  0.22602740   0.2671233
Balanced Accuracy     0.9489060  0.61441441   0.6775943
                     Class: van
Sensitivity           0.8611111
Specificity           0.9545455
Pos Pred Value        0.8611111
Neg Pred Value        0.9545455
```

```
Prevalence            0.2465753
Detection Rate        0.2123288
Detection Prevalence  0.2465753
Balanced Accuracy     0.9078283
```

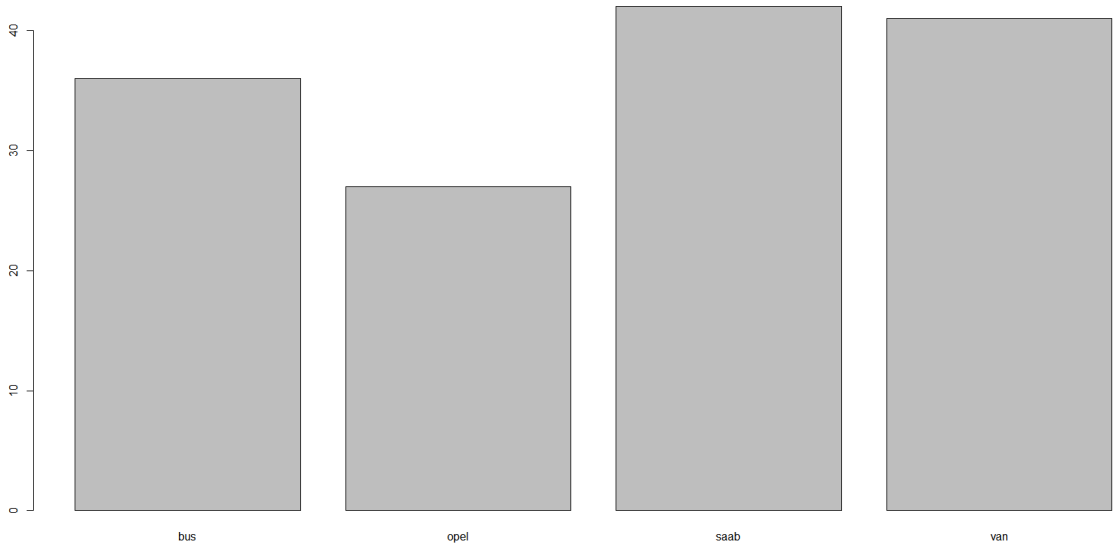
K=5




```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    2   1
      opel   0   13   14   2
      saab   1   19   23   0
      van    1    2    1  33

Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5973424
 Mcnemar's Test P-Value : 0.7973534

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571   0.3714286   0.5750000
Specificity           0.9639640   0.8558559   0.8113208
Pos Pred Value        0.8918919   0.4482759   0.5348837
Neg Pred Value        0.9816514   0.8119658   0.8349515
```

```
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2260274   0.0890411   0.1575342
Detection Prevalence  0.2534247   0.1986301   0.2945205
Balanced Accuracy     0.9534106   0.6136422   0.6931604
                     Class: van
Sensitivity           0.9166667
Specificity           0.9636364
Pos Pred Value        0.8918919
Neg Pred Value        0.9724771
Prevalence            0.2465753
Detection Rate        0.2260274
Detection Prevalence  0.2534247
Balanced Accuracy     0.9401515
```

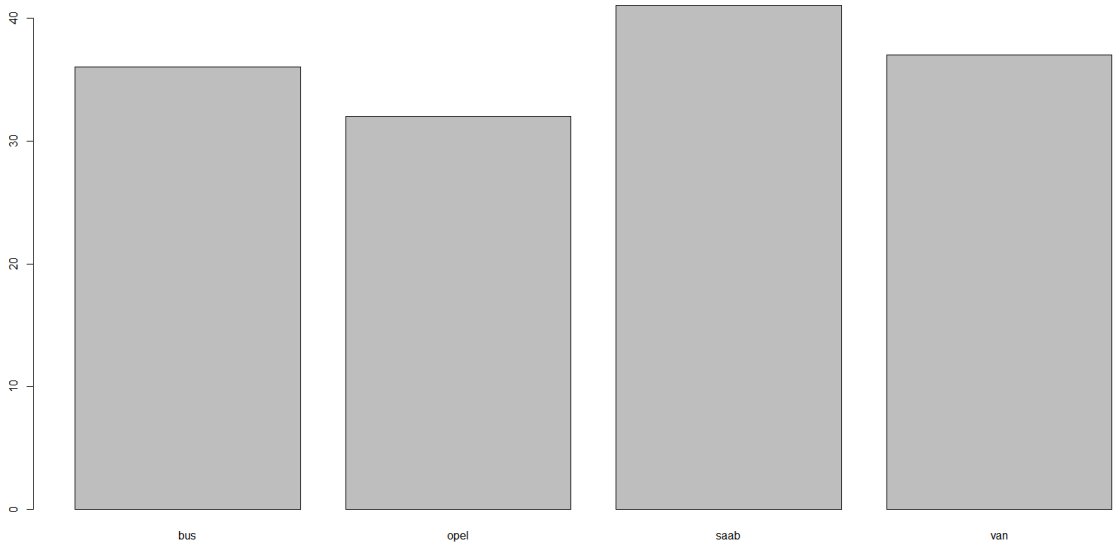
K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   32    0    2   2
      opel   1   12   14   0
      saab   1   18   22   1
      van    1    5    2  33

Overall Statistics

               Accuracy : 0.6780822
                 95% CI : (0.5958173, 0.7529494)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5699154
 Mcnemar's Test P-Value : 0.2770684
```

```
Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9142857  0.34285714   0.5500000
Specificity           0.9639640  0.86486486   0.8113208
Pos Pred Value        0.8888889  0.44444444   0.5238095
Neg Pred Value        0.9727273  0.80672269   0.8269231
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.2191781  0.08219178   0.1506849
Detection Prevalence  0.2465753  0.18493151   0.2876712
Balanced Accuracy     0.9391248  0.60386100   0.6806604
                     Class: van
Sensitivity           0.9166667
Specificity           0.9272727
Pos Pred Value        0.8048780
Neg Pred Value        0.9714286
Prevalence            0.2465753
Detection Rate        0.2260274
Detection Prevalence  0.2808219
Balanced Accuracy     0.9219697
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   32    0    2   2
      opel   1   16   12   3
      saab   1   17   23   0
      van    1    2    3  31
```

```
Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5975946
 Mcnemar's Test P-Value : 0.4542528

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9142857   0.4571429   0.5750000
Specificity           0.9639640   0.8558559   0.8301887
Pos Pred Value        0.8888889   0.5000000   0.5609756
Neg Pred Value        0.9727273   0.8333333   0.8380952
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2191781   0.1095890   0.1575342
Detection Prevalence  0.2465753   0.2191781   0.2808219
Balanced Accuracy     0.9391248   0.6564994   0.7025943
                     Class: van
Sensitivity           0.8611111
Specificity           0.9454545
Pos Pred Value        0.8378378
Neg Pred Value        0.9541284
Prevalence            0.2465753
Detection Rate        0.2123288
Detection Prevalence  0.2534247
Balanced Accuracy     0.9032828
```

3.7. Test5

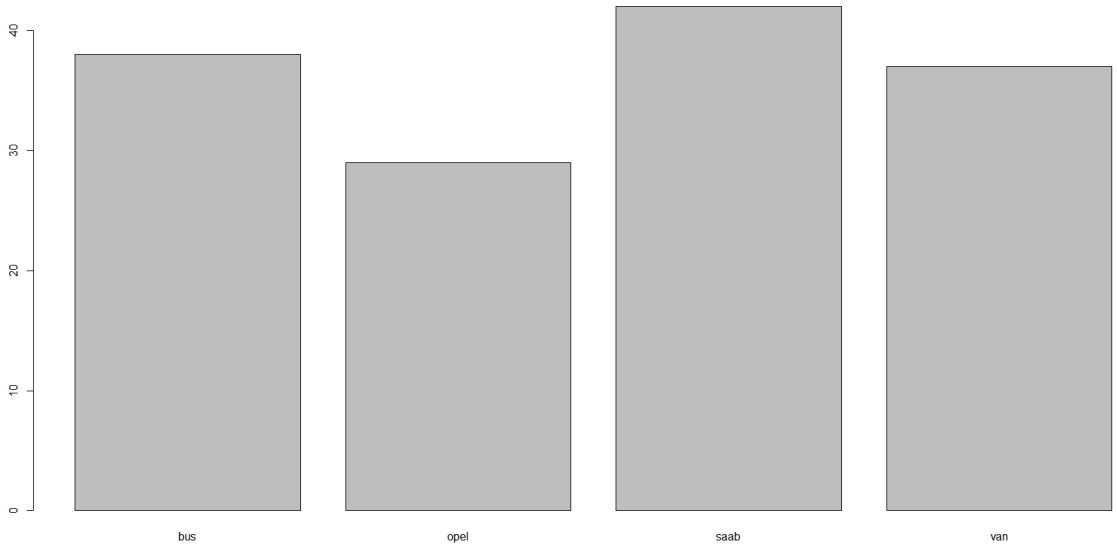
```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,6,7,8,9,10,11,12,19)]
head(Newvehicles)
str(Newvehicles)


#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:12], normalize))

summary(vehiclesN)


vehiclesNnew <- cbind(vehiclesN,random_class[13])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]

#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)
```

:tive 03 latest Test5 norm.R

Source on Save

```
set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Plotting accuracy vs K vlue graph
plot(fit.knn)

#Using selected K values

set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-13]
testset_without_class<-testset[-13]

#Labels
trainset_labels <- vehiclesNnew[1:700, 13]
testset_labels <- vehiclesNnew[701:846, 13]


#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)

#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

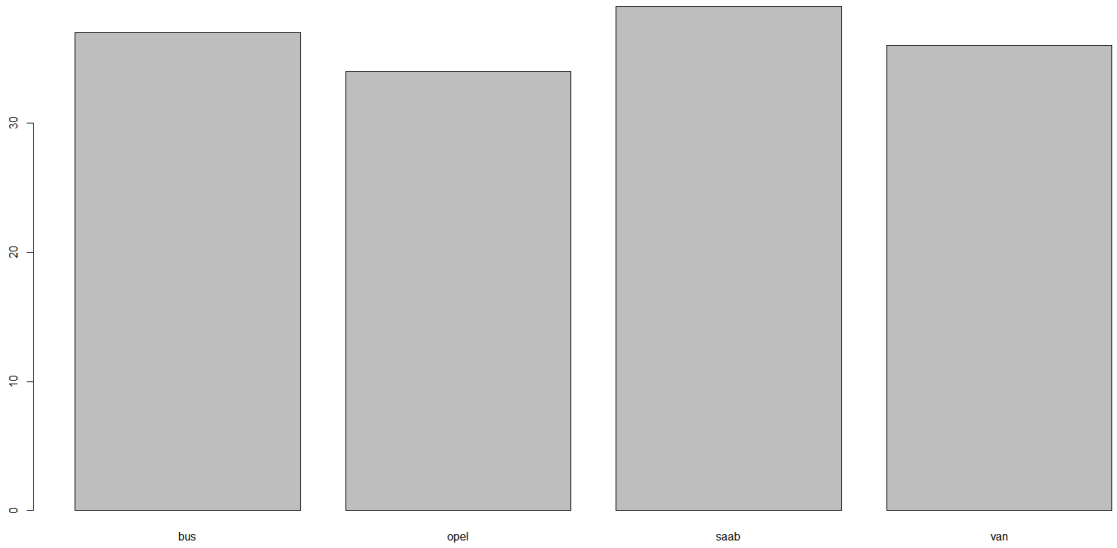
K=3




```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    2   2
      opel   0   10   16   3
      saab   1   20   21   0
      van    1    4    1  31

Overall Statistics

               Accuracy : 0.6506849
                 95% CI : (0.5674717, 0.7276452)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5334294
 Mcnemar's Test P-Value : 0.7763621

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571  0.28571429   0.5250000
Specificity           0.9549550  0.82882883   0.8018868
Pos Pred Value        0.8684211  0.34482759   0.5000000
Neg Pred Value        0.9814815  0.78632479   0.8173077
Prevalence            0.2397260  0.23972603   0.2739726
Detection Rate        0.2260274  0.06849315   0.1438356
Detection Prevalence  0.2602740  0.19863014   0.2876712
Balanced Accuracy     0.9489060  0.55727156   0.6634434
                     Class: van
Sensitivity           0.8611111
Specificity           0.9454545
```

```
Pos Pred Value        0.8378378
Neg Pred Value        0.9541284
Prevalence            0.2465753
Detection Rate        0.2123288
Detection Prevalence  0.2534247
Balanced Accuracy     0.9032828
```

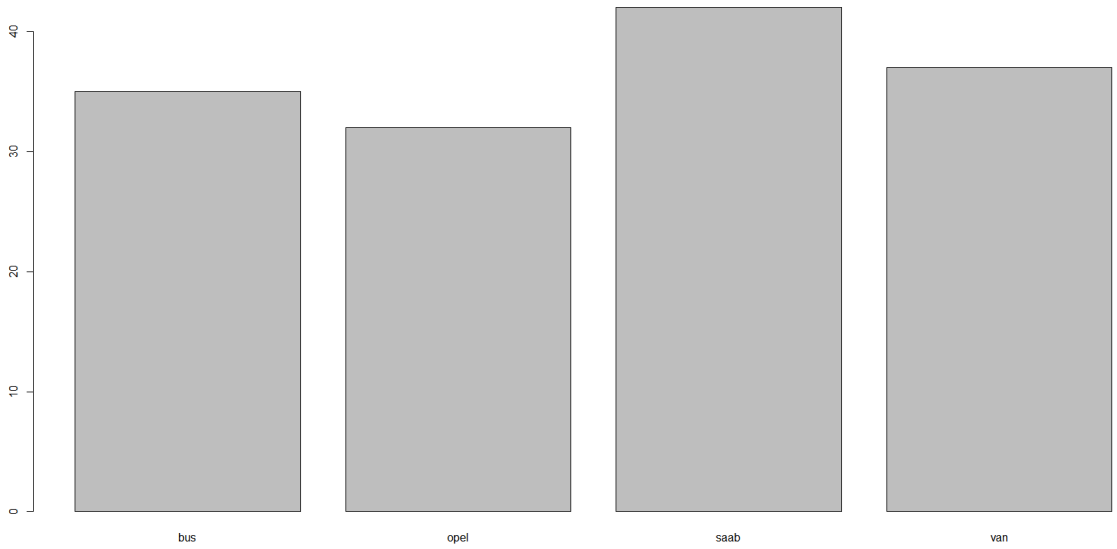
K=5




```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    2   1
      opel   0   17   16   1
      saab   1   16   21   1
      van    1    1    1  33

Overall Statistics

               Accuracy : 0.7123288
                 95% CI : (0.6316608, 0.7841615)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.6161502
 Mcnemar's Test P-Value : 0.9697879

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571   0.4857143   0.5250000
Specificity           0.9639640   0.8468468   0.8301887
```

```
Pos Pred Value         0.8918919   0.5000000   0.5384615
Neg Pred Value         0.9816514   0.8392857   0.8224299
Prevalence             0.2397260   0.2397260   0.2739726
Detection Rate         0.2260274   0.1164384   0.1438356
Detection Prevalence   0.2534247   0.2328767   0.2671233
Balanced Accuracy      0.9534106   0.6662806   0.6775943
                     Class: van
Sensitivity            0.9166667
Specificity            0.9727273
Pos Pred Value         0.9166667
Neg Pred Value         0.9727273
Prevalence             0.2465753
Detection Rate         0.2260274
Detection Prevalence   0.2465753
Balanced Accuracy      0.9446970
```

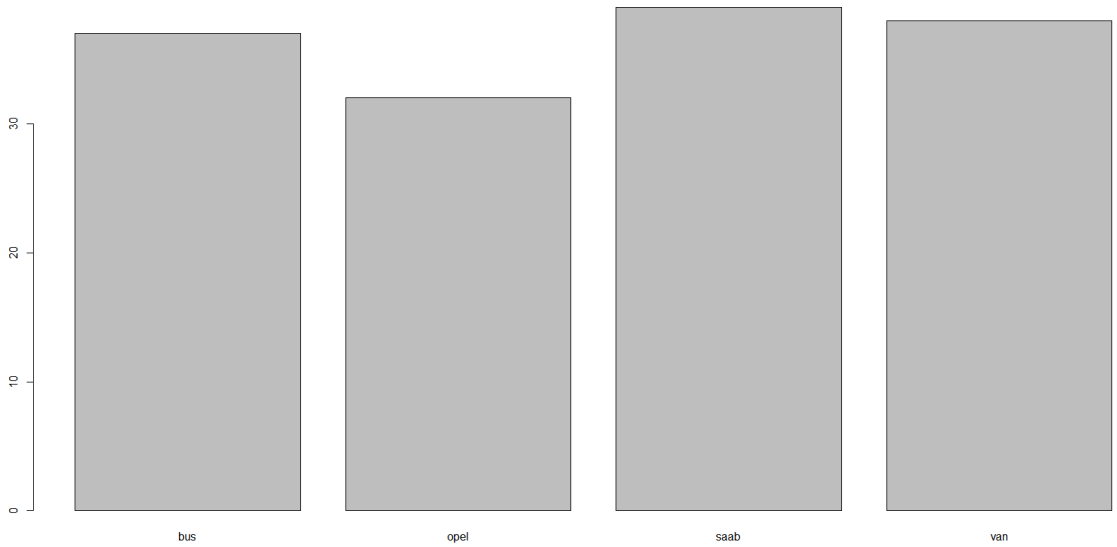
K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   32    0    2   1
      opel   1   15   13   3
      saab   1   18   23   0
      van    1    2    2  32

Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022
```

```
                 Kappa : 0.5974685
 Mcnemar's Test P-Value : 0.6307974

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9142857   0.4285714   0.5750000
Specificity           0.9729730   0.8468468   0.8207547
Pos Pred Value        0.9142857   0.4687500   0.5476190
Neg Pred Value        0.9729730   0.8245614   0.8365385
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2191781   0.1027397   0.1575342
Detection Prevalence  0.2397260   0.2191781   0.2876712
Balanced Accuracy     0.9436293   0.6377091   0.6978774
                     Class: van
Sensitivity           0.8888889
Specificity           0.9545455
Pos Pred Value        0.8648649
Neg Pred Value        0.9633028
Prevalence            0.2465753
Detection Rate        0.2191781
Detection Prevalence  0.2534247
Balanced Accuracy     0.9217172
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    0    2   2
      opel   0   15   15   2
      saab   1   18   20   0
      van    1    2    3  32
```

```
Overall Statistics

               Accuracy : 0.6849315
                 95% CI : (0.6029484, 0.7592301)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022204

                  Kappa : 0.5795405
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: bus Class: opel Class: saab
Sensitivity           0.9428571   0.4285714   0.5000000
Specificity           0.9639640   0.8468468   0.8207547
Pos Pred Value        0.8918919   0.4687500   0.5128205
Neg Pred Value        0.9816514   0.8245614   0.8130841
Prevalence            0.2397260   0.2397260   0.2739726
Detection Rate        0.2260274   0.1027397   0.1369863
Detection Prevalence  0.2534247   0.2191781   0.2671233
Balanced Accuracy     0.9534106   0.6377091   0.6603774
                     Class: van
Sensitivity           0.8888889
Specificity           0.9454545
Pos Pred Value        0.8421053
Neg Pred Value        0.9629630
Prevalence            0.2465753
Detection Rate        0.2191781
Detection Prevalence  0.2602740
Balanced Accuracy     0.9171717
```

3.8. Test6

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

Objective 03 latest Test2 norm.R | Objective 03 latest Test3 norm.R | Objective 03 latest Test4 norm.R | Objective 03 lat

Source on Save

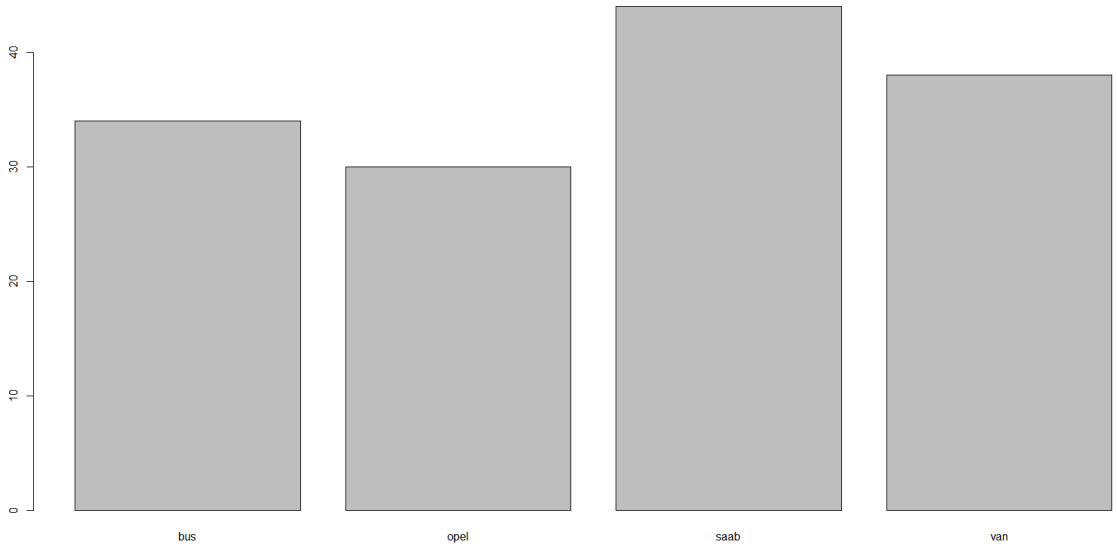
```r
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,19)]
head(Newvehicles)
str(Newvehicles)

#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')


#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:15], normalize))

summary(vehiclesN)

vehiclesNnew <- cbind(vehiclesN,random_class[16])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
```

68:1 (Top Level)

```
#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]

#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)

summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Plotting accuracy vs K vlue graph
plot(fit.knn)

#Using selected K values

set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-16]
testset_without_class<-testset[-16]

#Labels
trainset_labels <- vehiclesNnew[1:700, 16]
testset_labels <- vehiclesNnew[701:846, 16]


#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)
```

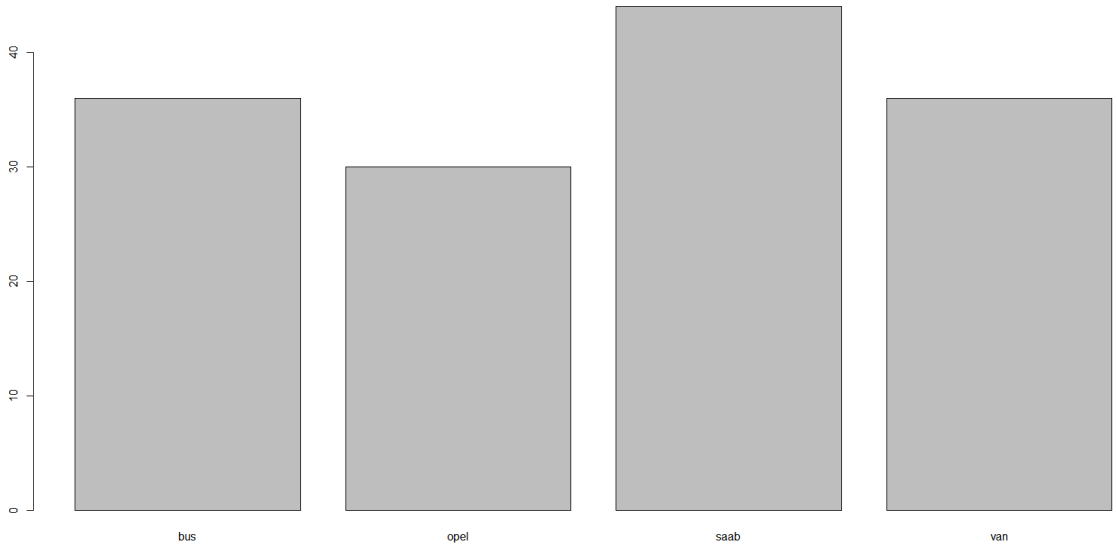
```
#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)

#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

K=3



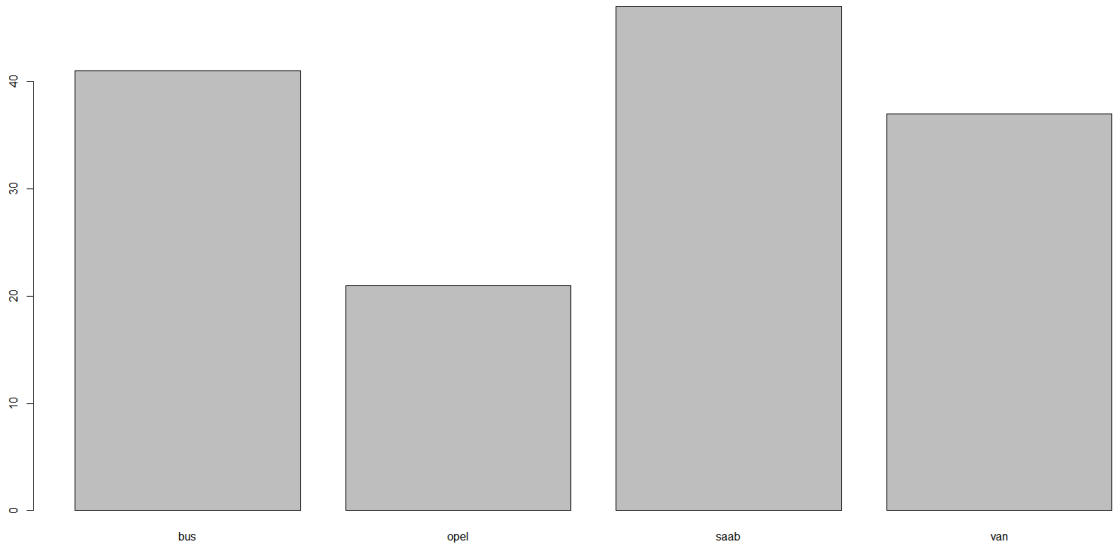

```
> # Confusion Test and matrix
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   31    0    1   2
      opel   2   17   11   0
      saab   2   14   27   1
      van    0    4    1  33

Overall Statistics

               Accuracy : 0.739726
                 95% CI : (0.6606922, 0.8087674)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.6521194
 Mcnemar's Test P-Value : 0.1915734

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.8857143   0.4857143   0.6750000  0.9166667
Specificity           0.9729730   0.8828829   0.8396226  0.9545455
Pos Pred Value        0.9117647   0.5666667   0.6136364  0.8684211
Neg Pred Value        0.9642857   0.8448276   0.8725490  0.9722222
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2123288   0.1164384   0.1849315  0.2260274
Detection Prevalence  0.2328767   0.2054795   0.3013699  0.2602740
Balanced Accuracy     0.9293436   0.6842986   0.7573113  0.9356061
```

K=5



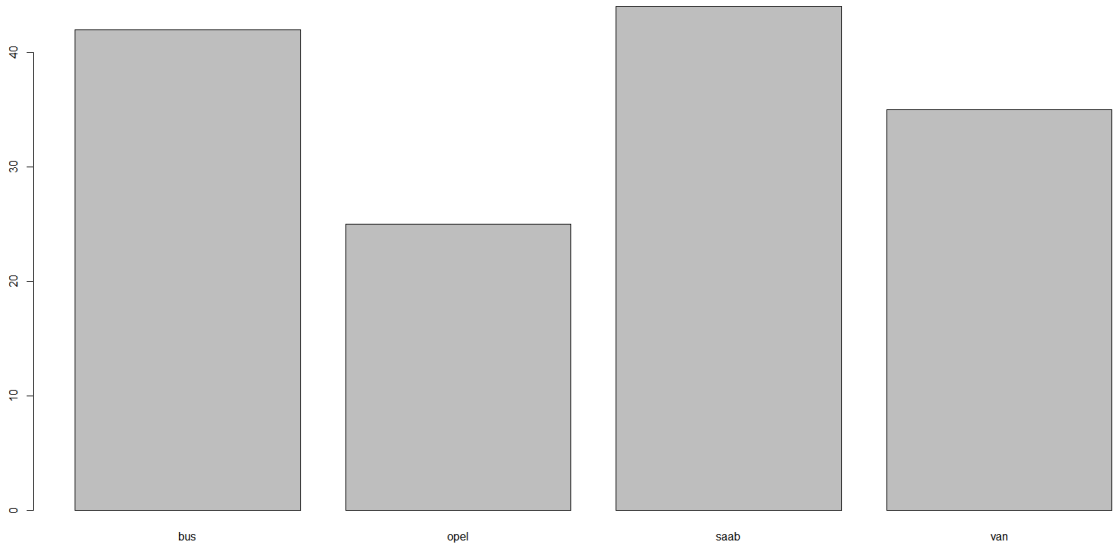

```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   31    0    3   2
      opel   1   21    8   0
      saab   3   12   27   2
      van    0    2    2  32

Overall Statistics

               Accuracy : 0.760274
                 95% CI : (0.6826931, 0.8269892)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.6796238
 Mcnemar's Test P-Value : 0.4459632

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.8857143   0.6000000   0.6750000  0.8888889
Specificity           0.9549550   0.9189189   0.8396226  0.9636364
Pos Pred Value        0.8611111   0.7000000   0.6136364  0.8888889
Neg Pred Value        0.9636364   0.8793103   0.8725490  0.9636364
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2123288   0.1438356   0.1849315  0.2191781
Detection Prevalence  0.2465753   0.2054795   0.3013699  0.2465753
Balanced Accuracy     0.9203346   0.7594595   0.7573113  0.9262626
```

K=7




```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    2    2   4
      opel   0   16    5   0
      saab   2   13   31   1
      van    0    4    2  31

Overall Statistics

               Accuracy : 0.760274
                 95% CI : (0.6826931, 0.8269892)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.000000000000000222

                  Kappa : 0.6793021
 Mcnemar's Test P-Value : 0.03090227

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9428571   0.4571429   0.7750000  0.8611111
Specificity           0.9279279   0.9549550   0.8490566  0.9454545
Pos Pred Value        0.8048780   0.7619048   0.6595745  0.8378378
Neg Pred Value        0.9809524   0.8480000   0.9090909  0.9541284
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2260274   0.1095890   0.2123288  0.2123288
Detection Prevalence  0.2808219   0.1438356   0.3219178  0.2534247
Balanced Accuracy     0.9353925   0.7060489   0.8120283  0.9032828
```

K=9



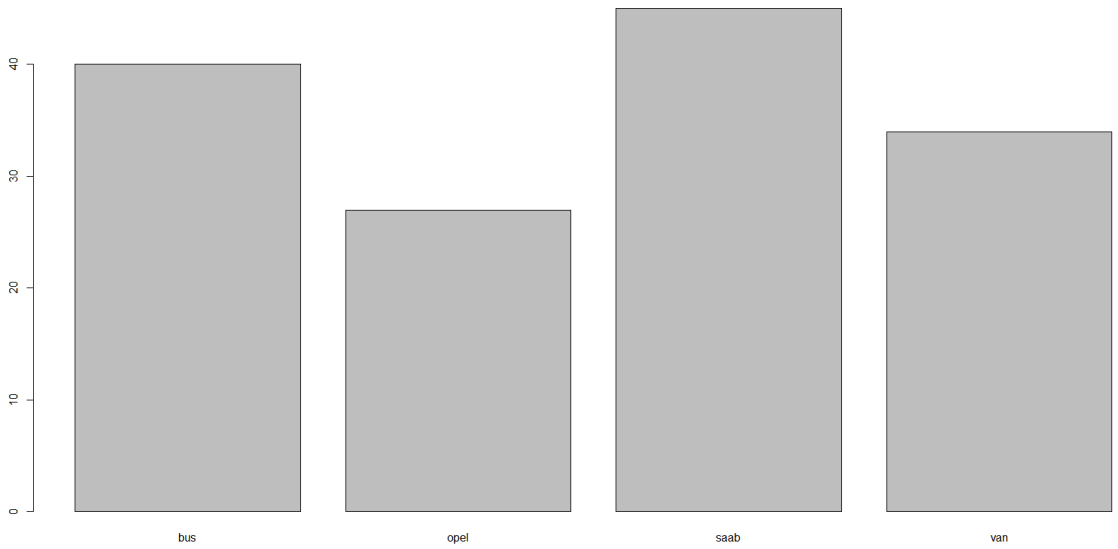

```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    1    3   5
      opel   0   16    8   1
      saab   2   15   27   0
      van    0    3    2  30

Overall Statistics

               Accuracy : 0.7260274
                 95% CI : (0.6461349, 0.796507)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.000000000000000222

                  Kappa : 0.6338788
 Mcnemar's Test P-Value : 0.07868527

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9428571   0.4571429   0.6750000  0.8333333
Specificity           0.9189189   0.9189189   0.8396226  0.9545455
Pos Pred Value        0.7857143   0.6400000   0.6136364  0.8571429
Neg Pred Value        0.9807692   0.8429752   0.8725490  0.9459459
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2260274   0.1095890   0.1849315  0.2054795
Detection Prevalence  0.2876712   0.1712329   0.3013699  0.2397260
Balanced Accuracy     0.9308880   0.6880309   0.7573113  0.8939394
```

3.9. Test7

```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)
original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1:19)]
head(Newvehicles)
str(Newvehicles)

#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)

#how the species are spread
random_class %>% ggvis(~Comp, ~Circ, fill = ~factor(Class)) %>%  layer_points()

#Check whether data is uniformely distributed in plot
melted_class = melt(random_class) #melt function
tail(melted_class)
qplot(x=value, data=melted_class) + facet_wrap(~variable, scales='free')

#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:18], normalize))

summary(vehiclesN)


vehiclesNnew <- cbind(vehiclesN,random_class[19])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]

#dimension checking of trin and test dataset
dim(trainset);
dim(testset);
```

```
#Check for NULL values in normalized data
anyNA(vehiclesNnew)

summary(vehiclesNnew)

#Using selected K values

set.seed(1234)

#Removing Class Attribute for training and testing data
trainset_without_class<-trainset[-19]
testset_without_class<-testset[-19]

#Labels
trainset_labels <- vehiclesNnew[1:700, 19]
testset_labels <- vehiclesNnew[701:846, 19]

#k=3
knn_prediction1 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=3)
CrossTable(x=testset_labels, y=knn_prediction1,prop.chisq=FALSE)
plot(knn_prediction1)

#k=5
knn_prediction2 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=5)
CrossTable(x=testset_labels, y=knn_prediction2,prop.chisq=FALSE)
plot(knn_prediction2)

#k=7
knn_prediction3 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=7)
CrossTable(x=testset_labels, y=knn_prediction3,prop.chisq=FALSE)
plot(knn_prediction3)

#k=9
knn_prediction4 <- knn(train = trainset_without_class, test = testset_without_class,cl = trainset_labels, k=9)
CrossTable(x=testset_labels, y=knn_prediction4,prop.chisq=FALSE)
plot(knn_prediction4)

# Confusion Test and matrix
confusionMatrix(knn_prediction1, testset_labels)
confusionMatrix(knn_prediction2, testset_labels)
confusionMatrix(knn_prediction3, testset_labels)
confusionMatrix(knn_prediction4, testset_labels)
```

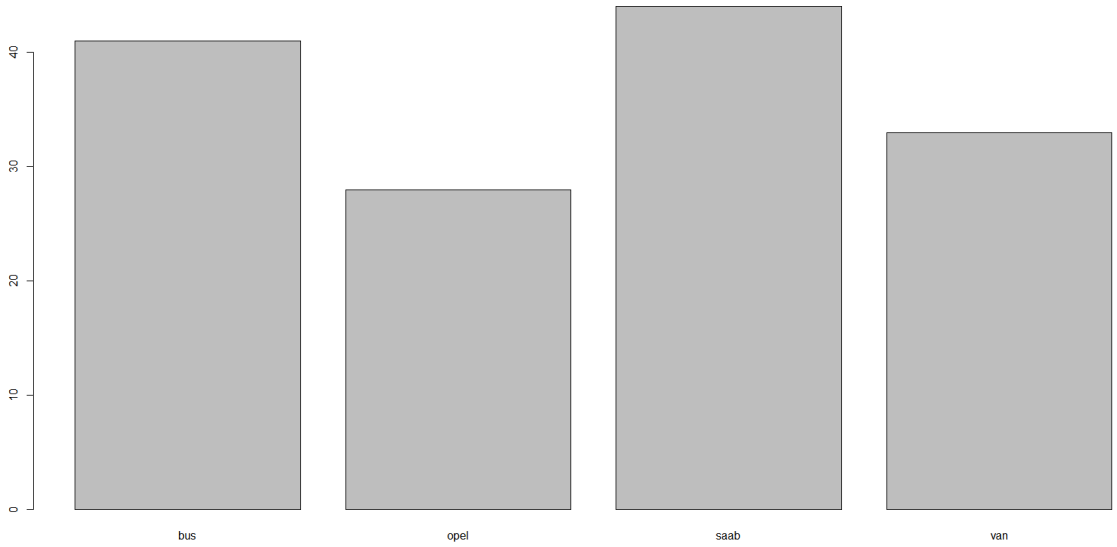
K=3




```
> # Confusion Test and matrix
```

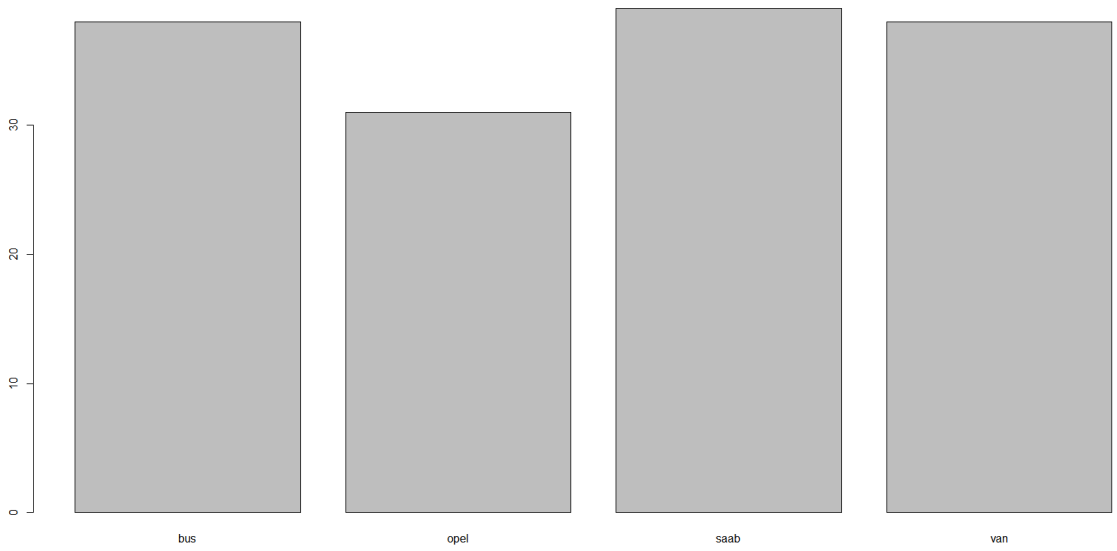
```
> confusionMatrix(knn_prediction1, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   34    0    3   3
      opel   1   13   10   3
      saab   0   18   26   1
      van    0    4    1  29

Overall Statistics

               Accuracy : 0.6986301
                 95% CI : (0.6172663, 0.7717348)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5971656
 Mcnemar's Test P-Value : 0.1508712

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9714286   0.3714286   0.6500000  0.8055556
Specificity           0.9459459   0.8738739   0.8207547  0.9545455
Pos Pred Value        0.8500000   0.4814815   0.5777778  0.8529412
Neg Pred Value        0.9905660   0.8151261   0.8613861  0.9375000
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2328767   0.0890411   0.1780822  0.1986301
Detection Prevalence  0.2739726   0.1849315   0.3082192  0.2328767
Balanced Accuracy     0.9586873   0.6226512   0.7353774  0.8800505
```

K=5



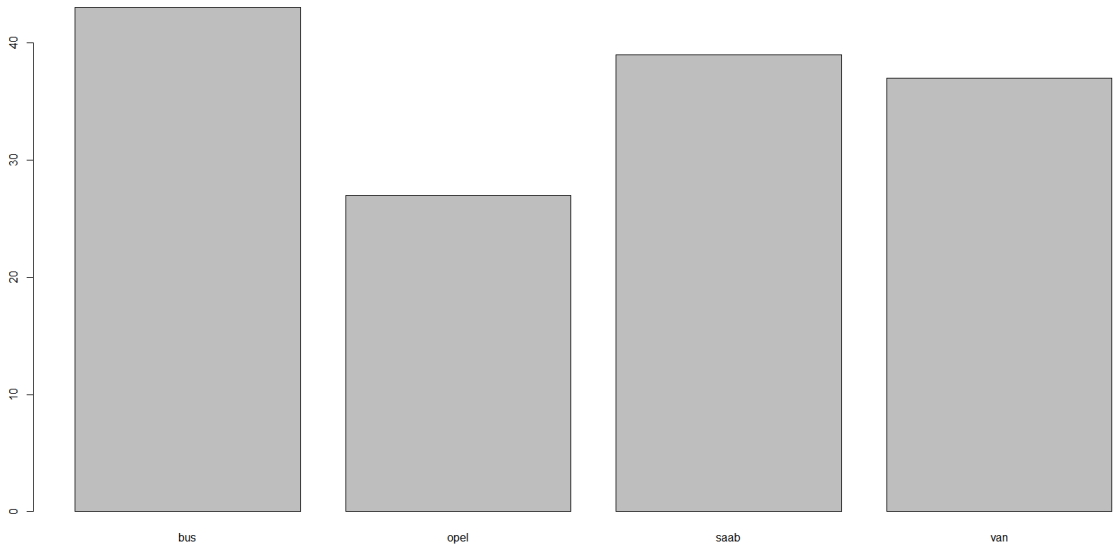
```
> confusionMatrix(knn_prediction2, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   34    0    3   4
      opel   1   16   11   0
      saab   0   17   24   3
      van    0    2    2  29

Overall Statistics

               Accuracy : 0.7054795
                 95% CI : (0.6244538, 0.7779581)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.000000000000000222

                  Kappa : 0.606469
 Mcnemar's Test P-Value : 0.07447582

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9714286   0.4571429   0.6000000  0.8055556
Specificity           0.9369369   0.8918919   0.8113208  0.9636364
Pos Pred Value        0.8292683   0.5714286   0.5454545  0.8787879
Neg Pred Value        0.9904762   0.8389831   0.8431373  0.9380531
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2328767   0.1095890   0.1643836  0.1986301
Detection Prevalence  0.2808219   0.1917808   0.3013699  0.2260274
Balanced Accuracy     0.9541828   0.6745174   0.7056604  0.8845960
```

K=7



```
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   32    1    2   3
      opel   2   14   13   2
      saab   0   16   21   2
      van    1    4    4  29

Overall Statistics

               Accuracy : 0.6575342
                 95% CI : (0.5745318, 0.7339979)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5429788
 Mcnemar's Test P-Value : 0.5467649

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9142857  0.40000000   0.5250000  0.8055556
Specificity           0.9459459  0.84684685   0.8301887  0.9181818
Pos Pred Value        0.8421053  0.45161290   0.5384615  0.7631579
Neg Pred Value        0.9722222  0.81739130   0.8224299  0.9351852
Prevalence            0.2397260  0.23972603   0.2739726  0.2465753
Detection Rate        0.2191781  0.09589041   0.1438356  0.1986301
Detection Prevalence  0.2602740  0.21232877   0.2671233  0.2602740
Balanced Accuracy     0.9301158  0.62342342   0.6775943  0.8618687
```

K=9




```
> confusionMatrix(knn_prediction4, testset_labels)
Confusion Matrix and Statistics
```

```
          Reference
Prediction bus opel saab van
      bus   35    2    2   4
      opel   0   14   11   2
      saab   0   15   23   1
      van    0    4    4  29

Overall Statistics

               Accuracy : 0.6917808
                 95% CI : (0.6100979, 0.765492)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.00000000000000022

                  Kappa : 0.5887066
 Mcnemar's Test P-Value : 0.0858734

Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity           1.0000000  0.40000000   0.5750000  0.8055556
Specificity           0.9279279  0.88288288   0.8490566  0.9272727
Pos Pred Value        0.8139535  0.51851852   0.5897436  0.7837838
Neg Pred Value        1.0000000  0.82352941   0.8411215  0.9357798
Prevalence            0.2397260  0.23972603   0.2739726  0.2465753
Detection Rate        0.2397260  0.09589041   0.1575342  0.1986301
Detection Prevalence  0.2945205  0.18493151   0.2671233  0.2534247
Balanced Accuracy     0.9639640  0.64144144   0.7120283  0.8664141
```

3.10. Experiment with various K values for different input options

The highest accuracy was obtained with the use of 15 attributes. Hence, cross validation and confusion matrix was created for 15 attributes.

K	No. of attributes						
	3	**5**	**8**	**10**	**12**	**15**	**18**
3	Kappa : 0.3768516 Accuracy : 0.5342466	Kappa : 0.5141583 Accuracy : 0.6369863	Kappa : 0.597645 Accuracy : 0.6986301	Kappa : 0.5704538 Accuracy : 0.6780822	Kappa : 0.5334294 Accuracy : 0.6506849	Kappa : 0.6521194 Accuracy : 0.739726	Kappa : 0.5971656 Accuracy : 0.6986301
5	Kappa : 0.4592253 Accuracy : 0.5958904	Kappa : 0.4698554 Accuracy	Kappa : 0.6068633 Accuracy	Kappa : 0.5973424 Accuracy	Kappa : 0.6161502 Accuracy	Kappa : 0.6796238 Accuracy	Kappa : 0.606469

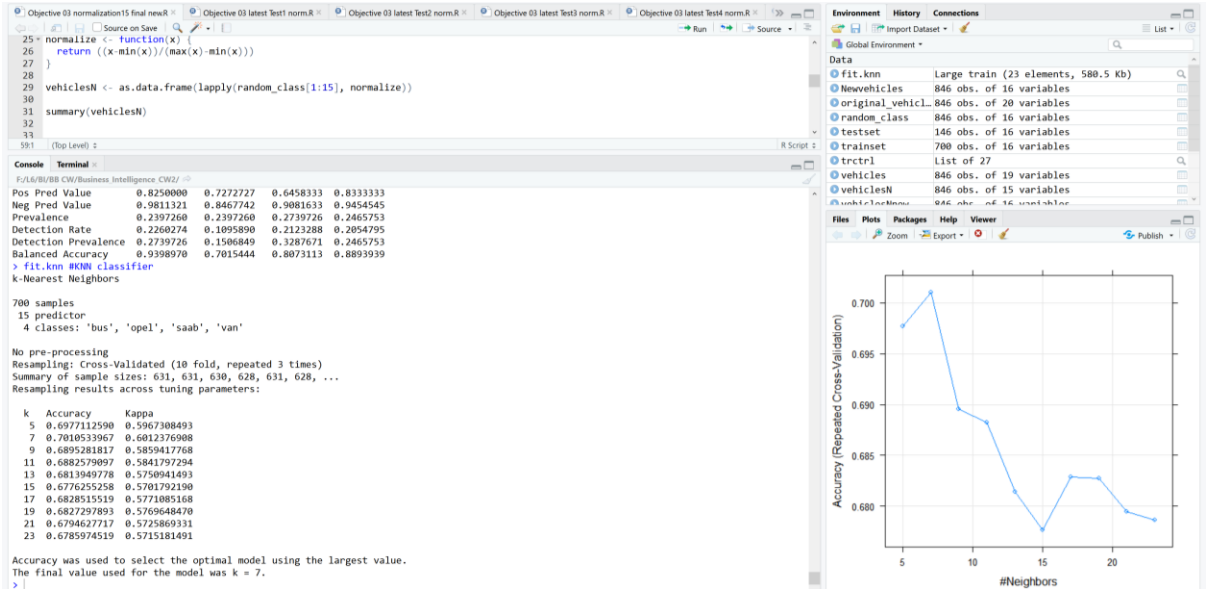
		: 0.6027397	: 0.7054795	: 0.6986301	: 0.7123288	: 0.760274	Accuracy : 0.7054795
7	Kappa : 0.4506115 Accuracy : 0.5890411	Kappa : 0.5336632 Accuracy : 0.6506849	Kappa : 0.5978717 Accuracy : 0.6986301	Kappa : 0.5699154 Accuracy : 0.6780822	Kappa : 0.5974685 Accuracy : 0.6986301	Kappa : 0.6793021 Accuracy : 0.760274	Kappa : 0.5429788 Accuracy : 0.6575342
9	Kappa : 0.4512997 Accuracy : 0.5890411	Kappa : 0.5341009 Accuracy : 0.6506849	Kappa : 0.6160301 Accuracy : 0.7123288	Kappa : 0.5975946 Accuracy : 0.6986301	Kappa : 0.5795405 Accuracy : 0.6849315	Kappa : 0.6338788 Accuracy : 0.7260274	Kappa : 0.5887066 Accuracy : 0.6917808

Code for 15 attributes

```
library(class)
library(gmodels)
library(ggvis)
library(caret)

set.seed(1234)

original_vehicles<-read.csv("vehicles.csv")
#Remove 1st column
vehicles<- original_vehicles[2:20]

#Dataset with Class column ane selected columns
Newvehicles<-vehicles[,c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,19)]
head(Newvehicles)
str(Newvehicles)


#Shuffling data
random_class<-Newvehicles[sample(1:nrow(Newvehicles)), ]
head(random_class) #row no. are displayed

str(random_class)
```

```
#Normalization for randomized numerrical columns
normalize <- function(x) {
  return ((x-min(x))/(max(x)-min(x)))
}

vehiclesN <- as.data.frame(lapply(random_class[1:15], normalize))

summary(vehiclesN)

vehiclesNnew <- cbind(vehiclesN,random_class[16])

#Data partioning
trainset<-vehiclesNnew[1:700, ]
testset<-vehiclesNnew[701:846,]

#dimension checking of trin and test dataset
dim(trainset);
dim(testset);

#Check for NULL values in normalized data
anyNA(vehiclesNnew)

summary(vehiclesNnew)

#Training and preprocessing using caret
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

set.seed(1234)
fit.knn <- train(Class ~., data = trainset, method = "knn",
                 trControl =trctrl,
                 tuneLength = 10) #cross validation

fit.knn #KNN classifier

#Plotting accuracy vs K vlue graph
plot(fit.knn)

#Predict class for testset
set.seed(1234)
test_pred <- predict(fit.knn, newdata = testset)
#Check accuracy with statistics
confusionMatrix(test_pred, testset$Class)
```

KNN Result with Cross Validation Method



3.11. K-NN Results Discussion

When it comes to the results discussion the evaluation process can be carried out. When fit.knn is to be evaluated a further observation can be further drawn that the accuracy along with the kappa factors have significantly changed according to the various K values. Finally, a conclusion can be draw saying that the highest accuracy is gained when k=7 for 15 attributes.

3.12. Confusion Matrix Results

Below are the results for 15 attributes when K=7

```
> confusionMatrix(knn_prediction3, testset_labels)
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   33    2    2   4
      opel   0   16    5   0
      saab   2   13   31   1
      van    0    4    2  31

Overall Statistics

               Accuracy : 0.760274
                 95% CI : (0.6826931, 0.8269892)
    No Information Rate : 0.2739726
    P-Value [Acc > NIR] : < 0.000000000000000222

                  Kappa : 0.6793021
 Mcnemar's Test P-Value : 0.03090227

Statistics by Class:
```

```
                     Class: bus Class: opel Class: saab Class: van
Sensitivity           0.9428571   0.4571429   0.7750000  0.8611111
Specificity           0.9279279   0.9549550   0.8490566  0.9454545
Pos Pred Value        0.8048780   0.7619048   0.6595745  0.8378378
Neg Pred Value        0.9809524   0.8480000   0.9090909  0.9541284
Prevalence            0.2397260   0.2397260   0.2739726  0.2465753
Detection Rate        0.2260274   0.1095890   0.2123288  0.2123288
Detection Prevalence  0.2808219   0.1438356   0.3219178  0.2534247
Balanced Accuracy     0.9353925   0.7060489   0.8120283  0.9032828
```

3.13. Cross Validation Results

Different training sessions were carried out and looked at to generate k-NN with the help of data partitioning.

```
sss
36  #Data partioning
37  trainset<-vehiclesNnew[1:700, ]
38  testset<-vehiclesNnew[701:846,]
```

Once the code was executed cross validation was carried out to validate the results.

```
49  #Training and preprocessing using caret
50  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
51
52  set.seed(1234)
53  fit.knn <- train(Class ~., data = trainset, method = "knn",
54                   trControl =trctrl,
55                   tuneLength = 10) #cross validation
56
```

Note: The above code is run during the process of cross validation.

It can be seen that 10 times the fold cross validation process has been carried out.

Once the code is executed the cross validation would then be displayed in the fit.knn result

```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 631, 631, 630, 628, 631, 628, ...
Resampling results across tuning parameters:
```

REFERENCES

1) Frank, R. J., Davey, N. and Hunt, S. P. (2001) *Time Series Prediction and Neural Networks*, Journal of Intelligent and Robotic Systems. Available at: https://link.springer.com/content/pdf/10.1023%2FA%3A1012074215150.pdf (Accessed: 16 December

2) Jason Brownlee (2016) *Time Series Forecasting as Supervised Learning*. Available at: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/ (Accessed: 16 December 2018).

3) Johansson, S. and Nafar, S. (2017) *Effective Sampling and Windowing for an Artificial Neural Network Model Used in Currency Exchange Rate Forecasting SHAYAN NAFAR KTH ROYAL INSTITUTE OF TECHNOLOGY SCHOOL OF ENGINEERING SCIENCES*. Available at: www.kth.se (Accessed: 16 December 2018).

4) Nowrouz Kohzadi *et al.* (1996) 'A comparison of artificial neural network and time series models for forecasting commodity prices', *Neurocomputing*, 10, pp. 169–181. Available at: https://ac.els-cdn.com/0925231295000208/1-s2.0-0925231295000208-main.pdf?_tid=0f154d69-8770-46b2-966f-b354ff925c44&acdnat=1544970538_321e7e1196f9db8e19aa3f036a0d8f39 (Accessed: 16 December 2018).

5) Tran, H. D., Muttil, N. and Perera, B. J. C. (2015) 'Selection of significant input variables for time series forecasting', *Environmental Modelling & Software*, 64, pp. 156–163. doi: 10.1016/j.envsoft.2014.11.018.