**Implement a KNN model to classify the animals in to categories.**

**Logic:** This problem statement can be solved using the KNN algorithm that will classify the

animals in to categories.

**Algorithm:**

* Calculate the distance from x to all points in zoo data set.
* Sort the points in zoo data by increasing distance from x.
* Predict the majority label of the k closest points.

Note that the value of k effects the results, its ideal to test the model for different values of k for better results and there by a better model.

**Data**

**Zoo** **Identification Database**  contains 18 attributes/variables. The response is Zoo type which has 7 values.

table(Zoo\_data1$type)

1 2 3 4 5 6 7

41 20 5 13 4 8 10

**Import the dataset**

After importing the dataset, let’s take a look at the structure of the dataset:

The **Zoo** data set contain 101 observations and 18 variables

Note that, the ‘Type’ variable is our output variable or the target variable. The value of the Type variable represents **Zoo** classification.

**Data Normalization**

Always normalize the data set so that the output remains unbiased. To explain this, let’s take a look at the first few observations in our data set.

> head(Zoo\_data)

animal.name hair feathers eggs milk airborne aquatic predator toothed backbone

1 aardvark 1 0 0 1 0 0 1 1 1

2 antelope 1 0 0 1 0 0 0 1 1

3 bass 0 0 1 0 0 1 1 1 1

4 bear 1 0 0 1 0 0 1 1 1

5 boar 1 0 0 1 0 0 1 1 1

6 buffalo 1 0 0 1 0 0 0 1 1

breathes venomous fins legs tail domestic catsize type

1 1 0 0 4 0 0 1 1

2 1 0 0 4 1 0 1 1

3 0 0 1 0 1 0 0 4

4 1 0 0 4 0 0 1 1

5 1 0 0 4 1 0 1 1

6 1 0 0 4 1 0 1 1

**# create normalization function**

normalize <- function(x) {

return ((x - min(x)) / (max(x) - min(x)))

}

In the below code , we’re storing the normalized data set in the ‘Zoo\_data1\_n’ variable and also we’re removing the ‘Type’ variable since it’s the response variable that needs to be predicted.

Zoo\_data1\_n <- as.data.frame(lapply(Zoo\_data1[1:16], normalize))

**Data Splicing**

After cleaning the data set and formatting it, the next step is data splicing. Data splicing basically involves splitting the data set into training and testing data set.

**#random sampling**

#random sampling

n <- nrow(Zoo\_data1\_n)

n1 <- n\*0.8

n2 <- n-n1

train\_index <- sample(1:n,n1)

zoo\_train <- Zoo\_data1[train\_index, ]

zoo\_test <- Zoo\_data1[-train\_index, ]

After deriving the training and testing data set, the below code is going to create a separate data frame for the ‘Type’ variable so that our final outcome can be compared with the actual value.

**#Creating seperate dataframe for 'Type' feature which is our target.**

zoo\_train\_labels <- Zoo\_data1[train\_index,17]

zoo\_test\_labels <- Zoo\_data1[-train\_index,17]

**Building a Machine Learning model**

At this stage, we have to build a model by using the training data set. Since we’re using the KNN algorithm to build the model, we must first install the ‘class’ package provided by R. This package has the KNN function in it:

Predict the the data set using KNN function

zoo\_test\_pred <- knn(train = zoo\_train, test = zoo\_test,

cl = zoo\_train\_labels, k=1)

**#Error in prediction**

error <- mean(zoo\_test\_pred!=zoo\_test\_labels)

**Model Evaluation**

After building the model, it is time to calculate the accuracy of the created models:

We can check the predicted outcome against the actual value in tabular form:

table(zoo\_test\_pred ,zoo\_test\_labels)

> table(zoo\_test\_pred ,zoo\_test\_labels)

zoo\_test\_labels

zoo\_test\_pred 1 2 3 4 6

1 13 0 0 0 0

2 0 3 0 0 0

3 0 0 1 0 0

4 0 0 1 2 0

5 0 0 0 0 0

6 0 0 0 0 1

7 0 0 0 0 0

We can also use the confusion matrix to calculate the accuracy. To do this we must first install the infamous Caret package

Now, let’s use the confusion matrix to calculate the accuracy of the KNN model with **K value set to 1**

confusionMatrix(table(zoo\_test\_pred, zoo\_test\_labels))

Confusion Matrix and Statistics

zoo\_test\_labels

zoo\_test\_pred 1 2 3 4 6

1 13 0 0 0 0

2 0 3 0 0 0

3 0 0 1 0 0

4 0 0 1 2 0

5 0 0 0 0 0

6 0 0 0 0 1

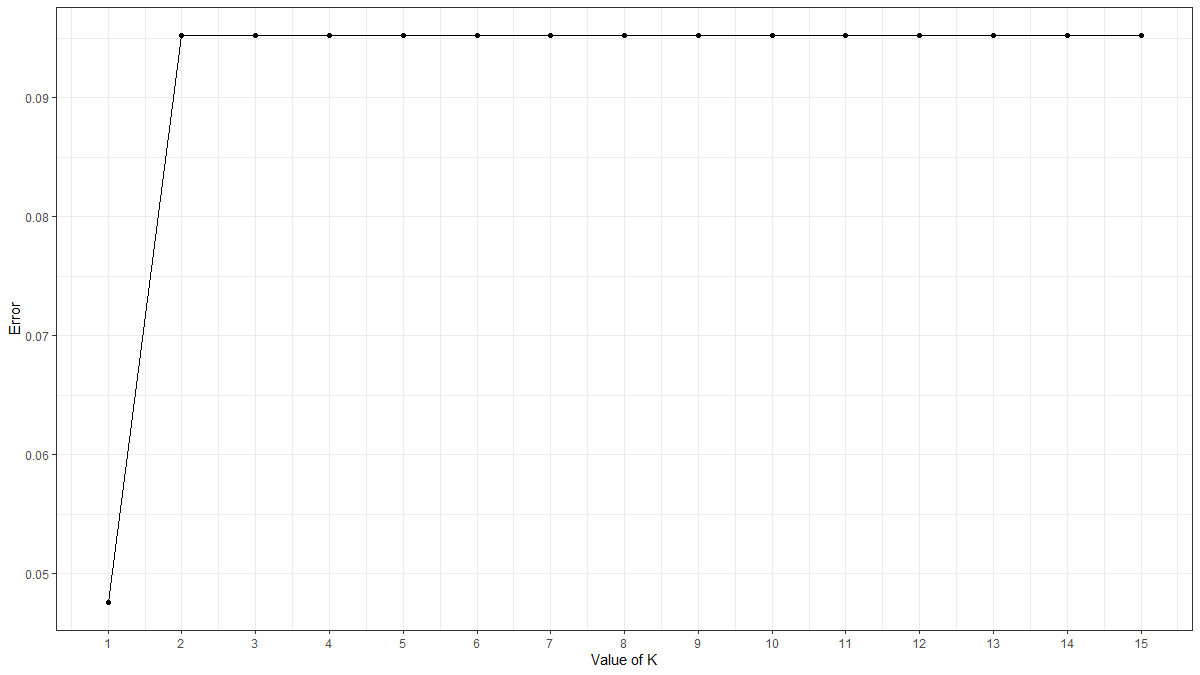
7 0 0 0 0 0

Overall Statistics

Accuracy : 0.9523

So, from the output, we can see that our model predicts the outcome with an accuracy of 95%

In order to improve the accuracy of the model, created a loop that calculates the accuracy of the KNN model for ‘K’ values ranging from 1 to 10. This way you can check which ‘K’ value will result in the most accurate model.



**R-Code:**

Zoo\_data <- read.csv("C:/RAVI/Data science/Assignments/Module 18 KNN/KNN Assignment2 dataset/Zoo.csv/Zoo.csv")

View(Zoo\_data)

attach(Zoo\_data)

# drop the animal.name feature

Zoo\_data1 <- Zoo\_data[ ,2:18]

View(Zoo\_data1)

str(Zoo\_data1)

table(Zoo\_data1$type)

summary(Zoo\_data)

summary(Zoo\_data[c("feathers","toothed","domestic","breathes","tail")])

head(Zoo\_data)

str(Zoo\_data)

#Data Visualization

install.packages('ggplot2') #for Data Visualization

library(ggplot2)

plot(Zoo\_data1)

install.packages('corrplot') #Correlation Plot

library(corrplot)

corrplot(cor(Zoo\_data1))

# create normalization function

normalize <- function(x) {

return ((x - min(x)) / (max(x) - min(x)))

}

Zoo\_data1\_n <- as.data.frame(lapply(Zoo\_data1[1:16], normalize))

Zoo\_data1\_n

summary(Zoo\_data1\_n$aquatic)

# create training and test datasets

#random sampling

n <- nrow(Zoo\_data1\_n)

n1 <- n\*0.8

n2 <- n-n1

train\_index <- sample(1:n,n1)

zoo\_train <- Zoo\_data1[train\_index, ]

zoo\_test <- Zoo\_data1[-train\_index, ]

#Creating seperate dataframe for 'Type' feature which is our target.

zoo\_train\_labels <- Zoo\_data1[train\_index,17]

zoo\_test\_labels <- Zoo\_data1[-train\_index,17]

#---- Training a model on the data ----

#Find the number of observation

NROW(zoo\_train\_labels)

sqrt(80) # k=9

# load the "class" library

install.packages("class") ##KNN

library(class)

zoo\_test\_pred <- knn(train = zoo\_train, test = zoo\_test,

cl = zoo\_train\_labels, k=1)

#Error in prediction

error <- mean(zoo\_test\_pred!=zoo\_test\_labels)

error

install.packages('caret')

library(caret)

##--------Evaluating model performance ----

#Calculate the proportion of correct classification for k = 1

# Check prediction against actual value in tabular form for k=1

table(zoo\_test\_pred ,zoo\_test\_labels)

confusionMatrix(table(zoo\_test\_pred, zoo\_test\_labels))

zoo\_test\_pred <- NULL

error\_rate <- NULL

for (i in 1:15) {

zoo\_test\_pred <- knn(train = zoo\_train, test = zoo\_test,cl = zoo\_train\_labels,k=i)

error\_rate[i] <- mean(zoo\_test\_pred!=zoo\_test\_labels)

}

knn\_error <- as.data.frame(cbind(k=1:15,error\_type =error\_rate))

#K Value by Visualization

install.packages('ggplot2') #for Data Visualization

library(ggplot2)

ggplot(knn\_error,aes(k,error\_type))+

geom\_point()+

geom\_line() +

scale\_x\_continuous(breaks=1:15)+

theme\_bw() +

xlab("Value of K") +

ylab('Error')

zoo\_test\_pred <- knn(train = zoo\_train, test = zoo\_test,

cl = zoo\_train\_labels, k=2)

zoo\_test\_predOO

#Error in prediction

error <- mean(zoo\_test\_pred!=zoo\_test\_labels)

error

confusionMatrix(table(zoo\_test\_pred,zoo\_test\_labels))

#########################################

library(gmodels)

CrossTable(x = zoo\_test\_labels, y = zoo\_test\_pred,

prop.chisq=FALSE)