Data Mining and Machine Learning Lab 4.

Instructions: Create a file called xxxxxxxx.doc where <xxxxxxxx> is your UCD student number. Write your answers in this file and save it to your own computer so you don't lose your answers. Then upload to the moodle before the end of the lab.

At the top of the file, fill in your details below (delete the 'x' where your information goes):

Name: x
BDIC Student Number: x
UCD Student Number: x

Using the k-Nearest Neighbors algorithm (kNN) to predict upcoming generator failures.

1. Load the dataset

Create a folder called Lab4 on your desktop. Download 'generators .csv' from moodle into this folder. Create a text file called lab4-knn.r in this folder. Write the lines of code below into the file and save the file. If you double click on the file it should automatically open in Rstudio. If not try right clicking and select 'open with Rstudio'.

```
rm(list=ls()) #removes (almost) all objects from the current workspace (R memory)

#if you set the seed you will get the same results every time, if not, you will get differ-
ent results every time.
set.seed(123)   #try putting a # in front of this line and run the code a few times to see
what happens, then remove the # and run it a few times to see the difference…

generators <- read.csv('generators.csv')
```

If this does not work try either:

```
> generators <- read.csv('~/Desktop/Lab4/generators.csv')
```

or

```
> generators<-read.csv('~\\Desktop\\Lab4\\generators.csv')
```

Then run the file by typing:

```
> source("lab4-knn.r")
```

You can save all the code you write today in this file and rerun it using the source command. This will be very useful to you for your assignment.

Check that the data has loaded correctly:

```
>  generators
```

This dataset contains measurements of:

1. The revolutions per minute (RPM) that power station generators are running at
2. The amount of vibration in the generators (VIBRATION)
3. An indicator to show whether the generators proved to be working or faulty the day after these measurements were taken (STATUS: good or faulty)

If power station administrators could predict upcoming generator failures before the generators actually fail, they could improve power station safety and save money on maintenance

2. Load the library “class”

We will use the knn function in the package ‘class’ to train a k-nearest neighbors model. The knn() function identifies the k-nearest neighbors using Euclidean distance where k is a user-specified number.

```
> library("class")
```

If it is not installed, install it:

```
> install.packages("class")
```

then load it again:

```
> library("class")
```

3. Get summary statistics and create a Data Quality Report

```
> summary(generators)

> library(Hmisc)

> describe(generators)
```

Are there any data quality issues?

4. Normalise the data

First we will normalise the data that will make up the training and test sets in the range [-1,1].

Write your own normalize function:

```
normalize <- function(x) {
     num <- x - min(x)
     denom <- max(x) - min(x)
     return (2*(num/denom)-1)
}
```

You can add this to your lab4-knn.r script. Use this function to normalise you data:

```
> generators['RPM'] <- as.data.frame(lapply(generators['RPM'], normalize))
> generators['Vibration'] <- as.data.frame(lapply(generators['Vibration'], normalize))
```

Check that the data has been normalised correctly:

```
> summary(generators)
```

5. Convert the Status label into a factor

Factors are variables in R which take on a limited number of different values, often referred to as categorical variables. Storing data as factors insures that the modelling functions will treat such data correctly. Factors in R are stored as a vector of integer values with a corresponding set of character values to use when the factor is displayed. The factor function is used to create a factor.

```
> generators$Status<-factor(generators$Status,labels=c('faulty','good'))
```

6. k-Nearest Neighbors algorithm

Have a look at the knn help file for the knn function. Try to understand what you need to provide for input.

```
> ?knn
```

Usage

knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)

Can you work out what these inputs are?

train	matrix or data frame of training set cases.
test	matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
cl	factor of true classifications of training set
k	number of neighbours considered.

You have to provide the training set, test set, and the classifications of training set all at the same time. For most other prediction algorithms, we build the prediction model on the training set in the first step, and then use the model to test our predictions on the test set in the second step. However, the knn function does both in a single step.

7. Training And Test Sets

In order to assess your model's performance later, you will need to divide the data set into two parts: a training set and a test set. The first is used to train the system, while the second is used to evaluate the predictive model. In this lab we will split the data into a training set (75% of the data) and a test set (25% of the data). The function createDataPartition can be used to create a stratified random sample of the data into training and test sets. See ?createDataPartition for more details of the function

```
> inTraining <- createDataPartition(generators$Status, p = .75, list = FALSE)#create a
training set with 75% of the date
> generators_train <- generators[ inTraining,]#this is the training set
> generators_test  <- generators[-inTraining,]#this is the test set
```

For best results the number of instances of both classes needs to be present at more or less the same ratio in your training and test sets. You can check this using summary():

```
> summary(generators_train)
> summary(generators_test)
```

8. Training a knn model on data

```
> generators_test_pred <- knn(train = generators_train[,2:3], test = generators_test[,2:3],
cl = generators_train[,4], k=10)
```

We can check the accuracy on the test set by creating a confusion matrix:

```
> cm = as.matrix(table(Actual = generators_test[,4], Predicted = generators_test_pred))

> cm
```

9. Model Evaluation

Calculate the following Performance Measures (check how to calculate these from your lecture notes):

classification accuracy

misclassification rate

TPR

TNR

FPR

FNR

precision

recall

F1-measure

average class accuracy

Save the code you have written into your script file (lab4-knn.r) so you can easily re-run the script using different values of k. Also, copy the code into your answer sheet.

10. Model Evaluation with different values of k (Optional, if you have time)

See if you can improve the performance by changing the value of k.

Create a table using a number of the performance measures above and record how the values change as you change the value of k (from k = 3 to k = 10 for example).

What is the best value of k? Why?

11. Please fill out the feedback on todays lab in Moodle