Data Mining and Machine Learning Lab 8.

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 a Support Vector Machine to predict upcoming generator failures.

1. Load the dataset

Create a folder called Lab8 on your desktop. Download 'generators.csv' from moodle into this folder. Create a text file called lab8.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()) #This will remove (almost) everything in the working environment before you start

#if you set the seed you will get the same results every time, if not, you will get different 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/Lab8/generators.csv')
```

or

```
> generators<-read.csv('~\\Desktop\\Lab8\\generators.csv')
```

Then run the file by typing:

```
> source("lab8.r")
```

You can save all the code you write today in this file and rerun it using the source command.

Check that the data has loaded correctly:

```
>  generators
```

This dataset contains measurements of:

- The revolutions per minute (RPM) that power station generators are running at
- The amount of vibration in the generators (VIBRATION)
- 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 “caret”

We will use the train function in the package ‘caret’ to train the SVM model. The caret library does a lot more than just SVM, for more details on caret see https://topepo.github.io/caret/index.html

```
> library('caret')
```

This should have been installed already, if not, install it:

```
> install.packages('caret')
```

then load it again.

3. 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 lab8.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)
```

4. Convert the Status label into a factor

Factors are variables in R which take on a limited number of different values, often re-

ferred 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'))
```

5. 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
> training <- generators[ inTraining,]#this is the training set
> testing  <- 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(training)
> summary(testing)
```

6. Training a SVM model on the data

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

```
> ?train
```

Can you work out what some these inputs are? (This is a bit more difficult than last week!)

First we set up the control parameters for the train function see ?trainControl for more details. Here we are setting up a 10-fold cross-validation training (check your lecture notes if you need to remind yourself what this is):

```
> ctrl <- trainControl(method = "cv", number = 10, search = "grid", savePred=T)
```

The train function uses ctrl to set up a grid of tuning parameters for our SVM. Here we use a 'svmLinear' kernel we could also try a 'svmRadial' or 'svmPoly' for example. See http://topepo.github.io/caret/train-models-by-tag.html#Support_Vector_Machines for a full list of available kernels.

```
> model <- train(training[,2:3], training$Status, method = 'svmLinear', metric =
'Accuracy', trControl = ctrl)

> print(model)
```

We can test our model with the independent test set we created earlier:

```
> pred <- predict(model,testing[,2:3])
```

7. Model Evaluation

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

```
> cm = as.matrix(table(Actual = testing$Status, Predicted = pred))

> print(cm)
```

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

classification accuracy

misclassification rate

TRP

TNR

FPR

FNR

precision

recall

F1-measure

average class accuracy

Save the code you have written into your script file (lab8.r) so you can easily re-run the script using different kernels.

Question 1: Which kernel gives the best performance: 'svmLinear', 'svmRadial' or 'svmPoly'? Copy the results for each kernel here.