INSTRUMENTATION

R coding use to do the SVM

```
>library(kernlab)

>### read data

>mydata<- read.csv('NApo.csv')

>summary(mydata)
```

In R we need to declare the package that we want to use by using the library fuction.

To import data into R we used function read.csv to read the data that we had saved in csv format. Summary function will gives information of the data such as mean, min, mod and other information.

```
### determine the pH selection

>pH<- mydata[1:120, 2,]

>ytrain<-mydata[1:120, 1:1]

>pH<-cbind(pH,pH)

>bestmodel<-ksvm (pH,ytrain,type="C-svc",  kernel='vanilladot', cross=10)

>bestmodel
```

This part is where the run the linear kernel to see the error produce by each parameter to determine the suitability of the parameter to be use as the input. In the first line we declare the parameter column and row of testing data and then in the second line declare the labels of classification for testing data. Then we bind the parameter with itself before the linear SVM is run. Other parameter is done the same way and we need to define the row and the column of each parameter.

```
>library(kernlab)

 ### read data

>mydata<- read.csv('NApo.csv')

>summary(mydata)
```

In the csv file we had determine which parameter need to be combined together by doing forward selection. Then data is imported in to R. Actually based on the parameter selection made before we can just use the cbind function to bind the best parameter together and run for the kernel test.

```
### set data to be training and test

>xtrain<- mydata[1:120, 2:4]

>ytrain<- mydata[1:120, 1:1]

>xtest<- mydata[121:147, 2:4]

>ytest<- mydata[121:147, 1:1]

### label for train and test data

>ytrain<- factor(ytrain, levels = c ("High","Medium","Low"),labels=c("H", "M", "L"))

>ytest<- factor(ytest, levels = c ("High","Medium","Low"),labels=c("H", "M", "L"))
```

In the first until fourth line train data for the input is set from data 1 until data 120 which start from column 2 while testing data is starting after training data. The label is same as training and testing data but it start from column 1. Next in fifth line we labeled the classification of DO as H for class high, M for class Medium and L for low class.

```
>table(ytrain)

>table(ytest)
```

Table function is used to see the value of each class in train and test data.

```
###Kernel selection (Each kernel is set with default value)

>model<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='vanilladot',cross=10)

>model2<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='rbf',cross=10)

>model3<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='polydot',cross=10)

>model4<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='laplacedot', cross=10)

>model5<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='tanhdot', cross=10)

>model6<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='besseldot', cross=10)

>model7<-ksvm (ytrain~.,data=xtrain,type="C-svc",  kernel='anovadot', cross=10)
```

Based on the kernel used we assign it with different name such as model, model2 and as follow. Each is done using the default value which is usually the value is 1.

```
>model

>model2

>model3

>model4

>model5

>model6

>model7
```

By calling the assign name we can see the error produce for each parameter and the best kernel is chosen when it resulted with least error.

```
###loop function to determined best kernel parameter value

>clist<- c(1:20)

>clist<- clist /10

>sigmalist<- clist



>cv_err<- matrix(0,nrow=length(clist),ncol=length (sigmalist))

+for (i in seq(length(clist))) {

+C <- clist[i]

+for (j in seq(length(sigmalist))) {

+model<-ksvm(ytrain~.,data=xtrain,type="C-svc",          kernel='anovadot',
+kpar=list(sigma=sigmalist[j],degree=1),  C=clist[i],cross=10)

+cv_err[i,j] <- cross(model)

+}

+}

>cv_err
```

After selecting the kernel we need to find the best parameter value as this can improve our result. By doing the SVM model using default value we still do not know whether the value is the best value and if it is the best combination that can gives us least error. For the best kernel use in this project is Anova kernel. In Anova the parameter used are sigma, C and degree. We use default value for degree value firstly. In the first until third line we assign the value to be used in finding the best sigma and C value. Next in fourth

line we assign which from sigma and C value to be in row and which to in the column line. We used for loop to find the best generated sigma and C value and input it in the model that has been chosen. In the tenth line from the row and column of sigma and C value we assign it to display the error produce from the combination of both parameter values.

```
>bestmodel<-ksvm (ytrain~.,data=xtrain,type="C-svc", kernel='anovadot', kpar=list(sigma=1,degree=1.5), C=24,cross=10)

>bestmodel
```

From the loop function we choose the best parameter value with least error to be input into our model.

```
>pred.ksvm<- predict(bestmodel, xtest)

>pred.ksvm

>ytest
```

The predict function is used to predict our test data with the model that has been created before. When calls the assigned name for predict function it will show the prediction made on the test data. ytest is called to see the comparison between the predicted and actual data class label.

```
>table(pred.ksvm, ytest)

>sum(pred.ksvm==ytest)/length(ytest)
```

The table function is used to generate the cross tabulation table of actual and predicted data. Then to calculate the accuracy we use the sum function.

APPENDIX

Appendix A

SVM on R Tutorial

Some facts about SVM on R:

- R is light – it doesn't come with all functions.
- Specialised functions come in packages – you only add the packages that you need.
- SVM functions are bundled in the "kernlab" package.
- Thus, prior to run SVM functions, we need to download the 'kernlab" package on our machine.

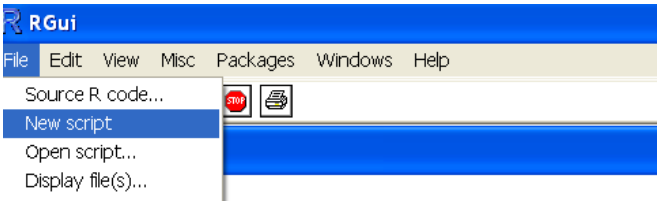
(1) In order to download "kernlab" package on our machine, type the following in the terminal:

install.packages("kernlab")

You would then need to choose the mirror. Once the installation is completed, you would be prompted accordingly.

(2) In this tutorial, we shall write all our R codes in scripts. Do the following:

File -> New Scripts



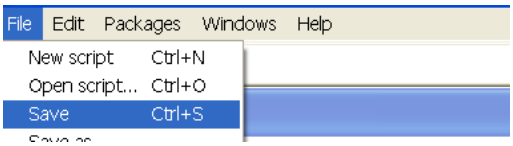
A new window of R Editor will be opened.

(3) Type the following codes in the R editor:

```
#This is my 1st attempt to run SVM on R

#attaching the kernlab package

library(kernlab)
```

(4) Save the file. Do the following:

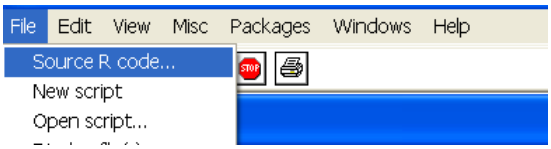
File -> Save



Save the file as ICIMU.R

(5) Invoke the file. Do the following:

File ->Source R code…



Chose the file ICIMU.R

Our tutorial:

- In this tutorial, we shall be doing things step by step.
- It is essential that every step is saved and compiled accordingly.
- The comments help you to understand each given step.

(6) Go back to your ICIMU.R file. Append the following codes in the R editor:

```
#These codes create two classes of 50 specimens each,
# namelyblueClass and redClass.
#Each class has two features, namely f1 and f2
#Both f1 and f2 are of normal distribution
#The blue class has a mean of 1 and sd of 0.2
#for its f1 and f2.
#The red class has a mean of 1.5 and sd of 0.2
#for its f1 and f2.


n <- 50


f1blueclass <- rnorm(n, mean = 1, sd = 0.2)
f2blueclass <- rnorm(n, mean = 1, sd = 0.2)
f1redclass <- rnorm(n, mean = 1.5, sd = 0.2)
```

```
f2redclass <- rnorm(n, mean = 1.5, sd = 0.2)


blueclass<- cbind(f1blueclass, f2blueclass)

redclass<- cbind(f1redclass, f2redclass)
```

Do not forget to save the file.

(7) Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>blueclass

>redclass
```

(8) Let us visualize the data. Type the following in the terminal:

```
>plot(blueclass, col = "red")

>plot(redclass, col = "blue")
```

Some facts on rnorm(n, mean, sd) function:

- This function creates a random n samples from a distribution with the given mean and standard deviation
- The values are different from one person to another
- The values are different from one run to another

(9) Go back to your ICIMU.R file. Append the following codes in the R editor:

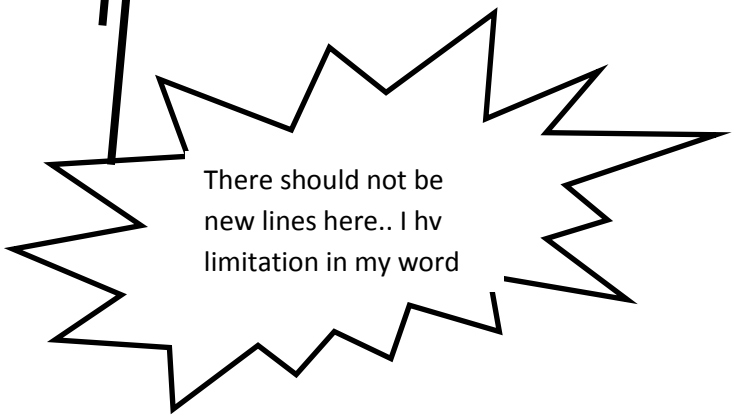
```
#Prepare data for SVM

#Data – regardless the number of features is often known as x

x <- rbind(blueclass, redclass)



#Generate the labels for the classes

#Labels are often known as y

#For blue class, we assign the value 1

#For red class, we assign the value -1

y<- matrix(c(rep(1,50),rep(-1,50)))
```

Do not forget to save the file.

(10) Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>x
>y
>plot(x,col=ifelse(y>0,"blue","red"), xlab = "feature2",
ylab="feature1")
>legend("topleft",c('Blue Class','Red
Class'),col=c("blue","red"),pch=1,text.col=c("blue","red"))
```



(11) Go back to your ICIMU.R file. Append the following codes in the R editor:

```
# Prepare a training and a test set randomly from the data set#


# ntrain is the total number of training samples
# Usually 80% of data is used for training
# whilst 20% of data is used for testing
ntrain<- round(n*0.8)


# tindex lists the indices of training samples (randomly chosen)
```

```
tindexblue<- sample(n,ntrain)
tindexred<- 50 + sample(n,ntrain)
tindex<- c(tindexblue, tindexred)
xtrain<- x[tindex,]
xtest<- x[-tindex,]
ytrain<- y[tindex]
ytest<- y[-tindex]
istrain=rep(0,2*n)
istrain[tindex]=1
```

Do not forget to save the file.

(12) Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

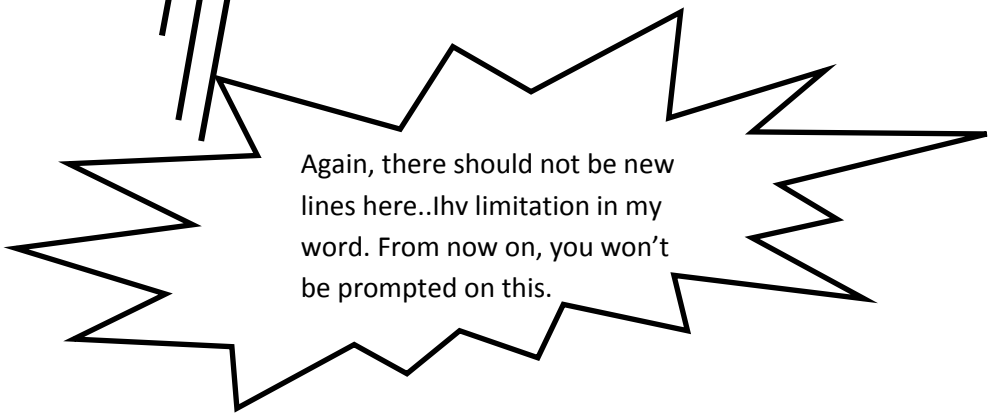
```
>tindexblue
>tindexred
>tindex
>xtrain
>xtest
>ytrain
>ytest
>istrain
```

Some facts on sample(n, x) function:

- This function selects random x numbers from a 0 to n integer list
- x < n
- There should not be duplicate values of x

(13) Let us visualize the data. Type the following in the terminal:

```
>plot(x,col=ifelse(y>0, "blue", "red"),pch=ifelse(istrain==1,1,2),
xlab = "feature2", ylab="feature1")
>legend("topleft",c('Blue Class Train','Blue Class Test','Red Class
Train','Red Class Test'),col=c("blue", "blue", "red",
"red"),pch=c(1,2,1,2),text.col=c("blue", "blue", "red", "red"))
```



(14) Now we are ready to run the SVM. Let us first build the SVM model using linear kernel on out training data. Go back to your ICIMU.R file. Append the following codes in the R editor:

```
# Train the SVM
# Assign our model to an object known as svm_model
svm_model<- ksvm(xtrain,ytrain,type="C-
svc",kernel='vanilladot',C=1,scaled=c())
```

Do not forget to save the file.

(15) Let us check our model. Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>svm_model
```

Some facts on ksvm(x, y ,...) function:

- This function builds the specified SVM model on our trained data x with label of classes y.
- In order to get more details on any function in R, just type help(function_name) on the terminal to go to the help page
- Kernel vanniladot is the linear kernel.
- By default, the C parameter is set to 1 (unless you set it differently).

- When you type the model name, you will be prompted on several information. The most important information is the training error rate.

(16) Let us now test the accuracy of our SVM model on the testing data. Go back to your ICIMU.R file. Append the following codes in the R editor:

```
# Predict labels on the testing data
ypred = predict(svm_model,xtest)


# Compute the model accuracy
# Our model is accurate if the predicted label = the actual label
# Otherwise, we have error
accuracy<- sum(ypred==ytest)/length(ytest)


# Compute at the prediction scores
# If the score is > 0, it belongs to class blue
# ifthescore is < 0, it belongs to class red
ypredscore = predict(svm_model,xtest,type="decision")
```

Do not forget to save the file.

(17) Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>ypredscore

>ypred

>ytest

>accuracy

>table(ypred, ytest)
```

(18) Let us repeat the above steps (training and testing) with different C parameters for our SVM models. Go back to your ICIMU.R file. Append the following codes in the R editor:

```
# Train the SVM with C = 20 & C = 30

svm_model2<- ksvm(xtrain,ytrain,type="C-

svc",kernel='vanilladot',C=20,scaled=c())

svm_model3<- ksvm(xtrain,ytrain,type="C-

svc",kernel='vanilladot',C=30,scaled=c())


# Predict labels (for model with C = 20 & C=30)

ypred2 = predict(svm_model2,xtest)

ypred3 = predict(svm_model3,xtest)


# Compute the model accuracy (for model with C = 20&C = 30)

accuracy2<- sum(ypred2==ytest)/length(ytest)

accuracy3<- sum(ypred3==ytest)/length(ytest)
```

Do not forget to save the file.

(19) Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>accuracy

> accuracy2

> accuracy3

>plot(svm_model,data=xtrain, xlab = "feature2", ylab="feature1")

>plot(svm_model2,data=xtrain, xlab = "feature2",

ylab="feature1")

>plot(svm_model3,data=xtrain, xlab = "feature2",

ylab="feature1")
```

You may re-run the experiment several times until you get different values of accuracy.

What we have done thus far:

- Basically we have divide our data (manually) into training and testing set.
- We built the model on the training data
- We investigate the accuracy by experimenting on the testing data.
- In research, data collection is expensive. Researches are often faced with the problems of data limitation.
- Cross validation is used when we have limited data.

- There is a build-in function for cross validation in R – simple, save us the trouble of dividing the data into separate training and testing set.

(20) Let us try the cross validation approach. Go back to your ICIMU.R file.

Append the following codes in the R editor:

```
# We perform cross validation on 5 folds data
svm_cv<- ksvm(x,y,type="C-
svc",kernel='vanilladot',C=1,scaled=c(),cross=5)
```

(21) Let us check the cross validation error rate.

Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>svm_cv
>cross(svm_cv)
```

Some facts aboutscross(svm_cv):

- It invokes the cross validation error rate from the object svm_cv.
- Training error rate is not as important as the cross validation error rate (i.e. there is no point getting a low training rate if it performs badly during cross validation testing)

(22) Tricks: We shall create a function that performs SVM cross validation in a loop for different parameter of C. Go back to your ICIMU.R file. Append the following codes in the R editor:

```
#Create a function that performs cross validation SVM repeatedly
#on a list of C parameters


clist<- c(1:20) #create an C list of integer from 1 to 20


#divide c list by 10
#thus our c parameter varry from 0.1 to 2.0
clist<- clist / 10


#create a list to hold the cross validation error
err<- numeric(length(clist))


#perform SVM in a loop
for (i in seq(length(clist))) {
        svm_cv_loop<- ksvm(x,y,type="C-
        svc",kernel='vanilladot',C=clist[i],scaled=c(),cross=5)
        err[i] <- cross(svm_cv_loop)
}
```

(23) Let us check the error rate performance.

Invoke the file ICIMU.R again. (Hint: Use the arrow key)

Type the following in the terminal:

```
>plot(clist,err,type='l',xlab="C",ylab="Error rate")
>cbind(clist, err)
```

What have given you thus far:

- Is the step by step exact codes (you just need to compile and run the codes)
- For the following session, you are required to add, change and modify the code yourselves.
- Good luck

(24) You have so have only tested the linear kernel. From now onwards you shall run your codes on non-linear kernel. Google package kernlab. What are other non-linear kernels available? Go back to your ICIMU.R file. Modify svm_model2 and svm_model3 using different kernels. Run. Compare the accuracy by using different kernels.

(Hint :- You may want to increase the overlapped of your data by increasing the standard deviation)

(25) You have so have only tested the linear kernel. From now onwards you shall run your codes on non-linear kernel. Google package kernlab. What are other non-linear kernels available? Go back to your ICIMU.R file. Modify svm_model2 and svm_model3 using different kernels. Run. Compare the accuracy by using different kernels.

(26) Try to make a nested loop for SVM RBF kernel. The outter loop should run with different value of C. Where as the inner loop should run with different value of sigma. You may want to view the help pages or the pdf documentation of kernlab for further elaboration on RBF kernel.

Appendix B

Data used for prediction

DO	pH	TEMP	COND
Medium	6.99	30.88	77
Medium	6.73	30.91	89
Medium	6.19	28.9	46
Medium	6.48	32.84	48
Medium	7.33	32.78	88
Medium	6.85	30.47	154
Medium	6.83	32.88	57
Medium	6.92	31.63	33
Medium	6.78	30.97	36
Medium	7.11	32.36	32
Medium	7.23	31.81	37
Medium	7.24	31.03	35
Medium	6.75	30.86	52
Medium	6.73	30.97	36
Medium	6.53	31.65	28
Medium	7.73	32.09	92
Medium	7.68	32.52	140
Medium	7.13	28.79	94
Medium	6.38	33.11	190
Medium	6.79	30.51	28
Medium	6.18	30.35	29
Medium	6.24	30.51	28
Medium	6.79	30.75	27
Medium	7.21	30.31	29
Medium	7.11	30.72	30
Medium	6.57	30.91	38
Medium	6.84	30.63	28
Medium	6.22	30.34	26
Medium	7.22	31.7	27
Medium	6.8	31.14	27
Medium	6.95	31.05	27
Medium	7.31	32.75	28
Medium	7.3	31.95	32
Medium	7.56	31.78	31
Medium	6.76	31.89	34
Medium	6.59	31.09	29
Medium	6.51	31.11	26
Medium	6.73	27.64	109
Medium	6.06	27.86	71
Medium	6.41	30.61	27
Low	7.64	32.96	139
Low	6.55	29.07	39
Low	6.86	30.48	93
Low	6.13	27.55	38
Low	6.51	31.17	28
Low	6.75	28.77	99
Low	6.25	27.67	72
Low	6.33	29.76	30
Low	6.41	27.59	32
Low	6.24	27.7	27
Low	5.27	27.67	24
Low	6.46	29.87	25
Low	7.33	28.19	90
Low	5.5	29.21	31
Low	5.61	30.2	31
Low	7.65	27.34	34
Low	7.6	29.99	40
Low	7.02	28.32	23
Low	6	30.11	33
Low	5.54	26.51	34
Low	5.59	30.04	35
Low	5.96	28.43	22
Low	5.84	28.62	31
Low	6.1	28.06	32
Low	6.01	27.39	49
Low	5.96	30.61	52
Low	6.09	28.02	39
Low	6.16	28.84	31
Low	6.71	29.32	36

Low	6.1	28.02	39
Low	6.67	28.4	44
Low	6.11	28.8	40
Low	5.5	27.09	36
Low	5.91	28.9	23
Low	5.84	27.87	29
Low	5.92	33.04	32
Low	6.32	29.32	33
Low	6.24	29.49	29
Low	6.51	29.34	33
Low	6.35	29.2	29
High	7.42	33.62	47
High	6.63	30.31	30
High	6.93	31.37	25
High	7.5	31.32	33
High	6.54	30.78	29
High	7.33	31.82	27
High	7.06	30.8	26
High	7.8	31.74	31
High	7.92	31.75	34
High	7.94	31.71	37
High	7.6	30.49	35
High	7.06	30.8	26
High	6.11	27.35	31
High	6.43	29	27
High	6.57	32.1	25
High	7.53	31.32	27
High	6.39	28.93	32
High	6.32	32.69	34
High	6.12	27.61	38
High	5.11	28.5	25
High	4.97	28.57	26
High	5.14	27.16	26
High	5.53	29.8	27
High	5.32	28.18	27
High	6.4	30.91	32
High	6.62	30.91	36
High	6.51	30.57	14
High	6.58	29.44	32

High	6.57	34.29	28
High	6.66	28.87	26
High	5.69	32.35	28
High	5.91	32.84	24
High	7.5	27.57	101
High	5.97	30.11	21
High	6.5	32.72	22
High	6.55	30.75	25
High	6.82	30.69	22
High	7.41	30.75	25
High	7.16	30.77	27
High	6.76	30.34	26
Medium	6.18	30.13	28
Medium	6.45	31.17	28
Medium	6.34	30.97	29
Medium	6.77	31.065	28
Medium	6.24	30.67	32
Medium	6.3	30.6	29
Medium	6.07	29.93	28
Medium	7.59	26.12	30
Medium	7.77	27.7	31
Low	5.79	28.46	26
Low	5.81	28.12	22
Low	6.41	29.88	22
Low	7.03	29.67	78
Low	6.66	28.37	33
Low	6.24	32.64	37
Low	6.15	30.95	25
Low	5.98	32.62	26
Low	6.9	29.82	69
High	6.67	30.72	26
High	6.68	30.47	25
High	7.01	33.47	23
High	7.26	30.71	125
High	6.64	30.91	43
High	6.71	30.28	26
High	6.42	32.86	27
High	7.89	31.54	21
High	7.42	32.54	23

Appendix C

Data eliminated for prediction

NH3-NL	TEMP	COND	SAL	TUR	NO3	PO4	E-coli	Coliform
0.03	30.88	77	0.03	23.7	0.04	0.01	100	20000
0.05	30.91	89	0.04	32	0.22	0.01	400	8700
0.09	28.9	46	0.02	29.2	0.05	0.01	2800	11000
0.03	32.84	48	0.02	15.1	0.01	0.01	140	7400
0.01	32.78	88	0.04	37.3	0.21	0.03	300	13000
0.01	30.47	154	0.07	56.2	1.06	0.13	100	3300
0.03	32.88	57	0.02	8.2	0.16	0.01	200	2800
0.01	31.63	33	0.01	4.7	0.17	0.04	80	1800
0.01	30.97	36	0.02	4.8	0.01	0.01	5400	11000
0.1	32.36	32	0.01	8.8	0.01	0.01	3000	10000
0.02	31.81	37	0.02	7.3	0.02	0.01	1800	7800
0.45	31.03	35	0.02	7.6	0.01	0.01	2100	6400
0.37	30.86	52	0.02	1.1	0.01	0.01	1800	8100
0.75	30.97	36	0.02	2.6	0.01	0.01	3000	7800
0.14	31.65	28	0.01	1.6	0.01	0.01	3000	6400
0.01	32.09	92	0.05	25.2	0.33	0.09	500	4500
0.05	32.52	140	0.06	27.2	0.47	0.07	300	4300
0.08	28.79	94	0.04	16.8	0.18	0.02	1600	5200
0.37	33.11	190	0.09	11.4	0.22	0.01	1300	1800
0.01	30.51	28	0.01	6.5	0.01	0.02	200	1300
0.01	30.35	29	0.01	6.5	0.05	0.23	500	1400
0.01	30.51	28	0.01	1.8	0.01	0.2	600	1900
0.01	30.75	27	0.01	11.3	0.01	0.3	100	1500
0.01	30.31	29	0.01	5.3	0.85	0.29	400	1700
0.01	30.72	30	0.01	5.9	0.04	0.23	300	2500
0.01	30.91	38	0.02	2.1	0.13	0.18	400	1000
0.01	30.63	28	0.01	7	0.01	0.19	300	1600
0.01	30.34	26	0.01	1	0.01	0.3	900	2400
0.01	31.7	27	0.01	6.4	0.01	0.07	70	170
0.2	31.14	27	0.01	9.1	0.46	0.01	20	110
0.26	31.05	27	0.01	3.8	0.75	0.01	60	310
0.5	32.75	28	0.01	8.3	0.53	0.01	60	230
0.62	31.95	32	0.01	8.7	0.78	0.12	60	180

0.33	31.78	31	0.01	11.4	0.47	0.03	0	20
0.04	31.89	34	0.01	1.5	0.29	0.01	120	340
0.08	31.09	29	0.01	2.2	0.15	0.01	30	130
0.08	31.11	26	0.01	3.1	0.11	0.01	40	410
0.02	27.64	109	0.05	129	0.8	0.08	2500	8400
0.01	27.86	71	0.03	13.2	0.03	0.07	3000	8500
0.5	30.61	27	0.01	3.5	0.01	0.04	400	2800
0.01	32.96	139	0.06	75.1	0.44	0.02	300	7800
0.01	29.07	39	0.02	39.8	0.07	0.01	400	14000
0.07	30.48	93	0.04	145	1.08	0.14	300	6000
0.01	27.55	38	0.02	28.9	0.25	0.01	200	2100
0.52	31.17	28	0.01	1.1	0.01	0.01	2100	7300
0.01	28.77	99	0.05	26.4	0.28	0.06	1300	8400
0.01	27.67	72	0.03	14.4	0.13	0.07	1600	5100
0.01	29.76	30	0.02	2	0.18	0.18	10	990
0.34	27.59	32	0.02	14.6	0.11	0.36	1100	3100
0.04	27.7	27	0.01	14.1	0.62	0.01	0	1000
0.03	27.67	24	0.01	0.8	0.71	0.01	100	2300
0.04	29.87	25	0.02	37	0.03	0.09	700	20800
0.01	28.19	90	0.04	125.5	0.76	0.05	100	5600
0.03	29.21	31	0.02	18.6	0.02	0.01	0	1800
0.01	30.2	31	0.03	2.2	0.31	0.01	0	4400
0.03	27.34	34	0.01	6.6	0.77	0.01	1300	10900
0.01	29.99	40	0.02	9.6	0.01	0.01	1000	14000
0.05	28.32	23	0.01	13	0.01	0.01	200	13000
0.81	30.11	33	0.01	8	0.31	0.01	100	4000
0.04	26.51	34	0.01	30	0.01	0.1	800	21000
0.01	30.04	35	0.01	30	0.01	0.1	2000	13000
0.01	28.43	22	0.01	7	0.01	0	100	7000
0.03	28.62	31	0.01	3	0.01	0	1000	13000
0.01	28.06	32	0.01	22.4	0.01	0.1	200	8100
0.01	27.39	49	0.02	33.7	0.01	0.05	2000	8000
0.01	30.61	52	0.02	36	0.01	0.07	4000	33000
0.01	28.02	39	0.02	29.1	0.01	0.05	600	2800
0.06	28.84	31	0.01	12.1	0.58	0.01	1100	7400
0.18	29.32	36	0.02	4.5	0.35	0.01	1200	7200
0.22	28.02	39	0.02	29.1	0.13	0.01	300	6100
0.5	28.4	44	0.02	27.8	1.06	0.06	200	800
0.23	28.8	40	0.02	2	0.75	0.01	1100	5500

0.23	27.09	36	0.02	5	0.93	0.01	900	10100
0.05	28.9	23	0.01	9.7	0.01	0.06	300	8000
0.01	27.87	29	0.01	17.4	0.01	0.04	400	8000
0.01	33.04	32	0.01	16.5	0.01	0.04	600	5000
0.01	29.32	33	0.01	8.9	0.02	0.05	200	3200
0.1	29.49	29	0.01	3.4	0.09	0.24	2000	4700
0.09	29.34	33	0.01	8.4	0.09	0.24	100	900
0.07	29.2	29	0.01	2.6	0.09	0.21	100	2100
0.01	33.62	47	0.02	10.9	0.01	0.01	100	2100
0.02	30.31	30	0.01	4	0.21	0.01	100	2000
0.01	31.37	25	0.02	1.7	0.01	0.01	300	1100
0.01	31.32	33	0.02	4.2	0.02	0.01	200	1200
0.01	30.78	29	0.02	0.8	0.02	0.01	200	1200
0.01	31.82	27	0.02	3.3	0.01	0.01	300	6600
0.01	30.8	26	0.02	0.4	0.01	0.01	500	1300
0.01	31.74	31	0.02	9.8	0.01	0.01	100	900
0.01	31.75	34	0.02	8.6	0.01	0.01	1100	2700
0.01	31.71	37	0.02	9.7	0.01	0.01	300	3400
0.01	30.49	35	0.02	1	0.01	0.01	100	31700
0.01	30.8	26	0.02	0.6	0.04	0.01	1200	2400
0.01	27.35	31	0.02	15.6	0.11	0.01	500	47100
0.01	29	27	0.02	1.1	0.01	0.07	2100	3300
0.01	32.1	25	0.02	4.2	0.02	0.08	100	2700
0.06	31.32	27	0.02	5.2	0.05	0.09	300	1200
0.01	28.93	32	0.02	1.8	0.01	0.08	100	1600
0.01	32.69	34	0.01	10.2	0.96	0.21	40	1560
0.03	27.61	38	0.02	16.2	0.11	0.01	1400	6400
0.01	28.5	25	0.01	1.5	0.09	0.01	1000	1900
0.01	28.57	26	0.01	3.1	0.04	0.01	1100	2200
0.01	27.16	26	0.01	0.7	0.31	0.01	50	1800
0.01	29.8	27	0.02	2.4	0.18	0.01	0	8100
0.01	28.18	27	0.02	2.7	0.53	0.01	200	5900
0.3	30.91	32	0.01	3.3	1.37	0.03	1000	13000
0.4	30.91	36	0.01	4	2.04	0.01	100	8000
0.13	30.57	14	0.02	5.2	0.66	0.01	100	9000
0.44	29.44	32	0.01	4.1	1.68	0.03	2000	18000
0.2	34.29	28	0.01	19	0.25	0.01	1000	31000
0.02	28.87	26	0.01	4	0.01	0.01	30	900
0.01	32.35	28	0.01	8.3	0.57	0.01	2000	6000

0.1	32.84	24	0.01	8.5	0.72	0.06	200	13800
0.21	27.57	101	0.05	36.5	0.24	0.05	200	31000
0.01	30.11	21	0.01	0.5	0.04	0.04	80	13300
0.01	32.72	22	0.01	8.7	0.24	0.01	2900	13900
0.01	30.75	25	0.01	5.7	0.03	0.09	164	12400
0.01	30.69	22	0.01	1.2	0.01	0.26	1328	24400
0.01	30.75	25	0.01	1.6	0.02	0.14	192	10900
0.01	30.77	27	0.01	2.5	0.01	0.22	384	15500
0.01	30.34	26	0.01	3	0.02	0.23	144	11300
0.01	30.13	28	0.02	1.3	0.01	0.12	500	1600
0.01	31.17	28	0.02	4	0.09	0.15	200	1500
0.01	30.97	29	0.02	6.4	0.71	0.03	20	1480
0.01	31.065	28	0.02	5.3	0.09	0.03	300	2500
0.01	30.67	32	0.02	1.1	0.18	0.12	300	3300
0.01	30.6	29	0.02	2.6	0.09	0.18	10	1180
0.01	29.93	28	0.02	1.5	0.75	0.21	500	3000
0.06	26.12	30	0.01	238	0.44	0.01	5200	16300
0.24	27.7	31	0.01	282.2	0.56	0.01	600	19100
0.06	28.46	26	0.01	1.1	0.09	0.31	1000	9000
0.01	28.12	22	0.01	74.2	0.37	0.02	1000	8600
0.02	29.88	22	0.01	16.6	0.02	0.01	2000	11000
0.01	29.67	78	0.03	30.8	0.01	0.02	100	7000
0.01	28.37	33	0.01	25.9	0.01	0.01	100	5000
0.01	32.64	37	0.02	10.4	0.01	0.01	16	700
0.01	30.95	25	0.01	9.2	0.01	0.07	100	900
0.01	32.62	26	0.01	1.8	0.01	0.04	100	3800
0.01	29.82	69	0.03	74.9	0.01	0.02	100	10000
0.01	30.72	26	0.01	1.1	0.01	0.2	160	3500
0.01	30.47	25	0.01	1.3	0.01	0.19	120	9400
0.01	33.47	23	0.01	4	0.04	0.01	1400	15100
0.05	30.71	125	0.06	15.7	0.01	0.05	100	284000
0.04	30.91	43	0.02	11	0.04	0.04	200	4600
0.04	30.28	26	0.01	6.9	0.01	0.03	400	17400
0.03	32.86	27	0.01	3.8	0.72	0.01	200	4700
0.21	31.54	21	0.01	12.2	0.01	0.01	1040	1300
0.08	32.54	23	0.01	11.7	0.01	0.01	800	1800

BIBLIOGRAPHY

1. Bouamar M., and Ladjal M., "Evaluation of the performances of ANN and SVM techniques used in water quality classification, "*Proceedings of ICECS'07*, 14.

2. Bouamar M., Ladjal M, Multisensor system using Support Vector Machines for water quality classification, *Proceedings of ISSPA'07, 9th IEEE International Symposium on Signal Processing and its Applications*, 12-15 Feb, Sharjah, UAE, 2007.

3. *DTREG*. (n.d.). Retrieved May 5, 2012 , from DTREG (Software For Predictive Modeling and Forecasting): http://www.dtreg.com.

4. He , T., & Chen, P. (2010). Prediction of water-quality based on wavelet transform using vector machine. *2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science.* 76 – 81.

5. Interim National Water Quality Standard in Malaysia.(n.d). Retrieved March 26, 2012, from WEPA (Water Environment Partnership in Asia): http://www.wepa-db.net.

6. Junsawang P, Pomsathit A, Areerachakul. Prediction of Dissolved Oxygen Using Artificial Neural Network. *2011 International Conference on Computer Communication and Management,Singapore,* 1-5.

7. Karatzoglou A.,Smola A., Hornik K. (2004). kernlab – An S4 Package for Kernel Methods in R. 11(9). 1-20.

8. Liao Y., Xu J., Wang Z. (2012). Application of biomonitoring and support vector machine in water quality assessment. 13(4): 327–334.

9. Liu JP, Chang MQ, Ma XY. Groundwater Quality Assessment Based on Support Vector Machine. HAIHE River Basin Research and Planning Approach-*Proceedings of 2009 International Symposium of HAIHE Basin Integrated Water and Environment Management*, Beijing, China. 2009, 173-178.

10. Medina, M. (2011). *Development and implementation of a water quality monitoring plan to support the creation of a geographic information system to assess water quality conditions of rivers in the state of veracruz in mexico.* (Master's thesis).

11. Naik, V. K. and Manjapp, S. (2010). Prediction of Dissolved Oxygen through Mathematical Modeling. Int. J. Environ. Res., 4(1), 153-160.

12. Najah, A., El-Shafie, A., Karim, O. A., and Jaafar, O. (2011)Integrated versus isolated scenario for prediction dissolved oxygen at progression of water quality monitoring stations, Hydrol. Earth Syst. Sci., 15, 2693–2708.

13. Najah, A., El-Shafie, A., Karim, O. A., Jaafar, O. El Shafie, H.O (2011).An application of different artificial intelligences techniques for water quality prediction.International Journal of the Physical Sciences, 6(22), 5298–5308.

14. Palani, S., Liong, S.Y., Tkalich, V., Palanichamy, J., (2008). Development of Neural Network Model for Dissolved Oxygen in Seawater. Indian Journal of Marine Sciences, 38(2), 151-159.

15. Prasad, M., W. Long, X. Zhang, R. Wood, and R. Murtugudde. 2011. Predicting dissolved oxygen in the Chesapeake Bay: Applications and implications. Aquatic Sciences—Research Across Boundaries: 1–15.