DATA ANALYTICS AND VISUALIZATION (UCS633)

LAB PROJECT - REPORT
SVM Classification on Diagnostic Cancer Data

Team No. 13

Siddhartha Khanooja (101410051)

Sreeparna Deb (101410053)

Sumit Dahiya (101410054)

Sunidhi Jalota (101410055)

Table of Contents

Introduction to the problem	3
Database Description	4
Attribute Information	5
A brief introduction to SVMs	6
Data preprocessing	7
Performance of the data analytics task	8
Accuracy and Confusion Matrix	12
Final Code	13

Introduction to the problem

We have a designed a simple model of classification using Support Vector Machine (SVM). The operation of classification and visualization was performed on RStudio, using R v3.4.0. The classification was performed on the Wisconsin Breast Cancer (Diagnostic) Dataset, hosted on the UCI ML Repository. The problem was, predicting whether a particular cancer cell would be benign or malignant based on the properties of the cell. We have also performed cross-validation on the dataset as a part of pre-processing to ensure better accuracy. To ensure better understanding of the work that we have done, a clear and concise explanation for each and every step performed, has been given.

Database Description

The Breast Cancer Wisconsin (Diagnostic) Dataset is a scientific database used for description of characteristics of cell nuclei present in image. The dataset is generated from digitized images of Fine Needle Aspirate (in which a needle is inserted into lumps of breast mass, and the tip of the needle is investigated using a microscope).

Data Set Characteristics:	Multivariate	**Number of Instances:**	699	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	11	**Date Donated**	1992-07-15
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	519444

Screenshot of the Dataset

samplecodenumber	clumpthickness	uniformcellsize	uniformcellshape	marginaladhesion	epithelial	barenuclei	blandchroatin	normalnucleoli	mitoses	classes
1000025	5	1	1	1	2	1	3	1	1	B
1002945	5	4	4	5	7	10	3	2	1	B
1015425	3	1	1	1	2	2	3	1	1	B
1016277	6	8	8	1	3	4	3	7	1	B
1017023	4	1	1	3	2	1	3	1	1	B
1017122	8	10	10	8	7	10	9	7	1	M
1018099	1	1	1	1	2	10	3	1	1	B
1018561	2	1	2	1	2	1	3	1	1	B
1033078	2	1	1	1	2	1	1	1	5	B
1033078	4	2	1	1	2	1	2	1	1	B
1035283	1	1	1	1	1	1	3	1	1	B
1036172	2	1	1	1	2	1	2	1	1	B
1041801	5	3	3	3	2	3	4	4	1	M
1043999	1	1	1	1	2	3	3	1	1	B
1044572	8	7	5	10	7	9	5	5	4	M
1047630	7	4	6	4	6	1	4	3	1	M
1048672	4	1	1	1	2	1	2	1	1	B
1049815	4	1	1	1	2	1	3	1	1	B
1050670	10	7	7	6	4	10	4	1	2	M
1050718	6	1	1	1	2	1	3	1	1	B
1054590	7	3	2	10	5	10	5	4	4	M
1054593	10	5	5	3	6	7	7	10	1	M
1056784	3	1	1	1	2	1	2	1	1	B
1057013	8	4	5	1	2	?	7	3	1	M
1059552	1	1	1	1	2	1	3	1	1	B
1065726	5	2	3	4	2	7	3	6	1	M
1066373	3	2	1	1	1	1	2	1	1	B
1066979	5	1	1	1	2	1	2	1	1	B

Attribute Information

#	Attribute (Column Name in Dataset)	Domain
1.	Sample Code Number (samplecodenumber)	ID Number
2.	Clump Thickness (clumpthickness)	1-10
3.	Uniformity of Cell Size (uniformcellsize)	1-10
4.	Uniformity of Cell Shape (uniformcellshape)	1-10
5.	Marginal Adhesion (marginaladhesion)	1-10
6.	Single Epithelial Cell Size (epithelial)	1-10
7.	Bare Nuclei (barenuclei)	1-10
8.	Bland Chromatin (blandchromatin)	1-10
9.	Normal Nucleoli (normalnucleoli)	1-10
10.	Mitoses (mitoses)	1-10
11.	Class (classes)	{M , B}

A brief introduction to SVMs

Support Vector Machine, or SVM, is a *supervised* machine learning algorithm which can be used for both classification or regression problems. However, it is mostly used for *classification* problems.

In this algorithm, we plot each attribute as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane(s) that differentiate the two (or more) classes very well.

To be more concise, it is simply a frontier which best segregates the two classes in the form of a hyperplane or a line.

SVM is generally used for small training data-sets, which have a large number of features.

Data preprocessing

We cannot start performing any operations on data without first “cleaning” it. Here are the steps performed to clean the data for this project:

1) Adding column headers - R will still work without this, as it will give default names to each of the columns (V0,V1,etc.) - but at the loss of clarity.
2) Preparing ‘class’ column for classification - The ‘class’ column initially contained values of {2, 4}, which meant that R would get confused whether to perform classification or regression. So, we changed the value of 2 to B (Benign) and 4 to M (Malignant) using Pandas, a Python package, and for extra security, converted the ‘class’ column into a factor in the R code.
3) Handling missing data - Missing data in this dataset was denoted by ‘?’. We decided to bypass this entirely. Instead of spending more memory (and time) on filling missing values with mean/median or using a naive Bayes classifier, we decided to ignore the missing values entirely, as explained in the R code.

Other steps like identifying and removing outliers, and handling noisy data, didn’t have to be performed due to the dataset already being pre-processed to a great extent before being uploaded on UCI.

Performance of the data analytics task

There are various implementations of the SVM in R, and we have used the **kernel** implementation of it - using the Caret function **train**, and method as "**SVMRadial**", which means that kernel used for SVM is the Gaussian radial basis function kernel.

A line-by-line explanation of the entire code follows, with the complete code given after the subsection. Output is provided wherever necessary.

Step 1) Loading libraries:

```
library("corrplot")
```
for plotting the correlation plot.
```
library("caret")
```
for createDataPartition(), confusionMatrix(), train() and trainControl().
```
library("kernlab")
```
for SVMRadial functionality.

Step 2) Loading data:

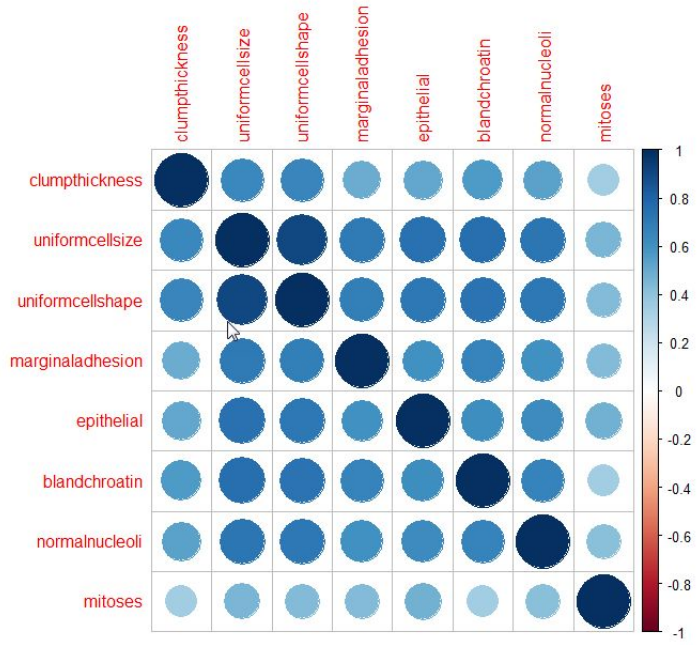
```
data2 <- read.csv("C:/Users/Dell/Desktop/Data Analytics Project/2.csv")
```

Step 3) Pre-Processing/Removing missing entries:

```
data2 =  data2[-c(25,42,141,147,160,166,237,251,277,294,296,299,317,323,413,619),]
```
here, we have removed the rows which have '?' as any of their attributes. We removed the entries since we had only 2% of missing data.

Step 4) Finding correlation between variables:

```
M <- cor(data2[ ,-c(1,7,11)])
```
removes variables which are to be classified ('classes'), redundant variables ('id'), and the variables which are factors ('barenuclei'), since they are redundant for the correlation plot.
```
corrplot(M)
```
plots correlation between variables - deeper colors imply stronger correlation.



Step 5) Define variable to be classified:

```
data2$classes <- as.factor(data2$classes)
```

this defines the last column as the variable to be classified.

Step 6) Summarize data (Optional step):

```
summary(data2)
```

```
 samplecodenumber   clumpthickness   uniformcellsize  uniformcellshape
 Min.   :   61634   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
 1st Qu.:  877617   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
 Median : 1171795   Median : 4.000   Median : 1.000   Median : 1.000
 Mean   : 1075790   Mean   : 4.391   Mean   : 3.122   Mean   : 3.193
 3rd Qu.: 1238620   3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000
 Max.   :13454352   Max.   :10.000   Max.   :10.000   Max.   :10.000

 marginaladhesion   epithelial       barenuclei   blandchromatin   normalnucleoli
 Min.   : 1.000   Min.   : 1.000   1      :395   Min.   : 1.000   Min.   : 1.000
 1st Qu.: 1.000   1st Qu.: 2.000   10     :127   1st Qu.: 2.000   1st Qu.: 1.000
 Median : 1.000   Median : 2.000   5      : 30   Median : 3.000   Median : 1.000
 Mean   : 2.817   Mean   : 3.199   2      : 29   Mean   : 3.433   Mean   : 2.864
 3rd Qu.: 4.000   3rd Qu.: 4.000   3      : 27   3rd Qu.: 5.000   3rd Qu.: 3.500
 Max.   :10.000   Max.   :10.000   8      : 20   Max.   :10.000   Max.   :10.000
                                   (Other): 55

    mitoses       classes
 Min.   : 1.000   B:450
 1st Qu.: 1.000   M:233
 Median : 1.000
 Mean   : 1.581
 3rd Qu.: 1.000
 Max.   :10.000
```

Step 7) Find the percentage of positive and negative elements in the column to be classified (Optional step):

```
prop.table(table(data2$classes))
```

express table entries (B,M) as fraction of table.

```
        B         M
0.658858 0.341142
```

Step 8) Split data into training set and testing set:

```
set.seed(312)
```

'seed' the random number generator, which in effect means you can reproduce the same results for training data and testing data (below) while running the program again and again.

```
index <- createDataPartition(data2$classes, p = 0.8, list = FALSE)
```

1) *data2$classes -> the target variable is classes.*
2) *p = 0.8 -> the train:test ratio is 80:20*
3) *list = FALSE -> index will NOT be a list, but a matrix instead (ex - int [1:547,1])*

```
traind <- data2[index, -1]
```

the first 'index' values are taken as training data. (from the above example, the first 547 values are chosen).

```
testd <- data2[-index, -1]
```

the rest of the values are taken as testing data.

Step 9) Fine-tune the training parameters:

```
fit <- trainControl(method="cv", number = 6, classProbs = TRUE,
summaryFunction = twoClassSummary)
```

1) *method = "cv" -> cross-validation, meaning that data is first trained, and then tested, for further accuracy.*
2) *number = 6 -> meaning that cross-validation will be applied for 6 iterations.*
3) *classProbs = TRUE -> class probabilities are calculated in each iteration.*
4) *summaryFunction = twoClassSummary -> summaryFunction() specifies metrics (performance measures). twoClassSummary means that sensitivity, specificity and area under ROC curve are computed.*

Step 10) Train the SVM Model:

```
SVMModel <- train(classes~., traind, method = "svmRadial",metric = "ROC",
preProcess = 'scale', trace = FALSE, trControl = fit)
```

1) *classes~. -> formula for prediction is classes ~ x1+x2+..., and so on.*
2) *traind -> the dataset for training.*
3) *method = "svmRadial" -> SVM is used for classification, and the kernel for SVM used is the radial kernel.*
4) *metric = "ROC" -> metric used for selecting the optimal model is the area under the ROC curve.*
5) *preProcess = "scale" -> before performing the iterations for SVM, each attribute is divided by its standard deviation.*
6) *trace = "false" -> turn off logging for this function.*
7) *trControl = fit -> fine-tune training using 'fit' already defined above.*

Step 11) Prediction using the SVM Model:

```
prediction <- predict(SVMModel, testd)
```

generates a column array, which has the predicted values of the column 'classes' in it.

We discuss the final results and the code in the subsequent sections.

Accuracy and Confusion Matrix

```
confusion <- confusionMatrix(prediction, testd$classes, positive = "M")
```

prediction -> the "expected" values / the data predicted

testd$classes -> the "actual" values / the data used

positive = "M" -> the "positive" class - this could be 'B' too, arbitrarily chosen.

```
Confusion Matrix and Statistics

          Reference
Prediction  B  M
         B 84  0
         M  6 46

               Accuracy : 0.9559
                 95% CI : (0.9064, 0.9836)
    No Information Rate : 0.6618
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9045
 Mcnemar's Test P-Value : 0.04123

            Sensitivity : 1.0000
            Specificity : 0.9333
         Pos Pred Value : 0.8846
         Neg Pred Value : 1.0000
             Prevalence : 0.3382
         Detection Rate : 0.3382
   Detection Prevalence : 0.3824
      Balanced Accuracy : 0.9667

       'Positive' Class : M
```

This means that we are getting an accuracy of over 95 percent on the dataset, primarily due to the reasons below:

1) Strong correlation between variables.
2) Fine-tuning the train() function using trainControl().
3) (Relatively) small dataset which is perfect for SVM.

The confusion matrix also is in line with our observations - showing that out of the training dataset of 136 observations, only 6 were predicted incorrectly.

Final Code

```
library("corrplot")
library("caret")
library("kernlab")
data2 <- read.csv("C:/Users/Dell/Desktop/Data Analytics Project/2.csv")
data2<- data2[-c(25,42,141,147,160,166,237,251,277,294,296,299,317,323,413,619), ]
M <- cor(data2[ ,-c(1,7,11)])
corrplot(M)
data2$classes <- as.factor(data2$classes)
summary(data2)
prop.table(table(data2$classes))
set.seed(312)
index <- createDataPartition(data2$classes, p = 0.8, list = FALSE)
traind <- data2[index, -1]
testd <- data2[-index, -1]
fit <- trainControl(method = "cv",
                            number = 6,
                            classProbs = TRUE,
                            summaryFunction = twoClassSummary)
SVMModel <- train(classes~.,
                  traind,
                  method = "svmRadial",
                  metric = "ROC",
                  preProcess = 'scale',
                  trace = FALSE,
                  trControl = fit)
prediction <- predict(SVMModel, testd)
confusion <- confusionMatrix(prediction, testd$classes, positive = "M")
confusion
```