Implementation of Support Vector Machine Classification using R package - Caret for Heart Disease Recognition Dataset.

Expt No: 7 (b) May 8,2019
Author: Subalakshmi Shanthosi S (186001008)

Aim

Implementation of Support Vector Machine(SVM) using R package- Classification and Regression Training(CARET) for Heart Disease Recognition Dataset.

Description

1. Support Vector Machine:
    - Support Vector Machine is a Supervised Learning Model.
    - SVM can be applied for both classification and regression algorithms but predominantly used for classification problems.
    - Support Vector Algorithm Working:
        - Input : Data points from the dataset (Heart Disease recognition dataset).
        - Output : Hyperplane - The line which best separates the tags.
        - Careful choice of Kernal function which decides the accuracy of the model.
    - Advantages of using SVM for classification:
        - High Dimensionality.
        - Memory Efficiency.
        - Versatility.
    - Disadvantages of using SVM:
        - Kernel Parameters Selection : SVM shows poor performance on higher dimensional data.
        - Non-Probabilistic : Effectiveness is less evident as the algorithm places few data points above and below the decision boundry which might lead to misclassification if the between class varients among points is less.
2. Classification hyperplane based on the data point's distribution is presented below:

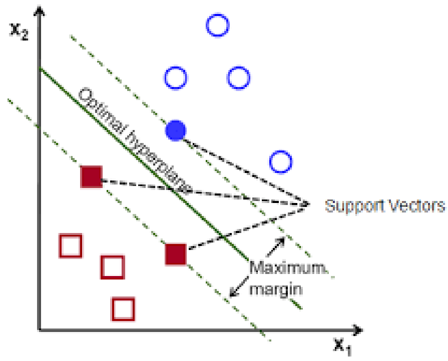

Figure 1: SVM Linear Model.

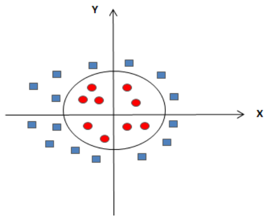

Figure 2: SVM Non Linear model by 3-D projection.

Tools and Packages

1. Tools
   - RStudio.
   - R Version 1.1.463
2. Support Vector Machine and Visualisation Packages
   -
   - ggplot2 (Visualisation)
   - GGally (Visualisation)

Dataset Description - Heart Disease Databases

- This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them.
- The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).
- Number of instances in Heart Disease Dataset:
  - Cleveland: 303
  - Hungarian: 294
  - Switzerland: 123
  - Long Beach VA: 200
- Attribute information:

Variable name	Short desciption	Variable name	Short description
age	Age of patient	thalach	maximum heart rate achieved
sex	Sex, 1 for male	exang	exercise induced angina (1 yes)
cp	chest pain	oldpeak	ST depression induc. ex.
trestbps	resting blood pressure	slope	slope of peak exercise ST
chol	serum cholesterol	ca	number of major vessel
fbs	fasting blood sugar larger 120mg/dl (1 true)	thal	no explanation provided, but probably thalassemia (3 normal; 6 fixed defect; 7 reversable defect)
restecg	resting electroc. result (1 anomality)	num	diagnosis of heart disease (angiographic disease status)

Figure 3: Heart Disease Detection Database important attributes.

Procedure

1. Split the data set as:
   - Training dataset.
   - Testing dataset.
2. Exploratory data Visualisation : To decide on the model to fit for a better precision in classification.
3. Feature Scaling and Model Fitting.
4. Calculate prediction and evaluate the SVM model/kernal accuracy.
5. Display the confusion matrix.

Support Vector Machine

- Support Vector Machine is a machine learning algorithm which:
  1. Solves classification problems.
  2. Uses flexible representation of decision boundary.
  3. Implements automatic complexity control to reduce overfitting.
  4. A single global minimum which can be found in polynomial time.

- Pseudocode :

- **Initialisation:**
  * For the specified kernel, and kernel parameters, compute the kernel of distances between the datapoints.
  * The main work here is the computation $K=XX^T$.
  * For the linear kernel, return K, for the polynomial of degree d return $\frac{1}{\sigma K^d}$.
  * For the RBF kernel, compute $K = \exp(-\frac{(x-x^{'})^2}{2\sigma^2})$.
- **Training**
  * Assemble the constraint set as matrices to solve:

$$min_x \frac{1}{2} x^T t_i t_j K_x + q^T x. \tag{1}$$

    subject to $G_x <= h$
    $A_x = b$
  * Pass these matrices to the solver.
  * Identify the support vectors as those that are within some specified distance of the closest point and dispose of the rest of the training data.
  * Calculate b^* using equation:

$$b^* = \frac{1}{N_s} \sum_{allsupportvectors} (t_j - \sum_{i=1}^{n} \lambda_i t_i x_i{}^T x_j). \tag{2}$$

- **Classification**
  * For the given test data **z**, Use the support vector to classify the data for the relevant kernal by :
    - Compute the inner product of the test data and the support vectors.
    - Perform the classification as:

$$\sum_{i=1}^{n} \lambda_i t_i K(x_i, z) + b^*. \tag{3}$$

      returning -
      The label (Hard Classification)
      The value(Soft Classification)

Algorithm 1: The Support Vector Algorithm

Confusion Matrix

- A confusion matrix is a table that can be generated for a classifier on a Data Set

  True Positives(TP)- These are the cases where the predicted and actual both are yes.
  True Negatives(TN)- These are the cases where the predicted value is no and actual value is yes.
  False Positive(FP)- These are the cases where the predicted value is yes and actual value is no.
  False Negative(FN)- These are the cases where prediction is no and actual value is no.

Coding

```
# Use library TeachingDemos to save output and commands
library(TeachingDemos)

txtStart("svmOutput.txt")

# SVM Classification using Linear Kernel

# Importing SVM library caret

library(caret)

# Loading CSV file to data frame

heart_df <- read.csv("/home/subalakshmi/PCP1211DALab/PCP1211ExptSevenB/heart_tidy.csv", se

# Showing the dataset description

# str(heart_df)

# head(heart_df)

# Split dataset for training and testing

set.seed(3033)
intrain <- createDataPartition(y = heart_df$V14, p= 0.7, list = FALSE)
training <- heart_df[intrain,]
testing <- heart_df[-intrain,]

# Printing the dimension
dim(training)
dim(testing)

# Preprocessing dataset - checking missing values
anyNA(heart_df)

# Summary Stats

summary(heart_df)

# Converting target to factor variable

training[["V14"]] = factor(training[["V14"]])

# Training SVM model

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3233)
```

```
svm_Linear <- train(V14 ~., data = training, method = "svmLinear",
                    trControl=trctrl,
                    preProcess = c("center", "scale"),
                    tuneLength = 10)

# Printing trained SVM model

# svm_Linear

# Predicting the model

test_pred <- predict(svm_Linear, newdata = testing)

# Printing the prediction

#str(test_pred)

# Confusion Matrix

confusionMatrix(factor(test_pred, levels = 1:148),
  factor(testing$V14, levels = 1:148))
# Parameter Tuning

grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))
set.seed(3233)
svm_Linear_Grid <- train(V14 ~., data = training, method = "svmLinear",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = grid,
                         tuneLength = 10)
plot(svm_Linear_Grid,main = "SVM Linear Grid")
#Prediction - with tuning exprementation

test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
#test_pred_grid

#Confusion matrix

confusionMatrix(factor(test_pred_grid, levels = 1:148),
  factor(testing$V14, levels = 1:148))

# SVM Classification for Non Linear Kernal - RBF
set.seed(3233)
svm_Radial <- train(V14 ~., data = training, method = "svmRadial",
                     trControl=trctrl,
                     preProcess = c("center", "scale"),
                     tuneLength = 10)

# Visualisation SVM -RBF Kernal
plot(svm_Radial, main= "SVM with RBF Kernal")

# Prediction of RBF trained model
test_pred_Radial <- predict(svm_Radial, newdata = testing)
confusionMatrix(factor(test_pred_Radial, levels = 1:148),
                factor(testing$V14, levels = 1:148))

# Tuning parameters of SVM - RBF
```

```
grid_radial <- expand.grid(sigma = c(0,0.01, 0.02, 0.025, 0.03, 0.04,
                                      0.05, 0.06, 0.07,0.08, 0.09, 0.1, 0.25, 0.5, 0.75,0.9
                           C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
                                 1, 1.5, 2,5))
set.seed(3233)
svm_Radial_Grid <- train(V14 ~., data = training, method = "svmRadial",
                         trControl=trctrl,
                         preProcess = c("center", "scale"),
                         tuneGrid = grid_radial,
                         tuneLength = 10)

#svm_Radial_Grid

# Visualisation
plot(svm_Radial_Grid, main="SVM RBF after tuning")
# Prediction with tuning
test_pred_Radial_Grid <- predict(svm_Radial_Grid, newdata = testing)

# Confusion Matrix
confusionMatrix(
  factor(test_pred_Radial_Grid, levels = 1:148),
  factor(testing$V14, levels = 1:148) )

txtStop()
```

Output

```
> library(caret)
> heart_df <- read.csv("/home/subalakshmi/PCP1211DALab/PCP1211ExptSevenB/heart_tidy.csv",
+ sep = ",", header = FALSE)
> str(heart_df)
'data.frame': 300 obs. of  14 variables:
 $ V1 : int  63 67 67 37 41 56 62 57 63 53 ...
 $ V2 : int  1 1 1 1 0 1 0 0 1 1 ...
 $ V3 : int  1 4 4 3 2 2 4 4 4 4 ...
 $ V4 : int  145 160 120 130 130 120 140 120 130 140 ...
 $ V5 : int  233 286 229 250 204 236 268 354 254 203 ...
 $ V6 : int  1 0 0 0 0 0 0 0 0 1 ...
 $ V7 : int  2 2 2 0 2 0 2 0 2 2 ...
 $ V8 : int  150 108 129 187 172 178 160 163 147 155 ...
 $ V9 : int  0 1 1 0 0 0 0 1 0 1 ...
 $ V10: num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ V11: int  3 2 2 3 1 1 3 1 2 3 ...
 $ V12: int  0 3 2 0 0 0 2 0 1 0 ...
 $ V13: int  6 3 7 3 3 3 3 3 7 7 ...
 $ V14: int  0 1 1 0 0 0 1 0 1 1 ...
> head(heart_df)
  V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
1 63  1  1 145 233  1  2 150  0 2.3   3   0   6   0
2 67  1  4 160 286  0  2 108  1 1.5   2   3   3   1
3 67  1  4 120 229  0  2 129  1 2.6   2   2   7   1
4 37  1  3 130 250  0  0 187  0 3.5   3   0   3   0
5 41  0  2 130 204  0  2 172  0 1.4   1   0   3   0
6 56  1  2 120 236  0  0 178  0 0.8   1   0   3   0
> set.seed(3033)
> intrain <- createDataPartition(y = heart_df$V14, p = 0.7, list = FALSE)
> training <- heart_df[intrain, ]
> testing <- heart_df[-intrain, ]
```

```
> dim(training)
[1] 210  14
> dim(testing)
[1] 90 14
> anyNA(heart_df)
[1] FALSE
> summary(heart_df)
       V1              V2              V3              V4              V5
 Min.   :29.00   Min.   :0.00   Min.   :1.000   Min.   : 94.0   Min.   :126.0
 1st Qu.:48.00   1st Qu.:0.00   1st Qu.:3.000   1st Qu.:120.0   1st Qu.:211.0
 Median :56.00   Median :1.00   Median :3.000   Median :130.0   Median :241.5
 Mean   :54.48   Mean   :0.68   Mean   :3.153   Mean   :131.6   Mean   :246.9
 3rd Qu.:61.00   3rd Qu.:1.00   3rd Qu.:4.000   3rd Qu.:140.0   3rd Qu.:275.2
 Max.   :77.00   Max.   :1.00   Max.   :4.000   Max.   :200.0   Max.   :564.0
       V6               V7               V8              V9               V10
 Min.   :0.0000   Min.   :0.0000   Min.   : 71.0   Min.   :0.0000   Min.   :0.00
 1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.8   1st Qu.:0.0000   1st Qu.:0.00
 Median :0.0000   Median :0.5000   Median :153.0   Median :0.0000   Median :0.80
 Mean   :0.1467   Mean   :0.9867   Mean   :149.7   Mean   :0.3267   Mean   :1.05
 3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:1.60
 Max.   :1.0000   Max.   :2.0000   Max.   :202.0   Max.   :1.0000   Max.   :6.20
      V11             V12             V13             V14
 Min.   :1.000   Min.   :0.00   Min.   :3.000   Min.   :0.00
 1st Qu.:1.000   1st Qu.:0.00   1st Qu.:3.000   1st Qu.:0.00
 Median :2.000   Median :0.00   Median :3.000   Median :0.00
 Mean   :1.603   Mean   :0.67   Mean   :4.727   Mean   :0.46
 3rd Qu.:2.000   3rd Qu.:1.00   3rd Qu.:7.000   3rd Qu.:1.00
 Max.   :3.000   Max.   :3.00   Max.   :7.000   Max.   :1.00
> training[["V14"]] = factor(training[["V14"]])
> trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
> set.seed(3233)
> svm_Linear <- train(V14 ~ ., data = training, method = "svmLinear",
+ trControl = trctrl, preProcess = c("center", "scale"), tuneLength = 10)
> svm_Linear
Support Vector Machines with Linear Kernel

210 samples
 13 predictor
  2 classes: '0', '1'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 189, 189, 189, 189, 189, 189, ...
Resampling results:

  Accuracy   Kappa
  0.7920635  0.581696

Tuning parameter 'C' was held constant at a value of 1
> test_pred <- predict(svm_Linear, newdata = testing)
> str(test_pred)
 Factor w/ 2 levels "0","1": 1 2 2 2 1 1 2 1 1 2 ...
> confusionMatrix(factor(test_pred, levels = 1:148), factor(testing$V14,
+ levels = 1:148))
Confusion Matrix and Statistics

Overall Statistics
```

```
               Accuracy : 1
                 95% CI : (0.8942, 1)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

> grid <- expand.grid(C = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
+ 1, 1.25, 1.5, 1.75, 2, 5))
> set.seed(3233)
> svm_Linear_Grid <- train(V14 ~ ., data = training, method = "svmLinear",
+ trControl = trctrl, preProcess = c("center", "scale"), tuneGrid = grid,
+ tuneLength = 10)
> plot(svm_Linear_Grid)
> test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
> test_pred_grid
 [1] 0 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1
[40] 1 0 0 1 0 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 0
[79] 0 1 0 1 1 0 1 0 0 0 1 0
Levels: 0 1
> confusionMatrix(factor(test_pred_grid, levels = 1:148), factor(testing$V14,
+ levels = 1:148))
Confusion Matrix and Statistics

Overall Statistics

               Accuracy : 1
                 95% CI : (0.8911, 1)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

> set.seed(3233)
> svm_Radial <- train(V14 ~ ., data = training, method = "svmRadial",
+ trControl = trctrl, preProcess = c("center", "scale"), tuneLength = 10)
> plot(svm_Radial)
> test_pred_Radial <- predict(svm_Radial, newdata = testing)
> confusionMatrix(factor(test_pred_Radial, levels = 1:148), factor(testing$V14,
+ levels = 1:148))
Confusion Matrix and Statistics
Overall Statistics

               Accuracy : 1
                 95% CI : (0.8911, 1)
    No Information Rate : 1
    P-Value [Acc > NIR] : 1

                  Kappa : NaN

 Mcnemar's Test P-Value : NA

> grid_radial <- expand.grid(sigma = c(0, 0.01, 0.02, 0.025, 0.03,
+ 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.25, 0.5, 0.75,
```

```
+ 0.9), C = c(0, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5,
+ 2, 5))
> set.seed(3233)
> svm_Radial_Grid <- train(V14 ~ ., data = training, method = "svmRadial",
+ trControl = trctrl, preProcess = c("center", "scale"), tuneGrid = grid_radial,
+ tuneLength = 10)
> svm_Radial_Grid
Support Vector Machines with Radial Basis Function Kernel

210 samples
 13 predictor
  2 classes: '0', '1'

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 189, 189, 189, 189, 189, 189, ...
Resampling results across tuning parameters:

  sigma  C     Accuracy   Kappa
  0.000  0.00        NaN          NaN
  0.000  0.01  0.5238095  0.000000000
  0.000  0.05  0.5238095  0.000000000
  0.000  0.10  0.5238095  0.000000000
  0.000  0.25  0.5238095  0.000000000
  0.000  0.50  0.5238095  0.000000000
  0.900  0.75  0.5492063  0.055825807
  0.900  1.00  0.5444444  0.055132187
  0.900  1.50  0.5555556  0.081488190
  0.900  2.00  0.5555556  0.081488190
  0.900  5.00  0.5555556  0.081488190

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.02 and C = 0.25.
> plot(svm_Radial_Grid)
> test_pred_Radial_Grid <- predict(svm_Radial_Grid, newdata = testing)
> confusionMatrix(factor(test_pred_Radial_Grid, levels = 1:148),
+ factor(testing$V14, levels = 1:148))
Confusion Matrix and Statistics
```

Result

Thus the implementation of Support Vector machine is executed successfully using R program.