SVM Classifier

Siddharth.S.Chandran (18BCE1003)

26/03/2021

1. Read the dataset and find the structure of it

```
library(readr)
data<-read_csv("C:/Users/Siddharth.S.Chandran/Desktop/diabetes.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Pregnancies = col_double(),
##   Glucose = col_double(),
##   BloodPressure = col_double(),
##   SkinThickness = col_double(),
##   Insulin = col_double(),
##   BMI = col_double(),
##   DiabetesPedigreeFunction = col_double(),
##   Age = col_double(),
##   Outcome = col_double()
## )
```

```
str(data)
```

```
## spec_tbl_df [768 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Pregnancies             : num [1:768] 6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : num [1:768] 148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : num [1:768] 72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : num [1:768] 35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : num [1:768] 0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num [1:768] 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num [1:768] 0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : num [1:768] 50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : num [1:768] 1 0 1 0 1 0 1 0 1 1 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Pregnancies = col_double(),
##   ..   Glucose = col_double(),
##   ..   BloodPressure = col_double(),
##   ..   SkinThickness = col_double(),
##   ..   Insulin = col_double(),
##   ..   BMI = col_double(),
##   ..   DiabetesPedigreeFunction = col_double(),
##   ..   Age = col_double(),
```

```
##   ..   Outcome = col_double()
##   .. )
```

Here we can see the outcome variable as categorical variable. We encode this features as a factor

```
data$Outcome = factor(data$Outcome, levels = c(0,1))
```

We divide the dataset into training and testing

```
library(caTools)
set.seed(123)
split = sample.split(data$Outcome, SplitRatio = 0.75)
training_set = subset(data, split == TRUE)
test_set = subset(data, split == FALSE)
head(training_set)
```

```
## # A tibble: 6 x 9
##   Pregnancies Glucose BloodPressure SkinThickness Insulin   BMI DiabetesPedigre~
##         <dbl>   <dbl>         <dbl>         <dbl>   <dbl> <dbl>            <dbl>
## 1           6     148            72            35       0  33.6            0.627
## 2           1      85            66            29       0  26.6            0.351
## 3           1      89            66            23      94  28.1            0.167
## 4           0     137            40            35     168  43.1            2.29
## 5          10     115             0             0       0  35.3            0.134
## 6           8     125            96             0       0   0              0.232
## # ... with 2 more variables: Age <dbl>, Outcome <fct>
```

The next step is to normalize the features of the training and testing data. This is done with the help of feature scaling.

```
cols<-c('Age', 'DiabetesPedigreeFunction','BMI', 'Insulin', 'SkinThickness','BloodPressure', 'Glucose',
training_set[cols] = scale(training_set[cols])
test_set[cols] = scale(test_set[cols])
```

Now we improve the predictive accuracy of the algorithm. After scaling the features we proceed to fitting the SVM classifier data to the training set

Linear kernel

```
library(e1071)
classifier = svm(formula = Outcome ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'linear')
summary(classifier)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = training_set, type = "C-classification",
##     kernel = "linear")
##
##
```

```
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  291
##
##  ( 146 145 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

The next line runs the classifier on the training set and test set so that the predictions can be made.

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
cols<-c('Age', 'DiabetesPedigreeFunction','BMI', 'Insulin', 'SkinThickness','BloodPressure', 'Glucose',
y_pred = predict(classifier, newdata = test_set[cols])
y_train_pred = predict(classifier, newdata = training_set[cols])
```

To find the accuracy of the model we will find the confusion matrix

```
cm1<-confusionMatrix(y_pred, test_set$Outcome)
cm1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 107  31
##          1  18  36
##
##                Accuracy : 0.7448
##                  95% CI : (0.677, 0.8048)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.003401
##
##                   Kappa : 0.4119
##
##  Mcnemar's Test P-Value : 0.086476
##
##             Sensitivity : 0.8560
##             Specificity : 0.5373
##          Pos Pred Value : 0.7754
##          Neg Pred Value : 0.6667
```

```
##              Prevalence : 0.6510
##          Detection Rate : 0.5573
##    Detection Prevalence : 0.7188
##       Balanced Accuracy : 0.6967
##
##        'Positive' Class : 0
##
```

```
cm2<-confusionMatrix(y_train_pred, training_set$Outcome)
cm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 339  87
##          1  36 114
##
##                Accuracy : 0.7865
##                  95% CI : (0.7507, 0.8193)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 9.281e-13
##
##                   Kappa : 0.5006
##
##  Mcnemar's Test P-Value : 6.533e-06
##
##             Sensitivity : 0.9040
##             Specificity : 0.5672
##          Pos Pred Value : 0.7958
##          Neg Pred Value : 0.7600
##              Prevalence : 0.6510
##          Detection Rate : 0.5885
##    Detection Prevalence : 0.7396
##       Balanced Accuracy : 0.7356
##
##        'Positive' Class : 0
##
```

Accruacy for the testing set -> 74.48% Accuaracy for the training set -> 78.65%

Radial kernel

```
library(e1071)
classifier = svm(formula = Outcome ~ .,
                 data = training_set,
                 type = 'C-classification',
                 kernel = 'radial')
summary(classifier)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = training_set, type = "C-classification",
```

```
##     kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  326
##
##  ( 159 167 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

The next line runs the classifier on the training set and test set so that the predictions can be made.

```
library(caret)
cols<-c('Age', 'DiabetesPedigreeFunction','BMI', 'Insulin', 'SkinThickness','BloodPressure', 'Glucose',
y_pred = predict(classifier, newdata = test_set[cols])
y_train_pred = predict(classifier, newdata = training_set[cols])
```

To find the accuracy of the model we will find the confusion matrix

```
cm1<-confusionMatrix(y_pred, test_set$Outcome)
cm1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106  29
##          1  19  38
##
##                Accuracy : 0.75
##                  95% CI : (0.6826, 0.8096)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 0.002076
##
##                   Kappa : 0.4301
##
##  Mcnemar's Test P-Value : 0.193931
##
##             Sensitivity : 0.8480
##             Specificity : 0.5672
##          Pos Pred Value : 0.7852
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6510
##          Detection Rate : 0.5521
##    Detection Prevalence : 0.7031
```

```
##       Balanced Accuracy : 0.7076
##
##        'Positive' Class : 0
##
```

```
cm2<-confusionMatrix(y_train_pred, training_set$Outcome)
cm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 345  76
##          1  30 125
##
##                Accuracy : 0.816
##                  95% CI : (0.7819, 0.8468)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5723
##
##  Mcnemar's Test P-Value : 1.238e-05
##
##             Sensitivity : 0.9200
##             Specificity : 0.6219
##          Pos Pred Value : 0.8195
##          Neg Pred Value : 0.8065
##              Prevalence : 0.6510
##          Detection Rate : 0.5990
##    Detection Prevalence : 0.7309
##       Balanced Accuracy : 0.7709
##
##        'Positive' Class : 0
##
```

Testing set accuracy -> 75% Training set accuracy -> 81.6%

Gaussian kernel