REPORT
HEART DISEASE RISK STUDY

Comparing classification models for predicting the risk of a heart attack.

KANAV MALIK

Contents

INTRODUCTION

Importance

In the United States one person dies every 36 seconds making heart disease one of the leading causes of death for men and women[1]. This disease has an estimated cost of 219 billion dollars each year in lost productivity, medicines, and health care costs[1]. The signs of a heart are silent leading the person to be unaware of it or the factors that lead up to it until it is to late. There are several different factors that can increase a person's risk of developing heart disease. Which is why applying statistical classification methods to these potential risk factors can be beneficial in helping doctors predict a patient's risk and lead to early detection of the disease. The analysis[2] was conducted using the open source data available from Cleveland Heart Dataset

Description

The Cleveland dataset consists of 76 diagnostics and histories of patients in relation to the presence or absence of heart disease. Of these 76 a subset of 13 diagnostic factors (Table 1) were chosen for the goal of building and comparing different classification models for predicting the outcome of heart disease.

Table 1. Summary and Description of 13 diagnostic factors in relation to the dependent variable of heart disease status. (Target 0 = Absence, 1 = Present)

Number	Predictor	Description
1	Age	Age of the patient in years
2	sex	Gender of the patient 1 = Male 0 = Female
3	cp	chest pain type (4 values) Value 1: typical angina Value 2: atypical angina Value 3: non-anginal pain Value 4: asymptomatic
4	trestbps	Resting blood pressure in mm/HG
5	chol	Serum cholesterol in mg/dl
6	fbs	Fasting blood sugar > 120 mg/dl, 1 = True 0 = False
7	restecg	Resting electrocardiographic results 0 = Normal 1 = Having ST-T wave abnormality 2 = Showing probable or definite left ventricular hypertrophy
8	thalach	Maximum heart rate achieved
9	exang	Exercise induced angina 1 = Yes 0 = No
10	oldpeak	ST depression induced by exercise relative to rest
11	slope	The slope of the peak exercise ST segment 1 = Upsloping 2 = Flat 3 = Downsloping
12	ca	number of major vessels (0-3) colored by fluoroscopy (4 missing values)
13	thal	Defect 3 = normal 6 = fixed defect 7 = reversable defect

METHODOLOGY

Part 1

Data Exploration / Logistic Regression

Data exploration will aide in summarizing the size, initial patterns, and accuracy of the heart dataset. By using correlation plot, bar graphs, and boxplots will help visualize possible relationships and interactions of the13 diagnostic factors with the target variable.

Part 2

Clustering

K-means unsupervised clustering would aid us in checking the optimal number of clusters for the observations.

K-Nearest Neighbors / Support Vector Machines

These 2 non-parametric distance-based approaches will be applied to the 80% sampled training dataset. The confusion matrix will be computed to compare the actual vs estimated results.

These methods work on scaled interval variables, so the interval variables have been scaled and the categorical variables have been converted to dummy variables. The results should make sense after the data has been modified. This is how the data looked before and after modification

ï..age	sex	cp	trestbps
47	1	3	108
51	1	3	125
53	1	4	123
55	1	4	140

→

ï..age	trestbps	sex.0	sex.1	cp.1	cp.2	cp.3	cp.4
-0.81358520	-1.30441437	0	1	0	0	1	0
-0.37163768	-0.36804014	0	1	0	0	1	0
-0.15066393	-0.47820181	0	1	0	0	0	1

Naïve Bayes Classifier using Kernel Density Estimate

It is a parametric based simple technique for constructing classifier which is based on a strong assumption that the value of a particular feature is independent of the value of any other feature , given the class variable. An advantage of this method is that it requires only a small training dataset to estimate the parameters necessary for classification.

DATA ANALYSIS

Part 1

Data Exploration / Logistic Regression

Clustering

Although, we are working on a supervised dataset, it is interesting to see the optimal number of clusters for our dataset.

The unsupervised K-means clustering gives the optimal number of clusters that are appropriate, given our dataset. Based on the data, the optimal number of clusters are 2.

Part 2

K-Nearest Neighbors

10-fold cross validation on the training dataset is done to get the optimal K as 10. The complete set of results can be seen in Appendix 2.

Support Vector Machines

10-fold cross validation on the training dataset is done to get the optimal parameters for “linear”, ”polynomial”, ”radial” density as 10. The complete set of results can be seen in Appendix 2.

Naïve Bayes Classifier

The complete set of results can be seen in Appendix 2.

Summary of results

Here is a summary of the results –

Method	Optimal Parameters	Train MSE	Test MSE	Test Sensitivity	Test Specificity
Logistic Regression	age, sex, cp, testbps, col, thalalach, slope, ca, thal		85.9%		
KNN	K = 10	86.9%	83.3%	88.6%	76.0%
SVM - Linear Kernel	cost = 0.1 , gamma = 0.05	86.5%	88.3%	91.4%	84.0%
SVM - Polynomial Kernel	cost = 0.1 ,degree = 1 , gamma = 0.05	86.5%	90.0%	97.1%	80.0%
SVM - Radial Kernel	cost = 0.85 , gamma = 0.05	94.5%	91.7%	97.1%	84.0%
Naive Bayes - KDE	prior prob(0) =0.525, prior prob(1) =0.475(comes from data)	84.4%	85.0%	85.7%	84.0%

CONCLUSION

The SVM radial kernel performs the best based on the Train and Test MSE's. It seems to be a good model for predicting the true positives and true negatives and scores better on all metrics of accuracy, sensitivity, specificity, false negatives. However, if we compare the models based on Specificity or False Positives rate, then logistic regression performs way better than any other model(91% specificity).

If the task if of predicting whether a person has the risk of heart attack, then we would like to have a low False positive rate i.e. low chances of incorrectly classifying patients at risk as not being at a risk of heart attack. So if the purpose is of having a low False positive rate, then logistic regression is the best model to use and if the purpose is increasing the sensitivity(low false negative rate), then SVM radial kernel is the best model to use. Finally, if the task is inference for research, then logistic regression is the only parametric method to use out of these methods.

APPENDIX

1. CDC. Heart Disease. In. (
2. Dua, D.a.G., Casey. (2017). {UCI} Machine Learning Repositor. In. (University of California, Irvine, School of Information and Computer Sciences.

Jimmy Teague, Kanav Malik

December 3,2020

Appendix 2

Jimmy Teague, Kanav Malik

December 3,2020

Libraries used

```
library(MASS)
library(klaR)
library(lattice)
library(ggplot2)
library(caret)
library(e1071)
library(cluster)
library(class)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(factoextra)
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
library(ggplot2)
```

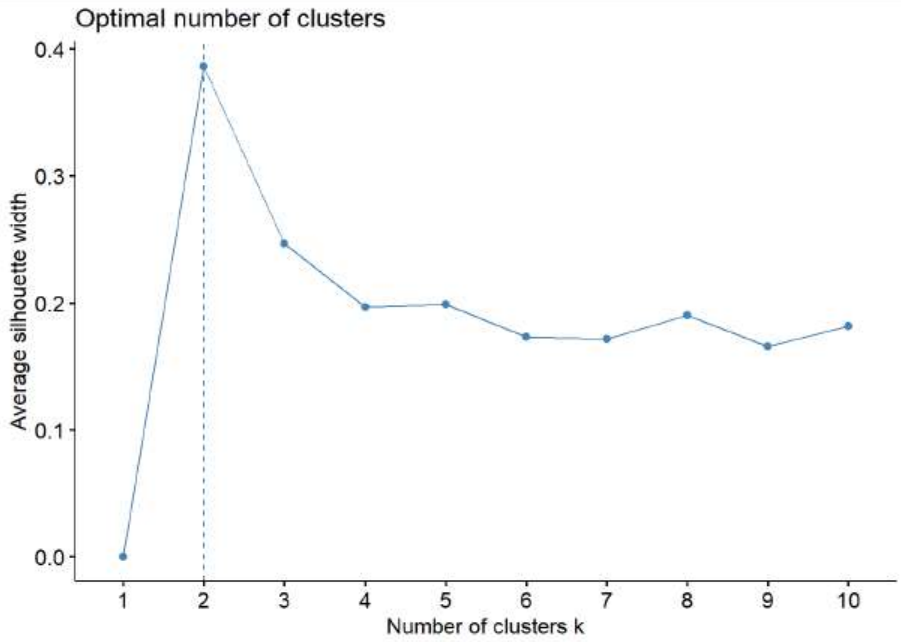
Heart Data

```
## 'data.frame':    297 obs. of  14 variables:
##  $ ï..age  : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : int  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : int  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : int  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : int  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : int  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : int  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : int  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : int  6 3 7 3 3 3 3 3 7 7 ...
##  $ target  : int  0 1 1 0 0 0 1 0 1 1 ...
```

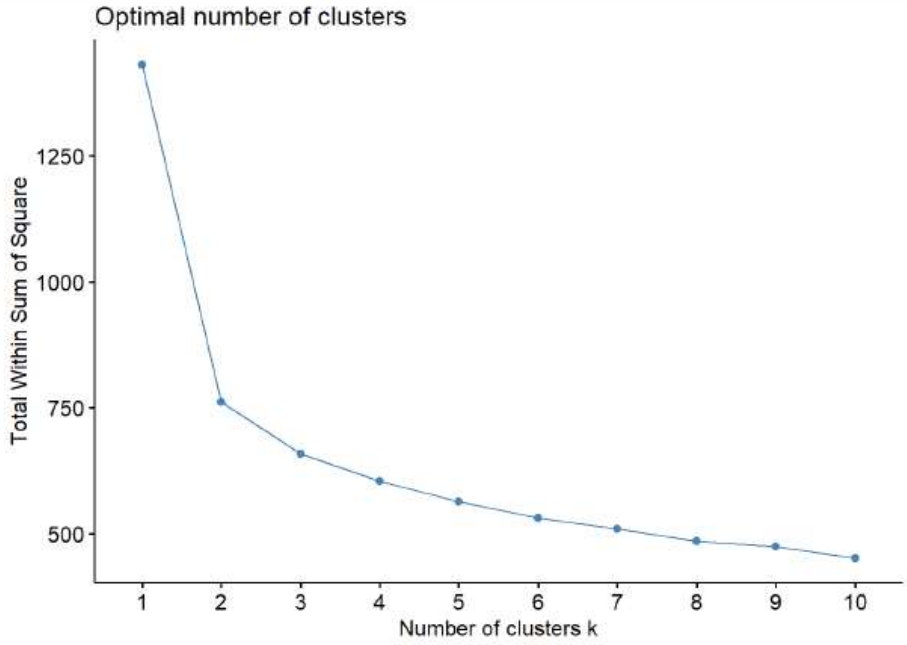
```
## 'data.frame':    297 obs. of  14 variables:
##  $ ï..age  : int  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: int  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : int  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach : int  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ target  : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 2 1 2 2 ...
```

Create a function to perform K means clustering for K= 1 to 5, and get TWSS

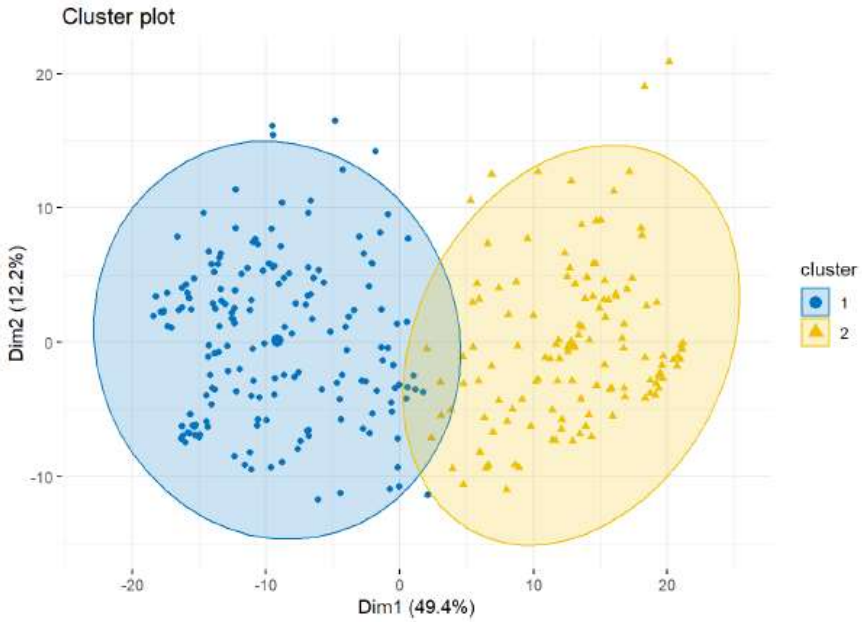
```
set.seed(2700)
disheart_1 <- daisy(Heart_Data,metric="gower")
heartmat1 <- as.matrix(disheart_1)

km.out2 <- eclust(heartmat1,FUNcluster = "kmeans",k=2,nstart=50,nboot=50)
```




```
fviz_nbclust(heartmat1,FUNcluster = kmeans,method = "wss")
```




```
fviz_cluster(km.out2,geom="point",ellipse.type = "norm",palette="jco",ggtheme = theme_minimal())
```



Validation set approach(Test/Train Split)

```
set.seed(2704)
roww <- nrow(Heart_Data)
coll <- ncol(Heart_Data)
numTrain <- floor(0.8 * roww) ##(Choose 80% data as training data)
numTest <- roww - numTrain
training <- Heart_Data[sample(roww, numTrain), ]
test <- Heart_Data[sample(roww, numTest), ]
```

Standardize the interval predictors of training and test data for knn and svm, and create dummy variables for the categorical variables

```
## Store original train,test data in a new train,test data which will be modified for knn,svm
training_knn_svm <- training
test_knn_svm <- test

## Standardize interval predictors of training,test data
training_knn_svm$ï..age <- scale(training_knn_svm$ï..age)
training_knn_svm$trestbps <- scale(training_knn_svm$trestbps)
training_knn_svm$chol <- scale(training_knn_svm$chol)
training_knn_svm$thalach <- scale(training_knn_svm$thalach)
training_knn_svm$oldpeak <- scale(training_knn_svm$oldpeak)
test_knn_svm$ï..age <- scale(test_knn_svm$ï..age)
test_knn_svm$trestbps <- scale(test_knn_svm$trestbps)
test_knn_svm$chol <- scale(test_knn_svm$chol)
test_knn_svm$thalach <- scale(test_knn_svm$thalach)
test_knn_svm$oldpeak <- scale(test_knn_svm$oldpeak)

## Create dummies for the categorical predictors
dummies_train <- dummyVars(~ ., data=training_knn_svm[,-c(1,4,5,8,10,14)])
c2_train <- predict(dummies_train, training_knn_svm[,-14])

dummies_test <- dummyVars(~ ., data=test_knn_svm[,-c(1,4,5,8,10,14)])
c2_test <- predict(dummies_test, test_knn_svm[,-14])

#Combine dummy and normalized data along with the target variable for train and test set
dum_norm_train <- as.data.frame(cbind(training_knn_svm[,c(14,1,4,5,8,10)], c2_train))
dum_norm_test <- as.data.frame(cbind(test_knn_svm[,c(14,1,4,5,8,10)], c2_test))
```

KNN

```
train.Predictors <- data.frame(dum_norm_train[,c(-1)]) ## training predictors
train.Response <- dum_norm_train[,"target"] ## training response variable
test.Predictors <- data.frame(dum_norm_test[,c(-1)]) ## testing predictors
test.Response <- dum_norm_test[,"target"] ## testing response variable

knn.best <- tune.knn(train.Predictors,train.Response,k=seq(2,50,by=1))
knn.best
```

```
##
## Parameter tuning of 'knn.wrapper':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  k
##  9
##
## - best performance: 0.1641304
```

```
knn.pred_train <- knn(train.Predictors,train.Predictors,train.Response,k=knn.best$best.parameters[,1])
knn.pred_test <- knn(train.Predictors,test.Predictors,train.Response,k=knn.best$best.parameters[,1])
confusionMatrix(knn.pred_train,as.factor(dum_norm_train$target))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 114  22
##          1   9  92
##
##                Accuracy : 0.8692
##                  95% CI : (0.8195, 0.9094)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7369
##
##  Mcnemar's Test P-Value : 0.03114
##
##             Sensitivity : 0.9268
##             Specificity : 0.8070
##          Pos Pred Value : 0.8382
##          Neg Pred Value : 0.9109
##              Prevalence : 0.5190
##          Detection Rate : 0.4810
##    Detection Prevalence : 0.5738
##       Balanced Accuracy : 0.8669
##
##        'Positive' Class : 0
##
```

```
paste("Train Accuracy Rate =",round(1 - mean(knn.pred_train != dum_norm_train$target),4)*100,"%")
```

```
## [1] "Train Accuracy Rate = 86.92 %"
```

```
confusionMatrix(knn.pred_test,as.factor(dum_norm_test$target))
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##         0 31  6
##         1  4 19
##
##               Accuracy : 0.8333
##                 95% CI : (0.7148, 0.9171)
##    No Information Rate : 0.5833
##    P-Value [Acc > NIR] : 3.191e-05
##
##                  Kappa : 0.6532
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.8857
##             Specificity : 0.7600
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.8261
##              Prevalence : 0.5833
##          Detection Rate : 0.5167
##    Detection Prevalence : 0.6167
##       Balanced Accuracy : 0.8229
##
##        'Positive' Class : 0
##
```

```
paste("Test Accuracy Rate =",round(1 - mean(knn.pred_test != dum_norm_test$target),4)*100,"%")
```

```
## [1] "Test Accuracy Rate = 83.33 %"
```

SVM

```
set.seed(403)

## Linear kernel
tune.out.linear <- tune.svm(target ~., data=dum_norm_train, kernel='linear', cost=seq(0.1,15,by=0.25),
                            gamma = seq(0.05,10,by=0.85))
summary(tune.out.linear$best.model)
```

```
##
## Call:
## best.svm(x = target ~ ., data = dum_norm_train, gamma = seq(0.05,
##     10, by = 0.85), cost = seq(0.1, 15, by = 0.25), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.05
##
## Number of Support Vectors:  89
##
##  ( 44 45 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
svm_predict_train <- predict(tune.out.linear$best.model, dum_norm_train[,-1])
svm_predict_test <- predict(tune.out.linear$best.model, dum_norm_test[,-1])

confusionMatrix(svm_predict_train, dum_norm_train$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 114  23
##          1   9  91
##
##                Accuracy : 0.865
##                  95% CI : (0.8148, 0.9058)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7283
##
##  Mcnemar's Test P-Value : 0.02156
##
##             Sensitivity : 0.9268
##             Specificity : 0.7982
##          Pos Pred Value : 0.8321
##          Neg Pred Value : 0.9100
##              Prevalence : 0.5190
##          Detection Rate : 0.4810
##    Detection Prevalence : 0.5781
##       Balanced Accuracy : 0.8625
##
##        'Positive' Class : 0
##
```

```
paste("Train accuracy rate for linear kernel =" ,
      round(1-mean(svm_predict_train != dum_norm_train[,"target"]),4)*100,"%")
```

```
## [1] "Train accuracy rate for linear kernel = 86.5 %"
```

```
confusionMatrix(svm_predict_test, dum_norm_test$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 32  4
##          1  3 21
##
##                Accuracy : 0.8833
##                  95% CI : (0.7743, 0.9518)
##     No Information Rate : 0.5833
##     P-Value [Acc > NIR] : 4.007e-07
##
##                   Kappa : 0.7586
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9143
##             Specificity : 0.8400
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.8750
##              Prevalence : 0.5833
##          Detection Rate : 0.5333
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.8771
##
##        'Positive' Class : 0
##
```

```
paste("Test accuracy rate for linear kernel =" ,
      round(1-mean(svm_predict_test != dum_norm_test[,"target"]),4)*100,"%")
```

```
## [1] "Test accuracy rate for linear kernel = 88.33 %"
```

```
#Polynomial kernel
tune.out.poly <- tune.svm(target ~., data=dum_norm_train,
                    kernel='polynomial', cost=seq(0.1,15,by=0.25), gamma = seq(0.05,10,by=0.85),
                    degree=seq(1,10,by = 1))
summary(tune.out.poly$best.model)
```

```
##
## Call:
## best.svm(x = target ~ ., data = dum_norm_train, degree = seq(1,
##     10, by = 1), gamma = seq(0.05, 10, by = 0.85), cost = seq(0.1,
##     15, by = 0.25), kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.1
##      degree:  1
##       gamma:  0.05
##      coef.0:  0
##
## Number of Support Vectors:  142
##
##  ( 71 71 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
svm_predict_train <- predict(tune.out.poly$best.model, dum_norm_train[,-1])
svm_predict_test <- predict(tune.out.poly$best.model, dum_norm_test[,-1])
confusionMatrix(svm_predict_train, dum_norm_train$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 117  26
##          1   6  88
##
##                Accuracy : 0.865
##                  95% CI : (0.8148, 0.9058)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7278
##
##  Mcnemar's Test P-Value : 0.0007829
##
##             Sensitivity : 0.9512
##             Specificity : 0.7719
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.9362
##              Prevalence : 0.5190
##          Detection Rate : 0.4937
##    Detection Prevalence : 0.6034
##       Balanced Accuracy : 0.8616
##
##        'Positive' Class : 0
##
```

```
paste("Train accuracy rate for polynomial kernel =" ,
      round(1-mean(svm_predict_train != dum_norm_train[,"target"]),4)*100,"%")
```

```
## [1] "Train accuracy rate for polynomial kernel = 86.5 %"
```

```
confusionMatrix(svm_predict_test, dum_norm_test$target)
```

```
## Confusion Matrix and Statistics
##
##          Reference
## Prediction  0  1
##          0 34  5
##          1  1 20
##
##                Accuracy : 0.9
##                  95% CI : (0.7949, 0.9624)
##     No Information Rate : 0.5833
##     P-Value [Acc > NIR] : 7.037e-08
##
##                   Kappa : 0.7895
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.9714
##             Specificity : 0.8000
##          Pos Pred Value : 0.8718
##          Neg Pred Value : 0.9524
##              Prevalence : 0.5833
##          Detection Rate : 0.5667
##    Detection Prevalence : 0.6500
##       Balanced Accuracy : 0.8857
##
##        'Positive' Class : 0
##
```

```
paste("Test accuracy rate for polynomial kernel =" ,
      round(1-mean(svm_predict_test != dum_norm_test[,"target"]),4)*100,"%")
```

```
## [1] "Test accuracy rate for polynomial kernel = 90 %"
```

```
## Radial kernel
tune.out.radial <- tune.svm(target ~., data=dum_norm_train,
                kernel='radial', cost=seq(0.1,15,by=0.25), gamma = seq(0.05,10,by=0.85))
summary(tune.out.radial$best.model)
```

```
##
## Call:
## best.svm(x = target ~ ., data = dum_norm_train, gamma = seq(0.05,
##     10, by = 0.85), cost = seq(0.1, 15, by = 0.25), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.85
##       gamma:  0.05
##
## Number of Support Vectors:  160
##
##  ( 83 77 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

```
svm_predict_train <- predict(tune.out.radial$best.model, dum_norm_train[,-1])
svm_predict_test <- predict(tune.out.radial$best.model, dum_norm_test[,-1])
confusionMatrix(svm_predict_train, dum_norm_train$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 123  13
##          1   0 101
##
##                Accuracy : 0.9451
##                  95% CI : (0.908, 0.9705)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8897
##
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 1.0000
##             Specificity : 0.8860
##          Pos Pred Value : 0.9044
##          Neg Pred Value : 1.0000
##              Prevalence : 0.5190
##          Detection Rate : 0.5190
##    Detection Prevalence : 0.5738
##       Balanced Accuracy : 0.9430
##
##        'Positive' Class : 0
##
```

```
paste("Train accuracy rate for radial kernel =" ,
      round(1-mean(svm_predict_train != dum_norm_train[,"target"]),4)*100,"%")
```

```
## [1] "Train accuracy rate for radial kernel = 94.51 %"
```

```
confusionMatrix(svm_predict_test, dum_norm_test$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 34  4
##          1  1 21
##
##                Accuracy : 0.9167
##                  95% CI : (0.8161, 0.9724)
##     No Information Rate : 0.5833
##     P-Value [Acc > NIR] : 1.042e-08
##
##                   Kappa : 0.8256
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.9714
##             Specificity : 0.8400
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.9545
##              Prevalence : 0.5833
##          Detection Rate : 0.5667
##    Detection Prevalence : 0.6333
##       Balanced Accuracy : 0.9057
##
##        'Positive' Class : 0
##
```

```
paste("Test accuracy rate for radial kernel =" ,
      round(1-mean(svm_predict_test != dum_norm_test[,"target"]),4)*100,"%")
```

```
## [1] "Test accuracy rate for radial kernel = 91.67 %"
```

Naive Bayes

```
### KDC ###
nb1 <- NaiveBayes(target ~.,data=training, usekernel=T)

pred_train <- predict(nb1, training[,1:13])
pred_test <- predict(nb1, test[,1:13])

confusionMatrix(pred_train$class,training$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 108  22
##          1  15  92
##
##                Accuracy : 0.8439
##                  95% CI : (0.7913, 0.8876)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6866
##
##  Mcnemar's Test P-Value : 0.3239
##
##             Sensitivity : 0.8780
##             Specificity : 0.8070
##          Pos Pred Value : 0.8308
##          Neg Pred Value : 0.8598
##              Prevalence : 0.5190
##          Detection Rate : 0.4557
##    Detection Prevalence : 0.5485
##       Balanced Accuracy : 0.8425
##
##        'Positive' Class : 0
##
```

```
paste("Train Accuracy Rate =",round(1 - mean(pred_train$class != training$target),4)*100,"%")
```

```
## [1] "Train Accuracy Rate = 84.39 %"
```

```
confusionMatrix(pred_test$class,test$target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 30  4
##          1  5 21
##
##                Accuracy : 0.85
##                  95% CI : (0.7343, 0.929)
##     No Information Rate : 0.5833
##     P-Value [Acc > NIR] : 8.415e-06
##
##                   Kappa : 0.6932
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8571
##             Specificity : 0.8400
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.8077
##              Prevalence : 0.5833
##          Detection Rate : 0.5000
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8486
##
##        'Positive' Class : 0
##
```

```
paste("Test Accuracy Rate =",round(1 - mean(pred_test$class != test$target),4)*100,"%")
```

```
## [1] "Test Accuracy Rate = 85 %"
```