Project 5

Support Vector Machine

Brooke O'Donnell

November 22nd, 2020

In this project we will train some support vector machines using "e1071" and "tidymodels" ("kernlab" as the engine) to predict whether patients have heart disease or not. The variables in the data set of "heart.csv" are listed below; and the target variable is "target".

- age : age in years sex : (1 = male; 0 = female) *cp : chest pain type
- trestbps : resting blood pressure (in mm Hg on admission to the hospital)
- chol : serum cholestoral in mg/dl
- fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg : resting electrocardiographic results
- thalach : maximum heart rate achieved
- exang : exercise induced angina (1 = yes; 0 = no)
- oldpeak : ST depression induced by exercise relative to rest
- slope : the slope of the peak exercise ST segment
- ca : number of major vessels (0–3) colored by flourosopy
- thal : 1 = normal; 2 = fixed defect; 3 = reversable defect
- target : 1 = disease; 0 = no disease

Note that variables such as "sex" and "exang" are already dummy variables; there is no need to recode them. Please also treat the "thal" variable as numeric.

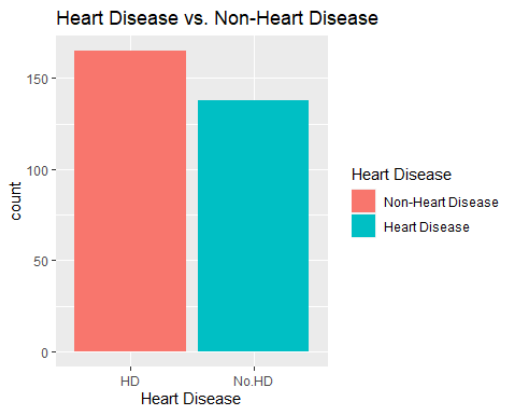
```
setwd("C:/Data Mining/Data Sets")

mydf1<-mydf1 %>% mutate(target=recode_factor(target, '1'="HD",
                                '0'="No.HD"))
prop.table(table(mydf1$target))
      HD     No.HD
 0.5445545 0.4554455

# Acceptable—both close to .5

# Barplot for target varaible

ggplot(mydf1, aes(x=target, fill=target))+
  geom_bar()+
  xlab("Heart Disease")+
  ylab("count")+
  ggtitle("Heart Disease vs. Non-Heart Disease")+
  scale_fill_discrete(name= "Heart Disease", labels=c("Non-Heart Disease", "Heart Disease"))
```




```
# Normalize feature variables
norm.func<-function(x){(x-min(x))/(max(x)-min(x))}

mydf2<-as.data.frame(lapply(mydf1[,-14], norm.func))

mydf2$target<-mydf1$target

library(rsample)
set.seed(123)
df.split<-initial_split(mydf2, prop=.7, strata=target)

df.train<-training(df.split)
df.test<-testing(df.split)
```

1. Tune the "cost" parameter for an SVM with a linear kernel using the "e1071" package; and estimate the best model based on the optimal "cost."
   - What is the optimal "cost?
     - The optimal cost is 0.5.

```
set.seed(123)
tune.1<-e1071::tune(svm, target~.,
          data=df.train,
          kernel='linear',
           ranges=list(cost=c(0.01, 0.1, 0.5, 1, 3, 5, 10, 100)))
summary(tune.1)
 Parameter tuning of 'svm':
 - sampling method: 10-fold cross validation
 - best parameters:
  cost
   0.5
 - best performance: 0.1170996
 - Detailed performance results:
   cost     error dispersion
 1 1e-02 0.1590909 0.08187841
 2 1e-01 0.1357143 0.09663828
 3 5e-01 0.1170996 0.08086250
```

```
4 1e+00 0.1218615 0.08389319
5 3e+00 0.1218615 0.08389319
6 5e+00 0.1218615 0.08389319
7 1e+01 0.1218615 0.08389319
8 1e+02 0.1173160 0.08693290
```

- How many support vectors are there in your machine?
  - There are 78 support vectors in this machine.

```
library(e1071)
library(caret)
set.seed(123)
model.1<-svm(target~.,
             data=df.train,
             kernel='linear',
             cost=0.5,
             scale=TRUE)

summary(model.1)
Call:
 svm(formula = target ~ ., data = df.train, kernel = "linear", cost = 0.5,
    scale = TRUE)
 Parameters:
    SVM-Type:  C-classification
  SVM-Kernel:  linear
        cost:  0.5
 Number of Support Vectors:  78
  ( 39 39 )
 Number of Classes:  2
 Levels:
  HD No.HD

# Prediction

pred.yltr<-predict(model.1, df.train)
pred.ylts<-predict(model.1, df.test)
```

- Cross-validate the model by obtaining confusion matrices for both training and testing sets?

```
confusionMatrix(pred.yltr, df.train$target)

Confusion Matrix and Statistics
          Reference
 Prediction  HD No.HD
     HD    108    16
     No.HD   8    81
               Accuracy : 0.8873
                 95% CI : (0.837, 0.9265)
```

```
    No Information Rate : 0.5446
    P-Value [Acc > NIR] : <2e-16
                  Kappa : 0.7713
 Mcnemar's Test P-Value : 0.153
            Sensitivity : 0.9310
            Specificity : 0.8351
         Balanced Accuracy : 0.8830
          'Positive' Class : HD
```

```
confusionMatrix(pred.ylts, df.test$target)
```

```
 Confusion Matrix and Statistics
           Reference
 Prediction HD No.HD
     HD    38     8
     No.HD 11    33
               Accuracy : 0.7889
                 95% CI : (0.6901, 0.8679)
    No Information Rate : 0.5444
    P-Value [Acc > NIR] : 1.206e-06
                  Kappa : 0.5769
 Mcnemar's Test P-Value : 0.6464
            Sensitivity : 0.7755
            Specificity : 0.8049
         Balanced Accuracy : 0.7902
          'Positive' Class : HD
```

- Judging from the two sets of "accuracy," and "sensitivity" do you think overfitting is a concern? (Please show only necessary output, not the two matrices in their entirety)
  - The training set preforms better in 'accuracy' and 'sensitivity' than the testing set therefore there might be problems of overfitting.

'Linear'	Training	Testing
Accuracy	0.8873	0.7889
Sensitivity	0.9310	0.7755

2. Tune both "cost" and "degree" parameters for an SVM with a polynomial kernel using the "e1071" package; and estimate the best model based on the optimal "cost" and "degree" pair.
   - What are the values of the optimal "cost" and "degree?"
     - optimal cost=3 | optimal degree=3

```
library(e1071)
set.seed(123)
tune.2<-e1071::tune(svm, target~.,
          data=df.train,
          kernel='polynomial',
          ranges=list(cost=c(0.01, 0.1, 0.5, 1, 3, 5, 10, 100),
```

```
                    degree=c(2,3,4)))
summary(tune.2)
 Parameter tuning of 'svm':
 - sampling method: 10-fold cross validation
  - best parameters:
  cost degree
     3      3
 - best performance: 0.1640693
 - Detailed performance results:
    cost degree     error dispersion
1  1e-02      2 0.4543290 0.11511702
2  1e-01      2 0.4448052 0.11264311
3  5e-01      2 0.3554113 0.09918365
4  1e+00      2 0.3510823 0.11842638
5  3e+00      2 0.3329004 0.06925505
6  5e+00      2 0.3329004 0.06925505
7  1e+01      2 0.3426407 0.05389615
8  1e+02      2 0.3989177 0.11266529
9  1e-02      3 0.4543290 0.11511702
10 1e-01      3 0.2943723 0.12396802

etc…
```

- How many support vectors are there in your machine?
  - There are 124 support vectors in this machine.

```
library(e1071)
set.seed(123)
model.op.2<-e1071::svm(target~.,
           data=df.train,
           kernel='polynomial',
           degree=3,
           cost=3)
summary(model.op.2)
 Call:
 svm(formula = target ~ ., data = df.train, kernel = "polynomial",
    degree = 3, cost = 3)
 Parameters:
    SVM-Type:  C-classification
  SVM-Kernel:  polynomial
        cost:  3
      degree:  3
      coef.0:  0
 Number of Support Vectors:  124
 ( 67 57 )
 Number of Classes:  2
 Levels:   HD No.HD
```

```
# Prediction

pred.2.tr<-predict(model.op.2, df.train)
pred.2.ts<-predict(model.op.2, df.test)
```

– Cross-validate the model by obtaining confusion matrices for both training and testing sets?

```
confusionMatrix(pred.2.tr, df.train$target)

 Confusion Matrix and Statistics
          Reference
 Prediction  HD No.HD
     HD    114     6
     No.HD   2    91
               Accuracy : 0.9624
                 95% CI : (0.9273, 0.9836)
    No Information Rate : 0.5446
    P-Value [Acc > NIR] : <2e-16
                  Kappa : 0.924
 Mcnemar's Test P-Value : 0.2888
            Sensitivity : 0.9828
            Specificity : 0.9381
         Balanced Accuracy : 0.9605
          'Positive' Class : HD

confusionMatrix(pred.2.ts, df.test$target)

 Confusion Matrix and Statistics
          Reference
 Prediction HD No.HD
     HD    39    10
     No.HD 10    31
               Accuracy : 0.7778
                 95% CI : (0.6779, 0.8587)
    No Information Rate : 0.5444
    P-Value [Acc > NIR] : 3.689e-06
                  Kappa : 0.552
 Mcnemar's Test P-Value : 1
            Sensitivity : 0.7959
            Specificity : 0.7561
         Balanced Accuracy : 0.7760
          'Positive' Class : HD
```

– Judging from the two sets of "accuracy," and "sensitivity" do you think overfitting is a concern? (Please show only necessary output, not the two matrices in their entirety)

- Overfitting is still a concern because the testing set preforms worse compared to the training set when comparing the 'accuracy' and 'sensitivity'.

'Polynomial'	Training	Testing

Accuracy	0.9624	0.7778
Sensitivity	0.9828	0.7959

3. Tune both "cost" and "gamma" parameters for an SVM with a radial kernel using the "e1071" package; and estimate the best model based on the optimal "cost" and "gamma" pair.
   - What are the values of the optimal "cost" and "gamma?"
     - optimal cost=100 | optimal gamma=0.001

```
library(e1071)
set.seed(123)
tune.rad<-e1071::tune(svm, target~.,
          data=df.train,
          kernel='radial',
          ranges=list(cost=c(0.01, 0.1, 0.5, 1, 3, 5, 10, 100), gamma=c(0.001, 0.002, 0.005, 0.01)))

summary(tune.rad)
 Parameter tuning of 'svm':
 - sampling method: 10-fold cross validation
 - best parameters:
  cost gamma
   100 0.001
- best performance: 0.1218615
 - Detailed performance results:
    cost gamma     error dispersion
1  1e-02 0.001 0.4543290 0.11511702
2  1e-01 0.001 0.4543290 0.11511702
3  5e-01 0.001 0.4543290 0.11511702
4  1e+00 0.001 0.2051948 0.10206520
5  3e+00 0.001 0.1683983 0.08721389
6  5e+00 0.001 0.1590909 0.08187841
7  1e+01 0.001 0.1497835 0.08412092
8  1e+02 0.001 0.1218615 0.08606781
9  1e-02 0.002 0.4543290 0.11511702
10 1e-01 0.002 0.4543290 0.11511702
etc..
```

- How many support vectors are there in your SV machine?
  - 88

```
library(e1071)
set.seed(123)
model.rad<-e1071::svm(target~.,
          data=df.train,
          kernel='radial',
          gamma=0.001,
          cost=100)
```

```
summary(model.rad)
 Call:
 svm(formula = target ~ ., data = df.train, kernel = "radial", gamma = 0.001,
   cost = 100)
 Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
 Number of Support Vectors:  88

 ( 45 43 )
Number of Classes:  2
 Levels:  HD No.HD

# Prediction
pred.rad.tr<-predict(model.rad, df.train)
pred.rad.ts<-predict(model.rad, df.test)
```

– Cross-validate the model by obtaining confusion matrices for both training and testing sets?

```
confusionMatrix(pred.rad.tr, df.train$target)

 Confusion Matrix and Statistics
          Reference
 Prediction  HD No.HD
     HD    109    16
     No.HD   7    81
            Accuracy : 0.892
              95% CI : (0.8424, 0.9303)
    No Information Rate : 0.5446
    P-Value [Acc > NIR] : < 2e-16
                 Kappa : 0.7806
 Mcnemar's Test P-Value : 0.09529
            Sensitivity : 0.9397
            Specificity : 0.8351
         Balanced Accuracy : 0.8874
          'Positive' Class : HD

confusionMatrix(pred.rad.ts, df.test$target)

 Confusion Matrix and Statistics
          Reference
 Prediction HD No.HD
     HD    39     8
     No.HD 10    33
            Accuracy : 0.8
              95% CI : (0.7025, 0.8769)
    No Information Rate : 0.5444
```

```
 P-Value [Acc > NIR] : 3.697e-07
               Kappa : 0.5984
Mcnemar's Test P-Value : 0.8137
          Sensitivity : 0.7959
          Specificity : 0.8049
       Balanced Accuracy : 0.8004
       'Positive' Class : HD
```

- Judging from the two sets of "accuracy," and "sensitivity" do you think overfitting is a concern? (Please show only necessary output, not the two matrices in their entirety)
  - Overfitting is a concern because the training set preforms better than the testing set.

'Radial'	Training	Testing
Accuracy	0.892	0.8
Sensitivity	0.9397	0.7959

- Compare the 3 "best" SVM's obtained above, which would you adopt as a classifier? Please justify your decision. (Please keep in mind that all "best" classifiers involve some trade-off)

LIN-Training accuracy= 0.8873
LIN-Testing accuracy=0.7889
LIN-Training sensitivity=0.9310
LIN-Testing sensitivity=0.7755

POLY-Training accuracy=0.9624
POLY-Testing accuracy=0.7778
POLY-Training sensitivity=0.9828
POLY-Testing sensitivity=0.7959

RAD-Training accuracy=0.892
RAD-Testing accuracy=0.8
RAD-Training sensitivity=0.9397
RAD-Testing sensitivity=0.7959

- I will adopt the 'radial' SVM as my classifier because it has a high accuracy and sensitivity. Although it does not produce the highest accuracy and sensitivity from the three models it has less variance between the training and testing set. This means avoiding the problem of overfitting is more likely using the radial SVM compared to the linear and polynomial SVM.

4. Design a "tidymodels" procedure to tune SVM with a radial kernel (both "cost" and "gamma"). Please note that "tidymodels" uses "kernlab" package as the engine; and the equivalent of "gamma" in "kernlab" package is "rbf_sigma."

```
library(kernlab)
```

```
k_fold<-vfold_cv(df.train)
model.rec<-recipe(target~., df.train) %>%
  step_range(all_numeric())

m.metrics<-metric_set(accuracy, sens, spec, roc_auc)
m.control<-control_grid(save_pred=TRUE)

svm.mod<-svm_rbf(cost=tune(), rbf_sigma=tune()) %>% set_mode("classification") %>%
  set_engine("kernlab")

set.seed(123)
svm.tune<-tune_grid(svm.mod,
                    model.rec,
                    resamples=k_fold,
                    control=m.control,
                    metrics=m.metrics)
```

– Please identify the optimal model based on the criterion of "accuracy",

```
svm.tune %>% select_best(metric="accuracy")
```

cost	rbf_sigma	.config
9.957288	0.07305629	Model05

- Based on accuracy the optimal cost=9.96 | gamma=0.07

– estimate the model,

```
svm.mod2<-svm_rbf(cost=9.957288, rbf_sigma=0.07305629) %>%
  set_mode("classification") %>%
  set_engine("kernlab")

best.model<-workflow() %>%
  add_model(svm.mod2) %>%
  add_recipe(model.rec)

best.modr<-last_fit(best.model, df.split) %>%
  collect_predictions() %>%
  mutate(pred=if_else(.pred_HD>=0.5, "HD", "No.HD"),
         pred=as.factor(pred))
```

– obtain the confusion matrix using the testing set.

```
confusionMatrix(best.modr$pred, best.modr$target)

 Confusion Matrix and Statistics
           Reference
 Prediction HD No.HD
```

```
    HD    39     9
    No.HD 10    32
          Accuracy : 0.7889
            95% CI : (0.6901, 0.8679)
   No Information Rate : 0.5444
   P-Value [Acc > NIR] : 1.206e-06
             Kappa : 0.5753
Mcnemar's Test P-Value : 1
          Sensitivity : 0.7959
          Specificity : 0.7805
       Balanced Accuracy : 0.7882
       'Positive' Class : HD
```

– Present the “accuracy,” “sensitivity,” and “specificity” for the optimal model applied to the testing set. Please show your r-code, the optimal parameters after tuning, and the metrics from the confusion matrix.
  - Accuracy : 0.7889
  - Sensitivity : 0.7959
  - Specificity : 0.7805