

MODEL 1 SUPPORT VECTOR MACHINE MODEL

JUDISMA SALI

2022-12-16

```
# Helper packages
library(dplyr) # for data wrangling

##
## Attaching package: 'dplyr'

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

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

library(ggplot2) # for awesome graphics
library(rsample) # for data splitting
library(readr)   #load dataset

# Modeling packages
library(caret) # for classification and regression training

## Loading required package: lattice
library(kernlab) # for fitting SVMs

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##   alpha

library(modeldata) #for Failure.binary data
library(forcats)
library(pdp)       # for partial dependence plots, etc.
library(vip)       # for variable importance plots

##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##   vi
```

MODEL 1 SVM

Support Vector Machine (SVM) is a supervised learning technique that analyzes data and isolates patterns applicable to both classification and regression. The classifier is useful for choosing between two or more possible outcomes that depend on continuous or categorical predictor variables. Based on training and sample classification data, the SVM algorithm assigns the target data into any one of the given categories. The data is represented as points in space and categories are mapped in both linear and non-linear ways.

LOAD THE REPROCESSED DATASET

Note that we used the reprocessed data of radiomics_complete.csv (*RAD. NORMAL DATA.CSV*) in performing support vectore machine.

Radiomics Dataset 197 Rows (Observations) of 431 Columns (Variables) Failure.binary: binary property to predict

```
radiomicsdt<- read_csv("RAD. NORMAL DATA.CSV")

## Rows: 197 Columns: 431
## -- Column specification -----
## Delimiter: ","
## chr  (1): Institution
## dbl (430): Failure.binary, Failure, Entropy_cooc.W.ADC, GLNU_align.H.PET, Mi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
View(radiomicsdt)
head(radiomicsdt)
```

```
## # A tibble: 6 x 431
##   Institution Failure.~1 Failure Entro~2 GLNU~3 Min_h~4 Max_h~5 Mean~6 Varia~7
##   <chr>          <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 A              0    1.15    12.9  -0.433  -0.270  -0.257  -0.192  0.0509
## 2 A              1   -0.533   12.2  -1.02    0.671   0.405    0.490  0.687
## 3 A              0    2.24    12.8   0.179  -1.41   -1.57   -1.53  -1.57
## 4 A              1   -0.140   13.5   2.00   -0.218  0.0764  -0.153  0.0127
## 5 A              0    0.787   12.6   0.153  -1.06   -1.15   -1.45  -1.91
## 6 A              1   -2.80    13.2   0.391  -1.57   -1.91   -1.72  -1.84
## # ... with 422 more variables: Standard_Deviation_hist.PET <dbl>,
## #   Skewness_hist.PET <dbl>, Kurtosis_hist.PET <dbl>, Energy_hist.PET <dbl>,
## #   Entropy_hist.PET <dbl>, AUC_hist.PET <dbl>, H_suv.PET <dbl>,
## #   Volume.PET <dbl>, X3D_surface.PET <dbl>, ratio_3ds_vol.PET <dbl>,
## #   ratio_3ds_vol_norm.PET <dbl>, irregularity.PET <dbl>,
## #   tumor_length.PET <dbl>, Compactness_v1.PET <dbl>, Compactness_v2.PET <dbl>,
## #   Spherical_disproportion.PET <dbl>, Sphericity.PET <dbl>, ...
```

```
# Load Failure.binary data
```

```
radiomicsdt$Failure.binary=as.factor(radiomicsdt$Failure.binary)
```

Create training (80%) and test (20%) sets

```
set.seed(123) # for reproducibility
```

```
churn_split <- initial_split(radiomicsdt, prop = 0.8, strata = "Failure.binary")
split_train <- training(churn_split)
split_test <- testing(churn_split)
```

```
# Linear (i.e., soft margin classifier)
caret::getModelInfo("svmLinear")$svmLinear$parameters
```

```
##   parameter   class label
## 1          C numeric  Cost
```

```
# Polynomial kernel
caret::getModelInfo("svmPoly")$svmPoly$parameters
```

```
##   parameter   class          label
## 1   degree numeric Polynomial Degree
## 2    scale numeric           Scale
## 3      C numeric             Cost
```

```
# Radial basis kernel
caret::getModelInfo("svmRadial")$svmRadial$parameters
```

```
##   parameter   class label
## 1    sigma numeric  Sigma
## 2      C numeric  Cost
```

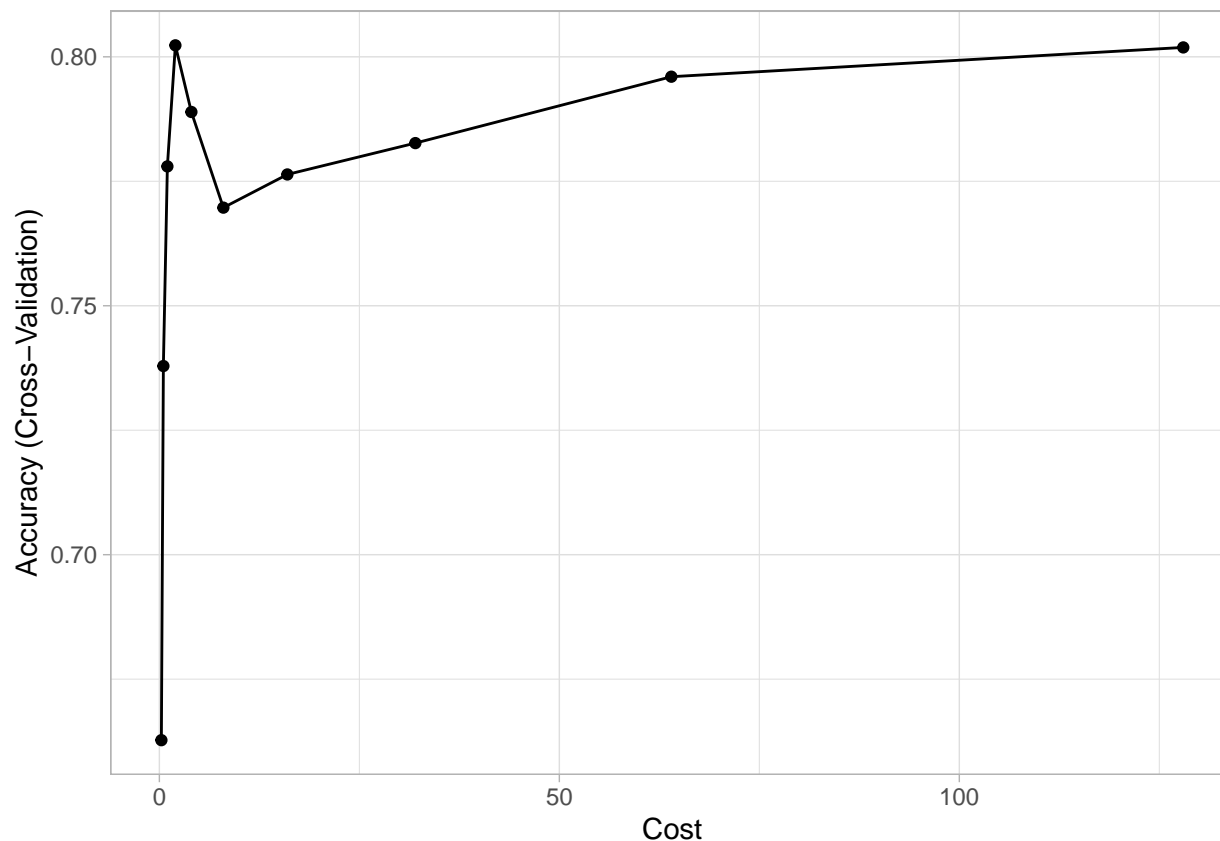
Run SVM Model in Training phase

Using `split_train`, we can tune an SVM model with radial basis kernel.

```
set.seed(1854) # for reproducibility
split_svm <- train(
  Failure.binary ~ .,
  data = split_train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)
```

Plot and print SVM model with with radial basis kernel.

```
# Plot results
ggplot(split_svm) + theme_light()
```



```
# Print results
split_svm$results
```

##	sigma	C	Accuracy	Kappa	AccuracySD	KappaSD
## 1	0.001998749	0.25	0.6627451	0.0000000	0.01891300	0.0000000
## 2	0.001998749	0.50	0.7378922	0.2715440	0.06418046	0.2198366
## 3	0.001998749	1.00	0.7779902	0.4565954	0.07142465	0.1608304
## 4	0.001998749	2.00	0.8023039	0.5196491	0.09057479	0.2186000
## 5	0.001998749	4.00	0.7889216	0.5030643	0.07639949	0.1942976
## 6	0.001998749	8.00	0.7697059	0.4653629	0.07092559	0.1830668
## 7	0.001998749	16.00	0.7763725	0.4861127	0.06283611	0.1498343
## 8	0.001998749	32.00	0.7826716	0.4985015	0.07602914	0.1806382
## 9	0.001998749	64.00	0.7960049	0.5248585	0.07147503	0.1670975
## 10	0.001998749	128.00	0.8018873	0.5429164	0.08701199	0.2010434

Control parameter

```
class.weights = c("No" = 1, "Yes" = 10)
```

```
# Control params for SVM
```

```
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary # also needed for AUC/ROC
)
```

```
split_train$Failure.binary=fct_recode(split_train$Failure.binary,No="0",Yes="1")
```

Print the AUC values during Training

```
# Tune an SVM
set.seed(5628) # for reproducibility
train_svm_auc <- train(
  Failure.binary ~ .,
  data = split_train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  metric = "ROC", # area under ROC curve (AUC)
  trControl = ctrl,
  tuneLength = 10
)

# Print results
train_svm_auc$results
```

```
##          sigma      C      ROC      Sens      Spec      ROCSD      SensSD
## 1  0.001697891  0.25  0.8102727  0.8445455  0.5033333  0.09982583  0.12592723
## 2  0.001697891  0.50  0.8102727  0.8536364  0.5033333  0.09982583  0.12708861
## 3  0.001697891  1.00  0.8323939  0.8827273  0.5233333  0.09919217  0.11244425
## 4  0.001697891  2.00  0.8520606  0.9036364  0.6033333  0.09942461  0.09988055
## 5  0.001697891  4.00  0.8582121  0.9236364  0.6366667  0.09545946  0.09679909
## 6  0.001697891  8.00  0.8729697  0.9427273  0.5766667  0.11486557  0.06542227
## 7  0.001697891 16.00  0.8901818  0.9327273  0.6366667  0.13222606  0.07892762
## 8  0.001697891 32.00  0.8830000  0.9418182  0.5933333  0.13402578  0.06886193
## 9  0.001697891 64.00  0.8812121  0.9418182  0.6133333  0.15158268  0.05019704
## 10 0.001697891 128.00 0.8659697  0.9236364  0.6133333  0.15790577  0.08454491
##          SpecSD
## 1  0.2224721
## 2  0.2224721
## 3  0.2403958
## 4  0.2157101
## 5  0.2235792
## 6  0.1937607
## 7  0.2027283
## 8  0.2968144
## 9  0.2563755
## 10 0.3182514
```

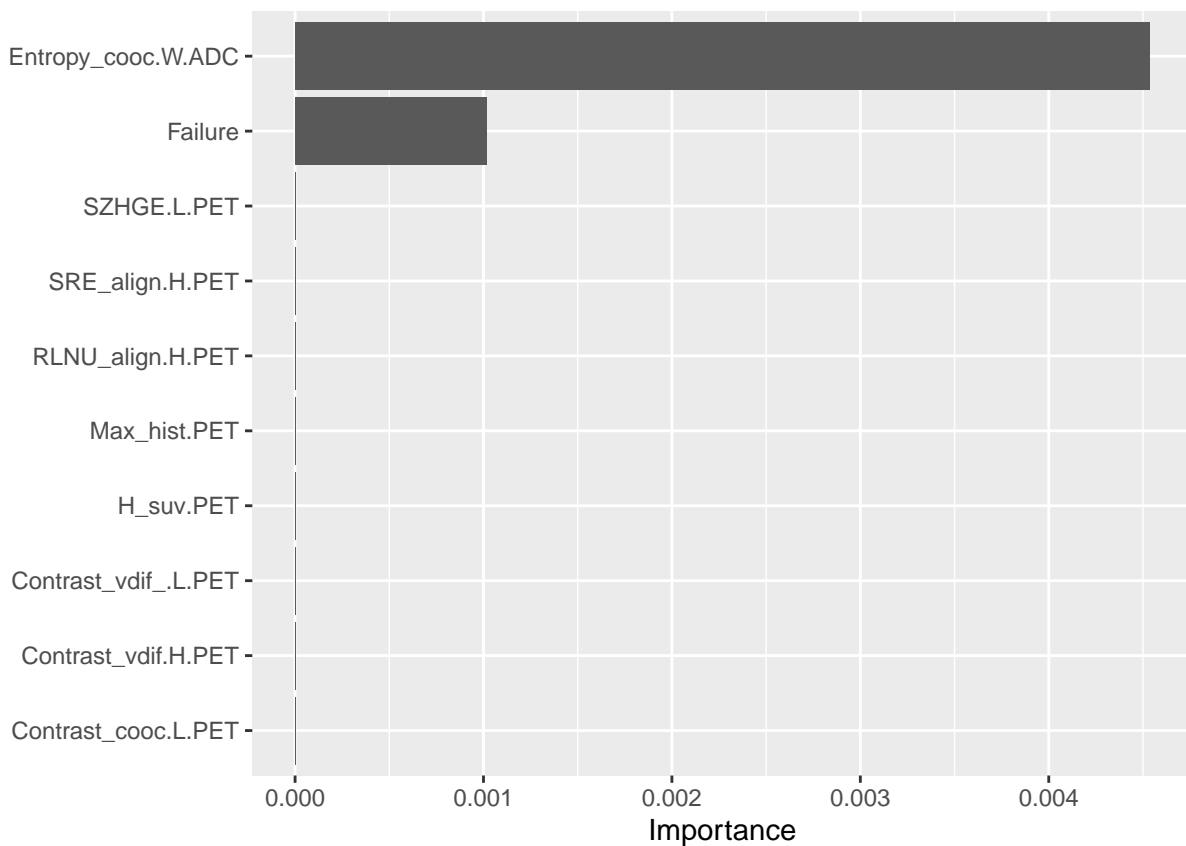
```
confusionMatrix(train_svm_auc)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##          Reference
## Prediction  No  Yes
##          No  61.8 12.1
##          Yes  4.5 21.7
##
```

```
## Accuracy (average) : 0.8344
```

Print the Top 20 important features during Training

```
prob_yes <- function(object, newdata) {  
  predict(object, newdata = newdata, type = "prob")[, "Yes"]  
}  
  
# Variable importance plot  
set.seed(2827) # for reproducibility  
vip(train_svm_auc, method = "permute", nsim = 5, train = split_train,  
     target = "Failure.binary", metric = "auc", reference_class = "Yes",  
     pred_wrapper = prob_yes)
```



Print the AUC values during Testing

```
split_test$Failure.binary=fct_recode(split_test$Failure.binary,No="0",Yes="1")  
  
# Tune an SVM with radial  
set.seed(5628) # for reproducibility  
test_svm_auc <- train(  
  Failure.binary ~ .,  
  data = split_test,  
  method = "svmRadial",
```

```

preProcess = c("center", "scale"),
metric = "ROC", # area under ROC curve (AUC)
trControl = ctrl,
tuneLength = 10
)

```

```

# Print results
test_svm_auc$results

```

##	sigma	C	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
## 1	0.001959001	0.25	0.6750000	0.9666667	0	0.2872013	0.1054093	0
## 2	0.001959001	0.50	0.5750000	0.9333333	0	0.3320577	0.1405457	0
## 3	0.001959001	1.00	0.6250000	1.0000000	0	0.3148829	0.0000000	0
## 4	0.001959001	2.00	0.3083333	0.9000000	0	0.3168372	0.2249829	0
## 5	0.001959001	4.00	0.3500000	0.9000000	0	0.4021547	0.2249829	0
## 6	0.001959001	8.00	0.3916667	0.9000000	0	0.3889881	0.2249829	0
## 7	0.001959001	16.00	0.3083333	0.9000000	0	0.3514740	0.2249829	0
## 8	0.001959001	32.00	0.4250000	0.8333333	0	0.3976202	0.2832789	0
## 9	0.001959001	64.00	0.3750000	0.9333333	0	0.3833937	0.1405457	0
## 10	0.001959001	128.00	0.4083333	0.8666667	0	0.3937200	0.2810913	0

```

confusionMatrix(test_svm_auc)

```

```

## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction  No  Yes
##           No 62.5 35.0
##           Yes 2.5 0.0
##
## Accuracy (average) : 0.625

```