Model 1 - SVM

Freddy A. Julkanain II

```r
pacman::p_load(
  pacman,
  tidyverse,
  rsample,
  caret,
  kernlab,
  modeldata,
  pdp,
  vip,
  ROCR,
  pROC
)
```

```r
# DATA
DF= read.csv("CleanedDF.csv")
```

```r
# Load Failure.binary data

DF$Failure.binary=as.factor(DF$Failure.binary)

set.seed(123)  # for reproducibility
indexing = initial_split(DF, prop = 0.8, strata = "Failure.binary")
split_train = training(indexing)
split_test  = testing(indexing)
```

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

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

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

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

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

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

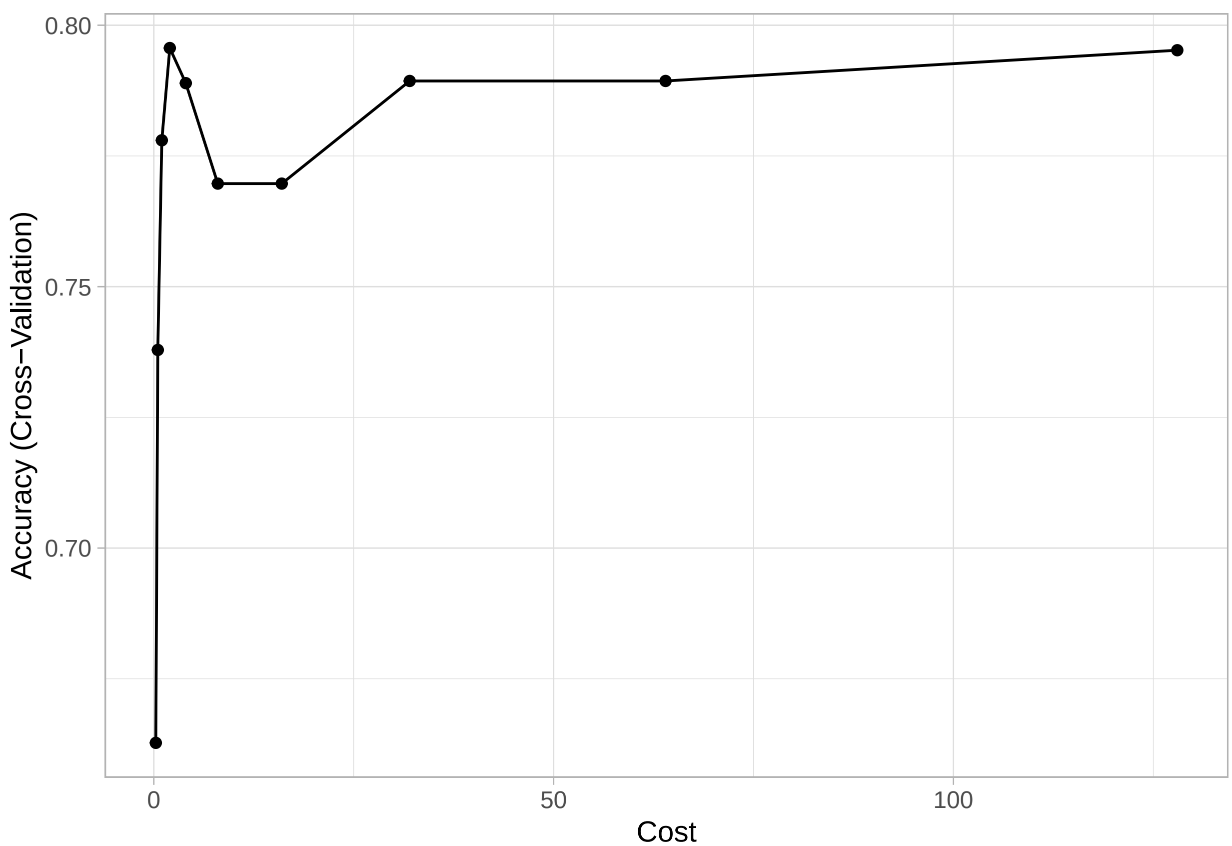
Run SVM Model in Training phase

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

```r
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.

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




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

```
##          sigma      C  Accuracy     Kappa AccuracySD   KappaSD
## 1  0.001994138   0.25 0.6627451 0.0000000 0.01891300 0.0000000
## 2  0.001994138   0.50 0.7378922 0.2715440 0.06418046 0.2198366
## 3  0.001994138   1.00 0.7779902 0.4565954 0.07142465 0.1608304
## 4  0.001994138   2.00 0.7956373 0.5022807 0.08771952 0.2093840
## 5  0.001994138   4.00 0.7889216 0.5030643 0.07639949 0.1942976
## 6  0.001994138   8.00 0.7697059 0.4653629 0.07092559 0.1830668
```

```
## 7  0.001994138  16.00 0.7697059 0.4752432 0.06358290 0.1437736
## 8  0.001994138  32.00 0.7893382 0.5122270 0.07412110 0.1716865
## 9  0.001994138  64.00 0.7893382 0.5130008 0.06712885 0.1548997
## 10 0.001994138 128.00 0.7952206 0.5320469 0.08968551 0.2000276
```

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  0.00169434   0.25 0.8084545 0.8445455 0.5033333 0.10201505 0.12592723
## 2  0.00169434   0.50 0.8084545 0.8536364 0.5033333 0.10201505 0.12708861
## 3  0.00169434   1.00 0.8342121 0.8827273 0.5233333 0.09643904 0.11244425
## 4  0.00169434   2.00 0.8467576 0.9036364 0.6033333 0.10718481 0.09988055
## 5  0.00169434   4.00 0.8616970 0.9236364 0.6366667 0.09000201 0.09679909
## 6  0.00169434   8.00 0.8708485 0.9327273 0.5766667 0.11876914 0.06330360
## 7  0.00169434  16.00 0.8898485 0.9327273 0.6366667 0.13325382 0.07892762
## 8  0.00169434  32.00 0.8810000 0.9418182 0.5933333 0.13521431 0.06886193
## 9  0.00169434  64.00 0.8773939 0.9318182 0.6133333 0.14989022 0.06670109
## 10 0.00169434 128.00 0.8699697 0.9236364 0.6133333 0.15711656 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
```

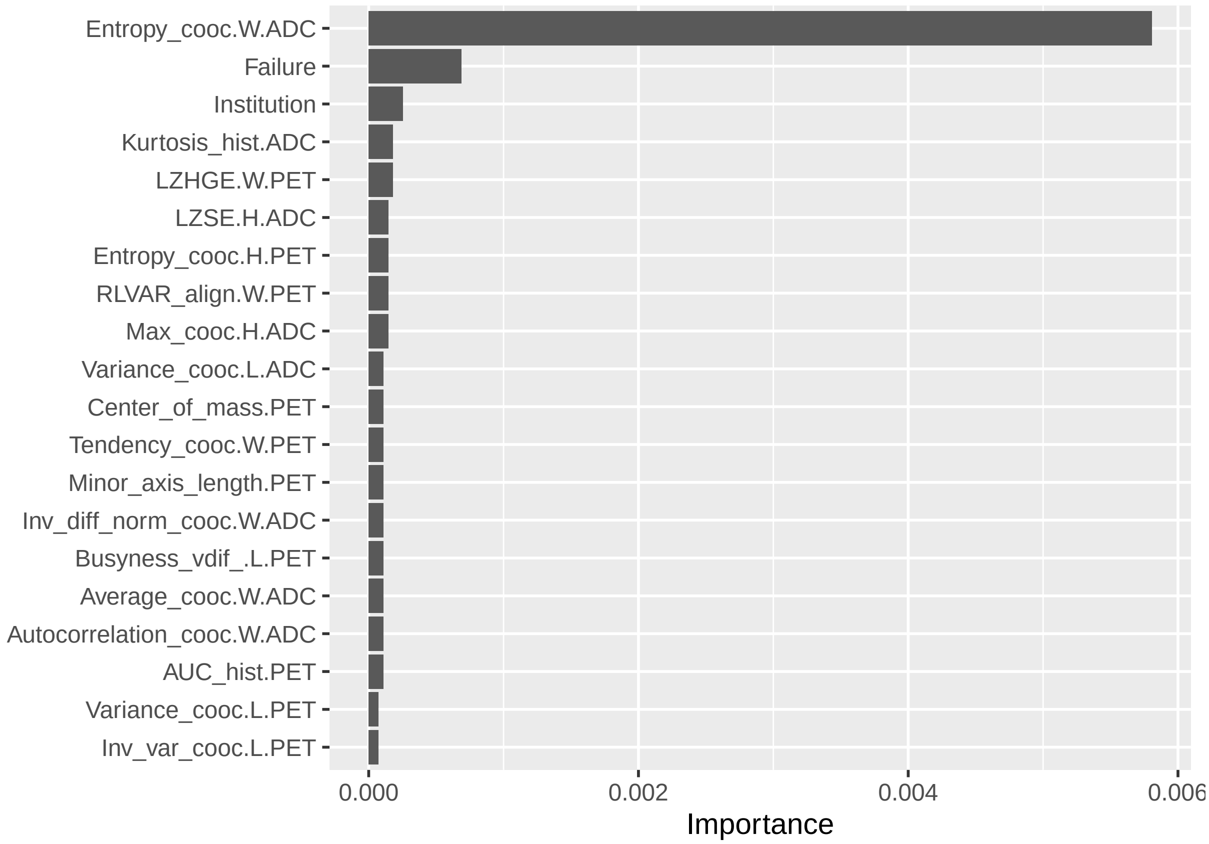
```
confusionMatrix(train_svm_auc)
```

```
## 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,num_features = 20)
```



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 0.001953177  0.25 0.6750000 0.9666667    0 0.2872013 0.1054093      0
## 2 0.001953177  0.50 0.5750000 0.9333333    0 0.3320577 0.1405457      0
## 3 0.001953177  1.00 0.6250000 1.0000000    0 0.3148829 0.0000000      0
## 4 0.001953177  2.00 0.3083333 0.9000000    0 0.3168372 0.2249829      0
## 5 0.001953177  4.00 0.3500000 0.9000000    0 0.4021547 0.2249829      0
## 6 0.001953177  8.00 0.3916667 0.9000000    0 0.3889881 0.2249829      0
```

```
## 7  0.001953177  16.00 0.3083333 0.9000000    0 0.3514740 0.2249829      0
## 8  0.001953177  32.00 0.4250000 0.8333333    0 0.3976202 0.2832789      0
## 9  0.001953177  64.00 0.3750000 0.9333333    0 0.3833937 0.1405457      0
## 10 0.001953177 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
```