# STATISTICAL COMPUTING FINAL PROJECT

Jobert Jay R. Pandan

2022-12-13

## MODEL 1: SVM MODEL

### Libraries Needed for Pre-processing

```
library(readr)
library(dplyr)
library(ggplot2) # for plotting
library(caret) # pre-processing and modeling
library(corrplot)
library(fastDummies) # for creating dummy variables
pacman::p_load(tidyverse)
pacman::p_load(bestNormalize)

# Modeling packages
library(caret)    # for classification and regression training
library(kernlab)  # for fitting SVMs
library(modeldata) #for Failure.binary data
library(forcats)

# Model interpretability packages
library(pdp)      # for partial dependence plots, etc.
library(vip)      # for variable importance plots
```

### DATA ACQUISITION AND PRE-PROCESSING

#### Dataset

*radiomics_completedata.csv* data was used for this project. The data has a 431 variables and 197 observations including *Failure.binary* as our target/response/outcome variable with its 430 predictors/features/independent variables. *Failure.binary* is a binary variable with 1 and 0 as its values.

```
radiomics = read.csv("radiomics_completedata.csv")
```

#### Data description

Variables *Institution* and *Failure.binary* needs to change its datatype into factors. `Institution` was re-coded into dummy variables

```
newdf = dummy_cols(radiomics, select_columns = "Institution" )
newdf = newdf[,-1]
str(newdf[431:434])
```

```
## 'data.frame':    197 obs. of  4 variables:
##  $ Institution_A: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Institution_B: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ Institution_C: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Institution_D: int  0 0 0 0 0 0 0 0 0 0 ...
```

```
newdf$Institution_A = as.factor(newdf$Institution_A)
newdf$Institution_B = as.factor(newdf$Institution_B)
newdf$Institution_C = as.factor(newdf$Institution_C)
newdf$Institution_D = as.factor(newdf$Institution_D)
newdf$Failure.binary = as.factor(newdf$Failure.binary)
str(newdf[1])
```

```
## 'data.frame':    197 obs. of  1 variable:
##  $ Failure.binary: Factor w/ 2 levels "0","1": 1 2 1 2 1 2 1 1 2 2 ...
```

### Null Values

```
sum(is.na(newdf))
```

```
## [1] 0
```

No null values found in the data.

### Normality of the data

To check the normality of the data, we used Shapiro-wilk test.

SHAPIRO-WILK TEST OF NORMALITY H_o = Data is normally distributed H_i = Data is not normally distributed

```
lshap <- lapply(newdf[2:430], shapiro.test) #applying shapiro-wilk test of normality to the data frame
l = 1:429
for (i in l) { #
  x = lshap[[i]]$p.value
  z= 1 +i
  if (lshap[[i]]$p.value >= 0.05){
    print(paste(x, "p-value for",colnames(newdf[z]) ))
 }
}
```

```
## [1] "0.135011298030283 p-value for Entropy_cooc.W.ADC"
```

Upon checking its normality, only the *Entropy_cooc.W.ADC* has p-value of 0.135 which tells that it follows normal distribution,the rest of the variables do not follow normal distribution. Transformation is required to the data to address the non-normality of the data. In this project we use *Ordered Quantile (ORQ) normalization transformation* to normalize the data.

```
newdf1 = newdf %>% select_if(is.numeric)
tempDF=apply(newdf1,2,orderNorm)
tempDF=lapply(tempDF, function(x) x$x.t)
tempDF=tempDF%>%as.data.frame()
norm_data = cbind(newdf[c('Failure.binary','Institution_A','Institution_B','Institution_C','Institution
```

Testing the normality of the data using the normalize data using **Ordered Quantile (ORQ) normalization transformation**.

```
lshap <- lapply(tempDF, shapiro.test) #applying shapiro-wilk test of normality to the data frame
l = 1:429
for (i in l) { #
  x = lshap[[i]]$p.value
  z= 1 +i
  if (lshap[[i]]$p.value <= 0.05){
    print(paste(x, "p-value for",colnames(newdf[z]) ))
```

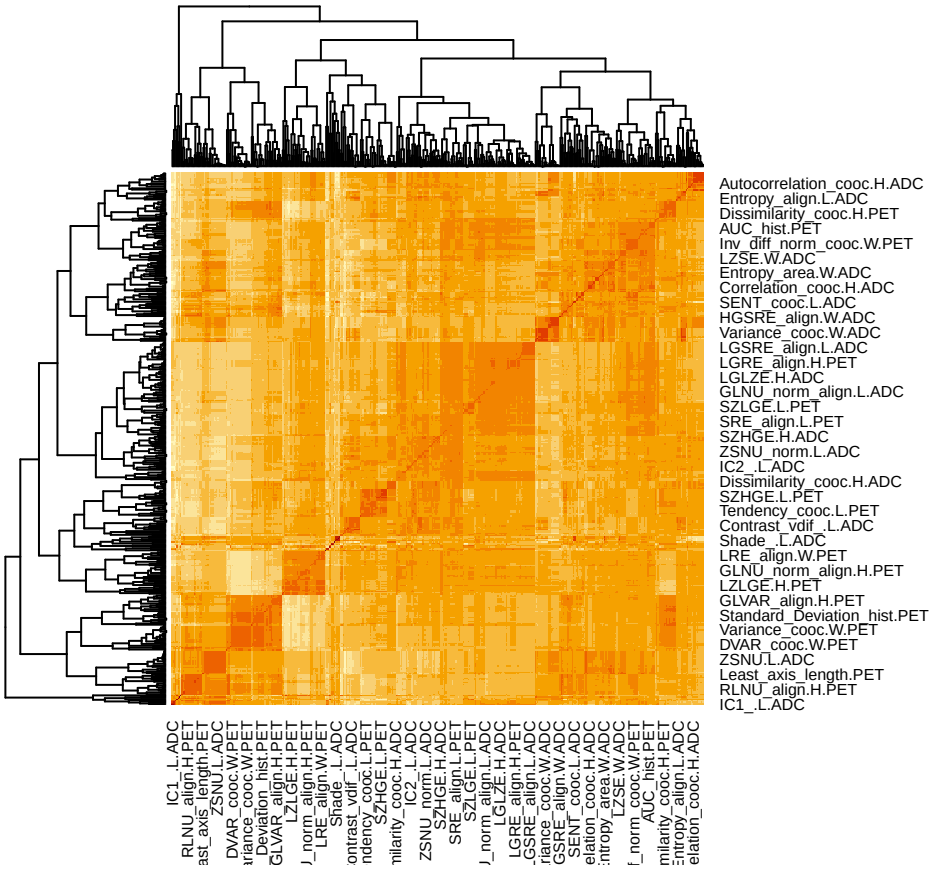
```
 }
}
```

Since all p-values are greater than 0.05 then we cannot reject the null hypothesis that the data by this time follows Normal distribution.

### Correlation

Checking the correlation of the data. By assumption that highly correlated data does not do good in model making, thus by removing this will improve the performance of our model. But in this project, removing highy correlated columns is not considered.

```
heatmap(corMatrix)
```




```
#ggcorrplot(corr <= 0.40 ,colors =  c("blue", "white","red"), ggtheme=theme_bw, lab = FALSE, title = "C
```

### Data Splitting

After pre-processing, we now split our data into 80% for training and 20% for testing. * **Training:** our training dataset has a 158 observations * **Testing:** our testing dataset has a 39 observations

```
set.seed(3333)
trainIndex <- createDataPartition(norm_data$Failure.binary, p = .80,
                                  list = FALSE,
                                  times = 1)
finaldata_train<- norm_data[ trainIndex,]
finaldata_test<- norm_data[-trainIndex,]
dim(finaldata_train) ; dim(finaldata_test)
```

```
## [1] 158 434
```

```
## [1]  39 434
```

## MODELING

```
# 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
```

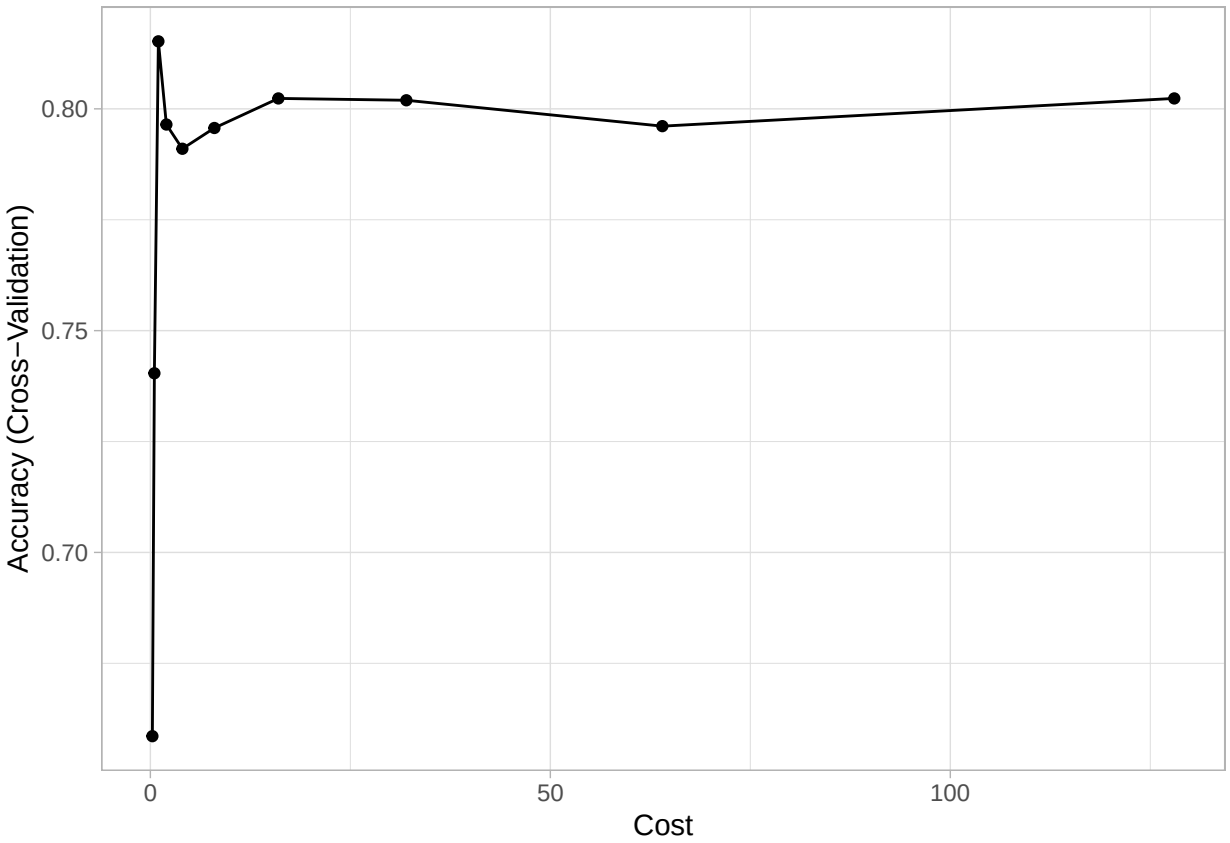
### Run SVM Model in Training phase

Tuning and fitting SVM Model using **radial kernel** to *finaldata_train* with 10 cross-validation.

```
set.seed(1854)  # for reproducibility
model_svm1 <- train(
  Failure.binary ~ .,
  data = finaldata_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(model_svm1) + theme_light()
```




Above plot is the accuracy obtained by level of cost, cost decides how much an SVM should be allowed to "bend" with the data. The plot suggest that 80% cross-validated accuracy is obtained 20-25 cost (low cost)

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
)

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

**Print the AUC values during Training**

```
# Tune an SVM
set.seed(5628)  # for reproducibility
model_svm1_auc <- train(
  Failure.binary ~ .,
  data = finaldata_train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  metric = "ROC",  # area under ROC curve (AUC)
```

```
  trControl = ctrl,
  tuneLength = 10
)

# Print results
model_svm1_auc$results
```

```
##         sigma      C       ROC      Sens      Spec      ROCSD     SensSD
## 1  0.00178247   0.25 0.8111212 0.8872727 0.6300000 0.10831656 0.09526918
## 2  0.00178247   0.50 0.8111212 0.8563636 0.6433333 0.10831656 0.09494090
## 3  0.00178247   1.00 0.8169091 0.9236364 0.6066667 0.09923458 0.07547827
## 4  0.00178247   2.00 0.8521212 0.9336364 0.5900000 0.10743591 0.08730744
## 5  0.00178247   4.00 0.8372727 0.9327273 0.5733333 0.11092629 0.07892762
## 6  0.00178247   8.00 0.8416970 0.9236364 0.6300000 0.10313564 0.07547827
## 7  0.00178247  16.00 0.8440909 0.9154545 0.6133333 0.09636991 0.10231929
## 8  0.00178247  32.00 0.8440303 0.9145455 0.6166667 0.08789598 0.08256672
## 9  0.00178247  64.00 0.8505455 0.9245455 0.6166667 0.08313380 0.09659492
## 10 0.00178247 128.00 0.8608485 0.9245455 0.6566667 0.08271337 0.09659492
##       SpecSD
## 1  0.2246808
## 2  0.2424413
## 3  0.2734327
## 4  0.2629604
## 5  0.2628430
## 6  0.2039426
## 7  0.2073942
## 8  0.1976716
## 9  0.1976716
## 10 0.1640912
```

```
confusionMatrix(model_svm1_auc)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  60.8 12.0
##        Yes  5.1 22.2
##
##  Accuracy (average) : 0.8291
```

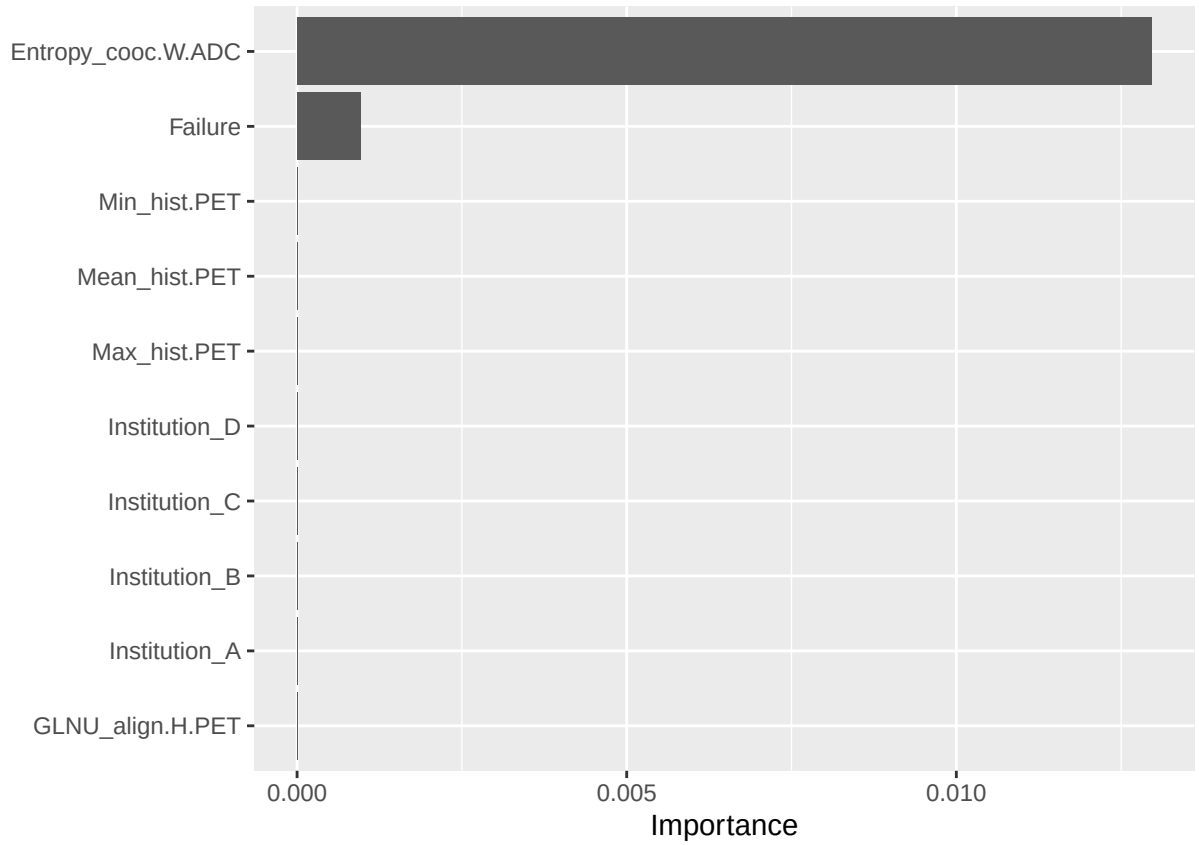
Confusion Matrix shows that prediction capability of our model achieved 82.91% accuracy and only 17% are misclassified.

### Print the Top 20 important features during Training

```
pred.prob <- function(object, newdata) {
  predict(object, newdata = newdata, type = "prob")[, "Yes"]
}

# Variable importance plot
set.seed(2827)  # for reproducibility
vip(model_svm1_auc, method = "permute", nsim = 5, train = finaldata_train,
```

```
    target = "Failure.binary", metric = "auc", reference_class = "Yes",
    pred_wrapper = pred.prob)
```




The top 3 most influential features in SVM model are **Entropy.cooc_W.ADC, Failure and Min_hist.PET**. Only in this model the *Min_hist.PET* considered as part of top 3 influential feature compare to *bagging model* and *Random Forest Model*.

**Print the AUC values during Testing**

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

# Tune an SVM with radial
set.seed(5628)  # for reproducibility
testing_svm1_auc <- train(
  Failure.binary ~ .,
  data = finaldata_test,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  metric = "ROC",  # area under ROC curve (AUC)
  trControl = ctrl,
  tuneLength = 10
)

# Print results
testing_svm1_auc$results
```

```
##          sigma      C       ROC      Sens Spec     ROCSD     SensSD     SpecSD
```

```
## 1  0.001717525   0.25 0.3916667 1.0000000 0.00 0.3380454 0.0000000 0.0000000
## 2  0.001717525   0.50 0.4916667 0.9333333 0.05 0.3567039 0.2108185 0.1581139
## 3  0.001717525   1.00 0.2916667 0.9166667 0.05 0.2812286 0.1800206 0.1581139
## 4  0.001717525   2.00 0.5583333 0.9666667 0.00 0.3143924 0.1054093 0.0000000
## 5  0.001717525   4.00 0.4750000 0.9166667 0.00 0.3192632 0.1800206 0.0000000
## 6  0.001717525   8.00 0.5416667 0.9666667 0.00 0.3540986 0.1054093 0.0000000
## 7  0.001717525  16.00 0.4250000 0.9500000 0.00 0.3479437 0.1581139 0.0000000
## 8  0.001717525  32.00 0.5083333 1.0000000 0.00 0.3567039 0.0000000 0.0000000
## 9  0.001717525  64.00 0.5083333 0.9666667 0.00 0.3567039 0.1054093 0.0000000
## 10 0.001717525 128.00 0.5583333 1.0000000 0.00 0.3514740 0.0000000 0.0000000
```

```
confusionMatrix(testing_svm1_auc)
```

```
## Cross-Validated (10 fold) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction   No  Yes
##        No  64.1 33.3
##        Yes  2.6  0.0
##
##  Accuracy (average) : 0.641
```

Unfortunately SVM modelling cannot provide good accuracy in predicting the classification of random case, it only has an accuracy of 64.1%, which means that 64.1% of the time the model willproedict correctly the Failure.binary '0'and '1'.