SVM for the Climate Model Simulation Crashes data.

Contents

1 INTRODUCTION

This project looks at the prediction of simulation outcomes of a Climate

Model Simulation Crashes data with the use of machine learning process. We are examining the relationship of some factors that influence the simulation outcome. For us to come to such a conclusion, we would have to subject the data to various inspections,

such as Data cleaning, Data exploration and Variable Selection for us to be able to construct an efficient Machine learning model to predict or forecast the simulation outcomes of a Climate Model Simulation Crashes data with 18 climate model parameters scaled in the interval [0,1]. The goal of this project is to use a machine learning classification model to predict simulation outcomes (target variable "outcome" with values 'fail (0)' or 'succeed (1)') based on input parameter values (columns 3-20) of the data set.

2 DATA COLLECTION/INPUT

The data must be read into R first in other to develop a machine learning model for the Climate Model Simulation Crashes.

```
dat <- read.table (file="http://archive.ics.uci.edu/ml/machine-learning-databases/00252/pop_failures.dat",header=TRUE)
dim(dat);
```

```
## [1] 540  21
```

```
head(dat)
```

```
##     Study Run vconst_corr   vconst_2   vconst_3   vconst_4   vconst_5   vconst_7
## 1       1   1  0.85903621 0.9278245 0.25286562 0.2988383 0.1705213 0.7359360
## 2       1   2  0.60604103 0.4577284 0.35944842 0.3069574 0.8433308 0.9348507
## 3       1   3  0.99759978 0.3732385 0.51739936 0.5049925 0.6189033 0.6055708
## 4       1   4  0.78340786 0.1040553 0.19753270 0.4218372 0.7420557 0.4908279
```

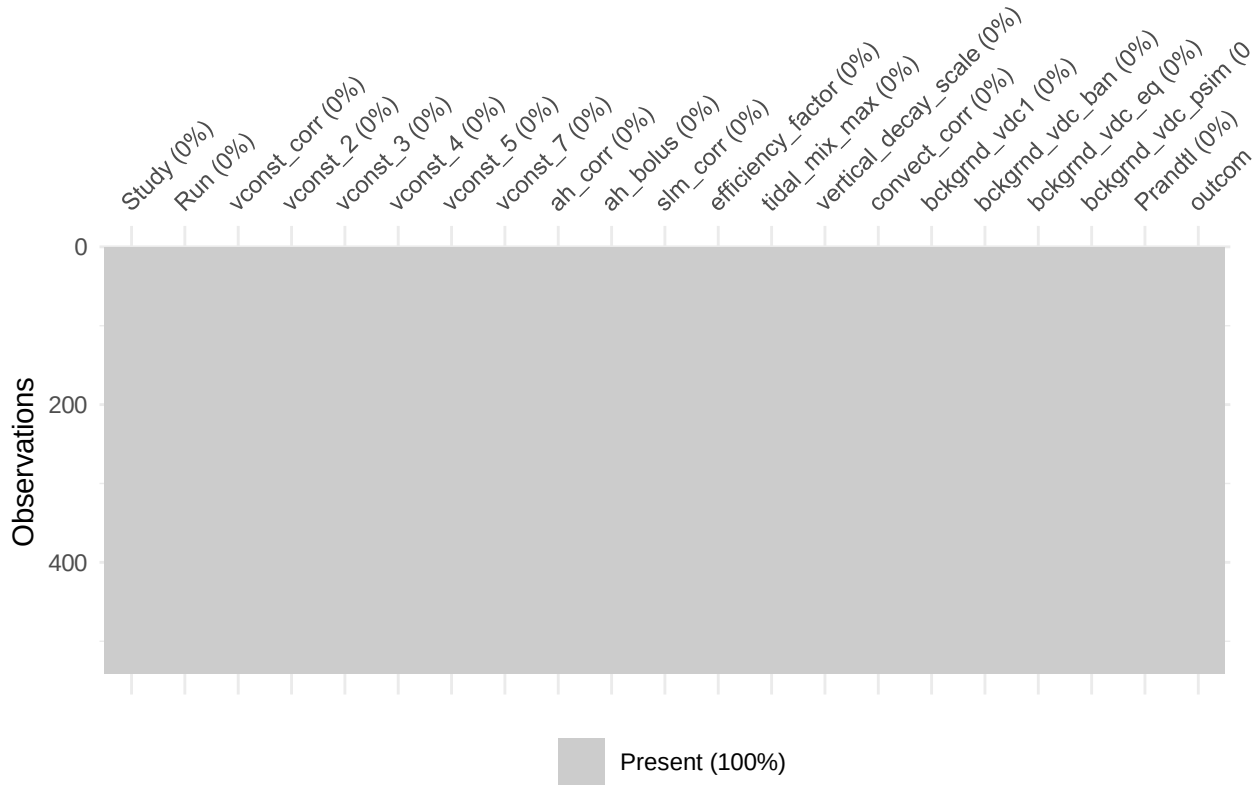
```
## 5       1    5  0.40624953  0.5131993 0.06181158 0.6358367 0.8447975 0.4415022
## 6       1    6  0.04137947  0.6290259 0.30338011 0.8134076 0.2228171 0.9712060
##         ah_corr   ah_bolus    slm_corr  efficiency_factor  tidal_mix_max
## 1 0.428325428  0.5679469 0.47436960            0.2456749      0.10422587
## 2 0.444572488  0.8280149 0.29661775            0.6168699      0.97578558
## 3 0.746225330  0.1959283 0.81566694            0.6793550      0.80341308
## 4 0.005525437  0.3921233 0.01001489            0.4714627      0.59787890
## 5 0.191926451  0.4875461 0.35853358            0.5515432      0.74387652
## 6 0.609778290  0.6478039 0.73791387            0.4409432      0.03598237
##    vertical_decay_scale convect_corr bckgrnd_vdc1 bckgrnd_vdc_ban bckgrnd_vdc_eq
## 1             0.8690907    0.99751850     0.4486201      0.30752179     0.8583104
## 2             0.9143437    0.84524714     0.8641519      0.34671269     0.3565734
## 3             0.6439952    0.71844113     0.9247751      0.31537141     0.2506424
## 4             0.7616588    0.36275056     0.9128191      0.97797118     0.8459212
## 5             0.3123494    0.65022283     0.5222610      0.04354476     0.3766601
## 6             0.6158677    0.01748718     0.9323195      0.32931804     0.9541228
##    bckgrnd_vdc_psim    Prandtl  outcome
## 1        0.7969972  0.8698930        0
## 2        0.4384472  0.5122561        1
## 3        0.2856355  0.3658580        1
## 4        0.6994309  0.4759867        1
## 5        0.2800978  0.1322829        1
## 6        0.1353789  0.2948048        1
```

The dimension shows we have 540 data size (in Rows) and 21 variables (in Columns) i.e 1 target variable and 18 predictors variable. The first 6 rows of the data set was displayed above. The target variable "outcome" is characterized as having value "0" as "fail" and value "1" as "succeed".Columns 3-20 contain numerical values of 18 climate model parameters scaled in the interval [0, 1];the scenario where the SVM model is suitable.

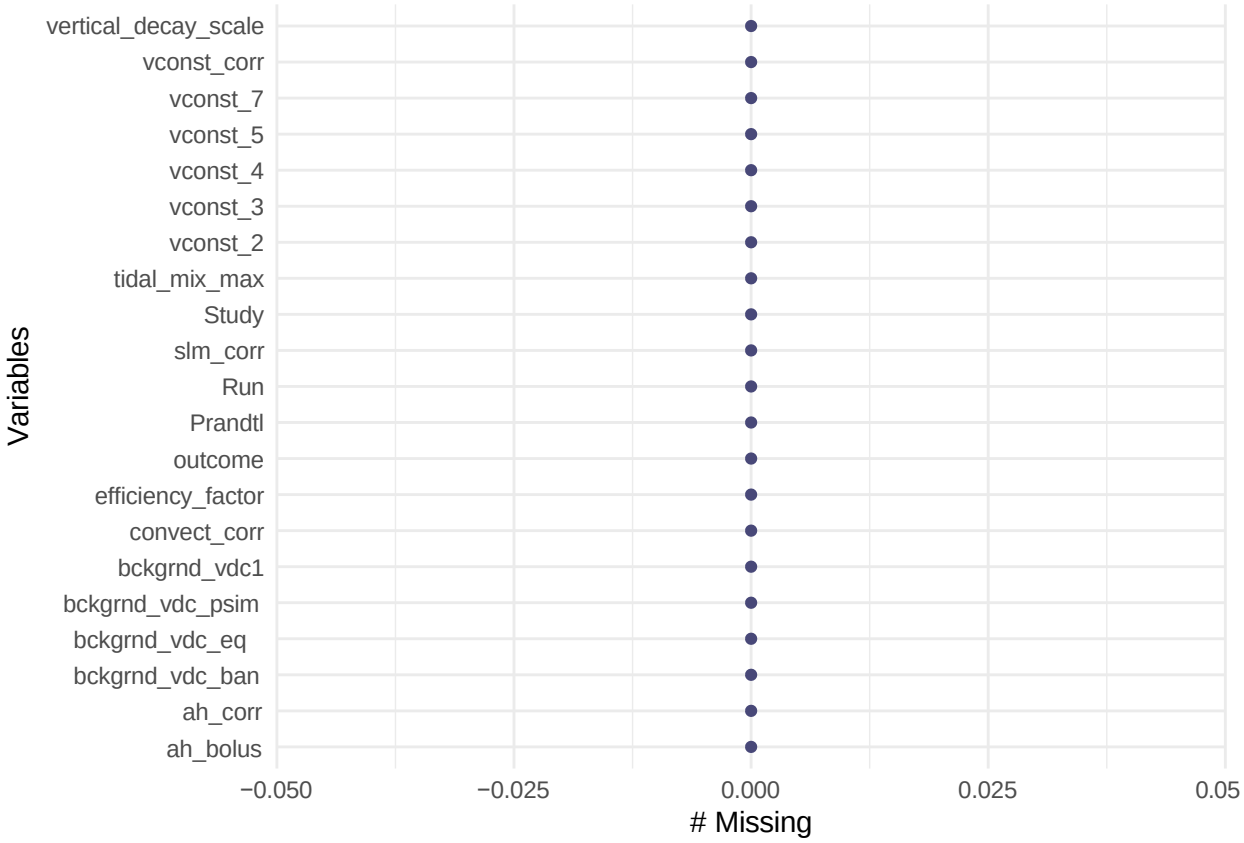
3 DATA PREPARATION

Inspecting the variable types, missing values and possibly wrong records, and other issues.

```
library(naniar)
vis_miss(dat)
```




```
gg_miss_var(dat)
```




```
data = dat
vnames <- colnames(data)
n <- nrow(data)
out <- NULL
for (j in 1:ncol(data)){
  vname <- colnames(data)[j]
  x <- as.vector(data[,j])
  n1 <- sum(is.na(x), na.rm=TRUE) # NA
  n2 <- sum(x=="NA", na.rm=TRUE) # "NA"
  n3 <- sum(x=="", na.rm=TRUE) # missing
  nmiss <- n1 + n2 + n3
  nmiss <- sum(is.na(x))
  ncomplete <- n-nmiss
  out <- rbind(out, c(col.num=j, v.name=vname, mode=mode(x),
                      n.level=length(unique(x)),
 ncom=ncomplete, nmiss= nmiss, miss.prop=nmiss/n))
}
out <- as.data.frame(out)
row.names(out) <- NULL
out
```

```
##      col.num                 v.name      mode n.level ncom nmiss miss.prop
## 1          1                  Study   numeric       3  540     0         0
## 2          2                    Run   numeric     180  540     0         0
## 3          3             vconst_corr numeric     540  540     0         0
## 4          4              vconst_2   numeric     540  540     0         0
## 5          5              vconst_3   numeric     540  540     0         0
## 6          6              vconst_4   numeric     540  540     0         0
## 7          7              vconst_5   numeric     540  540     0         0
## 8          8              vconst_7   numeric     540  540     0         0
## 9          9               ah_corr   numeric     540  540     0         0
## 10        10              ah_bolus   numeric     540  540     0         0
## 11        11              slm_corr   numeric     540  540     0         0
## 12        12      efficiency_factor  numeric     540  540     0         0
## 13        13          tidal_mix_max  numeric     540  540     0         0
## 14        14  vertical_decay_scale   numeric     540  540     0         0
## 15        15          convect_corr   numeric     540  540     0         0
## 16        16          bckgrnd_vdc1   numeric     540  540     0         0
## 17        17       bckgrnd_vdc_ban   numeric     540  540     0         0
## 18        18        bckgrnd_vdc_eq   numeric     540  540     0         0
## 19        19      bckgrnd_vdc_psim   numeric     540  540     0         0
## 20        20               Prandtl   numeric     540  540     0         0
## 21        21               outcome   numeric       2  540     0         0
```

```
dat= dat[, -c(1,2)]
dim(dat)
```

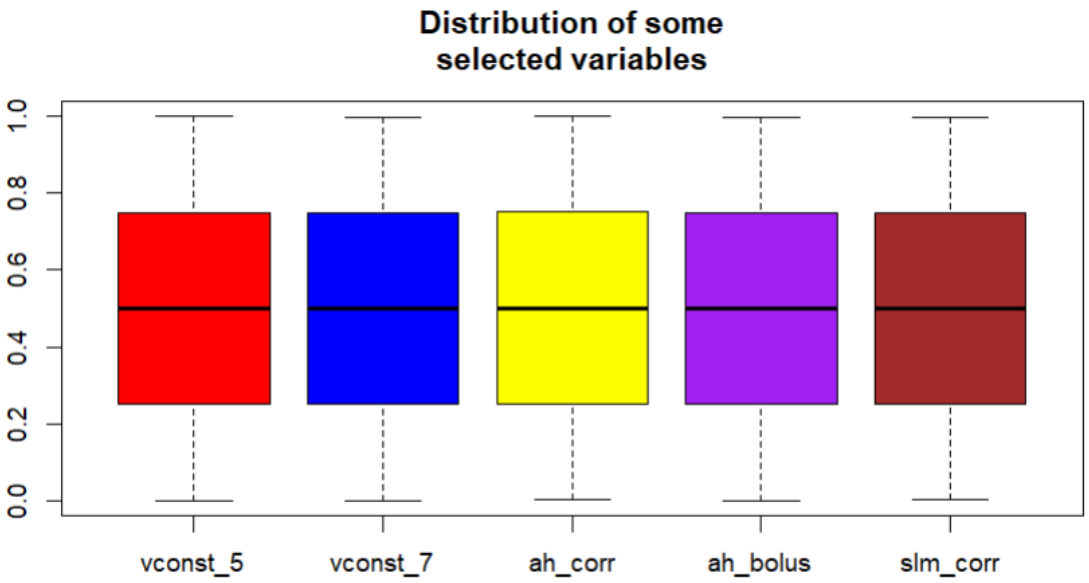
```
## [1] 540  19
```

All the variable types are continuous (scaled in interval [0,1]) except the Target variable which is binary (Categorical). The plot and the analysis above tells us there are no missing values in the data set and no presence of any wrong records. The first two columns of the data set were taken out since they are not important for the classification.

That leaves us with 19 variables (1 target and 18 predictors) for the classification.

```
boxplot(dat[,1:5],col=c("red","blue","yellow","purple","brown"),
        main="Distribution of some selected variables")
```

Distribution of some selected variables



There is no outlier in any of the predictor variables. The above box plot shows proof of some selected variables. The description of the data set also shows the predictor variables have been scaled in the interval [0,1], therefore it is expected of them not to contain any outlier.

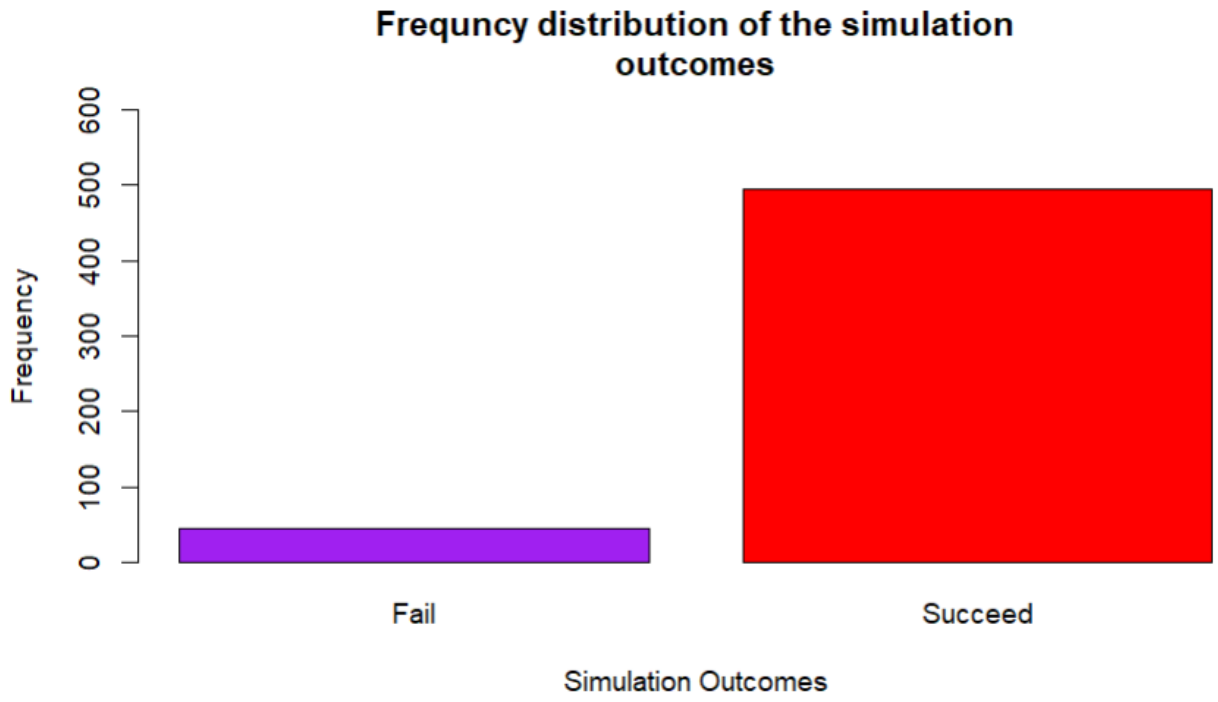
4 EXPLORATORY DATA ANALYSIS

In other to explore the frequency distribution of the output variable in the data set. We obtain the Bar plot of the target variable outcome. The association of the Target variable on some selected predicted variables will also be observed.

```
simulation_outcome=  dat$outcome
simulation_outcome<- ifelse(simulation_outcome==1, "Succeed", "Fail")
tab=table(simulation_outcome,useNA="no")
tab
```

```
## simulation_outcome
##       Fail  Succeed
##         46        494
```

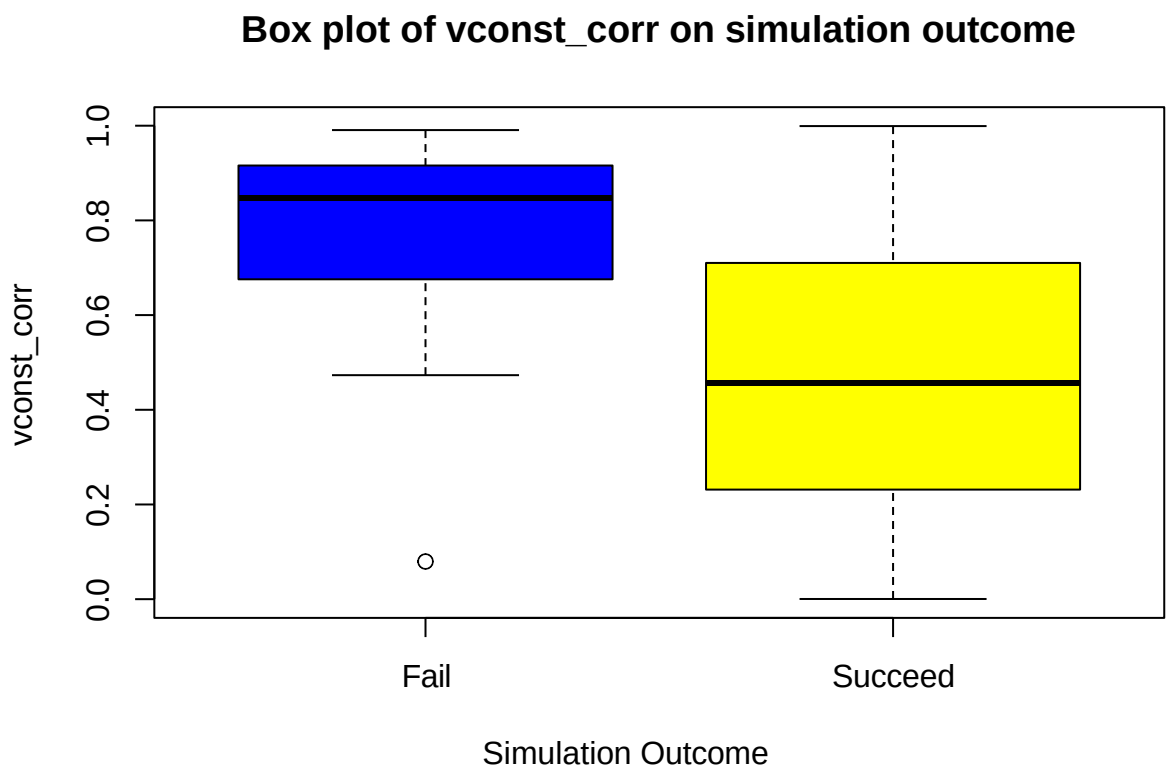
```
barplot(tab,names.arg=row.names(tab),col=c("purple","red"),ylab="Frequency",
xlab="Simulation Outcomes",
main="Frequency distribution of the simulation outcomes",ylim=c(0,600))
```



The plot above shows the frequency distribution of the target variable "outcome". We have 494 frequency of class "Success" and 46 frequency of class "Fail".

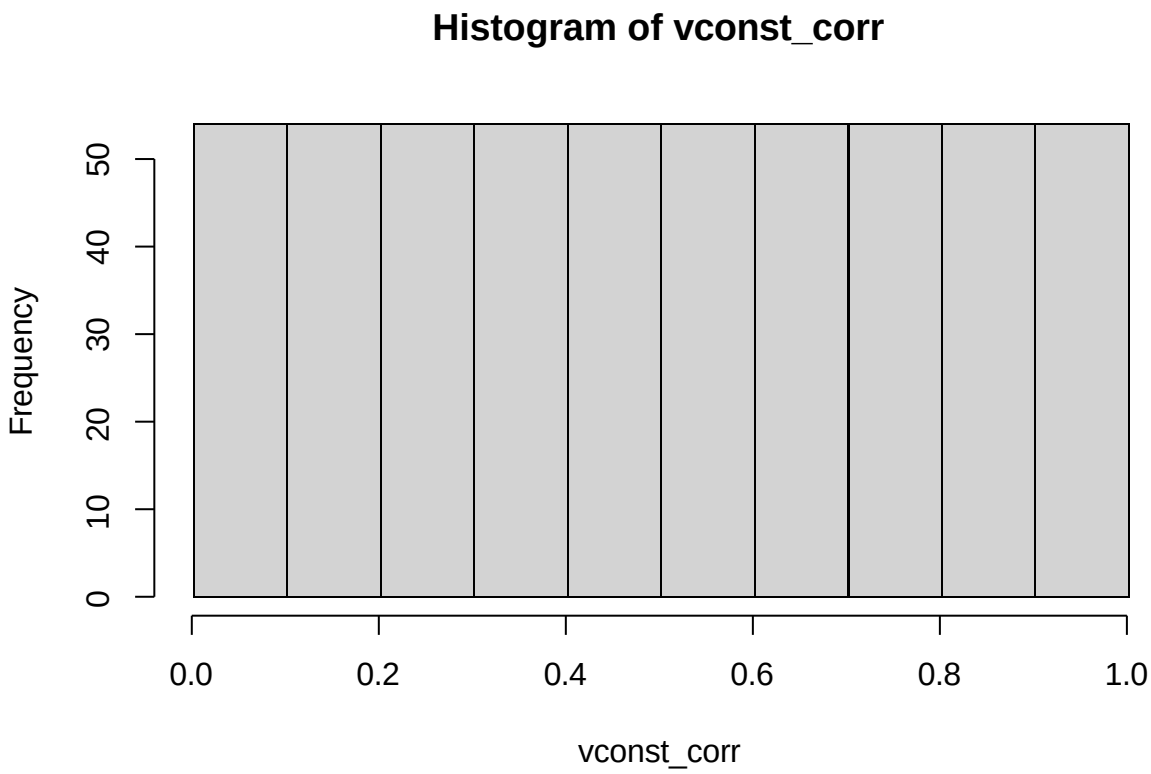
Simulation outcome and vconst_corr

```
boxplot<-boxplot(dat$vconst_corr~simulation_outcome,
xlab="Simulation Outcome", ylab="vconst_corr"
,main="Box plot of vconst_corr on simulation outcome"
,col=c("blue","yellow"))
```



The Boxplot above shows the distribution of the predictor variable vconst_corr on simulation outcome, which shows the presence of association between predictor variable vconst_corr on target variable "simulation outcome".

```
hist(dat$vconst_corr,main="Histogram of vconst_corr",xlab="vconst_corr")
```



The plot shows the distribution of the vconst_corr is uniform. Further evidence will be presented using the two normality test below.

```
library(nortest)
ad.test(dat$vconst_corr)
```

```
##
##   Anderson-Darling normality test
##
## data:   dat$vconst_corr
## A = 5.9651, p-value = 1.096e-14
```

```
shapiro.test(dat$vconst_corr)
```

```
##
##   Shapiro-Wilk normality test
##
## data:   dat$vconst_corr
## W = 0.95479, p-value = 8.313e-12
```

The p-value of both test are all less than alpha level of significance 0.05, which indicate that the distribution is not normal. Therefore the assumption of t-test is violated, then we use the non parametric alternative approach called wilcoxon rank sum test to examine the association between vconst_corr on simulation outcome.

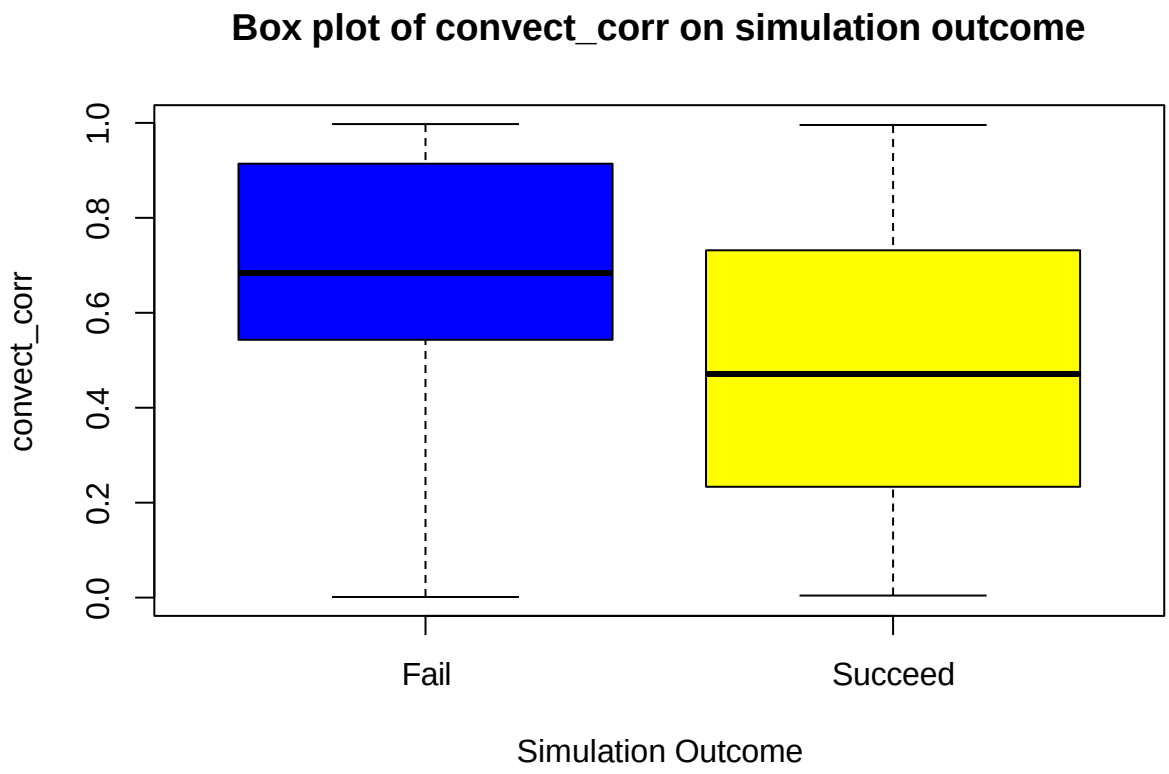
```
wilcox.test(dat$vconst_corr~simulation_outcome, data=dat, alternative = "two.sided"
            ,na.action = na.omit)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dat$vconst_corr by simulation_outcome
## W = 18518, p-value = 1.555e-12
## alternative hypothesis: true location shift is not equal to 0
```

The p-value of the test above which is less than alpha level of significance 0.05 indicate there exist a significance relationship between predictor variable vconst_corr on target variable "simulation outcome".

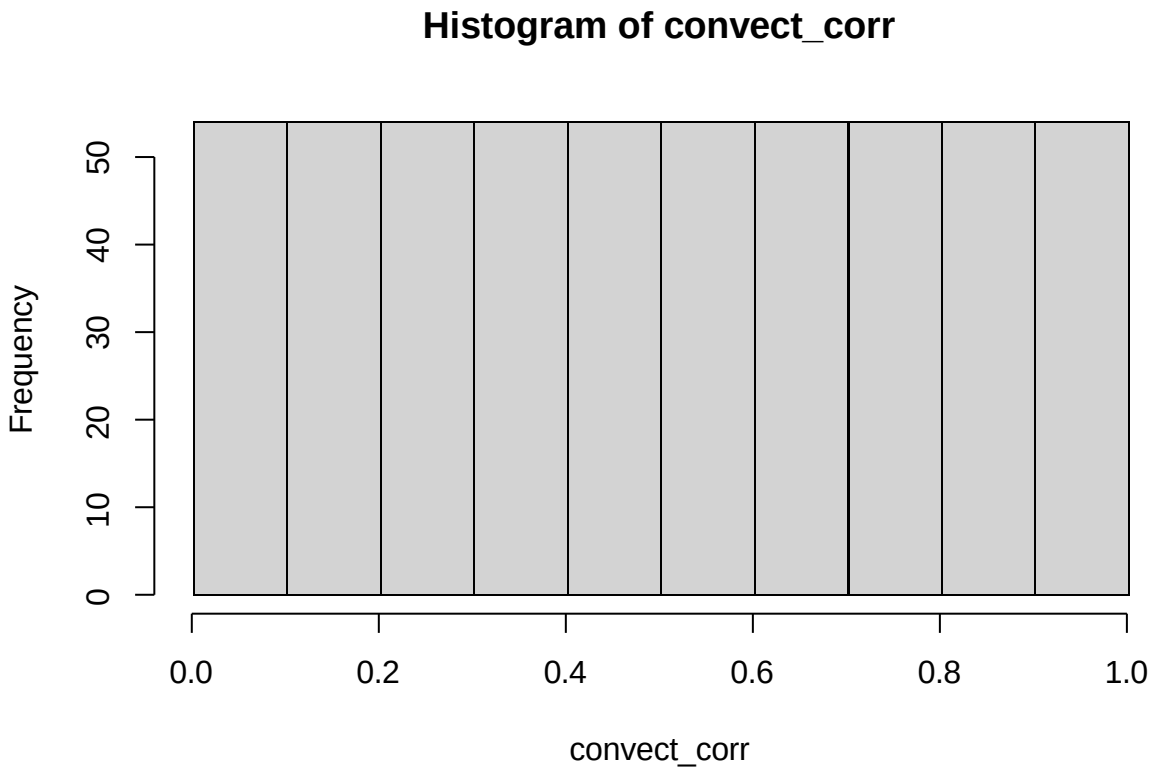
Simulation outcome and convect_corr

```
boxplot<-boxplot(dat$convect_corr~simulation_outcome
          ,xlab="Simulation Outcome", ylab="convect_corr"
         ,main="Box plot of convect_corr on simulation outcome"
         ,col=c("blue","yellow"))
```



The Boxplot above shows the distribution of the predictor variable convect_corr on simulation outcome, which indicate the presence of association between predictor variable convect_corr on target variable "simulation outcome".

```
hist(dat$convect_corr,main="Histogram of convect_corr",xlab="convect_corr")
```



The plot shows the distribution of the convect_corr is uniform. Further evidence will be presented using the two normality test below.

```
library(nortest)
ad.test(dat$convect_corr)
```

```
##
##   Anderson-Darling normality test
##
## data:   dat$convect_corr
## A = 5.9813, p-value = 1.002e-14
```

```
shapiro.test(dat$convect_corr)
```

```
##
##   Shapiro-Wilk normality test
##
## data:   dat$convect_corr
## W = 0.95468, p-value = 7.991e-12
```

The p-value of both test are all less than alpha level of significance 0.05, which indicate that the distribution is not normal. Therefore the assumption of t-test is violated, then we use the non parametric alternative approach called wilcoxon rank sum test to examine the association between convect_corr on simulation outcome.

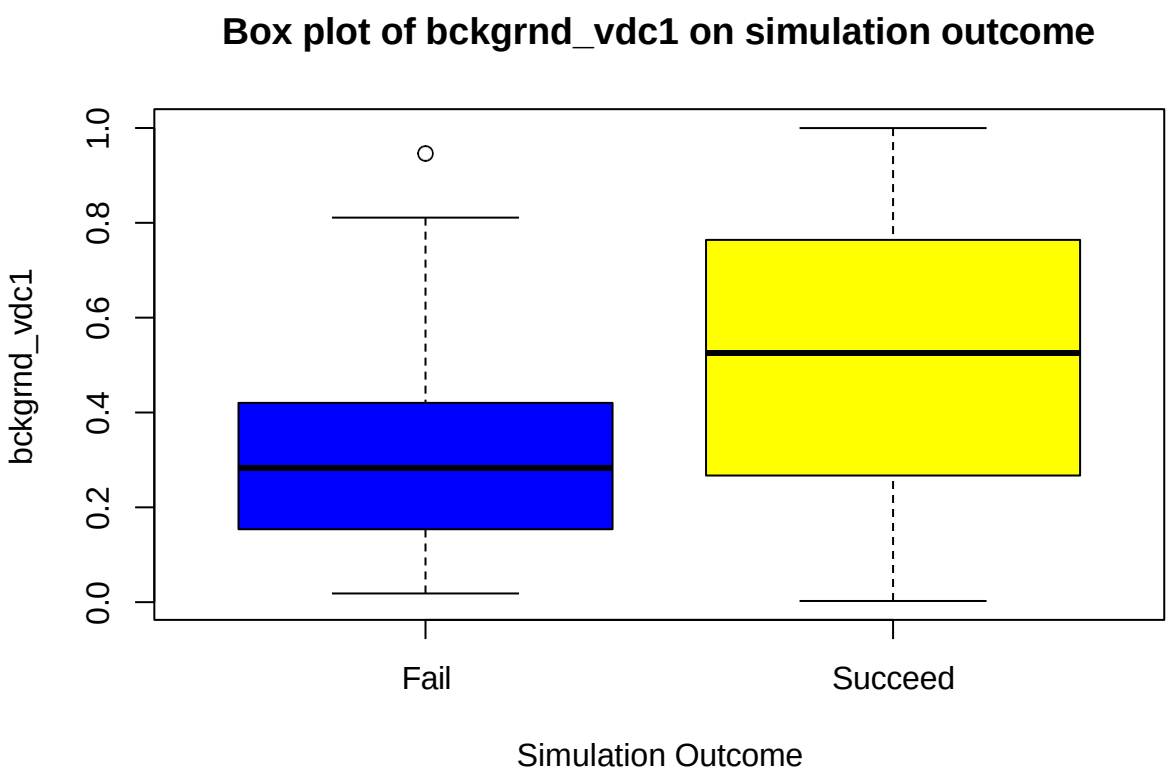
```
wilcox.test(dat$convect_corr~simulation_outcome,
                data=dat, alternative = "two.sided"
                ,na.action = na.omit)
```

```
##
##   Wilcoxon rank sum test with continuity correction
##
## data:    dat$convect_corr by simulation_outcome
## W = 15896, p-value = 7.499e-06
## alternative hypothesis: true location shift is not equal to 0
```

The p-value of the test above which is less than alpha level of significance 0.05 indicate there exist a significance relationship between predictor variable convect_corr on target variable "simulation outcome".

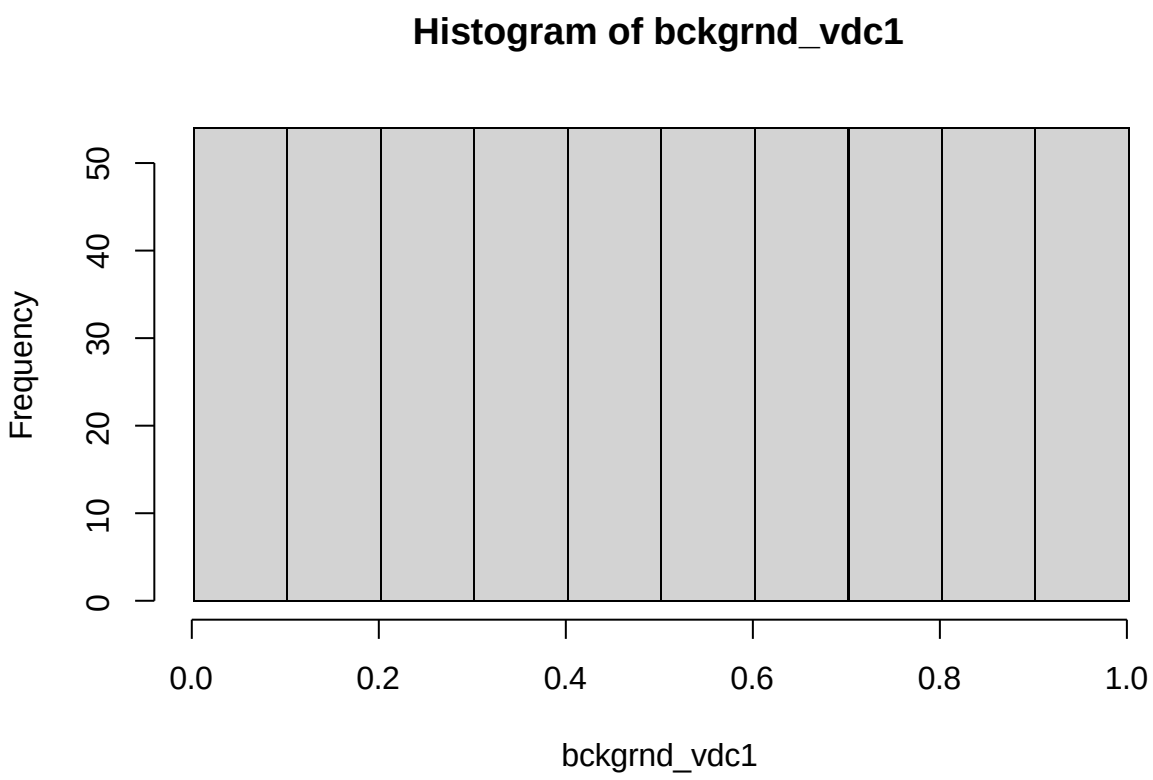
Simulation outcome and bckgrnd_vdc1

```
boxplot<-boxplot(dat$bckgrnd_vdc1~simulation_outcome,
        xlab="Simulation Outcome", ylab="bckgrnd_vdc1"
        ,main="Box plot of bckgrnd_vdc1 on simulation outcome",col=c("blue","yellow"))
```



The Boxplot above shows the distribution of the predictor variable bckgrnd_vdc1 on simulation outcome, which indicate the presence of association between predictor variable bckgrnd_vdc1 on target variable "simulation outcome".

```
hist(dat$bckgrnd_vdc1,main="Histogram of bckgrnd_vdc1",xlab="bckgrnd_vdc1")
```



The plot shows the distribution of the bckgrnd_vdc1 is uniform. Further evidence will be presented using the two normality test below.

```
library(nortest)
ad.test(dat$bckgrnd_vdc1)
```

```
##
##   Anderson-Darling normality test
##
## data:    dat$bckgrnd_vdc1
## A = 5.9659, p-value = 1.091e-14
```

```
shapiro.test(dat$bckgrnd_vdc1)
```

```
##
##   Shapiro-Wilk normality test
##
## data:    dat$bckgrnd_vdc1
## W = 0.95482, p-value = 8.411e-12
```

The p-value of both test are all less than alpha level of significance 0.05, which indicate that the distribution is not normal. Therefore the assumption of t-test is violated, then we use the non parametric alternative approach called wilcoxon rank sum test to examine the association between bckgrnd_vdc1 on simulation outcome.

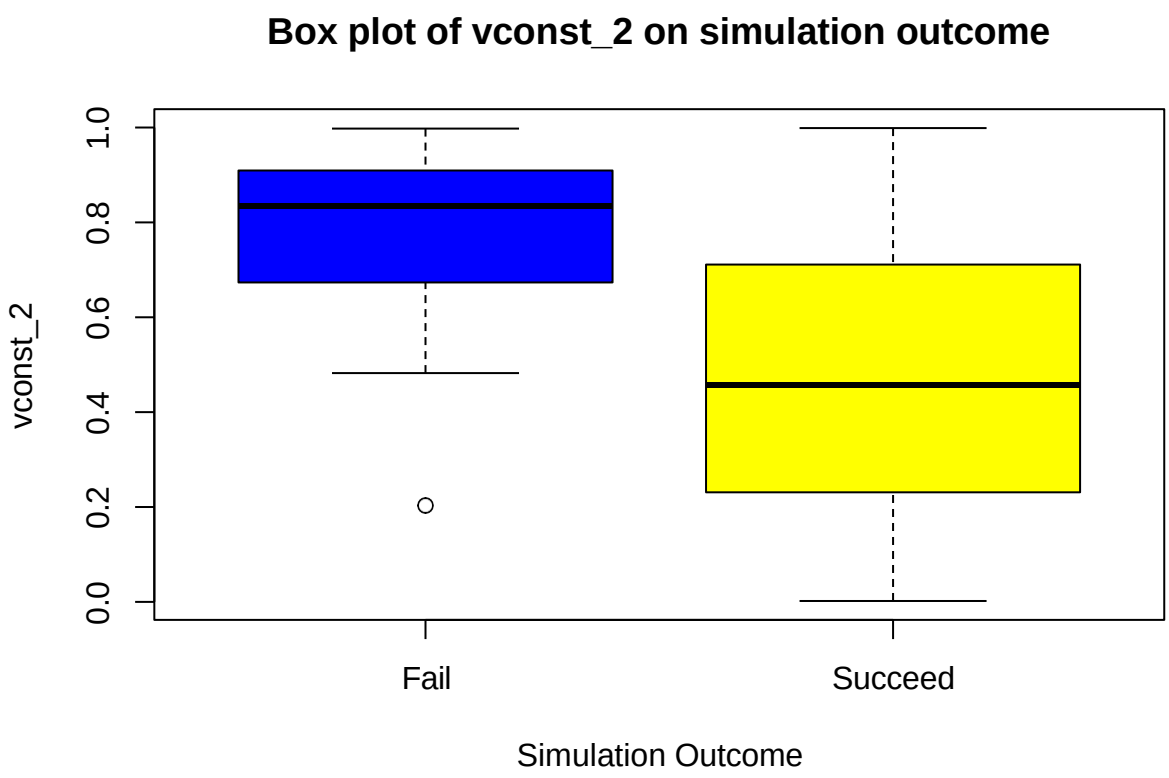
```
wilcox.test(dat$bckgrnd_vdc1~simulation_outcome,
                data=dat, alternative = "two.sided"
                ,na.action = na.omit)
```

```
##
##   Wilcoxon rank sum test with continuity correction
##
## data:     dat$bckgrnd_vdc1 by simulation_outcome
## W = 7029, p-value = 1.865e-05
## alternative hypothesis: true location shift is not equal to 0
```

The p-value of the test above which is less than alpha level of significance 0.05 indicate there exist a significance relationship between predictor variable bckgrnd_vdc1 on target variable "simulation outcome".

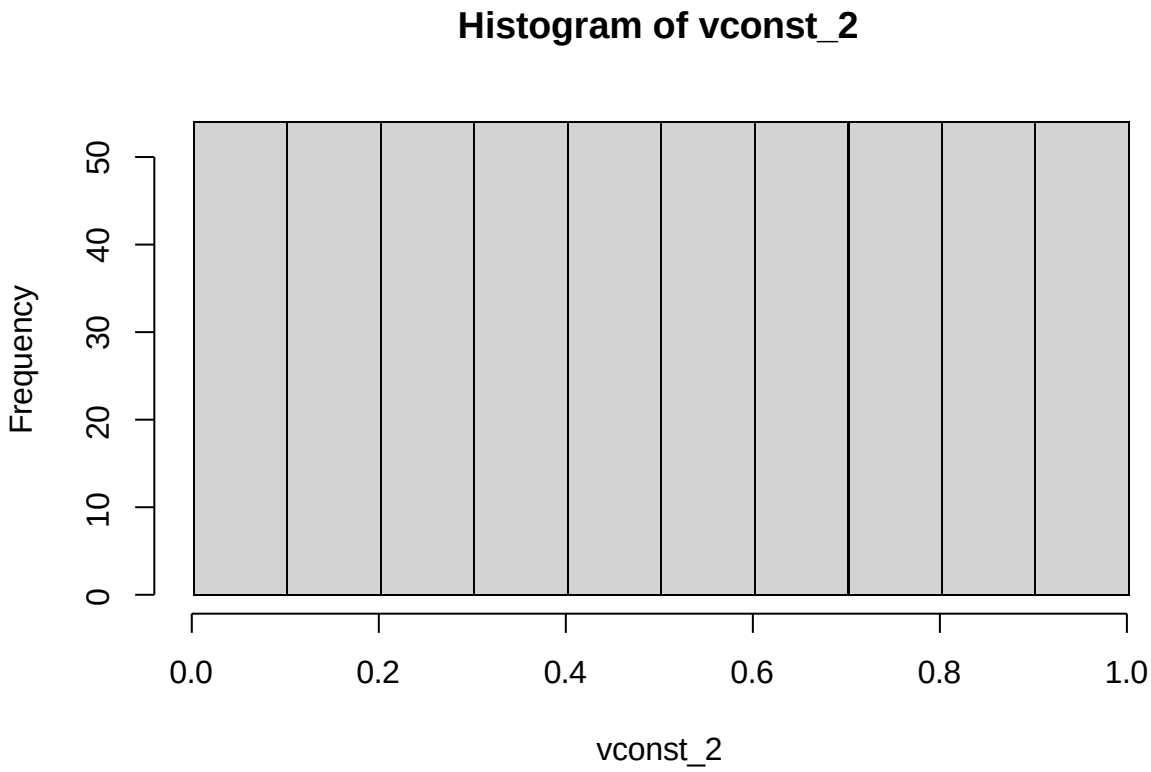
Simulation outcome and vconst_2

```
boxplot<-boxplot(dat$vconst_2~simulation_outcome
                        ,xlab="Simulation   Outcome",   ylab="vconst_2"
 ,main="Box plot of vconst_2 on simulation outcome",
 col=c("blue","yellow"))
```



The Boxplot above shows the distribution of the predictor variable vconst_2 on simulation outcome, which indicate the presence of association between predictor variable vconst_2 on target variable "simulation outcome".

```
hist(dat$vconst_2,main="Histogram of vconst_2",xlab="vconst_2")
```



The plot shows the distribution of the vconst_2 is uniform. Further evidence will be presented using the two normality test below.

```
library(nortest)
ad.test(dat$vconst_2)
```

```
##
##   Anderson-Darling normality test
##
## data:   dat$vconst_2
## A = 5.9754, p-value = 1.035e-14
```

```
shapiro.test(dat$vconst_2)
```

```
##
##   Shapiro-Wilk normality test
##
## data:   dat$vconst_2
## W = 0.95473, p-value = 8.131e-12
```

The p-value of both test are all less than alpha level of significance 0.05, which indicate that the distribution is not normal. Therefore the assumption of t-test is violated, then we use the non parametric alternative approach called wilcoxon rank sum test to examine the association between vconst_2 on simulation outcome.

```
wilcox.test(dat$vconst_2~simulation_outcome,
                data=dat, alternative = "two.sided"
                ,na.action = na.omit)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  dat$vconst_2 by simulation_outcome
## W = 18465, p-value = 2.264e-12
## alternative hypothesis: true location shift is not equal to 0
```

The p-value of the test above which is less than alpha level of significance 0.05 indicate there exist a significance relationship between predictor variable vconst_2 on target variable "simulation outcome".

5 DATA PARTITIONING

We will partition our data set into train and test with a ratio of approximately 2:1 in sample size. The train data will take 67% of the whole data set and the test takes will take 33% of the whole data set.

```
set.seed(123)
n <- nrow(dat)
split_data <- sample(x=1:2, size = n, replace=TRUE, prob=c(0.67, 0.33))
D1 <- dat[split_data == 1, ]
D2 <- dat[split_data == 2, ]
y.train <- D1$outcome
yobs <- D2$outcome
```

We have the training data set with "D1" as training input and "y.train" as training output. Finally we have the testing data set with "D2" as testing input and "yobs" as testing output.

6 MODEL BUILDING

Because the target variable is binary, we will use a Classifier machine model. In this project we will be using a Support Vector Machine classifier model which is would be suitable to the nature of our data set since the predictor variables have been scaled to interval [0,1]. We will explore the nonlinear nature of the Support Vector Machine using different types of kernel for the classification of the simulation outcome.

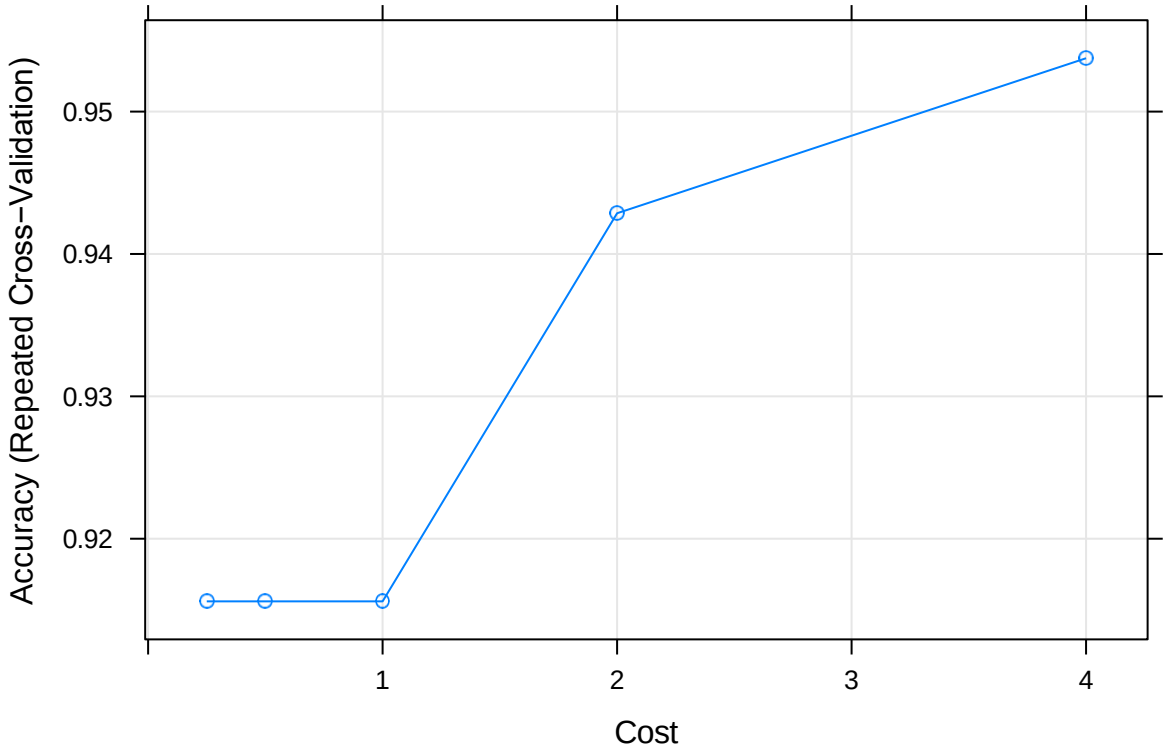
6.1 SVM Non-Linear with Radial Kernel function

```
library(caret) # Classification And Regression Training

set.seed(1492)
ctrl <- trainControl(method="repeatedcv",
                     number =10) # 10fold cross validation
svm_Radial <- train(factor(outcome)~., data = D1, method = "svmRadial",
                    trControl=ctrl,
                    tuneLength = 5)
svm_Radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 367 samples
##  18 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 329, 331, 331, 330, 330, 330, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.9155998  0.0000000
##   0.50  0.9155998  0.0000000
##   1.00  0.9155998  0.0000000
##   2.00  0.9428600  0.4643491
##   4.00  0.9537498  0.6097119
##
## Tuning parameter 'sigma' was held constant at a value of 0.02975469
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02975469 and C = 4.
```

After training the data using 10 fold cross validation, we have the above model summary.The final tuning parameters values used for the model were sigma = 0.02975469 and C = 4 having the highest accuracy of 95.4%.

```
plot(svm_Radial)
```



The plot tells us that with a significant increase in Cost values, the accuracy of the model also increases with the highest accuracy of 95.4% at C=4. After training the model, we have the prediction of the target variable using the test data below.

```
svm_pred_radial<- predict(svm_Radial, newdata = D2);
svm_pred_radial
```

```
##   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
## Levels: 0 1
```

```
svm_pred_radial <- as.numeric(as.character(svm_pred_radial))
MSE.f_radial <- mean((yobs-svm_pred_radial)^2)
MSE.f_radial
```

```
## [1] 0.06936416
```

An MSE of 7% makes us know the strength of trained model in predicting the original simulation outcome data. Although 7% MSE still seem high, more exploration of other SVM algorithm needs to be implemented to have a better model with minimal misclassification rate.

6.2 Model Evaluation

This decides if the model would perform better. Hence, it is important to consider the model outcomes according to every possible evaluation method. Applying different methods can provide different perspectives.

```
library(cvAUC)
svm_lin_AUC_radial <- ci.cvAUC(predictions = svm_pred_radial,
labels = yobs, folds=1:NROW(D2), confidence = 0.95)
svm_lin_AUC_radial
```

```
## $cvAUC
## [1] 0.6301688
##
## $se
## [1] 0.1503674
##
## $ci
## [1] 0.3354541 0.9248834
##
## $confidence
## [1] 0.95
```

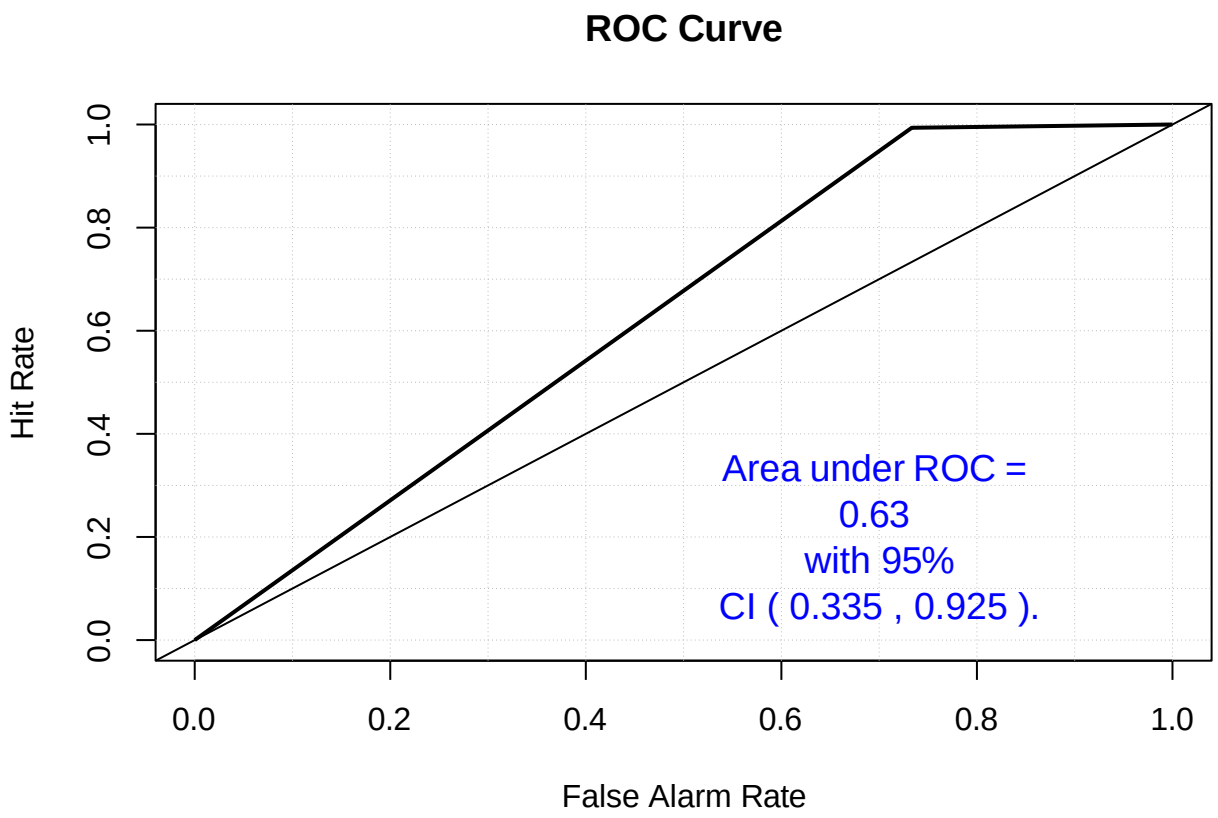
```
svm_lin_auc.ci_radial <- round(svm_lin_AUC_radial$ci, digits = 3)
```

AUC of 0.6301688 shows our fitted model has 63% ability to classify correctly the Simulation outcome is either "fail" or "succeed".
The confidence interval also indicate the true AUC falls within the interval (0.3354541 ,0.9248834). Therefore we are 95% confident that our AUC is accurate.

```
library(verification)
mod.svm_lin_radial<- verify(obs = yobs, pred = svm_pred_radial)
```

If baseline is not included, baseline values will be calculated from the sample obs

```
roc.plot(mod.svm_lin_radial, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = \n",
round(svm_lin_AUC_radial$cvAUC, digits = 3), "\n with 95% \nCI (",
 svm_lin_auc.ci_radial[1], ",", svm_lin_auc.ci_radial[2], ").", sep = " ")
, col="blue", cex =1.2)
```



The ROC curve shows the trade-off between sensitivity (or TPR) and False Positive Rate (1 – Specificity). It further indicates that the model performs better against the benchmark (50%) with total area of 0.63(63%). Though an AUC of 63% is not good enough.

6.3 SVM Non-Linear with Grid points and Radial Kernel function

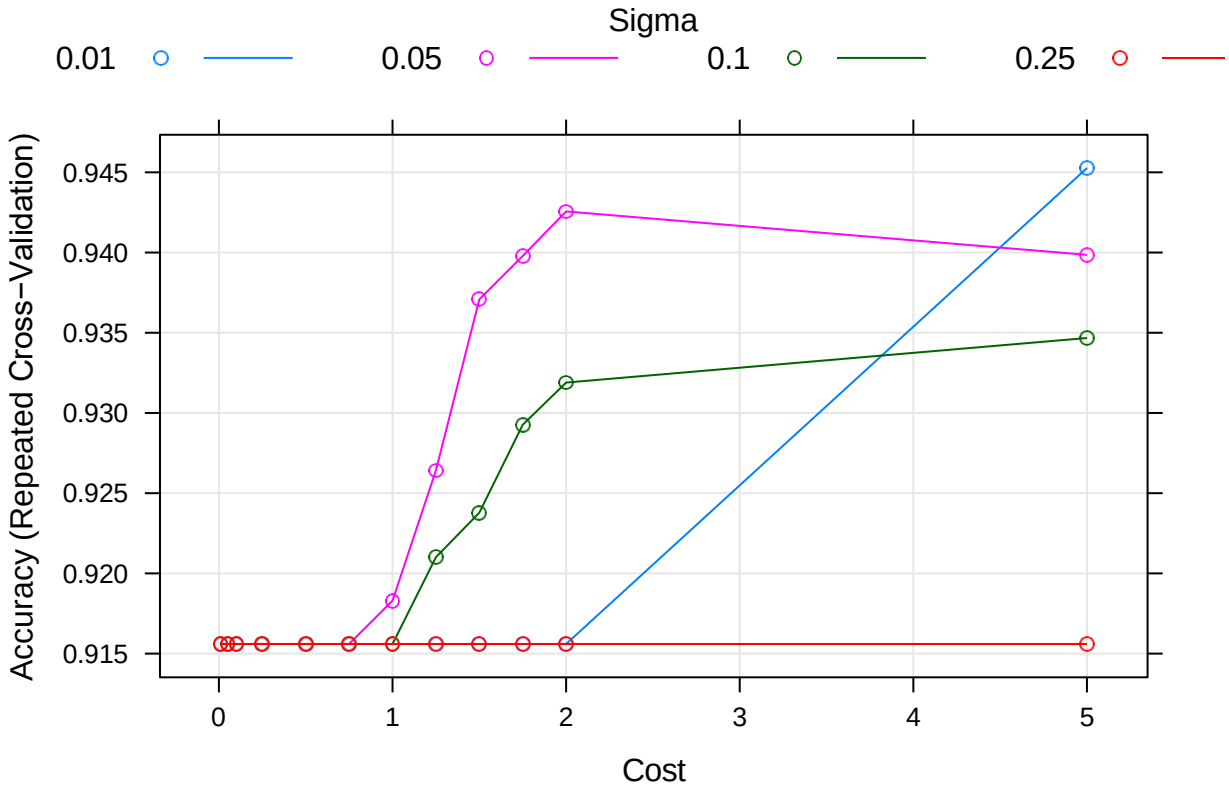
```
set.seed(3233)
ctrl <- trainControl(method="repeatedcv", # 10fold cross validation
        number =10)

grid <- expand.grid(C = c(0.01, 0.05, 0.1, 0.25, 0.5,
0.75, 1, 1.25, 1.5, 1.75, 2,5)
,sigma=c(0.01, 0.05, 0.1,0.25))
svm_nonLinear_Grid_radial <- train(factor(outcome) ~.,
data = D1, method = "svmRadial",trControl=ctrl,
tuneGrid = grid)
svm_nonLinear_Grid_radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 367 samples
##  18 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10-fold, repeated 1 times)
## Summary of sample sizes: 330, 331, 331, 330, 331, 330, ...
## Resampling results across tuning parameters:
##
##    C     sigma  Accuracy   Kappa
##    0.01  0.01   0.9155998  0.00000000
##    0.01  0.05   0.9155998  0.00000000
##    0.01  0.10   0.9155998  0.00000000
##    0.01  0.25   0.9155998  0.00000000
##    0.05  0.01   0.9155998  0.00000000
##    0.05  0.05   0.9155998  0.00000000
##    0.05  0.10   0.9155998  0.00000000
##    0.05  0.25   0.9155998  0.00000000
##    0.10  0.01   0.9155998  0.00000000
##    0.10  0.05   0.9155998  0.00000000
##    0.10  0.10   0.9155998  0.00000000
##    0.10  0.25   0.9155998  0.00000000
##    0.25  0.01   0.9155998  0.00000000
##    0.25  0.05   0.9155998  0.00000000
##    0.25  0.10   0.9155998  0.00000000
##    0.25  0.25   0.9155998  0.00000000
##    0.50  0.01   0.9155998  0.00000000
##    0.50  0.05   0.9155998  0.00000000
##    0.50  0.10   0.9155998  0.00000000
##    0.50  0.25   0.9155998  0.00000000
```

```
##    0.75  0.01  0.9155998  0.00000000
##    0.75  0.05  0.9155998  0.00000000
##    0.75  0.10  0.9155998  0.00000000
##    0.75  0.25  0.9155998  0.00000000
##    1.00  0.01  0.9155998  0.00000000
##    1.00  0.05  0.9183025  0.04788732
##    1.00  0.10  0.9155998  0.00000000
##    1.00  0.25  0.9155998  0.00000000
##    1.25  0.01  0.9155998  0.00000000
##    1.25  0.05  0.9264146  0.16380144
##    1.25  0.10  0.9210052  0.07861272
##    1.25  0.25  0.9155998  0.00000000
##    1.50  0.01  0.9155998  0.00000000
##    1.50  0.05  0.9370831  0.34222526
##    1.50  0.10  0.9237830  0.12643880
##    1.50  0.25  0.9155998  0.00000000
##    1.75  0.01  0.9155998  0.00000000
##    1.75  0.05  0.9397858  0.39011258
##    1.75  0.10  0.9292635  0.22215221
##    1.75  0.25  0.9155998  0.00000000
##    2.00  0.01  0.9155998  0.00000000
##    2.00  0.05  0.9425636  0.43793867
##    2.00  0.10  0.9318951  0.25951485
##    2.00  0.25  0.9155998  0.00000000
##    5.00  0.01  0.9452663  0.45158332
##    5.00  0.05  0.9398570  0.49963223
##    5.00  0.10  0.9346728  0.30734094
##    5.00  0.25  0.9155998  0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 5.
```

After training the data using 10 fold cross validation and grid points of tuning parameters Cost values(C) and sigma, we have the above model summary. The final tuning parameters values used for the model were sigma = 0.01 and C = 5 having the highest accuracy of 94.5%.

```
plot(svm_nonLinear_Grid_radial)
```



It can be deduced from the above plot that the Cost values are at their maximum best with small Sigma values. However, the model performs at its best with tuning parameters C=5 and sigma=0.01 with 94.5% accuracy.

```
svm_cLinear_pred_Grid_radial <- predict(svm_nonLinear_Grid_radial,
  newdata = D2)
svm_cLinear_pred_Grid_radial
```

```
##   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
## Levels: 0 1
```

```
svm_cLinear_pred_Grid_radial <- as.numeric(as.character(svm_cLinear_pred_Grid_radial))
MSE.b_Grid_radial <- mean((yobs-svm_cLinear_pred_Grid_radial)^2)
MSE.b_Grid_radial
```

```
## [1] 0.06358382
```

An MSE of 6% shows the strength of the trained model in predicting
the original simulation outcome data. With a reduction in MSE, the SVM nonlinear
with grid points of tuning parameters perform better than when we trained the model
without grid points of tuning parameters.
Also,further exploration of other SVM algorithm will still be implement
to have a better model with minimal MSE.

6.4 Model Evaluation

```
library(cvAUC)
svmc_AUC_Grid_radial <- ci.cvAUC(predictions = svm_cLinear_pred_Grid_radial,
   labels =yobs,
   folds=1:NROW(D2), confidence = 0.95)
svmc_AUC_Grid_radial
```

```
## $cvAUC
## [1] 0.6635021
##
## $se
## [1] 0.1506376
##
## $ci
## [1] 0.3682578 0.9587464
##
## $confidence
## [1] 0.95
```

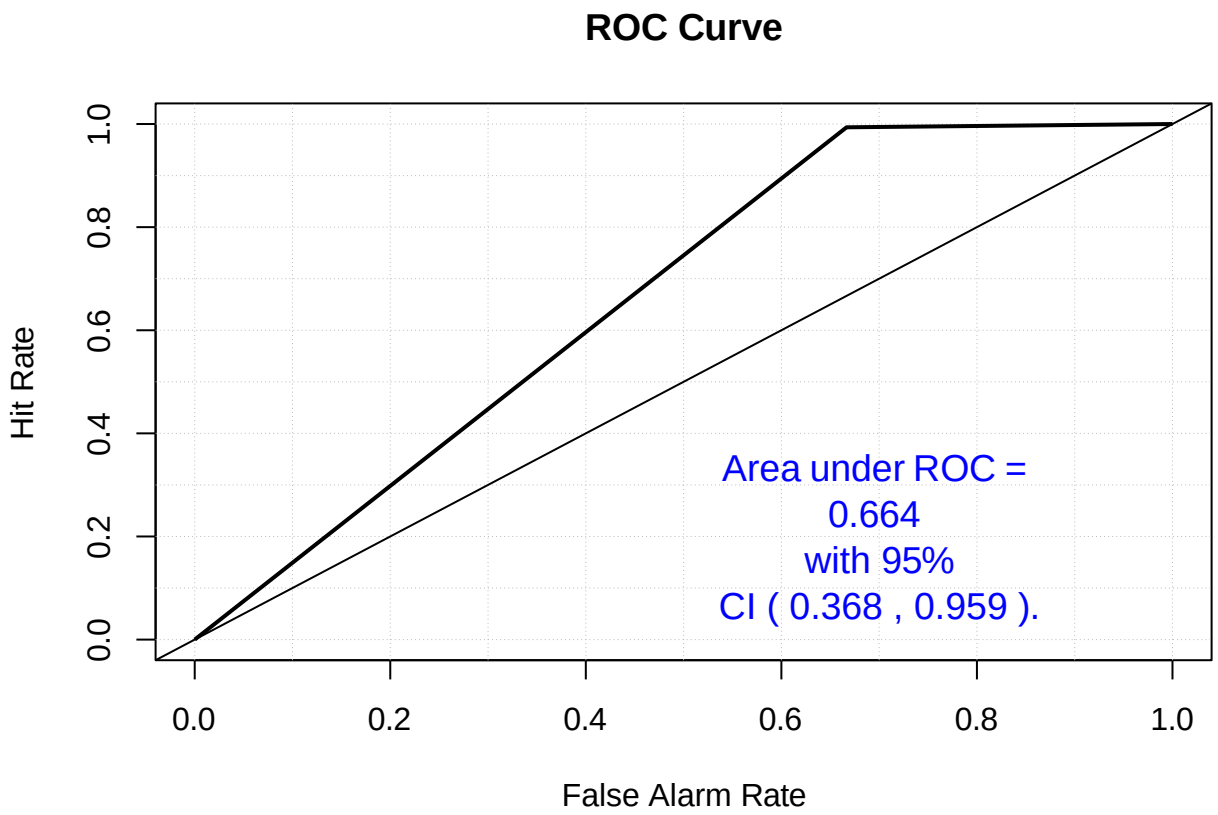
```
svmc_auc.ci_Grid_radial <- round(svmc_AUC_Grid_radial$ci, digits = 3)
```

AUC of 0.6635021 indicate that our fitted model has 66% ability to classify
correctly either the Simulation outcome is either “fail” or “succeed”.
The confidence interval also indicate the true AUC falls within the interval
(0.3682578,0.9587464). Therefore we are 95% confident that our AUC is accurate.

```
library(verification)
mod.svmc_Grid_radial <- verify(obs = yobs, pred = svm_cLinear_pred_Grid_radial)
```

```
## If baseline is not included, baseline values will be calculated from the sample obs
```

```
roc.plot(mod.svmc_Grid_radial, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = \n",
round(svmc_AUC_Grid_radial$cvAUC, digits = 3), "\n with 95% \nCI (",
svmc_auc.ci_Grid_radial[1], ",", svmc_auc.ci_Grid_radial[2], ").", sep = " "),
col="blue", cex =1.2)
```



The ROC curve above shows the trade-off between sensitivity (or TPR) and False Positive Rate(1 – Specificity). It further indicates that the model performs better against the bench mark (50%) with total area of 0.66(66%). Though an AUC of 66% is not good enough.

6.5 SVM Non-Linear with Grid points and Gaussian Radial basis method

We now walk through a faster algorithm using kernlab and e1071 packages for tuning the model parameters for better classification performance.

```
library(kernlab)
library("e1071")
set.seed(12345)
tobj_Gausian_Radial<- tune.svm(outcome ~ ., data = D1,
gamma = 10^c(-4:-1), cost = 10^c(-2:2), nrepeat=3,
tunecontrol = tune.control(sampling = "cross", cross=5))
tobj_Gausian_Radial
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
## gamma cost
##   0.1   10 ##
## - best performance: 0.05571038
```

```
bestGamma_Gausian_Radial <- tobj_Gausian_Radial$best.parameters[[1]]
bestC_Gausian_Radial <- tobj_Gausian_Radial$best.parameters[[2]]
```

After tuning the model with 5 fold cross validation, gamma = $[10^{-4},10^{-1}]$ and Cost = $[10^{-2},10^{2}]$. We have the optimal tuning parameters of *Sigma*(*Gamma*) = 0.1, *Cost* = 10.

Model summary below:

```
svm_probab_Gausian_Radial <- ksvm(outcome ~ ., data = D1,
type = "C-bsvc", kernel = "rbfdot",
kpar = list(sigma = bestGamma_Gausian_Radial), C = bestC_Gausian_Radial,
scaled=TRUE, prob.model = TRUE)
svm_probab_Gausian_Radial
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-bsvc (classification)
##  parameter : cost C = 10
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma = 0.1
##
```

```
## Number of Support Vectors : 188
##
## Objective Function Value : -52.1074
## Training error : 0
## Probability model included.
```

With predicted values:

```
svm_probab_pred_Gausian_Radial <- predict(svm_probab_Gausian_Radial, type="probabilities
svm_yhat_Gausian_Radial <- svm_probab_pred_Gausian_Radial[, 2]
svm_yhat_factor_Gausian_Radial <- ifelse(svm_yhat_Gausian_Radial>0.5, 1, 0)
svm_yhat_factor_Gausian_Radial
```

```
##   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
```

6.6 Model Evaluation

```
library(cvAUC)
svmprob_AUC_Gausian_Radial <- ci.cvAUC(predictions = svm_yhat_Gausian_Radial,
labels =yobs, folds=1:NROW(D2), confidence = 0.95)
svmprob_AUC_Gausian_Radial
```

```
## $cvAUC
## [1] 0.9054852
##
## $se
## [1] 0.03867006
##
## $ci
## [1] 0.8296933 0.9812772
##
## $confidence
## [1] 0.95
```

```
svmprob_auc.ci_Gausian_Radial <- round(svmprob_AUC_Gausian_Radial$ci,
  digits = 3)
```

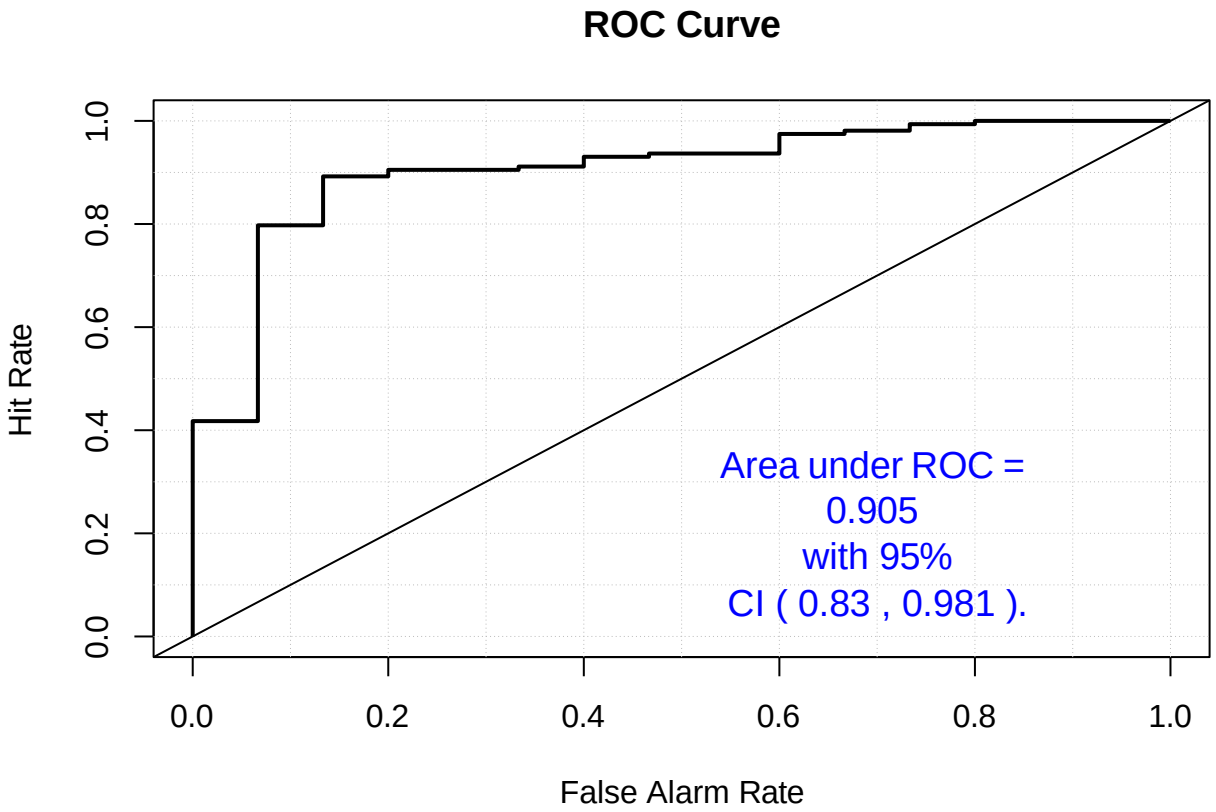
AUC of 0.9054852 indicates that our fitted model has 91% ability to classify correctly either the Simulation outcome is either "fail" or "succeed".

The confidence interval also shows the true AUC falls within the interval (0.8296933,0.9812772).Hence, we are 95% confident that our AUC is accurate.

```
library(verification)
mod.svmprob_Gausian_Radial <- verify(obs = yobs, pred = svm_yhat_Gausian_Radial)
```

```
## If baseline is not included, baseline values will be calculated from the sample obs
```

```
roc.plot(mod.svmprob_Gausian_Radial, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = \n",
round(svmprob_AUC_Gausian_Radial$cvAUC, digits = 3), "\n with 95% \nCI (",
svmprob_auc.ci_Gausian_Radial[1], ",",
svmprob_auc.ci_Gausian_Radial[2], ").", sep = " "),
col="blue", cex =1.2)
```



The ROC curve above shows the trade-off between sensitivity (or TPR) and False Positive Rate(1 – Specificity). It further indicates that the model performs

better against the bench mark (50%) with total area of 0.91(91%).It means the classification strength of our Machine learning model is 91%.

```
MSE.f_Gausian_Radial <- mean((yobs-svm_yhat_Gausian_Radial)^2)
MSE.f_Gausian_Radial
```

```
## [1] 0.06116435
```

```
(miss.rate_Gausian_Radial<- mean(yobs != svm_yhat_factor_Gausian_Radial))
```

```
## [1] 0.07514451
```

The MSE of our SVM model is 0.06116435 and Miss classification rate is 0.07514451. With a low MSE and Misclassification rate we can conclude that our machine learning model accuracy is high.

```
confusionMatrix(factor(yobs), factor(svm_yhat_factor_Gausian_Radial))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction 0   1
##            0   6   9
##            1   4 154
##
##                Accuracy : 0.9249
##                  95% CI : (0.8749, 0.9594)
##     No Information Rate : 0.9422
##     P-Value [Acc > NIR] : 0.8710
##
##                   Kappa : 0.4412
##
##  Mcnemar's Test P-Value : 0.2673
##
##             Sensitivity : 0.60000
##             Specificity : 0.94479
##          Pos Pred Value : 0.40000
##          Neg Pred Value : 0.97468
##              Prevalence : 0.05780
##          Detection Rate : 0.03468
##    Detection Prevalence : 0.08671
```

```
##          Balanced Accuracy : 0.77239
##
##           'Positive' Class : 0
##
```

From the result of the Confusion matrix, the positive is represented as "fail" with the value "0" and the negative is represented as "succeed" with value "1". Therefore, the Sensitivity (True Positive rate) also Known as "Recall" of 0.6000 (60%) calculated as $\frac{6}{6+4}$shows the model has a fair percentage of detecting a fail simulation outcome.
The Specificity (True Negative rate) of 0.94479 (94%) calculated as $\frac{154}{154+9}$ shows the model has a higher percentage of detecting a successful

simulation outcome of a Climate Model Simulation Crashes data.
Our trained SVM Model has an accuracy of 92.5% in terms of performance with a precision (Pos Pred Value) of 40% $\frac{6}{6+9}$ that indicate our Model has a high false positive rate (high wrong classification of failed simulation outcome).
Finally with a Negative Predicted Value of 97.4%$\frac{154}{154+4}$ that indicates our Model has a very low false negative rate (a very low wrong classification of successful simulation outcome).

6.7 SVM Non-Linear with Grid points and Bessel kernel method

```
set.seed(12345)
tobj_Bessel<- tune.svm(outcome ~ ., data = D1,degree=2:5
  ,gamma = 10^c(-4:-1), cost = 10^c(-2:2), nrepeat=3,
  tunecontrol = tune.control(sampling = "cross", cross=5))

tobj_Bessel
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  degree gamma cost
##          2   0.1   10 ##
## - best performance: 0.05571038
```

```
(bestDegree_Bessel<- tobj_Bessel$best.parameters[[1]])
```

```
## [1] 2
```

```
(bestGamma_Bessel <- tobj_Bessel$best.parameters[[2]])
```

```
## [1] 0.1
```

```
(bestC_Bessel <- tobj_Bessel$best.parameters[[3]])
```

```
## [1] 10
```

After tuning the model with 5 fold cross validation, gamma = $[10^{-4},10^{-1}]$ and Cost = $[10^{-2},10^{2}]$. We have the optimal tuning parameters of *Sigma*(*Gamma*) = 0.1, *Cost* = 10 and *Degree* = 2 .

Model summary1 below:

```
svm_probab_Bessel1<- ksvm(outcome ~ ., data = D1, type = "C-bsvc",
   kernel = "besseldot",
   kpar = list(sigma = bestGamma_Bessel,degree=bestDegree_Bessel,order=2),
   C = bestC_Bessel, scaled=TRUE, prob.model = TRUE)
svm_probab_Bessel1
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-bsvc (classification)
##  parameter : cost C = 10
##
## Bessel kernel function.
##  Hyperparameter : sigma = 0.1 order =  2 degree =  2
##
## Number of Support Vectors : 71
##
## Objective Function Value : -555.0981
## Training error : 0.076294
## Probability model included.
```

with *Order* = 2, we have a training error of 0.076294

Model summary2 below:

```
svm_probab_Bessel2<- ksvm(outcome ~ ., data = D1, type = "C-bsvc",
   kernel = "besseldot",
   kpar = list(sigma = bestGamma_Bessel,degree=bestDegree_Bessel,order=1),
   C = bestC_Bessel, scaled=TRUE, prob.model = TRUE)
svm_probab_Bessel2
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-bsvc (classification)
##  parameter : cost C = 10
##
## Bessel kernel function.
##  Hyperparameter : sigma = 0.1 order =  1 degree =  2
##
## Number of Support Vectors : 71
##
## Objective Function Value : -521.8385
## Training error : 0.054496
## Probability model included.
```

with *Order* = 1, we have a training error of 0.054496. We have an optimized *Order* = 1 with the minimum training error. *Order* > 2 gives a training error of 0.08447. Moving forward, we will be training the SVM model with the optimized tuning parameters *Sigma*(*Gamma*) = 0.1, *Cost* = 10, *Degree* = 2 and *Order* = 1.

With predicted values:

```
svm_probab_pred_Bessel2<-  predict(svm_probab_Bessel2,  type="probabilities"
, newdata=D2)
svm_yhat_Bessel2 <- svm_probab_pred_Bessel2[, 2]
svm_yhat_factor_Bessel2 <- ifelse(svm_yhat_Bessel2>0.5, 1, 0)
svm_yhat_factor_Bessel2
```

```
##   [1] 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
## [112] 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [149] 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
```

6.8 Model Evaluation

```
svmprob_AUC_Bessel2<- ci.cvAUC(predictions = svm_yhat_Bessel2,
labels =yobs, folds=1:NROW(D2), confidence = 0.95)
svmprob_AUC_Bessel2
```

```
## $cvAUC
## [1] 0.921519
##
## $se
## [1] 0.05053196
##
## $ci
## [1] 0.8224782 1.0000000
##
## $confidence
## [1] 0.95
```

```
(svmprob_auc.ci_Bessel2<- round(svmprob_AUC_Bessel2$ci, digits = 3))
```

```
## [1] 0.822 1.000
```

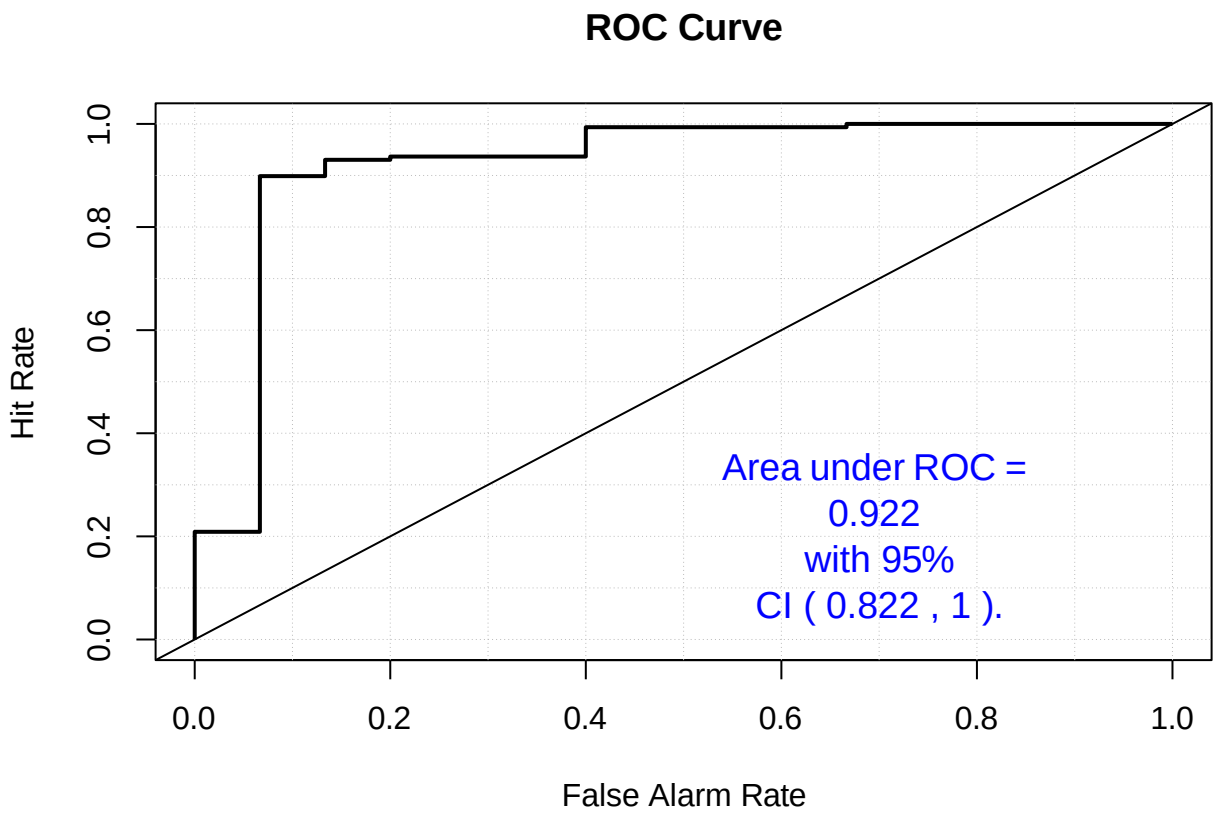
AUC of 0.921519 indicates that our fitted model has 92% ability to classify correctly either the Simulation outcome is either "fail" or "succeed".

The confidence interval also indicates the true AUC falls within the interval (0.8224782,1.0000000). Therefore we are 95% confident that our AUC is accurate.

```
library(verification)
mod.svmprob_Bessel2<- verify(obs = yobs, pred = svm_yhat_Bessel2)
```

```
## If baseline is not included, baseline values will be calculated from the sample obs
```

```
roc.plot(mod.svmprob_Bessel2, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = \n",
round(svmprob_AUC_Bessel2$cvAUC, digits = 3), "\n with 95% \nCI (",
svmprob_auc.ci_Bessel2[1], ",", svmprob_auc.ci_Bessel2[2], ").", sep = " "),
col="blue", cex =1.2)
```



The ROC curve above shows the trade-off between sensitivity (or TPR) and False Positive Rate(1 – Specificity). It further indicates that the model performs better against the bench mark (50%) with total area of 0.922(92%).Meaning the classification strength of our Machine learning model is 92%.

```
MSE.f_Bessel2 <- mean((yobs-svm_yhat_Bessel2)^2)
MSE.f_Bessel2
```

```
## [1] 0.04298274
```

```
(miss.rate_Bessel2 <- mean(yobs !=svm_yhat_factor_Bessel2))
```

```
## [1] 0.06358382
```

The MSE of our SVM model is 0.04298274 and Miss classification rate is 0.06358382. With a low MSE and Misclassification rate we can conclude that our machine learning model accuracy is high.

```
confusionMatrix(factor(yobs),  factor(svm_yhat_factor_Bessel2))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction 0   1
##            0    9    6
##            1    5 153
##
##                  Accuracy : 0.9364
##                    95% CI : (0.8891, 0.9678)
##       No Information Rate : 0.9191
##       P-Value [Acc > NIR] : 0.2493
##
##                     Kappa : 0.586
##
##   Mcnemar's Test P-Value : 1.0000
##
##               Sensitivity : 0.64286
##               Specificity : 0.96226
##            Pos Pred Value : 0.60000
##            Neg Pred Value : 0.96835
##                Prevalence : 0.08092
##            Detection Rate : 0.05202
##   Detection Prevalence : 0.08671 ##
          Balanced Accuracy : 0.80256
##
##          'Positive' Class : 0
##
```

From the result of the Confusion matrix, the positive is represented as "fail" with the value "0" and the negative is represented as "succeed" with value "1". Therefore, the Sensitivity (True Positive rate) also Known as "Recall" is 0.6429 (64%) calculated as $\frac{9}{9+5}$shows the model has a fair percentage of detecting a fail simulation outcome.

The Specificity (True Negative rate) of 0.96226(96%) calculated as $\frac{153}{153+6}$ shows the model has a higher percentage of detecting a successful simulation outcome of a Climate Model Simulation Crashes data.

Our trained SVM Model has an accuracy of 93.6% in terms of performance with a precision (Pos Pred Value) of 60% $\frac{9}{9+6}$ that indicate our Model has a high

false positive rate (high wrong classification of failed simulation outcome). Finally with a Negative Predicted Value of 97%$_{\frac{153}{153+5}}$ that indicate our Model has a very low false negative rate (a very low wrong classification of successful simulation outcome).

6.9 SVM Non-Linear with Grid points and AnovaRBF method

```
set.seed(12345)
tobj_ANovaRBF <- tune.svm(outcome ~ ., data = D1,degree=2:5
  ,gamma = 10^c(-4:-1), cost = 10^c(-2:2), nrepeat=3,
  tunecontrol = tune.control(sampling = "cross", cross=5))

tobj_ANovaRBF
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  degree gamma cost
##       2   0.1   10 ##
## - best performance: 0.05571038
```

```
(bestDegree_ANovaRBF<- tobj_ANovaRBF$best.parameters[[1]])
```

```
## [1] 2
```

```
(bestGamma_ANovaRBF <- tobj_ANovaRBF$best.parameters[[2]])
```

```
## [1] 0.1
```

```
(bestC_ANovaRBF <- tobj_ANovaRBF$best.parameters[[3]])
```

```
## [1] 10
```

After tuning the model with 5 fold cross validation, gamma = $[10^{-4},10^{-1}]$ and Cost = $[10^{-2},10^{2}]$. We have the optimal tuning parameters of *Sigma*(*Gamma*) = 0.1, *Cost* = 10 and *Degree* = 2 .

Model summary below:

```
svm_probab_ANovaRBF <- ksvm(outcome ~ ., data = D1, type = "C-bsvc",
  kernel = "anovadot",
  kpar = list(sigma = bestGamma_ANovaRBF,degree=bestDegree_ANovaRBF),
  C = bestC_ANovaRBF, scaled=TRUE, prob.model = TRUE)
svm_probab_ANovaRBF
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-bsvc (classification)
##  parameter : cost C = 10
##
## Anova RBF kernel function.
##  Hyperparameter : sigma = 0.1 degree =  2
##
## Number of Support Vectors : 56
##
## Objective Function Value : -2.4576
## Training error : 0.040872
## Probability model included.
```

With predicted values:

```
svm_probab_pred_ANovaRBF<-  predict(svm_probab_ANovaRBF,  type="probabilities"
, newdata=D2)
svm_yhat_ANovaRBF <- svm_probab_pred_ANovaRBF[, 2]
svm_yhat_factor_ANovaRBF <- ifelse(svm_yhat_ANovaRBF>0.5, 1, 0)
svm_yhat_factor_ANovaRBF
```

```
##   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

6.10 Model Evaluation

```
svmprob_AUC_ANovaRBF <- ci.cvAUC(predictions = svm_yhat_ANovaRBF,
labels =yobs, folds=1:NROW(D2), confidence = 0.95)
svmprob_AUC_ANovaRBF
```

```
## $cvAUC
## [1] 0.9278481
##
## $se
## [1] 0.04781827
##
## $ci
## [1] 0.834126 1.000000
##
## $confidence
## [1] 0.95
```

```
(svmprob_auc.ci_ANovaRBF<- round(svmprob_AUC_ANovaRBF$ci, digits = 3))
```

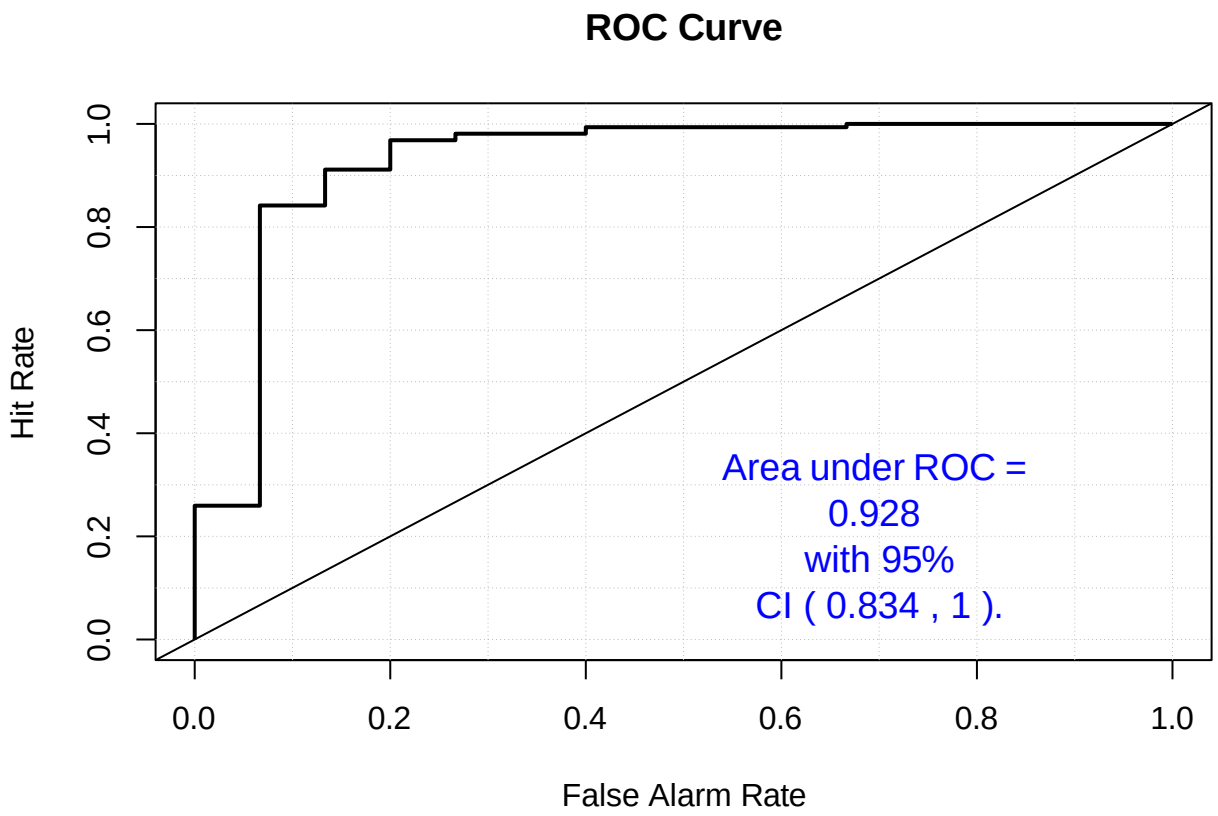
```
## [1] 0.834 1.000
```

AUC of 0.9278481 indicate that our fitted model has 93% ability to classify correctly either the Simulation outcome is either “fail” or “succeed”.
The confidence interval also indicate the true AUC falls within the interval (0.834126,1.0000000). Therefore we are 95% confident that our AUC is accurate.

```
library(verification)
mod.svmprob_ANovaRBF <- verify(obs = yobs, pred = svm_yhat_ANovaRBF)
```

```
## If baseline is not included, baseline values will be calculated from the sample obs
```

```
roc.plot(mod.svmprob_ANovaRBF, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = \n",
round(svmprob_AUC_ANovaRBF$cvAUC, digits = 3), "\n with 95% \nCI (",
svmprob_auc.ci_ANovaRBF[1], ",", svmprob_auc.ci_ANovaRBF[2], ").", sep = " "),
col="blue", cex =1.2)
```



The ROC curve above shows the trade-off between sensitivity (or TPR) and False Positive Rate(1 – Specificity). It further indicates that the model performs better against the bench mark (50%) with total area of 0.928(93%).This means the classification strength of our Machine learning model is 93%.

```
MSE.f_ANovaRBF <- mean((yobs-svm_yhat_ANovaRBF)^2)
MSE.f_ANovaRBF
```

```
##  [1]  0.03792806
```

```
(miss.rate_ANovaRBF <- mean(yobs !=svm_yhat_factor_ANovaRBF))
```

```
##  [1]  0.05202312
```

The MSE of our SVM model is 0.03792806 and Miss classification rate is 0.05202312. With a low MSE and Misclassification rate we can conclude that our machine learning model accuracy is high.

```
confusionMatrix(factor(yobs),  factor(svm_yhat_factor_ANovaRBF))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction 0   1
##            0    7    8
##            1    1  157
##
##                   Accuracy : 0.948
##                     95% CI : (0.9035, 0.9759)
##       No Information Rate : 0.9538
##       P-Value [Acc > NIR] : 0.7196
##
##                      Kappa : 0.5836
##
##   Mcnemar's Test P-Value : 0.0455
##
##                Sensitivity : 0.87500
##                Specificity : 0.95152
##             Pos Pred Value : 0.46667
##             Neg Pred Value : 0.99367
##                 Prevalence : 0.04624
##             Detection Rate : 0.04046
##   Detection Prevalence : 0.08671 ##
          Balanced Accuracy : 0.91326
##
##          'Positive' Class : 0
##
```

From the result of the Confusion matrix, the positive is represented as "fail" with the value "0" and the negative is represented as "succeed" with value "1". Therefore, the Sensitivity (True Positive rate) also Known as "Recall" is 0.8750 (88%) calculated as $\frac{7}{7+1}$shows the model has a perfect percentage of detecting a fail simulation outcome.

The Specificity(True Negative rate) of 0.9515(95%) calculated as $\frac{157}{157+8}$ shows the model has a higher percentage of detecting a successful simulation outcome of a Climate Model Simulation Crashes data.

Our trained SVM Model has an accuracy of 94.8% in terms of performance with a precision (Pos Pred Value) of 47% $\frac{7}{7+8}$ that indicate our Model has a high

false positive rate (high wrong classification of failed simulation outcome). Finally with a Negative Predicted Value of 99%$_{1\frac{157}{157+1}}$ that indicates our Model has a very low false negative rate (a very low wrong classification of successful simulation outcome).

7 MODEL COMPARISON

Three different kernel were used to train the SVM Model using kernlab and e1071 packages for tuning the model parameters for better classification performance. We have the analysis of the result below:

Evaluation metrics	Gaussian Radial basis	Bessel Kernel	AnovaRBF
Accuracy	0.9250	0.9360	0.9480
Sensitivity	0.6000	0.6429	0.8750
Specificity	0.9448	0.9623	0.9515
Precision	0.4000	0.6000	0.4667
Neg. Pred.Val	0.9747	0.9680	0.9937
MSE	0.0616	0.0430	0.0379
Miss.rate	0.0751	0.0636	0.0520
AUC	0.9055	0.9215	0.9278

Though AnovaRBF SVM appears to be the best model when avoiding false negative is prioritized over encountering false positive, that is if we are more interested in Recall (Sensitivity) than the Precision.
But, a model is considered a good fit if it performs well in all areas of evaluation metrics. Hence, we will consider Bessel kernel SVM has the best model because of its performance in all the evaluation metrics in the table above. There is a balanced trade off between its Precision and Recall(Sensitivity) when compared to the case of AnovaRBF.

8 SUMMARY AND CONCLUSION

Two different SVM algorithm were considered in training our machine learning model with the Climate Model Simulation Crashes data. In the first algorithm, we made use of a non linear kernel named Radial kernel function where we had two structure

of our tuning parameters. In the first case we supplied a tuning length of Cost values and in the second case we supplied Grid values of Cost and sigma. The SVM model with the grid points of turning parameter (Cost and Sigma) performs better than the one with a given tune length of Cost values in terms of AUC and MSE.
We then moved on with our second SVM algorithm using kernlab and e1071 packages for tuning the model parameters for better classification performance. Three different kernels were considered namely Gaussian Radial basis, Bessel Kernel, AnovaRBF. The three kernel performs well but Bessel kernel SVM perform best on the Climate Model Simulation Crashes data in terms of all the evaluation metrics considered.But if avoiding false negative is prioritized over encountering false positive then we can actually considered anovaRBF SVM has the best model.
In conclusion,the second SVM algorithm using kernlab and e1071 packages for tuning the model parameters for better classification performance appears to be the best for the classification of the simulation outcomes of a Climate Model Simulation Crashes data with 18 climate model parameters.