# Project-VI

*Md Al Masum Bhuiyan*

*December 12, 2018*

## Contents

# 1 Problem-1: Data Preparation

I begin the project VI by bringing the Climate Model Simulation Crashes data into Rmd. The data comes from the UCI machine learning repository. It contains 540 observations and 21 variables.

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

```
## [1] 540  21

##   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
## 1            0.8690907   0.99751850    0.4486201      0.30752179
## 2            0.9143437   0.84524714    0.8641519      0.34671269
## 3            0.6439952   0.71844113    0.9247751      0.31537141
## 4            0.7616588   0.36275056    0.9128191      0.97797118
## 5            0.3123494   0.65022283    0.5222610      0.04354476
## 6            0.6158677   0.01748718    0.9323195      0.32931804
##   bckgrnd_vdc_eq bckgrnd_vdc_psim   Prandtl outcome
## 1      0.8583104        0.7969972 0.8698930       0
## 2      0.3565734        0.4384472 0.5122561       1
## 3      0.2506424        0.2856355 0.3658580       1
## 4      0.8459212        0.6994309 0.4759867       1
## 5      0.3766601        0.2800978 0.1322829       1
## 6      0.9541228        0.1353789 0.2948048       1

## [1] FALSE
```

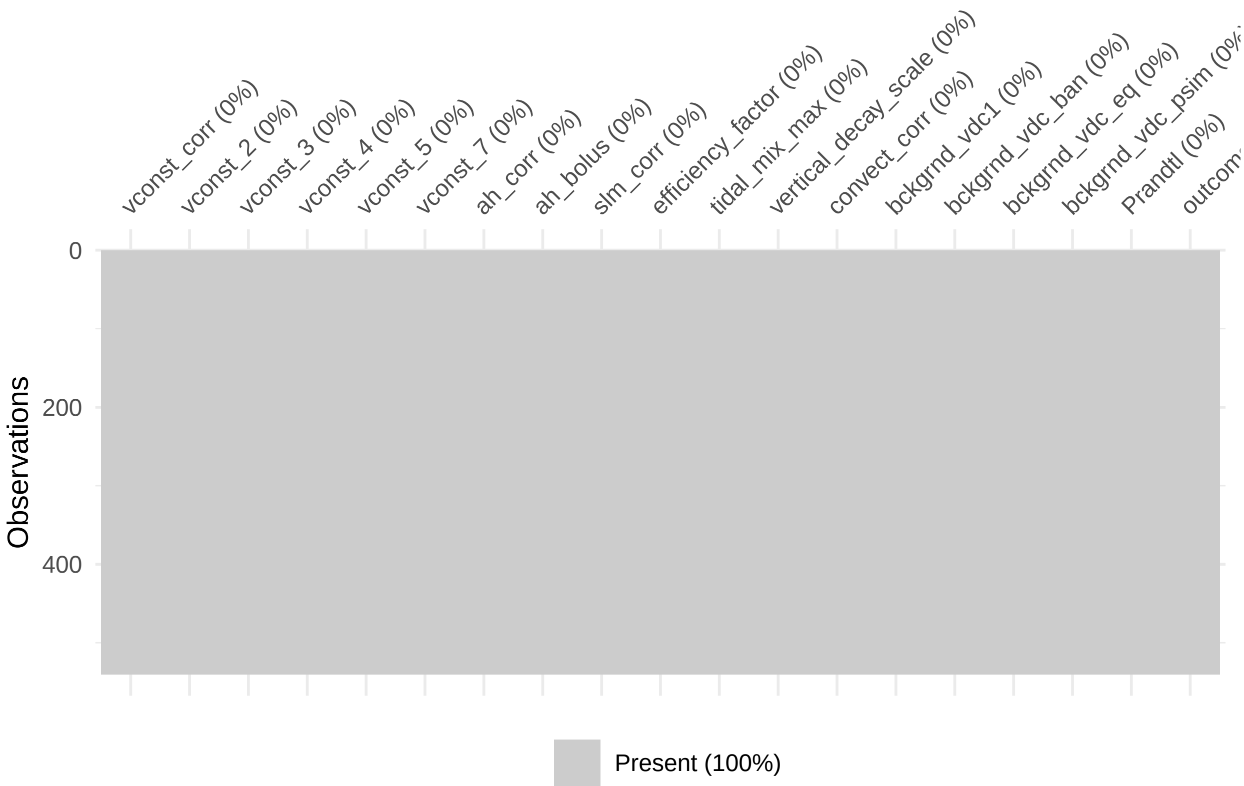
This dataset contains records of simulation crashes encountered during climate model uncertainty quantification (UQ) ensembles. I remove the first two columns and do the exploratory data analysis.

I remove the first two columns.

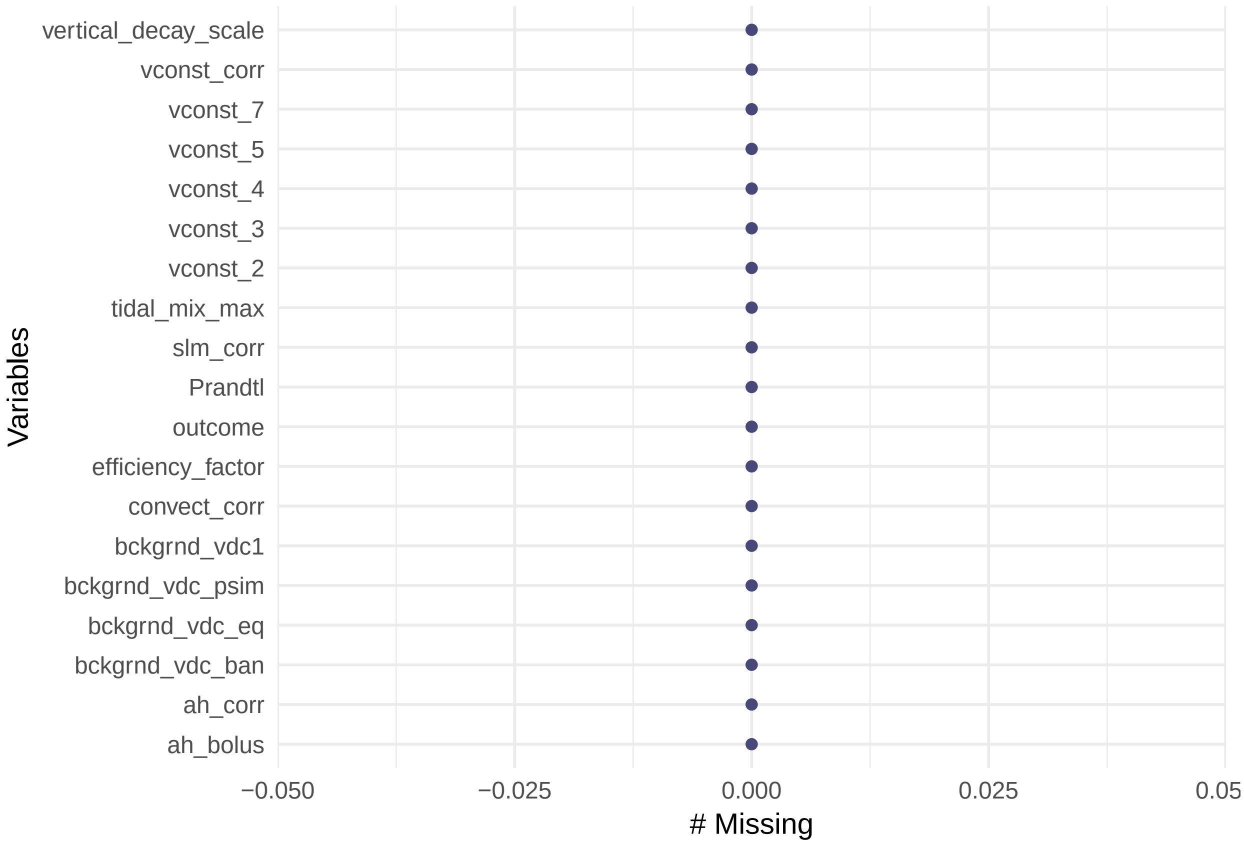
```
data <- dat[, c(-1,-2)]
```

Now I check the missing values.

```r
########  Missing values by visualization ###############
library(naniar)
vis_miss(data)
```

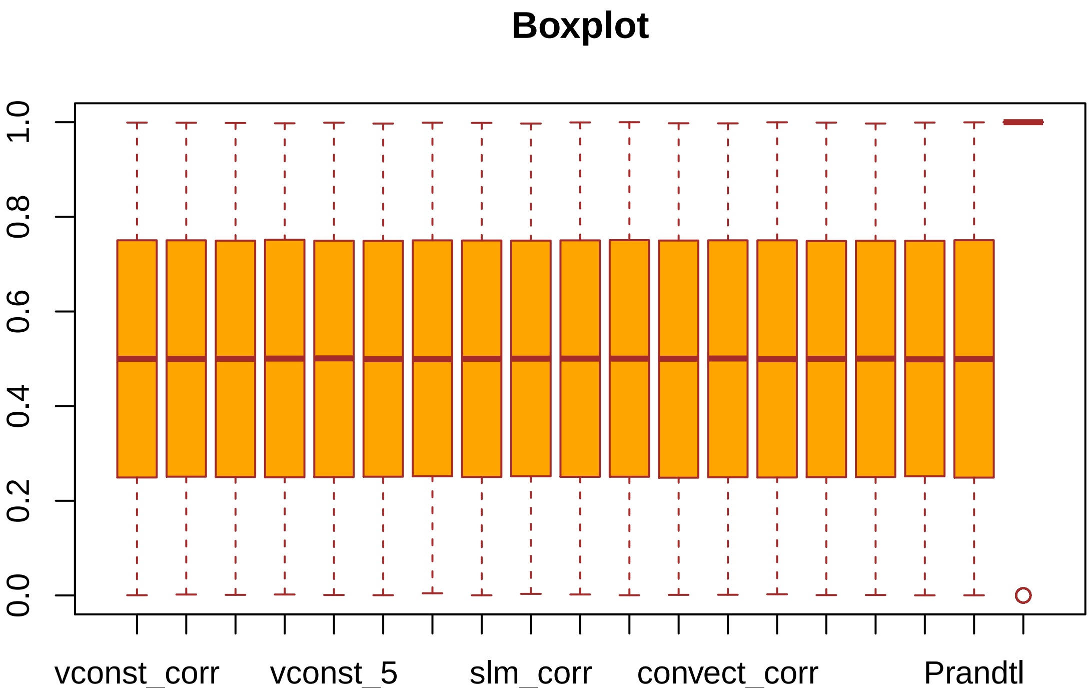



```r
gg_miss_var(data)
```



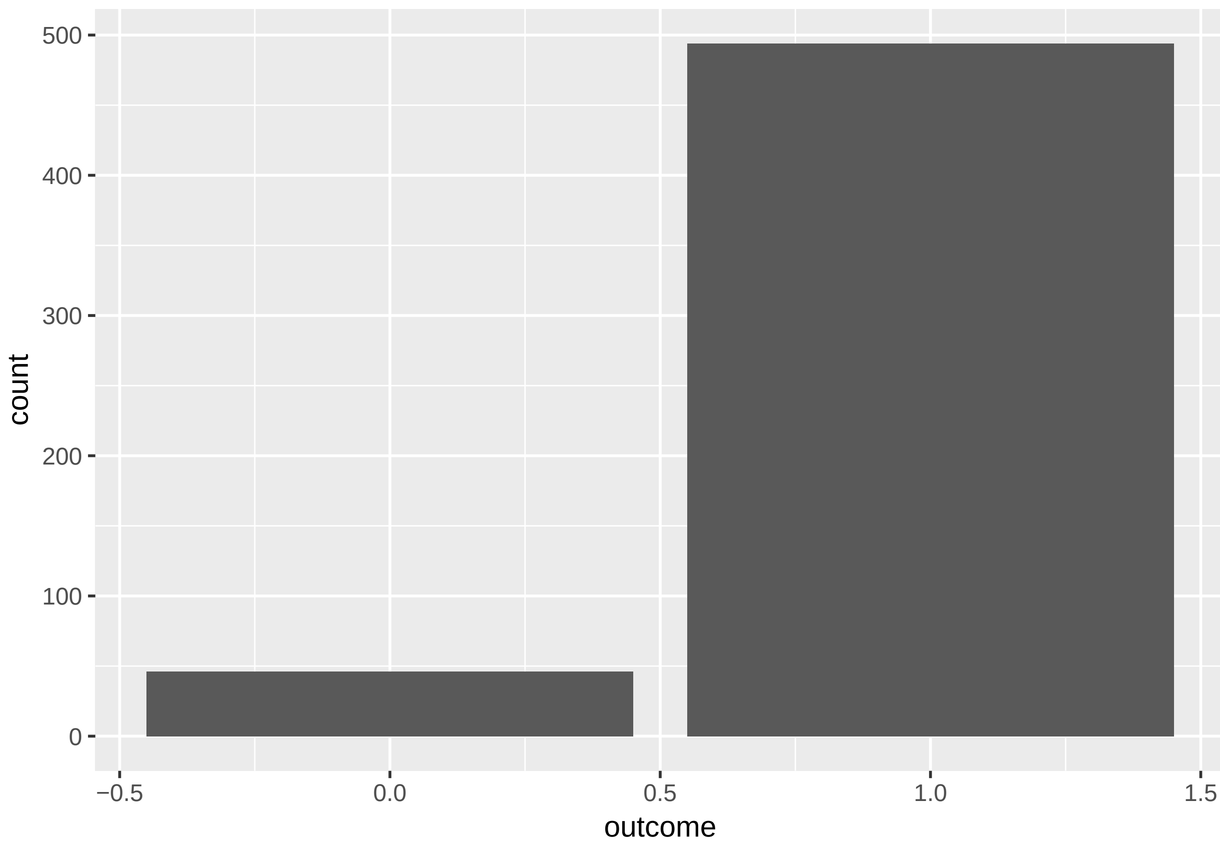

## 2 Problem-2: Exploratory Data Analysis

```
boxplot(data, main = "Boxplot", horizontal = F, col="orange",  border="brown")
```




Remarks: The box plot shows that the variablility of each predictors are same. Now I present another boxplot for outcome resonse and a predictor vconst_7.

```r
library(ggplot2)
ggplot(aes(x=outcome, fill = vconst_7), data = data)+geom_bar()
```



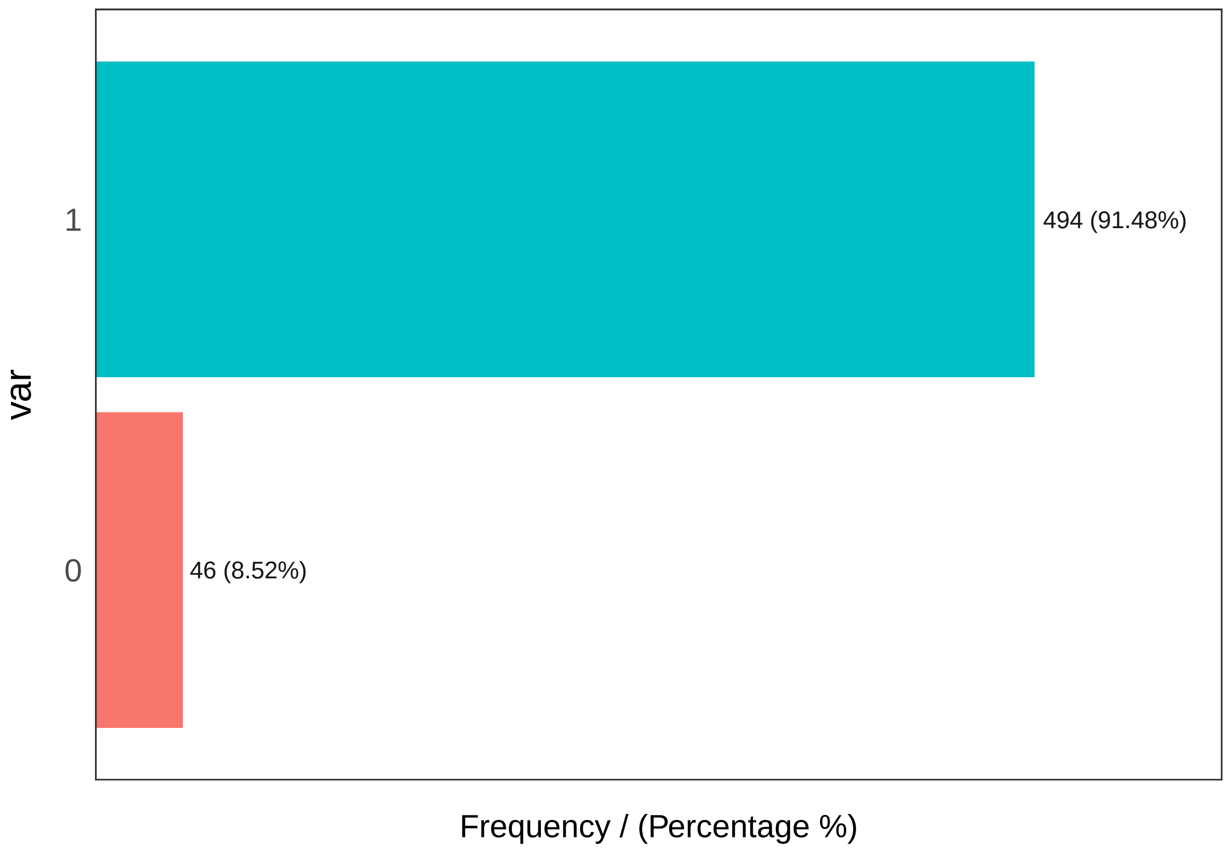
### 2.1 Frequency Distribution of the Target variable Outcome

The following code shows the frequency, percentage, cumulative percentage of the target variable Outcome..

```
library("funModeling")
```

```
## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, units

## funModeling v.1.6.8 :)
## Examples and tutorials at livebook.datascienceheroes.com
```

```
freq(data$outcome)
```

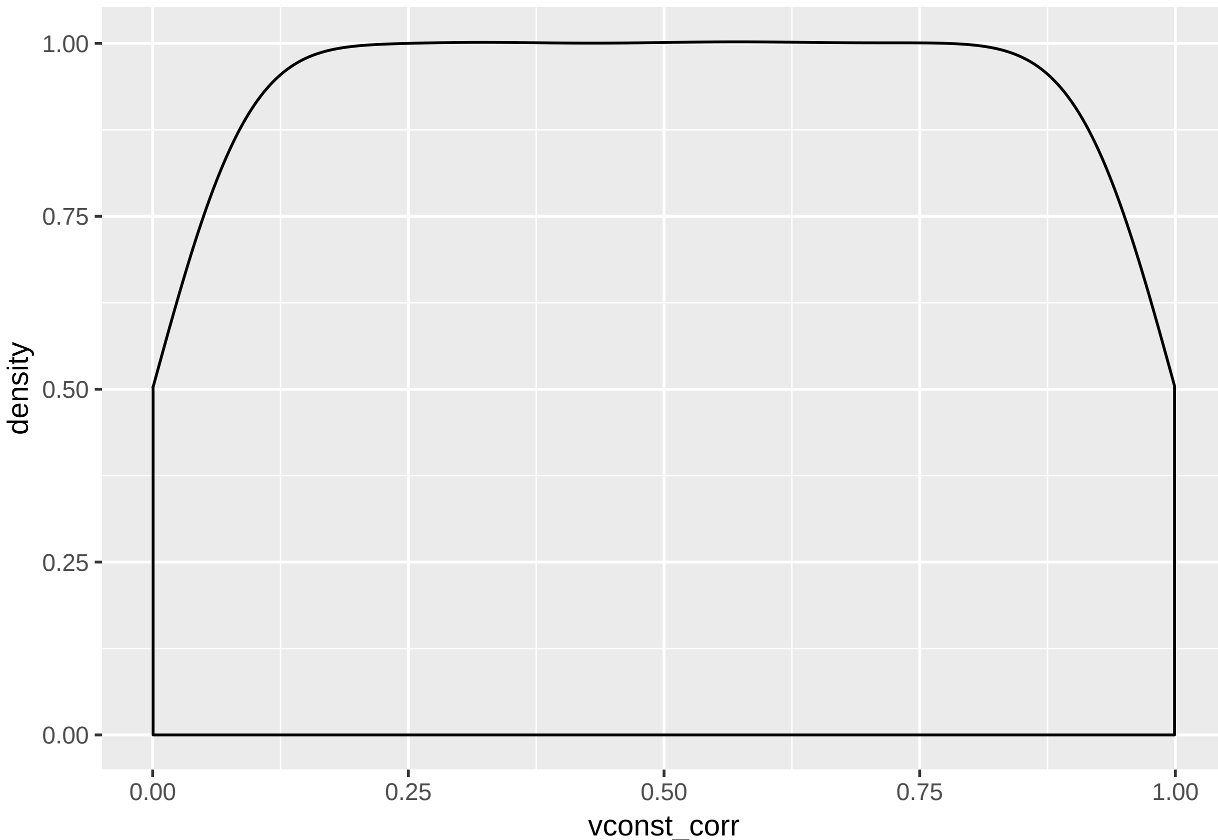



```
##   var frequency percentage cumulative_perc
```

```
## 1    1        494       91.48              91.48
## 2    0         46        8.52             100.00
```

## 2.2 Density plot

```r
library(ggplot2)
ggplot(aes(vconst_corr), data=data) + geom_density()
```




## 2.3 Correlation Matrix

From the corrleation matrix, we see the correlation among the variables. For exmaple, Vconst_1 and Vocnst-2 have the correlation values 3243, that is, positively correlated.

```r
cor(data, method = "pearson", use = "complete.obs")
```

```
##                         vconst_corr     vconst_2      vconst_3
## vconst_corr             1.000000000  0.0040390372  0.0093312390
## vconst_2                0.004039037  1.0000000000 -0.0004564662
## vconst_3                0.009331239 -0.0004564662  1.0000000000
## vconst_4               -0.018294219 -0.0006144773  0.0098992508
## vconst_5                0.018880130 -0.0082917043  0.0062887724
## vconst_7                0.001543507 -0.0243790678 -0.0015865605
## ah_corr                 0.003713983 -0.0051821138  0.0199408652
```

```
## ah_bolus            -0.012735004  0.0041791244  0.0044018146
## slm_corr             0.002336425 -0.0138597114 -0.0076953527
## efficiency_factor    0.010617309 -0.0110718393  0.0071000071
## tidal_mix_max       -0.014205434  0.0197060654 -0.0094284119
## vertical_decay_scale -0.008991676  0.0016229327 -0.0247021845
## convect_corr        -0.002980494  0.0026084619 -0.0206373211
## bckgrnd_vdc1        -0.002132558 -0.0147164701 -0.0042641402
## bckgrnd_vdc_ban     -0.002098679  0.0043858772 -0.0052104219
## bckgrnd_vdc_eq       0.015972793  0.0059985634 -0.0005588026
## bckgrnd_vdc_psim    -0.016631072  0.0042022103  0.0047712303
## Prandtl             -0.001466642  0.0091410713 -0.0013344059
## outcome             -0.304786983 -0.3023878482  0.0002271854
##                           vconst_4     vconst_5     vconst_7      ah_corr
## vconst_corr         -0.0182942192  0.018880130  1.543507e-03  0.003713983
## vconst_2            -0.0006144773 -0.008291704 -2.437907e-02 -0.005182114
## vconst_3             0.0098992508  0.006288772 -1.586561e-03  0.019940865
## vconst_4             1.0000000000  0.020504177  2.193088e-02  0.001804725
## vconst_5             0.0205041766  1.000000000  5.887138e-03 -0.003046706
## vconst_7             0.0219308802  0.005887138  1.000000e+00 -0.016770369
## ah_corr              0.0018047254 -0.003046706 -1.677037e-02  1.000000000
## ah_bolus            -0.0023344776  0.012453140 -2.164373e-02 -0.035497640
## slm_corr            -0.0017310089  0.003633894  1.243988e-03 -0.005119394
## efficiency_factor   -0.0047528309  0.001076585  1.512107e-02  0.009603721
## tidal_mix_max        0.0183200928  0.021353663  7.453114e-05 -0.006831726
## vertical_decay_scale -0.0100038424 -0.016311669  1.528827e-02  0.016502884
## convect_corr        -0.0067617001  0.021379515  7.035600e-03  0.002921182
## bckgrnd_vdc1         0.0204418668  0.009893622 -3.640898e-03  0.012447485
## bckgrnd_vdc_ban     -0.0010800378 -0.019179395 -7.897314e-03 -0.003368375
## bckgrnd_vdc_eq      -0.0092617101 -0.020752192 -6.575539e-03  0.007050740
## bckgrnd_vdc_psim    -0.0171472800 -0.009324311  1.320325e-02  0.002442509
## Prandtl              0.0050528658  0.012265166  8.411513e-03 -0.002380732
## outcome              0.0722970410  0.054390262  4.864631e-02  0.017048998
##                           ah_bolus     slm_corr efficiency_factor
## vconst_corr         -0.0127350037  0.002336425       0.010617309
## vconst_2             0.0041791244 -0.013859711      -0.011071839
## vconst_3             0.0044018146 -0.007695353       0.007100007
## vconst_4            -0.0023344776 -0.001731009      -0.004752831
## vconst_5             0.0124531397  0.003633894       0.001076585
## vconst_7            -0.0216437289  0.001243988       0.015121070
## ah_corr             -0.0354976401 -0.005119394       0.009603721
## ah_bolus             1.0000000000 -0.009403074       0.012259861
## slm_corr            -0.0094030741  1.000000000       0.008759613
## efficiency_factor    0.0122598609  0.008759613       1.000000000
## tidal_mix_max        0.0120047990  0.002574915      -0.017925605
## vertical_decay_scale -0.0039467690  0.002272225       0.018009324
## convect_corr        -0.0193073756  0.002633296       0.011924616
## bckgrnd_vdc1        -0.0106420189 -0.003043485      -0.034025929
## bckgrnd_vdc_ban      0.0048663286  0.006022810       0.003392878
```

```
## bckgrnd_vdc_eq        0.0323978926 -0.008446854       0.009924829
## bckgrnd_vdc_psim      0.0002592278 -0.002300814      -0.005241170
## Prandtl               0.0070551148  0.014280916      -0.004465106
## outcome               0.0038954358  0.048863670      -0.032364442
##                      tidal_mix_max vertical_decay_scale convect_corr
## vconst_corr          -1.420543e-02         -0.008991676 -0.002980494
## vconst_2              1.970607e-02          0.001622933  0.002608462
## vconst_3             -9.428412e-03         -0.024702184 -0.020637321
## vconst_4              1.832009e-02         -0.010003842 -0.006761700
## vconst_5              2.135366e-02         -0.016311669  0.021379515
## vconst_7              7.453114e-05          0.015288275  0.007035600
## ah_corr              -6.831726e-03          0.016502884  0.002921182
## ah_bolus              1.200480e-02         -0.003946769 -0.019307376
## slm_corr              2.574915e-03          0.002272225  0.002633296
## efficiency_factor    -1.792561e-02          0.018009324  0.011924616
## tidal_mix_max         1.000000e+00         -0.004328110  0.002345084
## vertical_decay_scale -4.328110e-03          1.000000000  0.010967371
## convect_corr          2.345084e-03          0.010967371  1.000000000
## bckgrnd_vdc1         -1.686416e-03         -0.008099906 -0.007046943
## bckgrnd_vdc_ban       1.761794e-02         -0.001098075 -0.013650361
## bckgrnd_vdc_eq       -9.566276e-03          0.007049969 -0.016532850
## bckgrnd_vdc_psim     -2.260626e-03         -0.015983717  0.006300329
## Prandtl              -1.191322e-02          0.006292207 -0.005097606
## outcome              -1.507052e-02         -0.045862217 -0.192892806
##                       bckgrnd_vdc1 bckgrnd_vdc_ban bckgrnd_vdc_eq
## vconst_corr          -0.0021325579   -0.0020986795   0.0159727928
## vconst_2             -0.0147164701    0.0043858772   0.0059985634
## vconst_3             -0.0042641402   -0.0052104219  -0.0005588026
## vconst_4              0.0204418668   -0.0010800378  -0.0092617101
## vconst_5              0.0098936216   -0.0191793953  -0.0207521918
## vconst_7             -0.0036408981   -0.0078973136  -0.0065755386
## ah_corr               0.0124474852   -0.0033683750   0.0070507404
## ah_bolus             -0.0106420189    0.0048663286   0.0323978926
## slm_corr             -0.0030434855    0.0060228097  -0.0084468541
## efficiency_factor    -0.0340259293    0.0033928784   0.0099248294
## tidal_mix_max        -0.0016864159    0.0176179416  -0.0095662761
## vertical_decay_scale -0.0080999062   -0.0010980751   0.0070499692
## convect_corr         -0.0070469427   -0.0136503606  -0.0165328497
## bckgrnd_vdc1          1.0000000000    0.0003010617   0.0100984372
## bckgrnd_vdc_ban       0.0003010617    1.0000000000  -0.0019247142
## bckgrnd_vdc_eq        0.0100984372   -0.0019247142   1.0000000000
## bckgrnd_vdc_psim      0.0058160811    0.0163438907   0.0191253312
## Prandtl              -0.0028796392    0.0037739752  -0.0008064784
## outcome               0.1842179984   -0.0283649869   0.0785044287
##                      bckgrnd_vdc_psim       Prandtl       outcome
## vconst_corr             -0.0166310716 -0.0014666423 -0.3047869831
## vconst_2                 0.0042022103  0.0091410713 -0.3023878482
## vconst_3                 0.0047712303 -0.0013344059  0.0002271854
```

```
## vconst_4              -0.0171472800  0.0050528658  0.0722970410
## vconst_5              -0.0093243109  0.0122651659  0.0543902621
## vconst_7               0.0132032517  0.0084115132  0.0486463050
## ah_corr                0.0024425089 -0.0023807322  0.0170489975
## ah_bolus               0.0002592278  0.0070551148  0.0038954358
## slm_corr              -0.0023008138  0.0142809159  0.0488636704
## efficiency_factor     -0.0052411699 -0.0044651060 -0.0323644420
## tidal_mix_max         -0.0022606260 -0.0119132212 -0.0150705214
## vertical_decay_scale  -0.0159837171  0.0062922071 -0.0458622169
## convect_corr           0.0063003294 -0.0050976057 -0.1928928063
## bckgrnd_vdc1           0.0058160811 -0.0028796392  0.1842179984
## bckgrnd_vdc_ban        0.0163438907  0.0037739752 -0.0283649869
## bckgrnd_vdc_eq         0.0191253312 -0.0008064784  0.0785044287
## bckgrnd_vdc_psim       1.0000000000  0.0127673739  0.0576844108
## Prandtl                0.0127673739  1.0000000000 -0.0269417345
## outcome                0.0576844108 -0.0269417345  1.0000000000
```

# 3 Problem-3: Data Partition

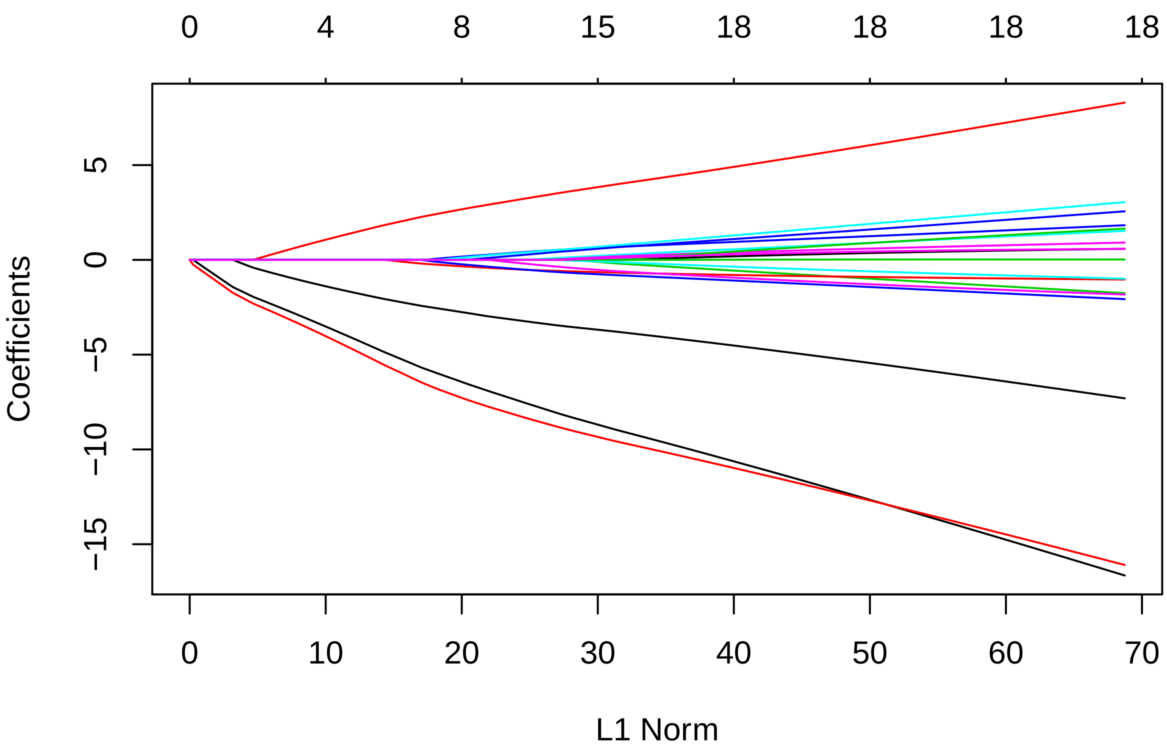
Now I randomly split the data D into the training set D1 and the test set D2 with a ratio of approximately 2:1 on the sample size. I used set.seed() to fix the random seed so that the results are easily reproducible.

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

# 4 Problem-4: Logistic Regression

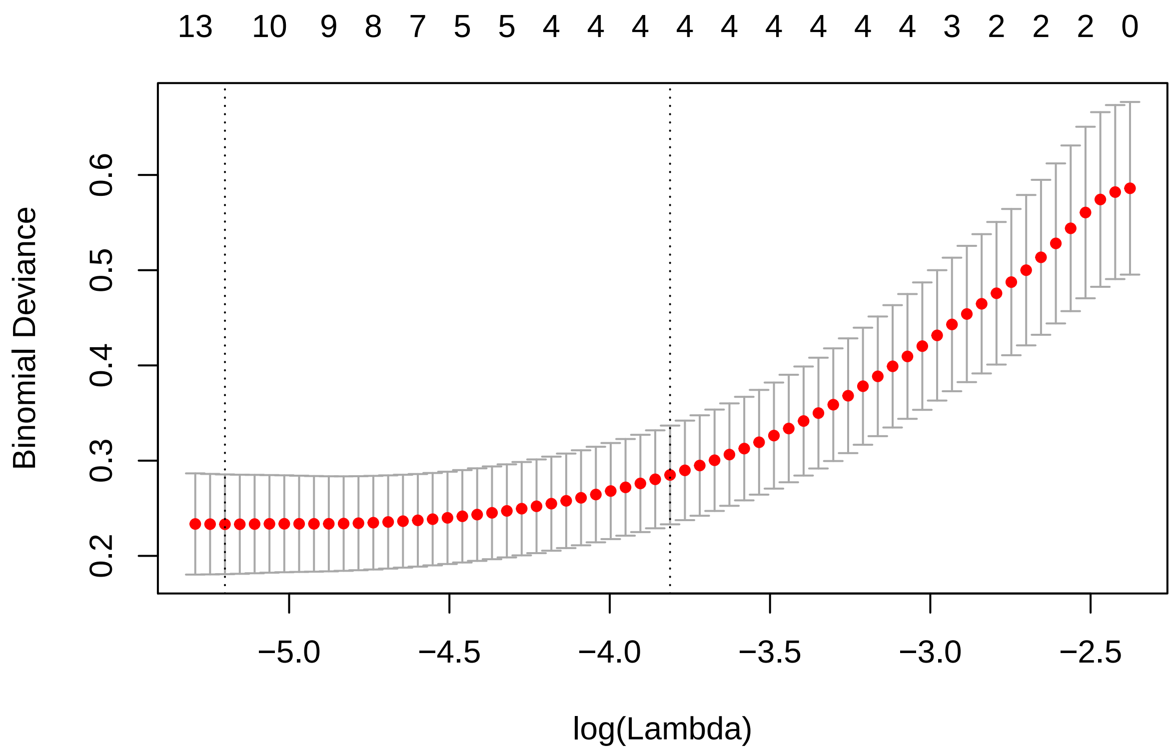
Here, I fit the regularized logistic regression using the training data D1 with Lasso model. In glmnet function, the family argument specify that we want a “binomial” model which tells glmnet() to fit a logistic function to the data.

```r
library(glmnet)
formula0 <- outcome~.
X <- model.matrix (as.formula(formula0), data = train)
y <- train$outcome
fit.lasso <- glmnet(x=X, y=y, family="binomial", alpha=1,
                    lambda.min = 1e-4, nlambda = 300, standardize=T, thresh =
                      1e-07, maxit=3000)
plot(fit.lasso)
```




Using cross validation to determine the optimal tuning parameter.

```
CV <- cv.glmnet(x=X, y=train$outcome, family="binomial", alpha = 1,
                lambda.min = 1e-4, nlambda = 200, standardize = T, thresh = 1e-07,
                maxit=1000)
plot(CV)
```



I select the best tuning parameter ($\lambda$) and use it in the final model.

```
b.lambda <- CV$lambda.1se; b.lambda
```

```
## [1] 0.02210913
```

```
fit.best <- glmnet(x=X, y=train$outcome, family="binomial", alpha = 1,
                   lambda=b.lambda, standardize = T, thresh = 1e-07,
                   maxit=1000)
fit.best$beta
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)           .
## vconst_corr          -3.910770
## vconst_2             -4.466996
## vconst_3              .
## vconst_4              .
## vconst_5              .
## vconst_7              .
## ah_corr               .
## ah_bolus              .
## slm_corr              .
## efficiency_factor     .
## tidal_mix_max         .
## vertical_decay_scale  .
## convect_corr         -1.600313
## bckgrnd_vdc1          1.296857
## bckgrnd_vdc_ban       .
## bckgrnd_vdc_eq        .
## bckgrnd_vdc_psim      .
## Prandtl               .
```

Here, I use the best lambda with fitted model to predict the test data.

```
X.test <- model.matrix (as.formula(formula0), data = test)
pred <- predict(fit.best, newx = X.test, s=b.lambda, type="response")
dim(pred)
```

```
## [1] 173   1
```

The missclassification rate is:

```
(miss.rate_a <- mean(yobs != (pred > 0.5)))
```

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

The Mean squared error is as:

```
MSE.a <- mean((yobs-pred)^2)
MSE.a
```

```
## [1] 0.05003158
```

Plotting ROC curve of the fit.best model.

```
library(cvAUC)
AUC <- ci.cvAUC(predictions = pred, labels = yobs, folds=1:NROW(test), confidence = 0.95)
AUC
```

```
## $cvAUC
## [1] 0.9067511
##
## $se
## [1] 0.06063456
##
## $ci
## [1] 0.7879095 1.0000000
##
## $confidence
## [1] 0.95
```

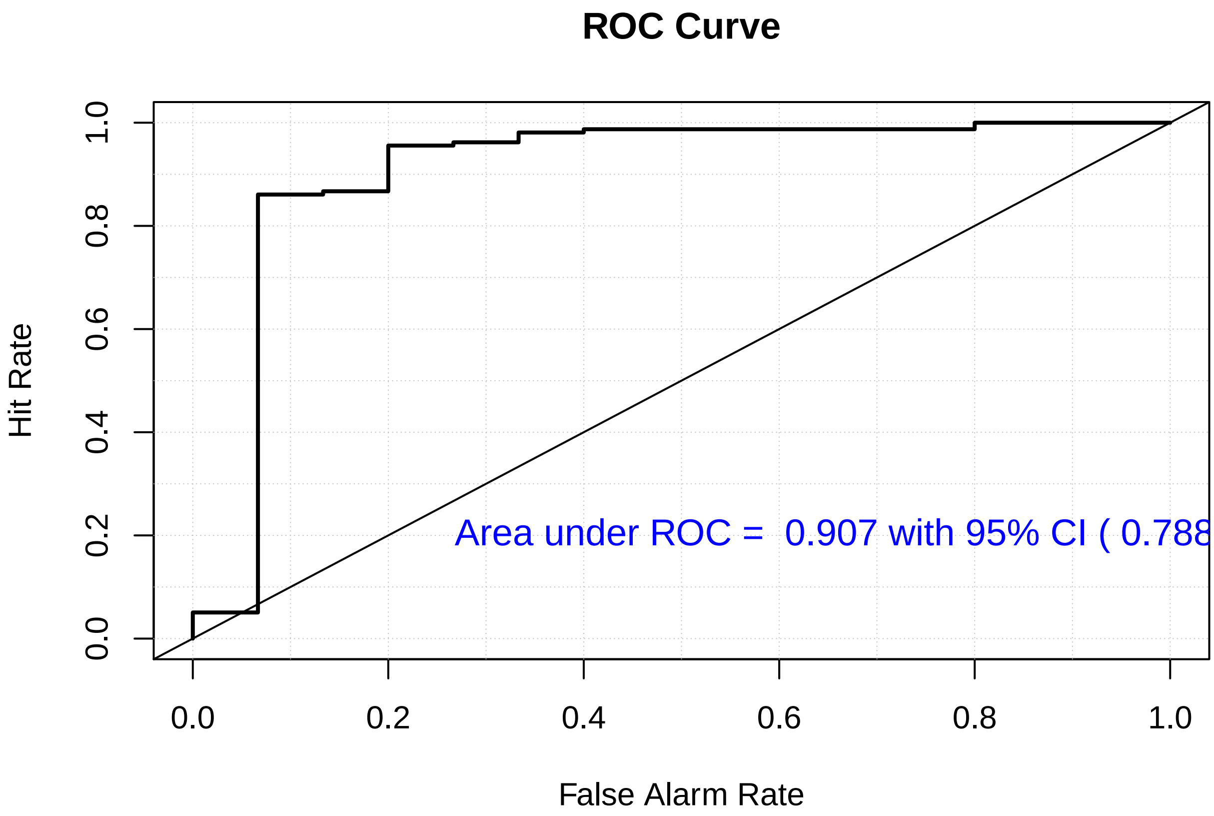
```
(auc.ci <- round(AUC$ci, digits = 3))
```

```
## [1] 0.788 1.000
```

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

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

```
roc.plot(mod.glm, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci[1], ",", auc.ci[2], ").", sep = " "), col="blue", cex =1.2)
```




The accuracy for logistic regression is 90.07%.

```r
library(caret)
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster
```

```r
pred1 <- ifelse(pred>0.5, 1, 0)
confusionMatrix(factor(pred1), factor(test$outcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   3   1
##          1  12 157
##
##                Accuracy : 0.9249
##                  95% CI : (0.8749, 0.9594)
##     No Information Rate : 0.9133
##     P-Value [Acc > NIR] : 0.354468
##
##                   Kappa : 0.2899
```

```
##  Mcnemar's Test P-Value : 0.005546
##
##             Sensitivity : 0.20000
##             Specificity : 0.99367
##          Pos Pred Value : 0.75000
##          Neg Pred Value : 0.92899
##              Prevalence : 0.08671
##          Detection Rate : 0.01734
##    Detection Prevalence : 0.02312
##       Balanced Accuracy : 0.59684
##
##        'Positive' Class : 0
##
```

# 5 Problem-5: Random Forest

I fit the random forest model with training data.

```
library(randomForest)
set.seed(123)
rf_fit <- randomForest(as.factor(outcome) ~.,
                    data=train,
                    importance=TRUE,
                    proximity=TRUE,
                    ntree=200)
rf_yhat <- predict(rf_fit, newdata=test, type="prob")[, 2]
```

Variable importance plot:

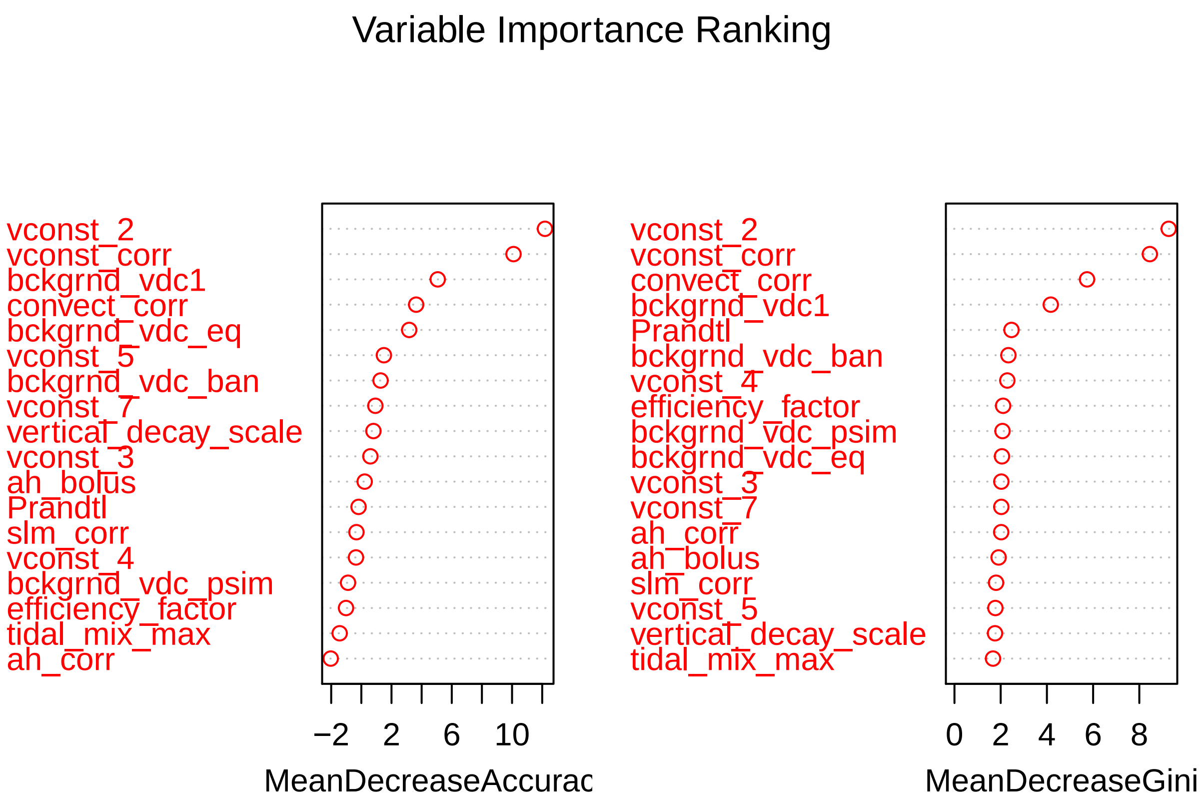
I plot the importance of varibles using Mean Decrease Accuracy and Mean Dcrease Gini.I see that the vconst_2 and vconst_corr are most important two variables from plots.

```
round(importance(rf_fit), 2)
```

```
##                          0     1 MeanDecreaseAccuracy MeanDecreaseGini
## vconst_corr           9.48  7.11                10.10             8.46
## vconst_2             13.29  7.28                12.18             9.27
## vconst_3             -0.95  0.99                 0.60             2.03
## vconst_4              0.34 -0.58                -0.36             2.29
## vconst_5             -1.08  2.05                 1.50             1.77
## vconst_7             -1.10  1.37                 0.93             2.03
## ah_corr              -1.85 -1.34                -2.03             2.02
## ah_bolus              0.23  0.05                 0.22             1.91
## slm_corr             -0.88  0.01                -0.33             1.80
## efficiency_factor    -1.94 -0.26                -1.02             2.10
## tidal_mix_max        -2.25 -0.27                -1.44             1.67
## vertical_decay_scale -0.13  0.83                 0.80             1.75
## convect_corr          3.69  1.59                 3.64             5.74
```

```
## bckgrnd_vdc1          6.30  2.21                5.07              4.17
## bckgrnd_vdc_ban       0.56  0.96                1.27              2.34
## bckgrnd_vdc_eq        2.53  2.12                3.17              2.06
## bckgrnd_vdc_psim      0.86 -1.40               -0.88              2.08
## Prandtl              -0.50 -0.03               -0.18              2.47
```

```
varImpPlot(rf_fit, main="Variable Importance Ranking", col="red")
```




The missclassification rate for random forest is:

```
(miss.rate_b <- mean(yobs != (rf_yhat> 0.5)))
```

```
## [1] 0.0867052
```

Mean squred error:

```
MSE.b <- mean((yobs-rf_yhat)^2)
MSE.b
```

```
## [1] 0.06026633
```

```
#######################  AUC ###############################
library(cvAUC)
rf_AUC <- ci.cvAUC(predictions = rf_yhat, labels =yobs, folds=1:NROW(test), confidence = 0.95)
rf_AUC
```

```
## $cvAUC
## [1] 0.8563291
##
```

```
## $se
## [1] 0.05479113
##
## $ci
## [1] 0.7489405 0.9637178
##
## $confidence
## [1] 0.95
```

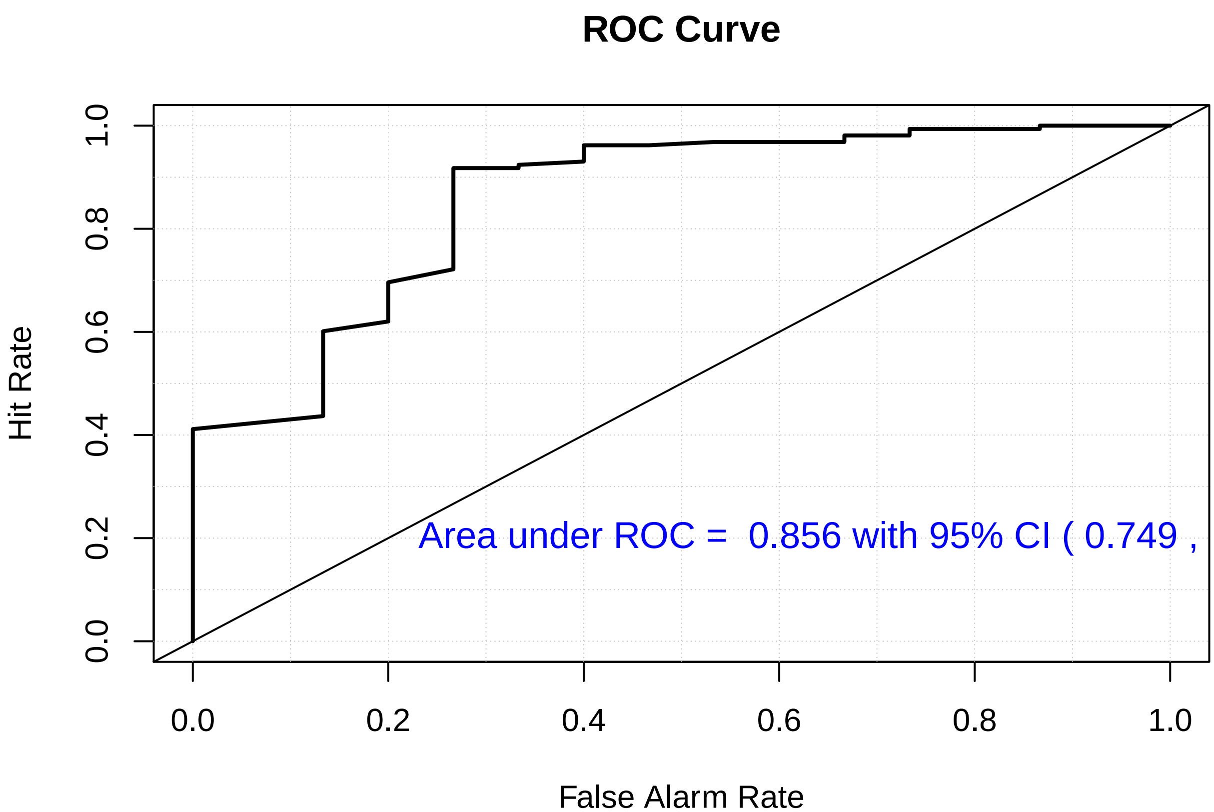
```
(rf_auc.ci <- round(rf_AUC$ci, digits = 3))
```

```
## [1] 0.749 0.964
```

```
library(verification)
mod.rf <- verify(obs = yobs, pred = rf_yhat)
```

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

```
roc.plot(mod.rf, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(rf_AUC$cvAUC, digits = 3), "with 95% CI ("
                         rf_auc.ci[1], ",", rf_auc.ci[2], ").", sep = " "), col="blue", cex =1
```




The accuracy is 85.60% for random forest model.

```
rf_yhat1 <- ifelse(rf_yhat>0.5, 1, 0)
confusionMatrix(factor(rf_yhat1), factor(test$outcome))
```

```
## Warning in confusionMatrix.default(factor(rf_yhat1), factor(test$outcome)):
```

```
## Levels are not in the same order for reference and data. Refactoring data
## to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   0   0
##          1  15 158
##
##                Accuracy : 0.9133
##                  95% CI : (0.861, 0.9507)
##     No Information Rate : 0.9133
##     P-Value [Acc > NIR] : 0.5681598
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 0.0003006
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.91329
##              Prevalence : 0.08671
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 0
##
```

# 6 Problem-3: Artificial Neural Network

## 6.1 ANN1: Single hidden layer

I first prepare the data for artificial neural network.

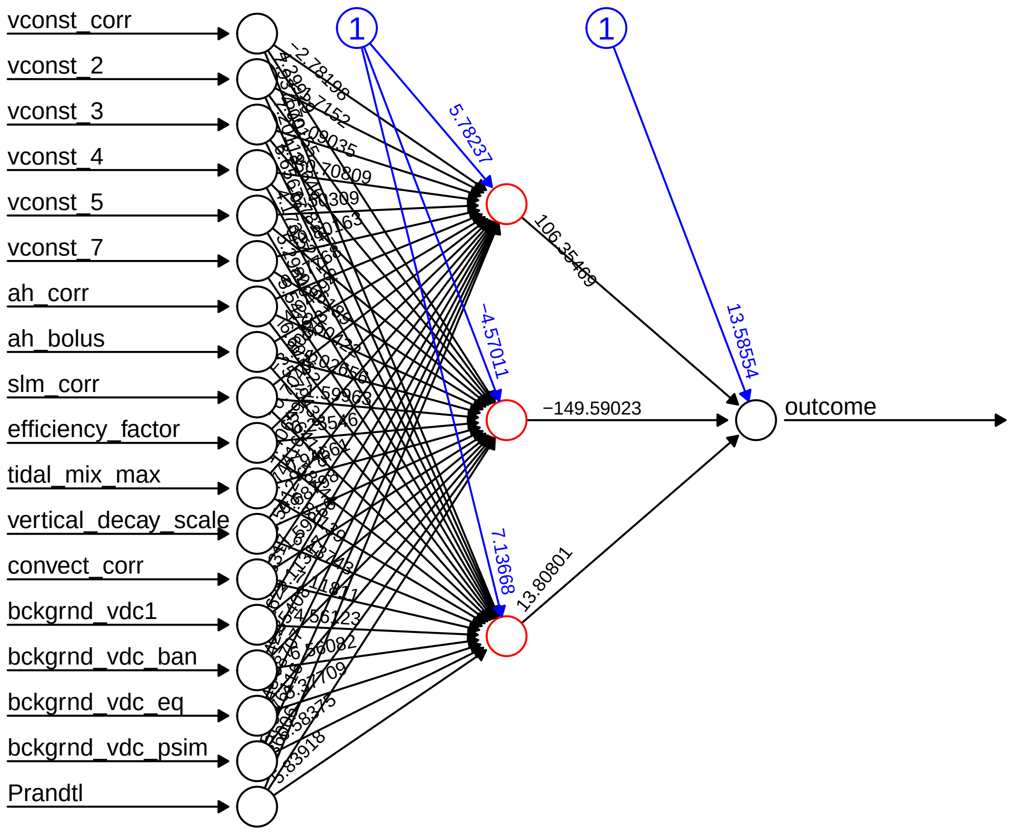
```
# DATA PREPARATION - NEED TO DEAL WITH CATEGORICAL PREDICTORS
X.dat <- as.data.frame(model.matrix(outcome~.-1, data=data))
dat1 <- data.frame(cbind(X.dat, outcome=data$outcome))
```

```
set.seed(123)
n <- nrow(dat1)
split_data <- sample(x=1:2, size = n, replace=TRUE, prob=c(0.67, 0.33))
train1 <- dat1[split_data == 1, ]
test1 <- dat1[split_data == 2, ]
yobs <- test1$outcome
test1 <- test1[ , -19]
```

I use 3 neurons for single layer.

```
library(neuralnet);
options(digits=3)
net1 <- neuralnet(outcome~vconst_corr+vconst_2+vconst_3+
                  vconst_4+vconst_5+vconst_7+ah_corr+ah_bolus+slm_corr+efficiency_factor+
                  tidal_mix_max+vertical_decay_scale+convect_corr+bckgrnd_vdc1+
                  bckgrnd_vdc_ban+bckgrnd_vdc_eq+bckgrnd_vdc_psim+Prandtl,
                  data=train1,
                  hidden=3, #1 hidden layer, 3 units
                  act.fct='logistic', err.fct="ce", linear.output=F, likelihood=TRUE)
```

```
# PLOT THE MODEL
plot(net1, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
     col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)
```




```
# PREDICTION
ypred_net1 <- compute(net1, covariate=test1)$net.result
```

The mean squared error is as:

```
MSE.c <- mean((yobs-ypred_net1)^2)
MSE.c
```

```
## [1] 0.06982044344
```

```
ypred_net11 <- ifelse(ypred_net1>0.5,1,0)
(miss.rate_c <- mean(yobs != ypred_net11))
```

```
## [1] 0.06936416185
```

```
#######################  AUC ###############################
library(cvAUC)
net_AUC <- ci.cvAUC(predictions = ypred_net1, labels =yobs, folds=1:NROW(test1), confidence = 0
net_AUC
```

```
## $cvAUC
## [1] 0.9506329114
##
## $se
## [1] 0.02299775766
##
## $ci
## [1] 0.9055581346 0.9957076881
##
## $confidence
## [1] 0.95
```

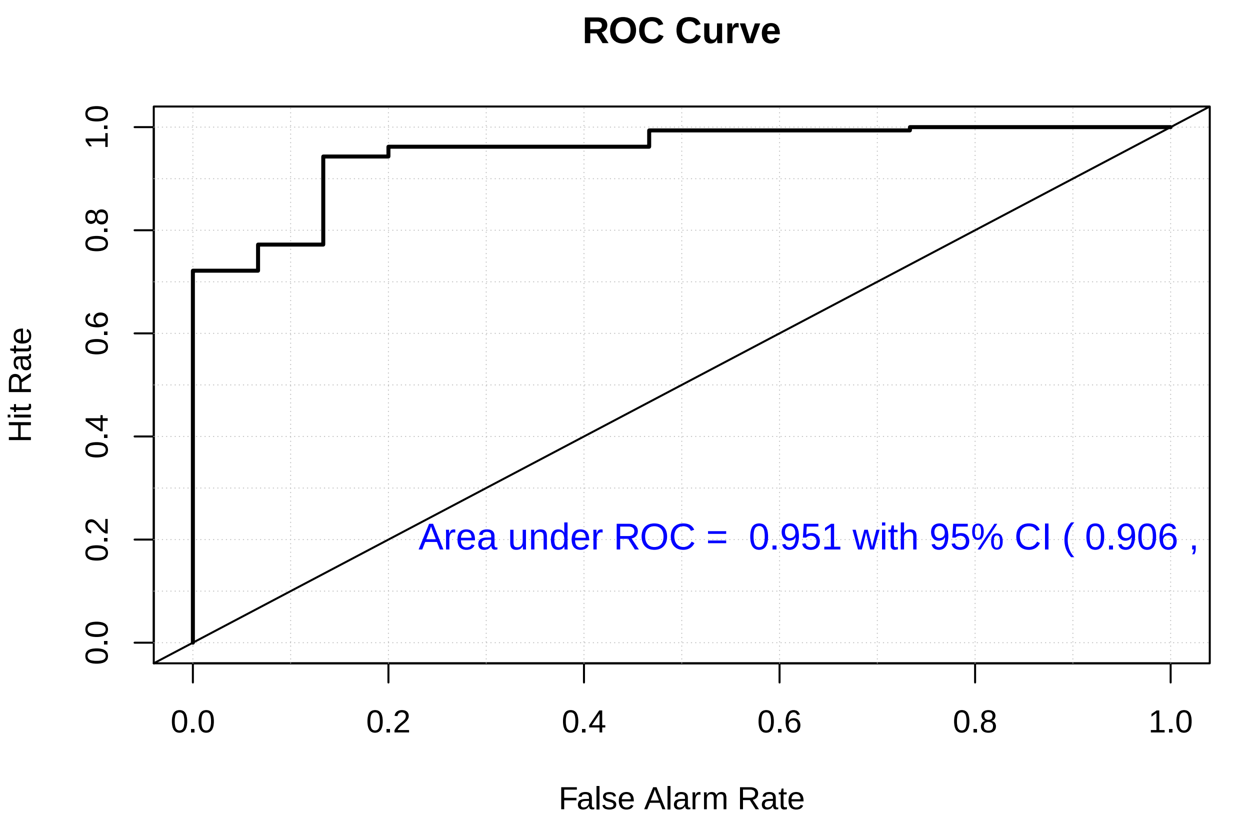
```
(net_auc.ci <- round(net_AUC$ci, digits = 3))
```

```
## [1] 0.906 0.996
```

```
library(verification)
mod.net1 <- verify(obs = yobs, pred = ypred_net1)
```

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

```
roc.plot(mod.net1, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(net_AUC$cvAUC, digits = 3), "with 95% CI (
                         net_auc.ci[1], ",", net_auc.ci[2], ").", sep = " "), col="blue", cex =
```




```r
confusionMatrix(factor(ypred_net11), factor(test$outcome))
```

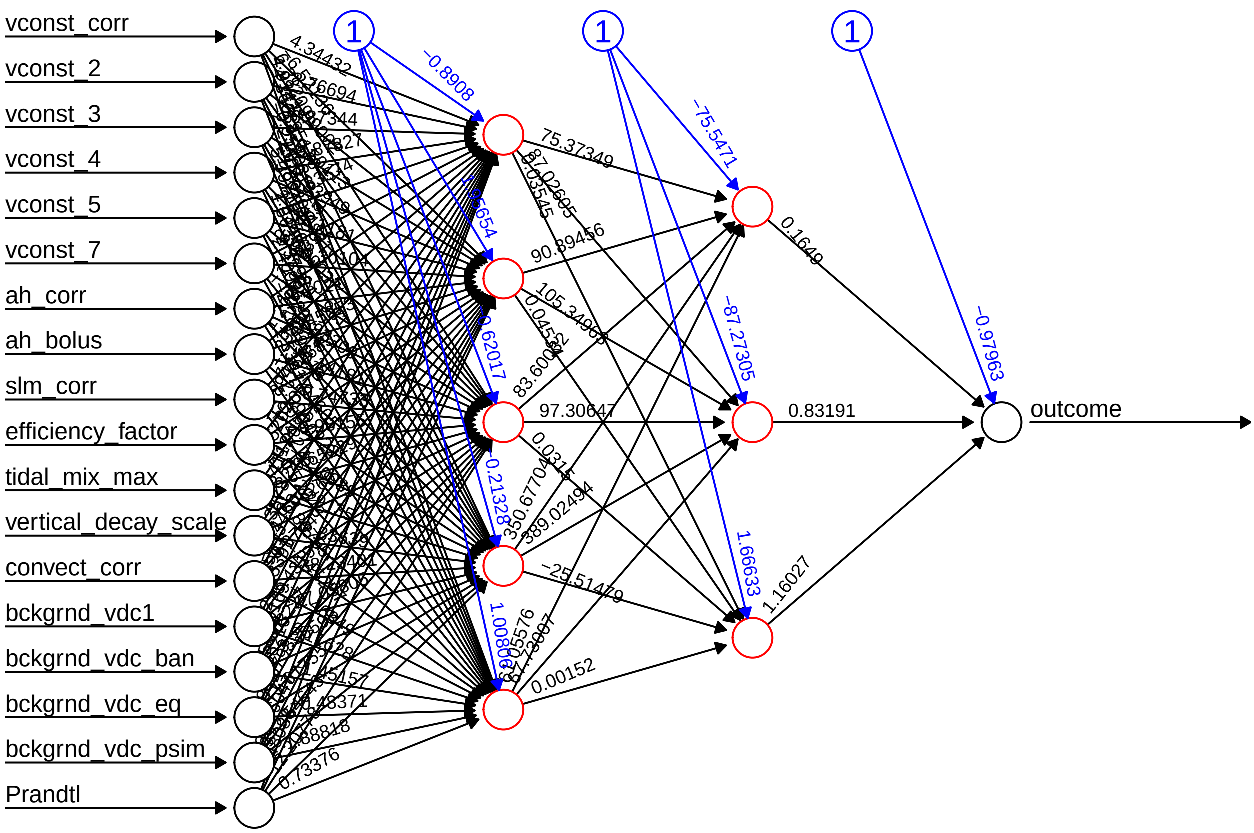
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   9   6
##          1   6 152
##
##                Accuracy : 0.9306358
##                  95% CI : (0.88197, 0.9636472)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.2563017
##
##                   Kappa : 0.5620253
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.60000000
##             Specificity : 0.96202532
##          Pos Pred Value : 0.60000000
##          Neg Pred Value : 0.96202532
##              Prevalence : 0.08670520
##          Detection Rate : 0.05202312
##    Detection Prevalence : 0.08670520
##       Balanced Accuracy : 0.78101266
```

```
##
##        'Positive' Class : 0
##
```

### 6.2 ANN2: Two hidden layers

I used two hidden layers( 5 neurons in first layer, and 3 neurons in second layer)

```
nn <- neuralnet(outcome~vconst_corr+vconst_2+vconst_3+
                     vconst_4+vconst_5+vconst_7+ah_corr+ah_bolus+slm_corr+efficiency_factor+
                     tidal_mix_max+vertical_decay_scale+convect_corr+bckgrnd_vdc1+
                     bckgrnd_vdc_ban+bckgrnd_vdc_eq+bckgrnd_vdc_psim+Prandtl, data=train1,
                   hidden=c(5,3),linear.output=T)
plot(nn, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
     col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)
```




```
# PREDICTION
ypred_nn <- compute(nn, covariate=test1)$net.result
```

```
ypred_nn1 <- ifelse(ypred_nn>0.5,1,0)
(miss.rate_d <- mean(yobs != ypred_nn1))
```

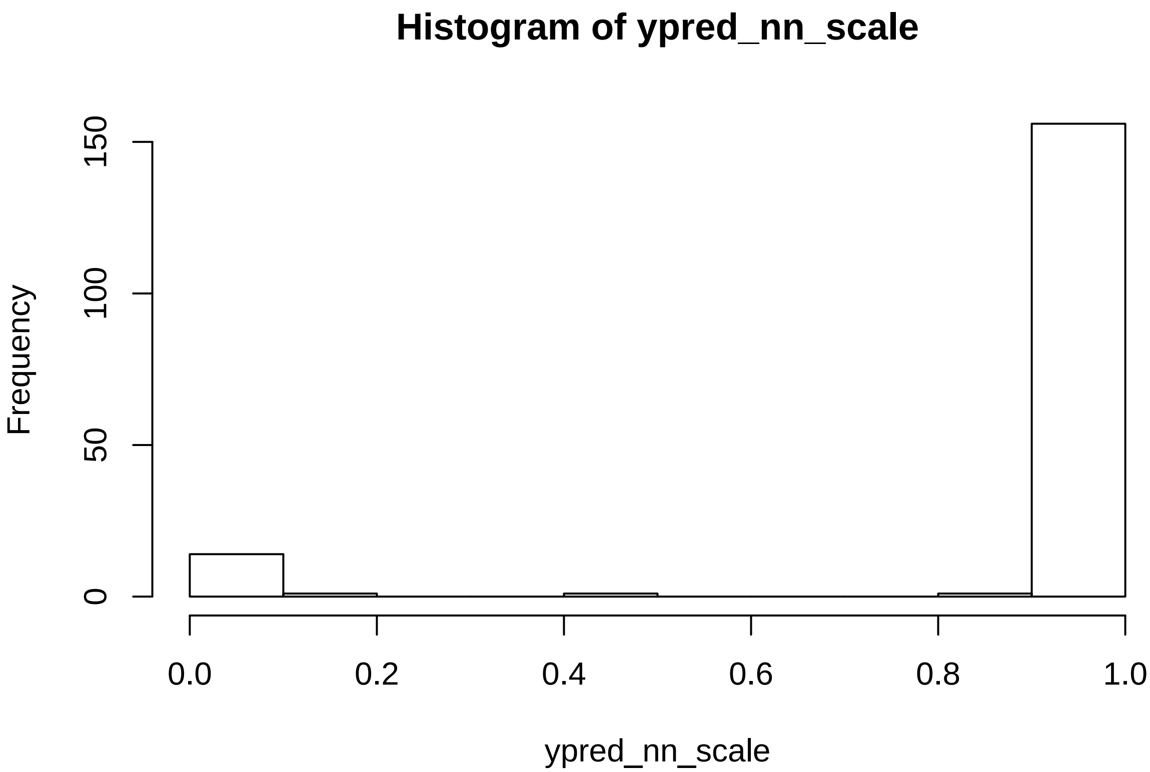
```
## [1] 0.08670520231
```

```
MSE.d <- mean((yobs-ypred_nn)^2)
MSE.d
```

```
## [1] 0.08478119453
```

Here I scaled the predicted probabilities between 0 and 1.

```
ypred_nn_scale <- scale(ypred_nn, center = min(ypred_nn), scale=max(ypred_nn)-min(ypred_nn))
hist(ypred_nn_scale)
```




```
#######################  AUC ###############################
library(cvAUC)
nn_AUC <- ci.cvAUC(predictions = ypred_nn_scale, labels =yobs, folds=1:NROW(test1), confidence
nn_AUC
```

```
## $cvAUC
## [1] 0.911814346
##
## $se
## [1] 0.04784049402
##
## $ci
## [1] 0.8180487007 1.0000000000
##
## $confidence
## [1] 0.95
```

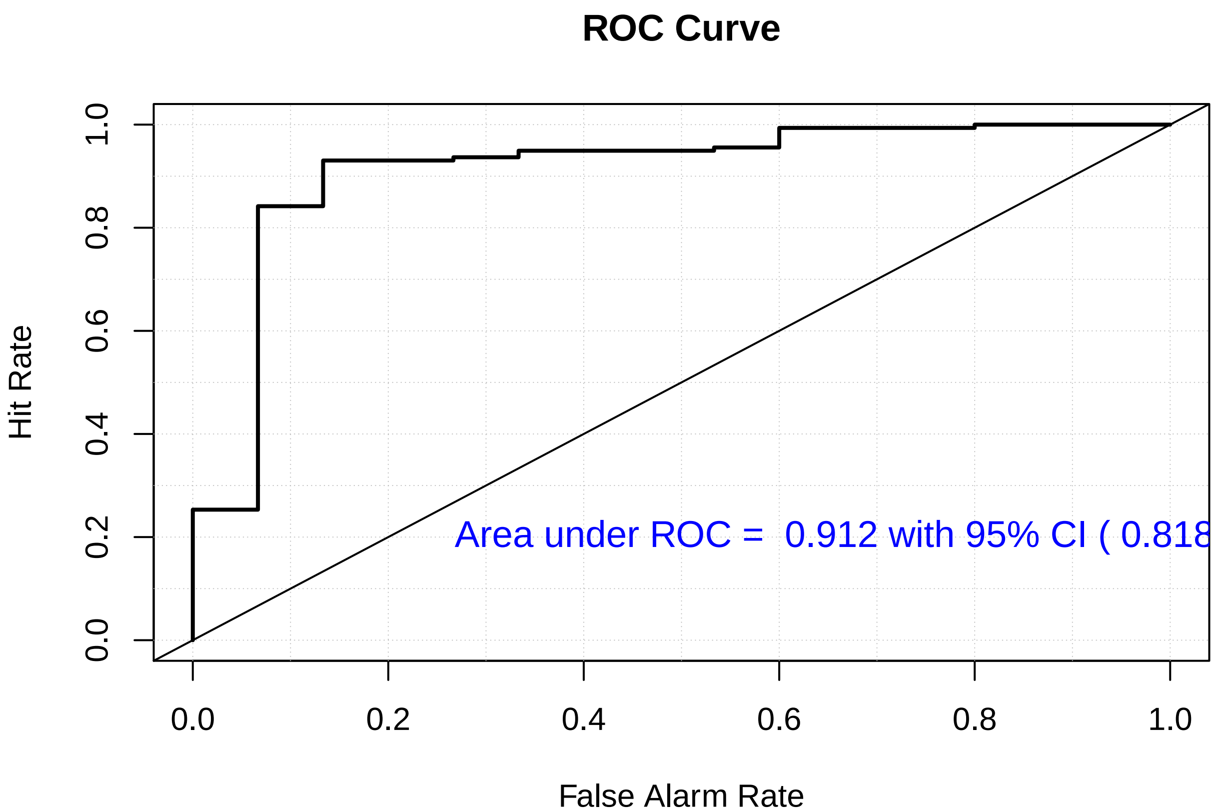
```r
(nn_auc.ci <- round(nn_AUC$ci, digits = 3))
```

```
## [1] 0.818 1.000
```

```r
library(verification)
mod.nn <- verify(obs = yobs, pred = ypred_nn_scale)
```

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

```r
roc.plot(mod.nn, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(nn_AUC$cvAUC, digits = 3), "with 95% CI ("
                         nn_auc.ci[1], ",", nn_auc.ci[2], ").", sep = " "), col="blue", cex =1
```




```r
confusionMatrix(factor(ypred_nn1), factor(test$outcome))
```

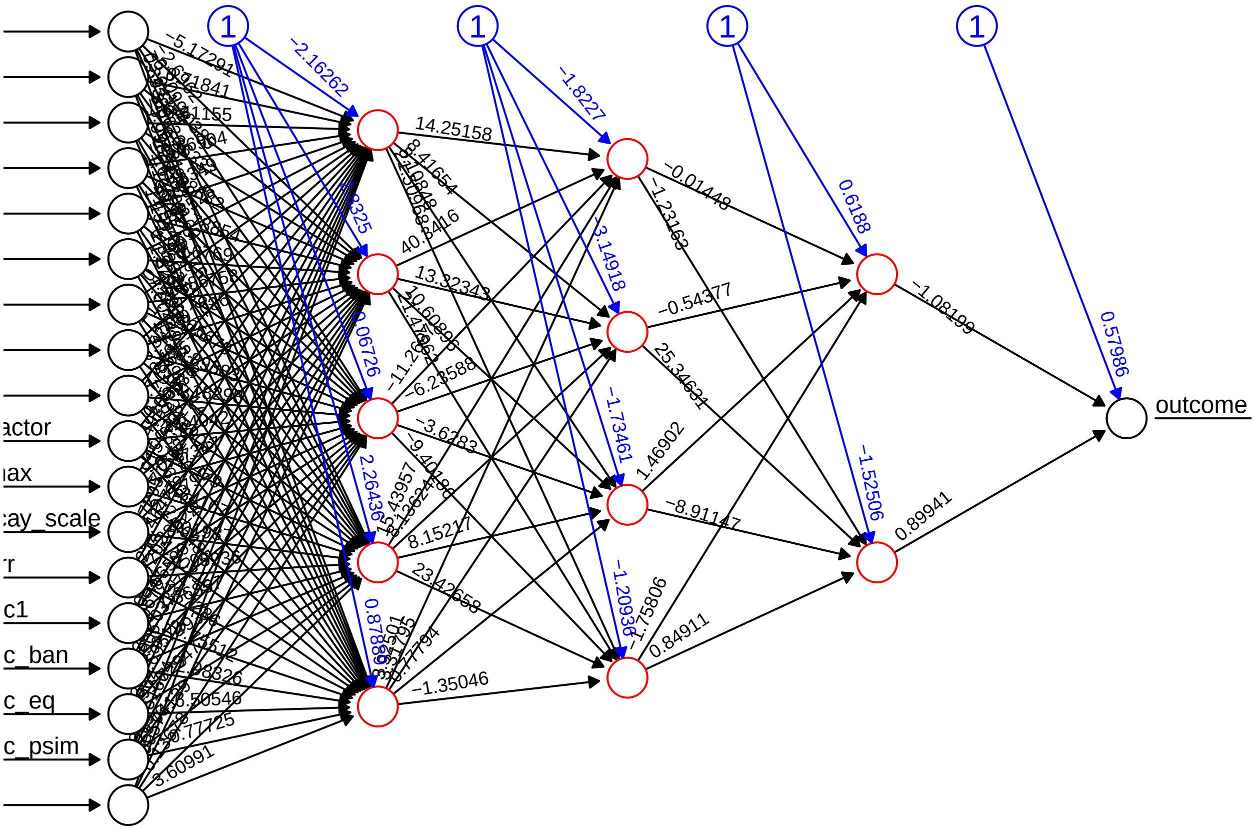
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   8   8
##          1   7 150
##
##                Accuracy : 0.9132948
##                  95% CI : (0.8610262, 0.9506581)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.5681598
```

```
##
##                   Kappa : 0.4685644
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.53333333
##             Specificity : 0.94936709
##          Pos Pred Value : 0.50000000
##          Neg Pred Value : 0.95541401
##              Prevalence : 0.08670520
##          Detection Rate : 0.04624277
##    Detection Prevalence : 0.09248555
##       Balanced Accuracy : 0.74135021
##
##        'Positive' Class : 0
##
```

### 6.3 ANN3: Three hidden layers

Here I used three hidden layers (5 neurons in first layer, 4 neurons in second layer, 2 neurons in third layer).

```
nn2 <- neuralnet(outcome~vconst_corr+vconst_2+vconst_3+
                  vconst_4+vconst_5+vconst_7+ah_corr+ah_bolus+slm_corr+efficiency_factor+
                  tidal_mix_max+vertical_decay_scale+convect_corr+bckgrnd_vdc1+
                  bckgrnd_vdc_ban+bckgrnd_vdc_eq+bckgrnd_vdc_psim+Prandtl, data=train1,
                hidden=c(5,4,2),linear.output=T)
plot(nn2, rep="best", show.weights=T, dimension=6.5, information=F, radius=.15,
     col.hidden="red", col.hidden.synapse="black", lwd=1, fontsize=9)
```




```r
# PREDICTION
ypred_nn2 <- compute(nn2, covariate=test1)$net.result
```

```r
ypred_nn22 <- ifelse(ypred_nn2>0.5,1,0)
(miss.rate_e <- mean(yobs != ypred_nn22))
```

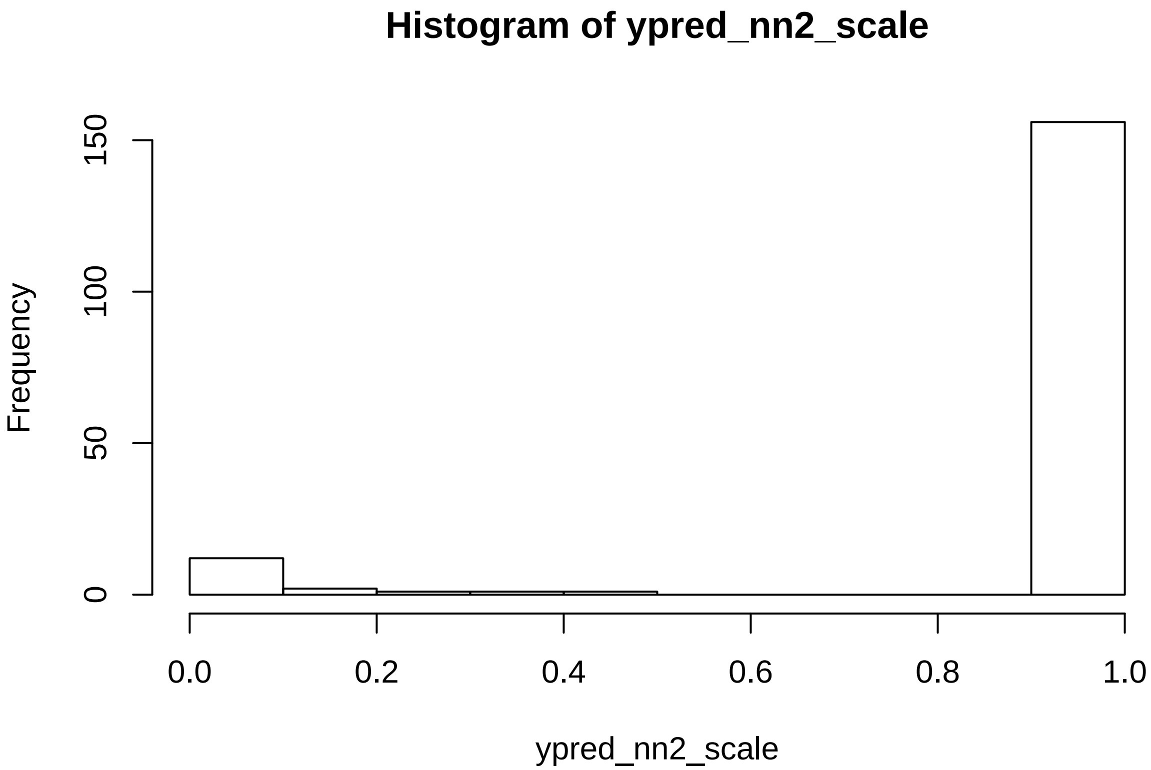
```
## [1] 0.05780346821
```

```r
MSE.e <- mean((yobs-ypred_nn2)^2)
MSE.e
```

```
## [1] 0.06238087088
```

Here I scaled the predicted probabilities between 0 and 1.

```r
ypred_nn2_scale <- scale(ypred_nn2, center = min(ypred_nn2), scale=max(ypred_nn2)-min(ypred_nn
hist(ypred_nn2_scale)
```




```
#######################  AUC ###############################
library(cvAUC)
nn2_AUC <- ci.cvAUC(predictions = ypred_nn2_scale, labels = yobs, folds=1:NROW(test1), confider
nn2_AUC
```

```
## $cvAUC
## [1] 0.7666666667
##
## $se
## [1] 0.09526080015
##
## $ci
## [1] 0.5799589292 0.9533744041
##
## $confidence
## [1] 0.95
```

```
(nn2_auc.ci <- round(nn2_AUC$ci, digits = 3))
```

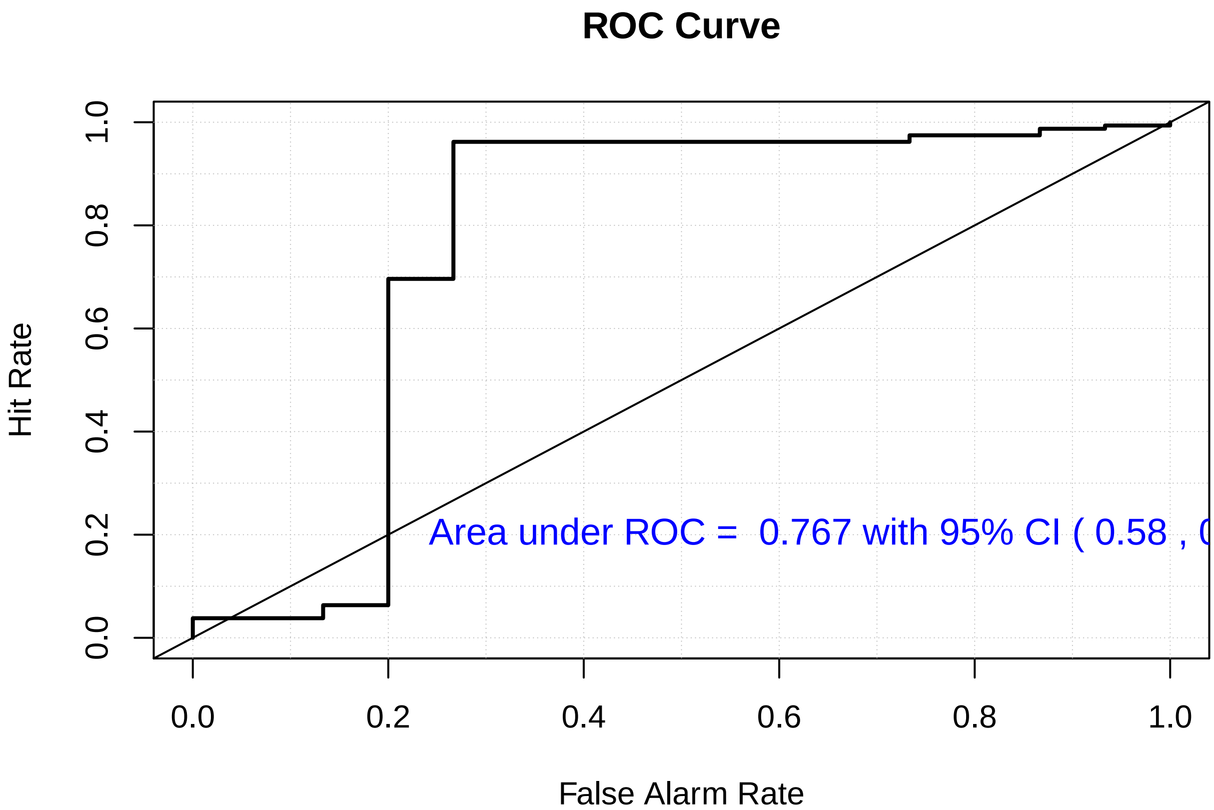
```
## [1] 0.580 0.953
```

```
library(verification)
mod.nn2 <- verify(obs = yobs, pred = ypred_nn2_scale)
```

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

```
roc.plot(mod.nn2, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(nn2_AUC$cvAUC, digits = 3), "with 95% CI ("
```

```
                              nn2_auc.ci[1], ",", nn2_auc.ci[2], ").", sep = " "), col="blue", cex =
```




```
confusionMatrix(factor(ypred_nn22), factor(test$outcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0  11   6
##          1   4 152
##
##                Accuracy : 0.9421965
##                  95% CI : (0.8962682, 0.9719361)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.1076656
##
##                   Kappa : 0.6557899
##  Mcnemar's Test P-Value : 0.7518296
##
##             Sensitivity : 0.73333333
##             Specificity : 0.96202532
##          Pos Pred Value : 0.64705882
##          Neg Pred Value : 0.97435897
##              Prevalence : 0.08670520
##          Detection Rate : 0.06358382
```

```
##    Detection Prevalence : 0.09826590
##       Balanced Accuracy : 0.84767932
##
##        'Positive' Class : 0
##
```

# 7 Support Vector Machine

I used five types of SVM linear techniques such as SVM linear, svm linear with grid, svm radial, svm radial grid, and Radial Basis kernel ”Gaussian”.

## 7.1 SVM-Liner

```
#############################  SVM Linear #############################
## SUPPORT VECTOR MACHINE MODEL
library(caret)
set.seed(1492)
# Setup for cross validation
ctrl <- trainControl(method="repeatedcv",   # 10fold cross validation
                     number =10)

set.seed(3233)

svm_Linear <- train(factor(outcome)~., data = train, method = "svmLinear",
                    trControl=ctrl,
                    tuneLength = 5) #5 arbitrary values of cost function C

svm_Linear
```

```
## Support Vector Machines with Linear Kernel
##
## 367 samples
##  18 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 331, 330, 330, 330, 330, 330, ...
## Resampling results:
##
##   Accuracy      Kappa
##   0.9425675676  0.553990734
##
## Tuning parameter 'C' was held constant at a value of 1
```

The confusion matrix produces the sensitivity, specificity, accuracy, and confidence interval etc.

```
svm_pred <- predict(svm_Linear, newdata = test);svm_pred
```

```
##   [1] 1 1 1 1 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
##  [36] 1 1 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 1
##  [71] 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 1 1 1 1 1
## [106] 1 1 1 0 1 1 1 1 0 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
## [141] 1 1 0 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 1 1 1 1
## Levels: 0 1
```

```
confusionMatrix(svm_pred, factor(test$outcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   9   5
##          1   6 153
##
##                Accuracy : 0.9364162
##                  95% CI : (0.8890805, 0.9678347)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.1728098
##
##                   Kappa : 0.5860344
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.60000000
##             Specificity : 0.96835443
##          Pos Pred Value : 0.64285714
##          Neg Pred Value : 0.96226415
##              Prevalence : 0.08670520
##          Detection Rate : 0.05202312
##    Detection Prevalence : 0.08092486
##       Balanced Accuracy : 0.78417722
##
##        'Positive' Class : 0
##
```

```
(miss.rate_f <- mean(yobs != svm_pred))
```

```
## [1] 0.06358381503
```

```
svm_pred <- as.numeric(as.character(svm_pred))
MSE.f <- mean((yobs-svm_pred)^2)
MSE.f
```

```
## [1] 0.06358381503
```

Plotting ROC curve of the fit.best model.

```r
library(cvAUC)
svm_lin_AUC <- ci.cvAUC(predictions = svm_pred, labels = yobs, folds=1:NROW(test), confidence =
svm_lin_AUC
```

```
## $cvAUC
## [1] 0.7841772152
##
## $se
## [1] 0.1344847533
##
## $ci
## [1] 0.5205919423 1.0000000000
##
## $confidence
## [1] 0.95
```

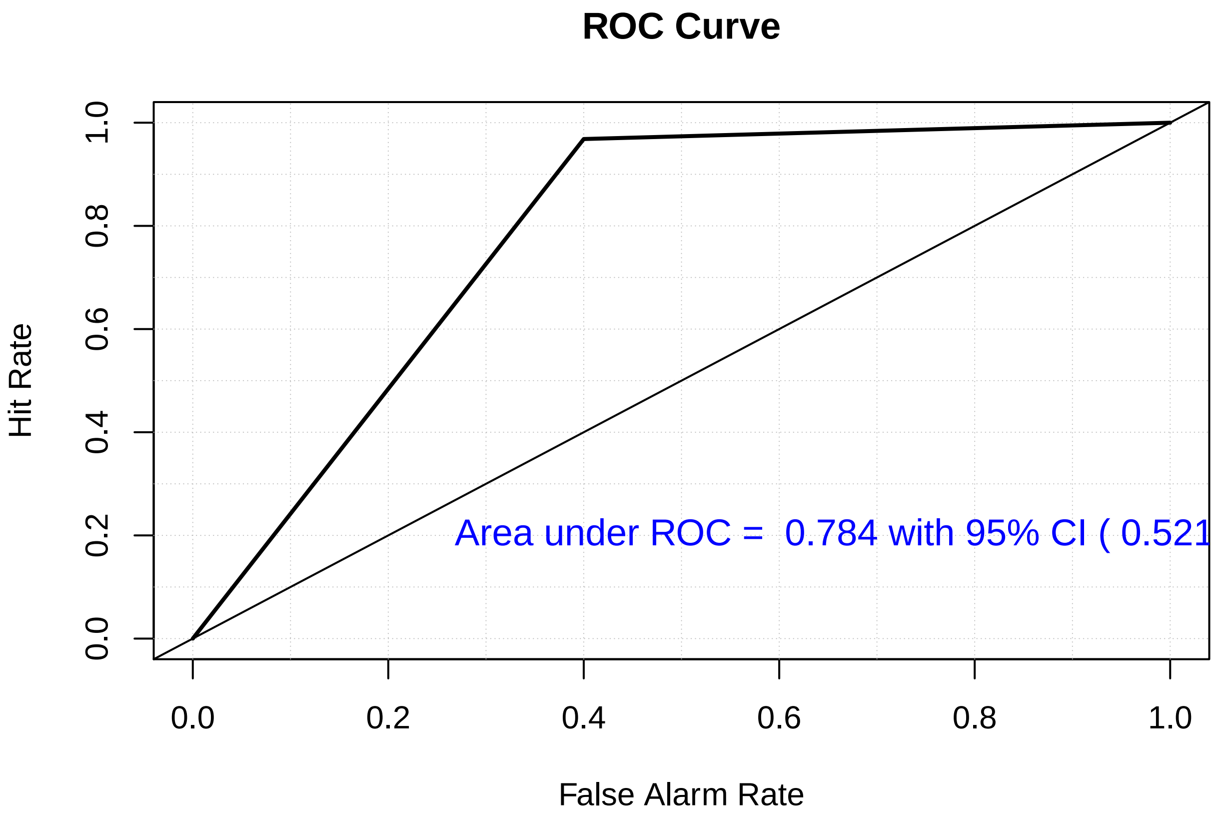
```r
(svm_lin_auc.ci <- round(svm_lin_AUC$ci, digits = 3))
```

```
## [1] 0.521 1.000
```

```r
library(verification)
mod.svm_lin <- verify(obs = yobs, pred = svm_pred)
```

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

```r
roc.plot(mod.svm_lin, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_lin_AUC$cvAUC, digits = 3), "with 95% C
                         svm_lin_auc.ci[1], ",", svm_lin_auc.ci[2], ").", sep = " "), col="blue
```




### 7.2 SVM-Linear-grid

```
############### C value in Linearclassifier ##############
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))
set.seed(3233)
svm_Linear_Grid <- train(factor(outcome) ~., data = train, method = "svmLinear",
                           trControl=ctrl,
                           tuneGrid = grid,
                           tuneLength = 5)

test_pred_grid <- predict(svm_Linear_Grid, newdata = test)
confusionMatrix(test_pred_grid, factor(test$outcome))
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    0   1
##          0    7   2
##          1    8 156
##
##                Accuracy : 0.9421965
##                  95% CI : (0.8962682, 0.9719361)
##     No Information Rate : 0.9132948
```

```
##     P-Value [Acc > NIR] : 0.1076656
##
##                   Kappa : 0.5543534
##  Mcnemar's Test P-Value : 0.1138463
##
##             Sensitivity : 0.46666667
##             Specificity : 0.98734177
##          Pos Pred Value : 0.77777778
##          Neg Pred Value : 0.95121951
##              Prevalence : 0.08670520
##          Detection Rate : 0.04046243
##    Detection Prevalence : 0.05202312
##       Balanced Accuracy : 0.72700422
##
##        'Positive' Class : 0
##
```

```
(miss.rate_g <- mean(yobs != test_pred_grid))
```

```
## [1] 0.05780346821
```

```
test_pred_grid <- as.numeric(as.character(test_pred_grid))
MSE.g <- mean((yobs-test_pred_grid)^2)
MSE.g
```

```
## [1] 0.05780346821
```

Plotting ROC curve of the fit.best model.

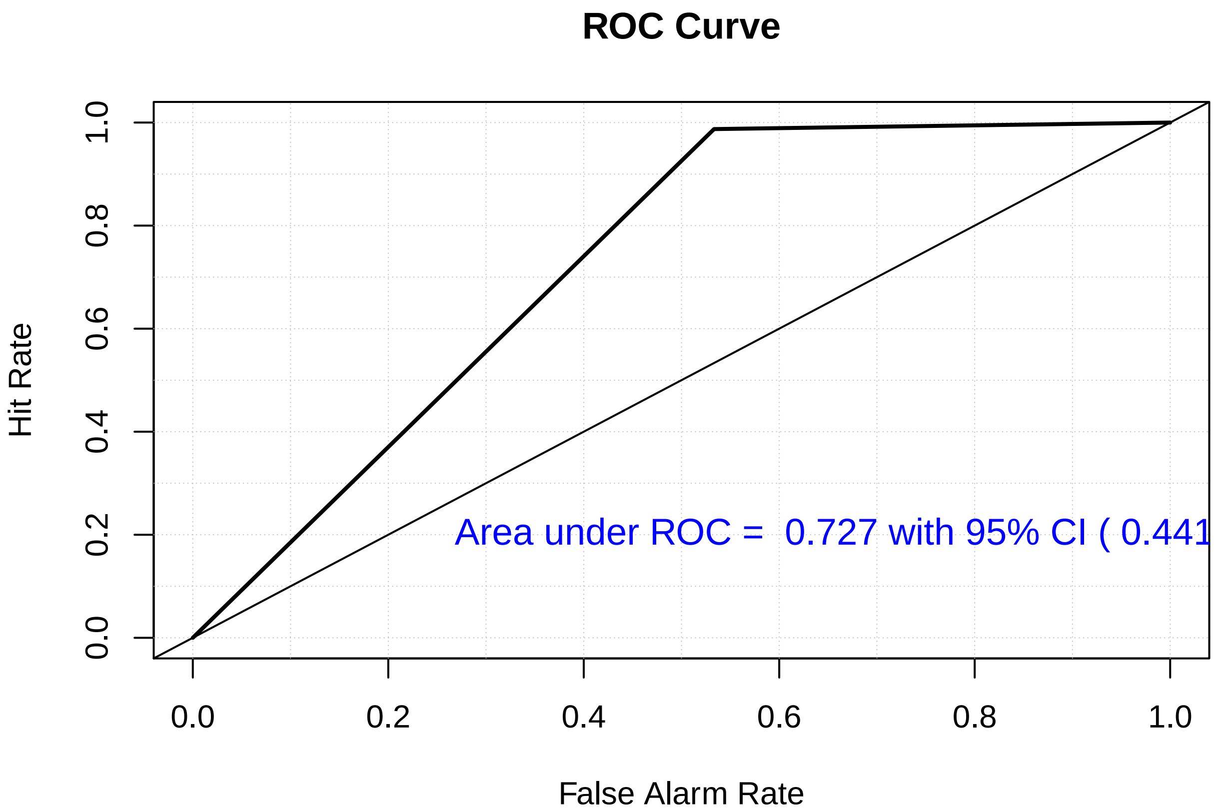
```
library(cvAUC)
svm_lin_g_AUC <- ci.cvAUC(predictions = test_pred_grid, labels = yobs, folds=1:NROW(test), con
(svm_lin_g_auc.ci <- round(svm_lin_g_AUC$ci, digits = 3))
```

```
## [1] 0.441 1.000
```

```
library(verification)
mod.svm_lin_g <- verify(obs = yobs, pred = test_pred_grid)
```

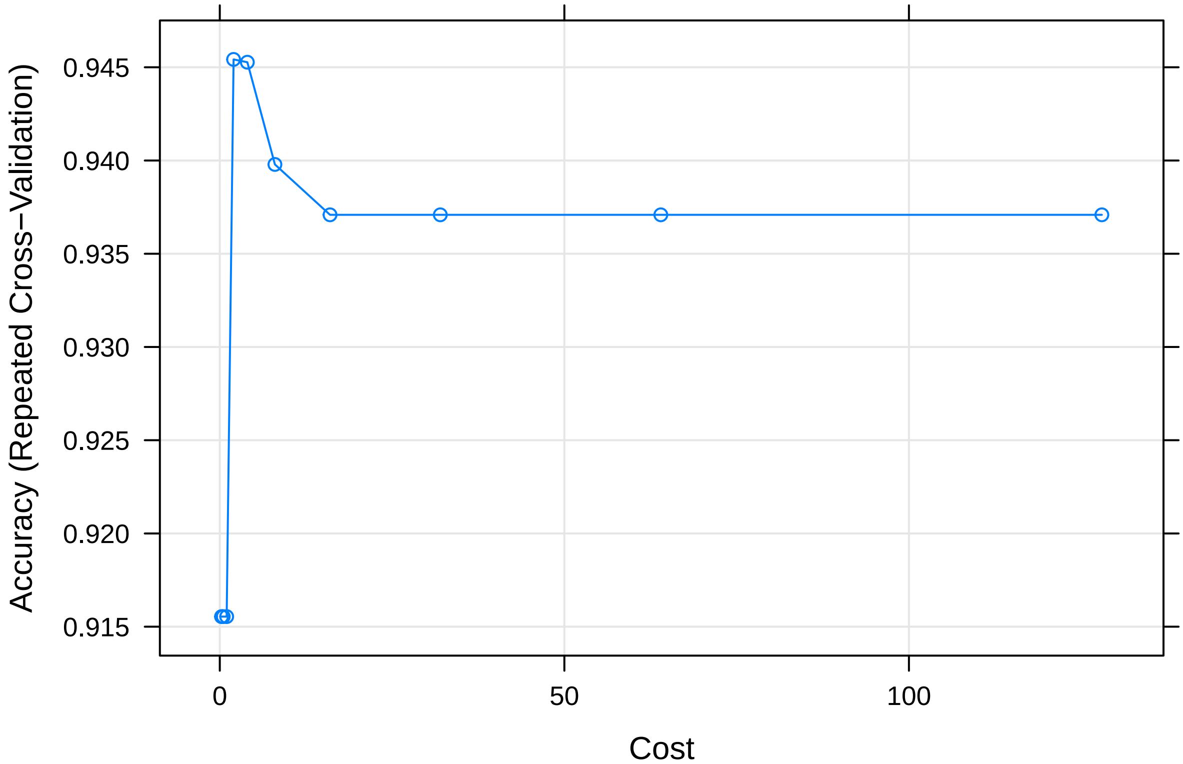
```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```

```
roc.plot(mod.svm_lin_g, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_lin_g_AUC$cvAUC, digits = 3), "with 95
                         svm_lin_g_auc.ci[1], ",", svm_lin_g_auc.ci[2], ").", sep = " "), col=
```




### 7.3 SVM-Radial

```
###############SVM Classifier using Non-Linear Kernel #############
set.seed(3233)
svm_Radial <- train(factor(outcome) ~., data = train, method = "svmRadial",#radial kernel
                      trControl=ctrl,
                      tuneLength = 10)
plot(svm_Radial)
```




```
test_pred_Radial <- predict(svm_Radial, newdata = test)
confusionMatrix(test_pred_Radial, factor(test$outcome))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   4   1
##          1  11 157
##
##                Accuracy : 0.9306358
##                  95% CI : (0.88197, 0.9636472)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.256301653
##
##                   Kappa : 0.3728097
##  Mcnemar's Test P-Value : 0.009374768
##
##             Sensitivity : 0.26666667
##             Specificity : 0.99367089
##          Pos Pred Value : 0.80000000
##          Neg Pred Value : 0.93452381
##              Prevalence : 0.08670520
##          Detection Rate : 0.02312139
##    Detection Prevalence : 0.02890173
##       Balanced Accuracy : 0.63016878
```

```
##
##        'Positive' Class : 0
##
```

```
(miss.rate_h <- mean(yobs != test_pred_Radial))
```

```
## [1] 0.06936416185
```

```
test_pred_Radial <- as.numeric(as.character(test_pred_Radial))
MSE.h <- mean((yobs-test_pred_Radial)^2)
MSE.h
```

```
## [1] 0.06936416185
```

Plotting ROC curve of the best model.

```
library(cvAUC)
svm_rad_AUC <- ci.cvAUC(predictions = test_pred_Radial, labels = yobs, folds=1:NROW(test), con
svm_rad_AUC
```

```
## $cvAUC
## [1] 0.6301687764
##
## $se
## [1] 0.1503673744
##
## $ci
## [1] 0.3354541382 0.9248834146
##
## $confidence
## [1] 0.95
```

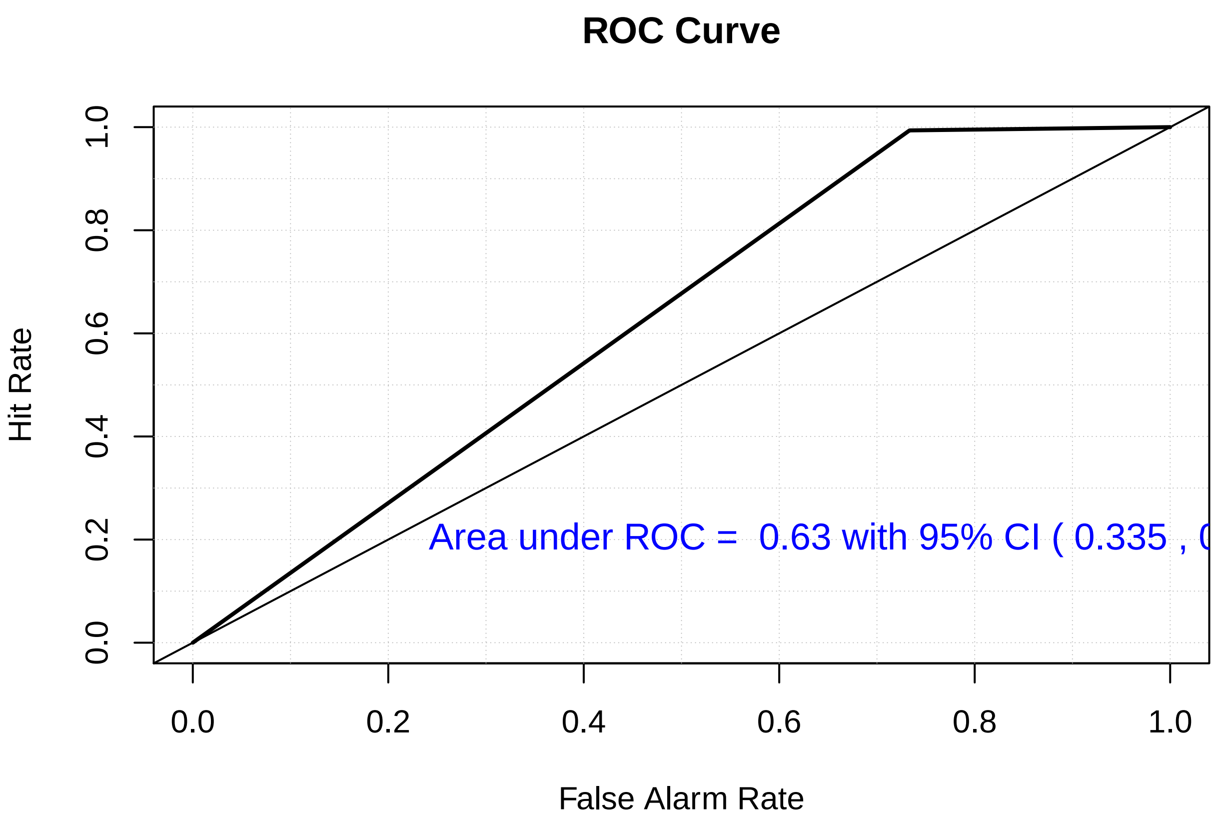
```
(svm_rad_auc.ci <- round(svm_rad_AUC$ci, digits = 3))
```

```
## [1] 0.335 0.925
```

```
library(verification)
mod.rad_h <- verify(obs = yobs, pred = test_pred_Radial)
```

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

```
roc.plot(mod.rad_h, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_rad_AUC$cvAUC, digits = 3), "with 95% 
                        svm_rad_auc.ci[1], ",", svm_rad_auc.ci[2], ").", sep = " "), col="blue
```




### 7.4 SVM-Radial-grid

```
grid_radial <- expand.grid(sigma = c(0.01, 0.02, 0.025, 0.03,
                                       0.05, 0.09, 0.1, 0.25, 0.5,
                                     0.75,0.9), C = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75,
                                   1, 1.5, 2,5 ))
```

```
set.seed(3233)
svm_Radial_Grid <- train(factor(outcome) ~., data = train, method = "svmRadial",
                           trControl=ctrl,
                           tuneGrid = grid_radial,
                           tuneLength = 9)
svm_Radial_Grid
```

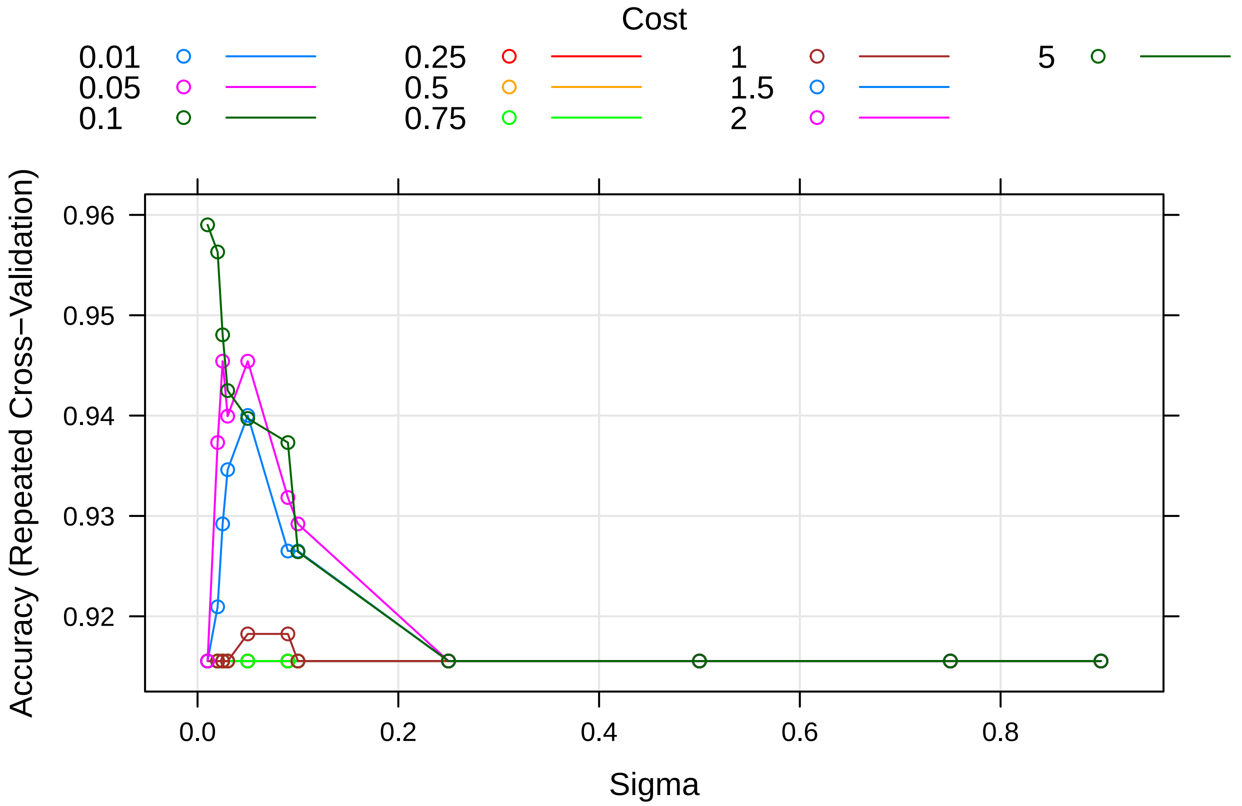
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 367 samples
##  18 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 331, 330, 330, 330, 330, 330, ...
## Resampling results across tuning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

```
##
##   sigma  C     Accuracy      Kappa
##   0.010  0.01  0.9155405405  0.00000000000
##   0.010  0.05  0.9155405405  0.00000000000
##   0.010  0.10  0.9155405405  0.00000000000
##   0.010  0.25  0.9155405405  0.00000000000
##   0.010  0.50  0.9155405405  0.00000000000
##   0.010  0.75  0.9155405405  0.00000000000
##   0.010  1.00  0.9155405405  0.00000000000
##   0.010  1.50  0.9155405405  0.00000000000
##   0.010  2.00  0.9155405405  0.00000000000
##   0.010  5.00  0.9590090090  0.63781707450
##   0.020  0.01  0.9155405405  0.00000000000
##   0.020  0.05  0.9155405405  0.00000000000
##   0.020  0.10  0.9155405405  0.00000000000
##   0.020  0.25  0.9155405405  0.00000000000
##   0.020  0.50  0.9155405405  0.00000000000
##   0.020  0.75  0.9155405405  0.00000000000
##   0.020  1.00  0.9155405405  0.00000000000
##   0.020  1.50  0.9209459459  0.09577464789
##   0.020  2.00  0.9373123123  0.33813425276
##   0.020  5.00  0.9563063063  0.63138650756
##   0.025  0.01  0.9155405405  0.00000000000
##   0.025  0.05  0.9155405405  0.00000000000
##   0.025  0.10  0.9155405405  0.00000000000
##   0.025  0.25  0.9155405405  0.00000000000
##   0.025  0.50  0.9155405405  0.00000000000
##   0.025  0.75  0.9155405405  0.00000000000
##   0.025  1.00  0.9155405405  0.00000000000
##   0.025  1.50  0.9292042042  0.21163421205
##   0.025  2.00  0.9454204204  0.44747236234
##   0.025  5.00  0.9480480480  0.58577562184
##   0.030  0.01  0.9155405405  0.00000000000
##   0.030  0.05  0.9155405405  0.00000000000
##   0.030  0.10  0.9155405405  0.00000000000
##   0.030  0.25  0.9155405405  0.00000000000
##   0.030  0.50  0.9155405405  0.00000000000
##   0.030  0.75  0.9155405405  0.00000000000
##   0.030  1.00  0.9155405405  0.00000000000
##   0.030  1.50  0.9346096096  0.29024692881
##   0.030  2.00  0.9399399399  0.38464997346
##   0.030  5.00  0.9424924925  0.53727195159
##   0.050  0.01  0.9155405405  0.00000000000
##   0.050  0.05  0.9155405405  0.00000000000
##   0.050  0.10  0.9155405405  0.00000000000
##   0.050  0.25  0.9155405405  0.00000000000
##   0.050  0.50  0.9155405405  0.00000000000
##   0.050  0.75  0.9155405405  0.00000000000
```

```
##    0.050  1.00  0.9182432432  0.04788732394
##    0.050  1.50  0.9400150150  0.36885964558
##    0.050  2.00  0.9454204204  0.45311146009
##    0.050  5.00  0.9397147147  0.49095405943
##    0.090  0.01  0.9155405405  0.00000000000
##    0.090  0.05  0.9155405405  0.00000000000
##    0.090  0.10  0.9155405405  0.00000000000
##    0.090  0.25  0.9155405405  0.00000000000
##    0.090  0.50  0.9155405405  0.00000000000
##    0.090  0.75  0.9155405405  0.00000000000
##    0.090  1.00  0.9182432432  0.04788732394
##    0.090  1.50  0.9265015015  0.17434607646
##    0.090  2.00  0.9318318318  0.30306103385
##    0.090  5.00  0.9373123123  0.35521499810
##    0.100  0.01  0.9155405405  0.00000000000
##    0.100  0.05  0.9155405405  0.00000000000
##    0.100  0.10  0.9155405405  0.00000000000
##    0.100  0.25  0.9155405405  0.00000000000
##    0.100  0.50  0.9155405405  0.00000000000
##    0.100  0.75  0.9155405405  0.00000000000
##    0.100  1.00  0.9155405405  0.00000000000
##    0.100  1.50  0.9265015015  0.17434607646
##    0.100  2.00  0.9292042042  0.22223340040
##    0.100  5.00  0.9264264264  0.21788557432
##    0.250  0.01  0.9155405405  0.00000000000
##    0.250  0.05  0.9155405405  0.00000000000
##    0.250  0.10  0.9155405405  0.00000000000
##    0.250  0.25  0.9155405405  0.00000000000
##    0.250  0.50  0.9155405405  0.00000000000
##    0.250  0.75  0.9155405405  0.00000000000
##    0.250  1.00  0.9155405405  0.00000000000
##    0.250  1.50  0.9155405405  0.00000000000
##    0.250  2.00  0.9155405405  0.00000000000
##    0.250  5.00  0.9155405405  0.00000000000
##    0.500  0.01  0.9155405405  0.00000000000
##    0.500  0.05  0.9155405405  0.00000000000
##    0.500  0.10  0.9155405405  0.00000000000
##    0.500  0.25  0.9155405405  0.00000000000
##    0.500  0.50  0.9155405405  0.00000000000
##    0.500  0.75  0.9155405405  0.00000000000
##    0.500  1.00  0.9155405405  0.00000000000
##    0.500  1.50  0.9155405405  0.00000000000
##    0.500  2.00  0.9155405405  0.00000000000
##    0.500  5.00  0.9155405405  0.00000000000
##    0.750  0.01  0.9155405405  0.00000000000
##    0.750  0.05  0.9155405405  0.00000000000
##    0.750  0.10  0.9155405405  0.00000000000
##    0.750  0.25  0.9155405405  0.00000000000
```

```
##   0.750  0.50  0.9155405405  0.00000000000
##   0.750  0.75  0.9155405405  0.00000000000
##   0.750  1.00  0.9155405405  0.00000000000
##   0.750  1.50  0.9155405405  0.00000000000
##   0.750  2.00  0.9155405405  0.00000000000
##   0.750  5.00  0.9155405405  0.00000000000
##   0.900  0.01  0.9155405405  0.00000000000
##   0.900  0.05  0.9155405405  0.00000000000
##   0.900  0.10  0.9155405405  0.00000000000
##   0.900  0.25  0.9155405405  0.00000000000
##   0.900  0.50  0.9155405405  0.00000000000
##   0.900  0.75  0.9155405405  0.00000000000
##   0.900  1.00  0.9155405405  0.00000000000
##   0.900  1.50  0.9155405405  0.00000000000
##   0.900  2.00  0.9155405405  0.00000000000
##   0.900  5.00  0.9155405405  0.00000000000
##
## 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.
```

```r
plot(svm_Radial_Grid)
```




```r
pred_Radial <- predict(svm_Radial_Grid, newdata = test)
confusionMatrix(pred_Radial, factor(test$outcome))
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0   5   1
##          1  10 157
##
##                Accuracy : 0.9364162
##                  95% CI : (0.8890805, 0.9678347)
##     No Information Rate : 0.9132948
##     P-Value [Acc > NIR] : 0.17280985
##
##                   Kappa : 0.448885
##  Mcnemar's Test P-Value : 0.01586133
##
##             Sensitivity : 0.33333333
##             Specificity : 0.99367089
##          Pos Pred Value : 0.83333333
##          Neg Pred Value : 0.94011976
##              Prevalence : 0.08670520
##          Detection Rate : 0.02890173
##    Detection Prevalence : 0.03468208
##       Balanced Accuracy : 0.66350211
##
##        'Positive' Class : 0
##
```

```r
(miss.rate_i <- mean(yobs != pred_Radial))
```

```
## [1] 0.06358381503
```

```r
pred_Radial <- as.numeric(as.character(pred_Radial))
MSE.i <- mean((yobs-pred_Radial)^2)
MSE.i
```

```
## [1] 0.06358381503
```

Plotting ROC curve of the fit.best model.

```r
library(cvAUC)
svm3_AUC <- ci.cvAUC(predictions = pred_Radial, labels = yobs, folds=1:NROW(test), confidence =
AUC
```

```
## $cvAUC
## [1] 0.9067510549
##
## $se
## [1] 0.0606345602
##
## $ci
## [1] 0.7879095006 1.0000000000
##
```

```
## $confidence
## [1] 0.95
```

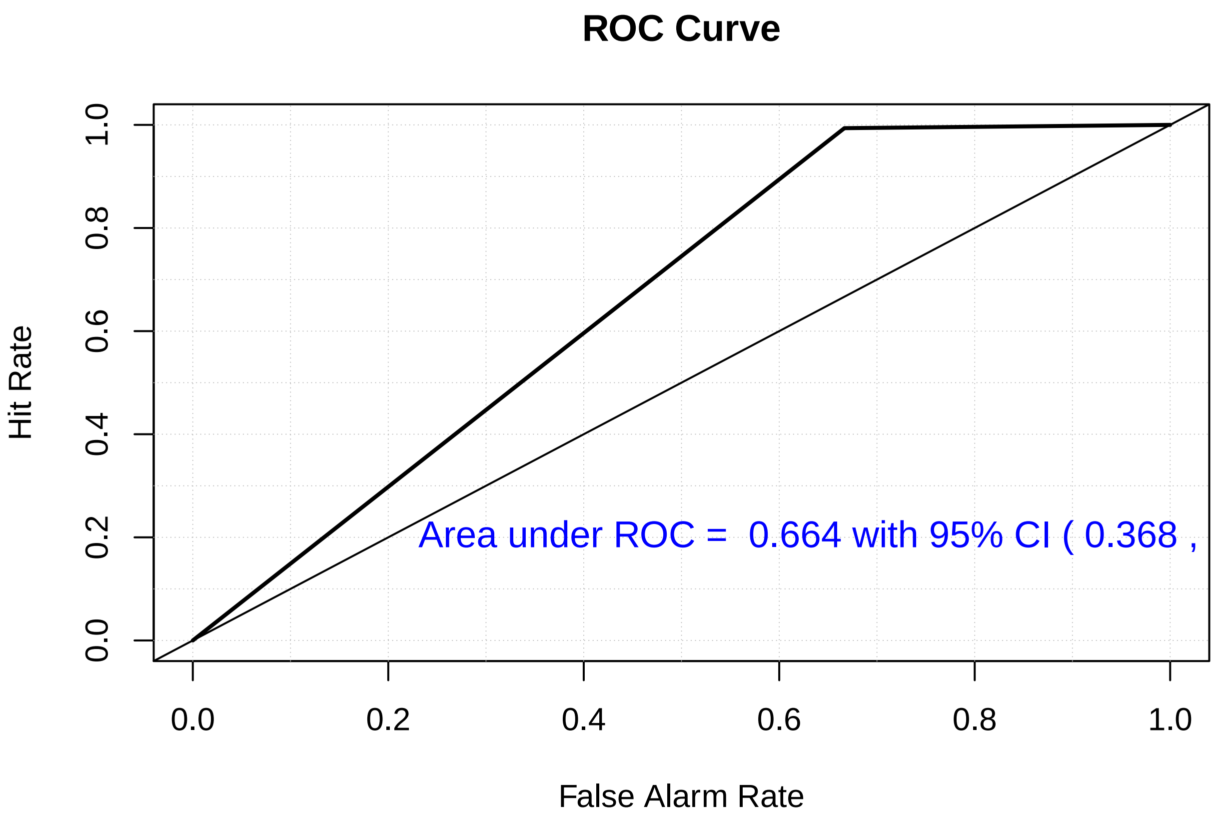
```
(svm3_auc.ci <- round(svm3_AUC$ci, digits = 3))
```

```
## [1] 0.368 0.959
```

```
library(verification)
mod.svm3 <- verify(obs = yobs, pred = pred_Radial)
```

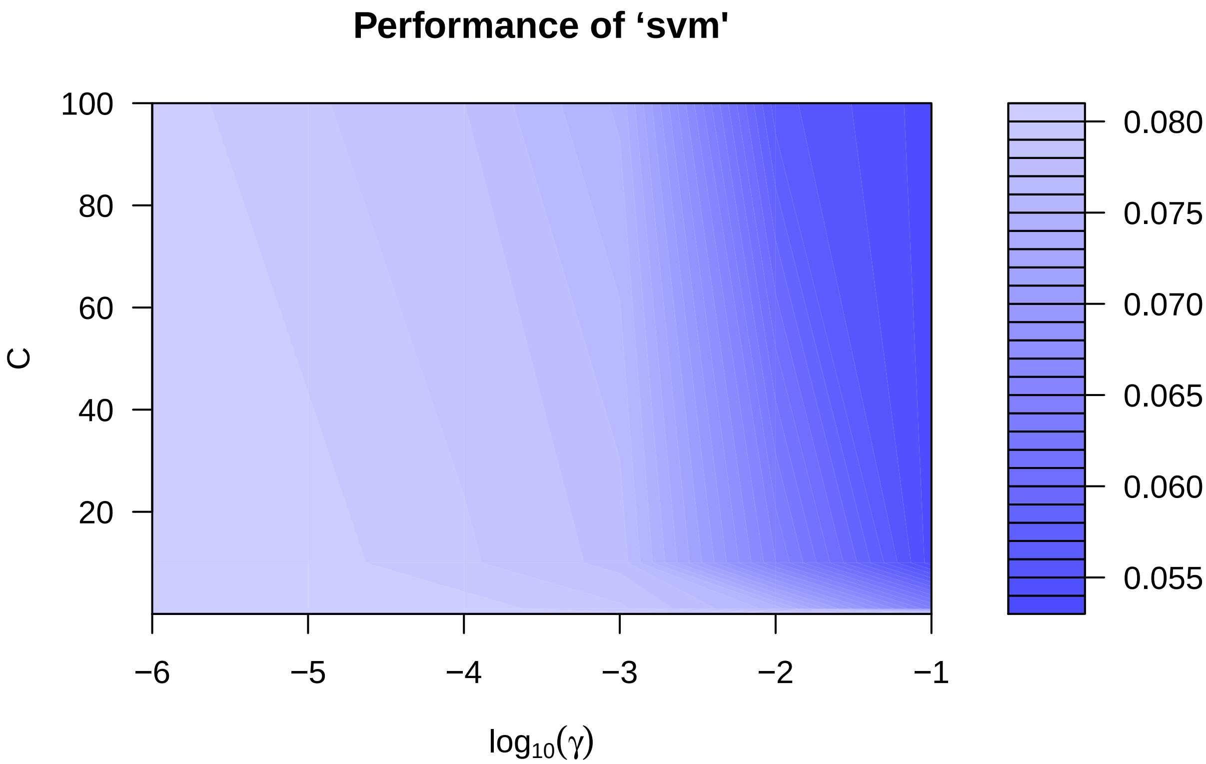
```
## If baseline is not included, baseline values  will be calculated from the  sample obs.
```

```
roc.plot(mod.svm3, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm3_AUC$cvAUC, digits = 3), "with 95% CI
                          svm3_auc.ci[1], ",", svm3_auc.ci[2], ").", sep = " "), col="blue", ce
```




### 7.5 SVM: Radial Basis kernel ”Gaussian”

```
library(kernlab)
library("e1071")
# USING ONLY PART OF THE DATA FOR TUNING FOR SHORTER COMPUTING TIME
tobj <- tune.svm(outcome ~ ., data = train, gamma = 10^(-6:-1), cost = 10^(-2:2), nrepeat=2,
                 tunecontrol = tune.control(sampling = "cross", cross=10))

plot(tobj, transform.x = log10, xlab = expression(log[10](gamma)), ylab = "C")
```




```
bestGamma <- tobj$best.parameters[[1]]
bestC <- tobj$best.parameters[[2]]
```

```
svm_probab  <- ksvm(outcome ~ ., data = train, type = "C-bsvc", kernel = "rbfdot",
    kpar = list(sigma = bestGamma), C = bestC,
    scaled=FALSE, prob.model = TRUE)
```

```
svm_probab_pred <- predict(svm_probab, type="probabilities", newdata=test)
svm_yhat <- svm_probab_pred[, 2]  # EXTRACT PREDICTED PROB FOR spam
```

```
MSE.j <- mean((yobs-svm_yhat)^2)
MSE.j
```

```
## [1] 0.0450621658
```

```
svm_yhat_factor <- ifelse(svm_yhat>0.5, 1, 0)
(miss.rate_j <- mean(yobs != svm_yhat_factor))
```

```
## [1] 0.06936416185
```

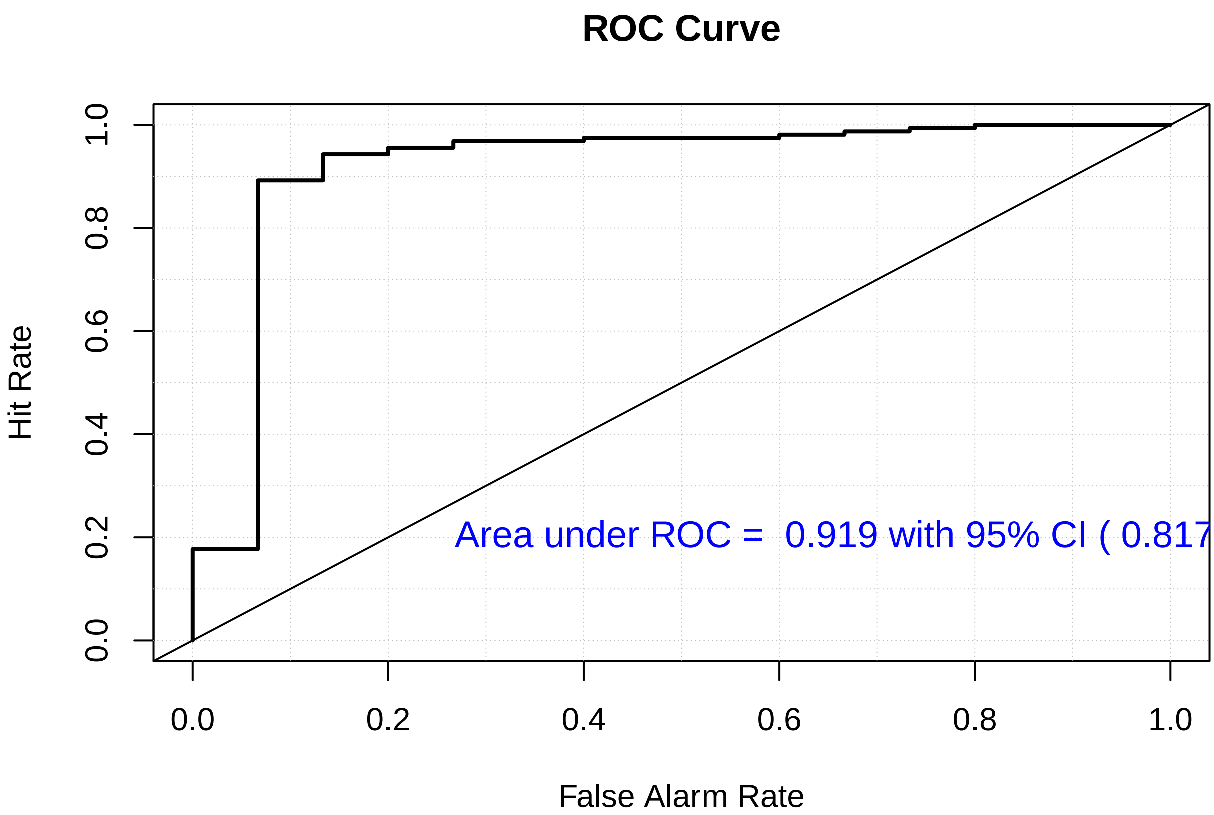
```
library(cvAUC)
svmprob_AUC <- ci.cvAUC(predictions = svm_yhat, labels =yobs, folds=1:NROW(test), confidence =
(svmprob_auc.ci <- round(svmprob_AUC$ci, digits = 3))
```

```
## [1] 0.817 1.000
```

```
library(verification)
mod.svmprob <- verify(obs = yobs, pred = svm_yhat)
```

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

```
roc.plot(mod.svmprob, plot.thres=NULL)
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svmprob_AUC$cvAUC, digits = 3), "with 95% C
                         svmprob_auc.ci[1], ",", svmprob_auc.ci[2], ").", sep = " "), col="blue
```




```
confusionMatrix(factor(yobs), factor(svm_yhat_factor))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   6   9
##          1   3 155
##
##                Accuracy : 0.9306358
##                  95% CI : (0.88197, 0.9636472)
##     No Information Rate : 0.9479769
##     P-Value [Acc > NIR] : 0.8816346
##
##                   Kappa : 0.4652241
##  Mcnemar's Test P-Value : 0.1489147
##
##             Sensitivity : 0.66666667
##             Specificity : 0.94512195
##          Pos Pred Value : 0.40000000
```

```
##          Neg Pred Value : 0.98101266
##              Prevalence : 0.05202312
##          Detection Rate : 0.03468208
##    Detection Prevalence : 0.08670520
##       Balanced Accuracy : 0.80589431
##
##        'Positive' Class : 0
##
```

# 8 Comparison among the fitted models

We see the MSE of ten fitting models that I used in this project.

```
(tabulate <- data.frame(
  Methods = c("Logistic regression","Random forest","ANN (single layer)", "ANN(two layer)", "AN
  Mean.squared.error = c(MSE.a, MSE.b, MSE.c, MSE.d, MSE.e, MSE.f, MSE.g, MSE.h, MSE.i, MSE.j)
  Missclassification.rate = c(miss.rate_a, miss.rate_b, miss.rate_c, miss.rate_d, miss.rate_e,
```

```
##                      Methods Mean.squared.error Missclassification.rate
## 1        Logistic regression      0.05003157544           0.07514450867
## 2              Random forest      0.06026632948           0.08670520231
## 3         ANN (single layer)      0.06982044344           0.06936416185
## 4             ANN(two layer)      0.08478119453           0.08670520231
## 5          ANN(Three layers)      0.06238087088           0.05780346821
## 6                 SVM-linear      0.06358381503           0.06358381503
## 7            SVM-linear-grid      0.05780346821           0.05780346821
## 8                 SVM-Radial      0.06936416185           0.06936416185
## 9                 SVM-Radial      0.06358381503           0.06358381503
## 10 SVM-Basis-Kernel_Gaussian      0.04506216580           0.06936416185
```

Remarks: The SVM-Basis-Kernel_Gaussian gives the lowest predicted mean squared error. The ANN with three layers have the lowest missclassificaiton rate.

Are under ROC curve:

```
(tabulate <- data.frame(
   Methods = c("Logistic regression","Random forest","ANN (single layer)", "ANN(two layer)", "A
  Model.Accuracy.percentage  = c(round(AUC$cvAUC, digits = 3),round(rf_AUC$cvAUC, digits =
                3),round(net_AUC$cvAUC, digits =
3),round(nn_AUC$cvAUC, digits = 3),round(nn2_AUC$cvAUC, digits = 3),round(svm_lin_AUC$cvAUC, di
```

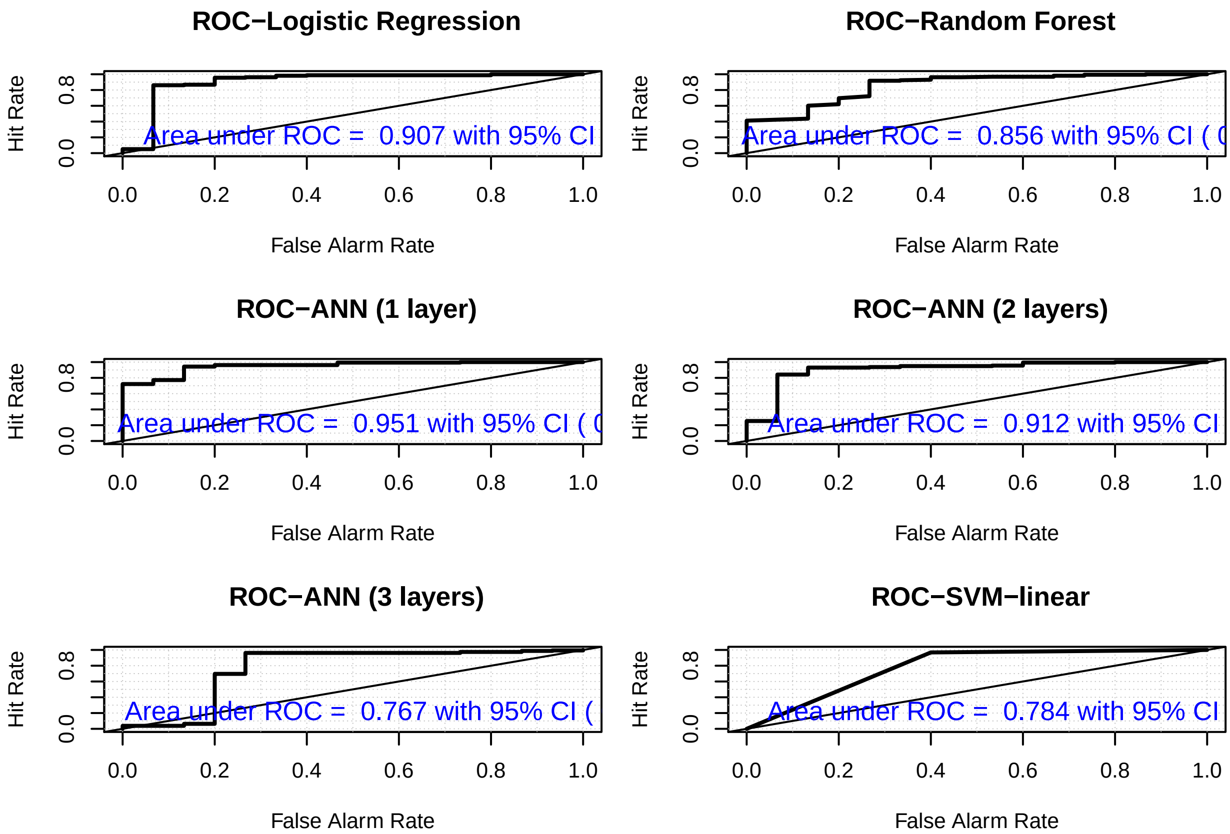
```
##                      Methods Model.Accuracy.percentage
## 1        Logistic regression                     0.907
## 2              Random forest                     0.856
## 3         ANN (single layer)                     0.951
## 4             ANN(two layer)                     0.912
## 5          ANN(Three layers)                     0.767
## 6                 SVM-linear                     0.784
## 7            SVM-linear-grid                     0.727
```

```
## 8                  SVM-Radial                             0.630
## 9                  SVM-Radial                             0.664
## 10 SVM-Basis-Kernel_Gaussian                              0.919
```

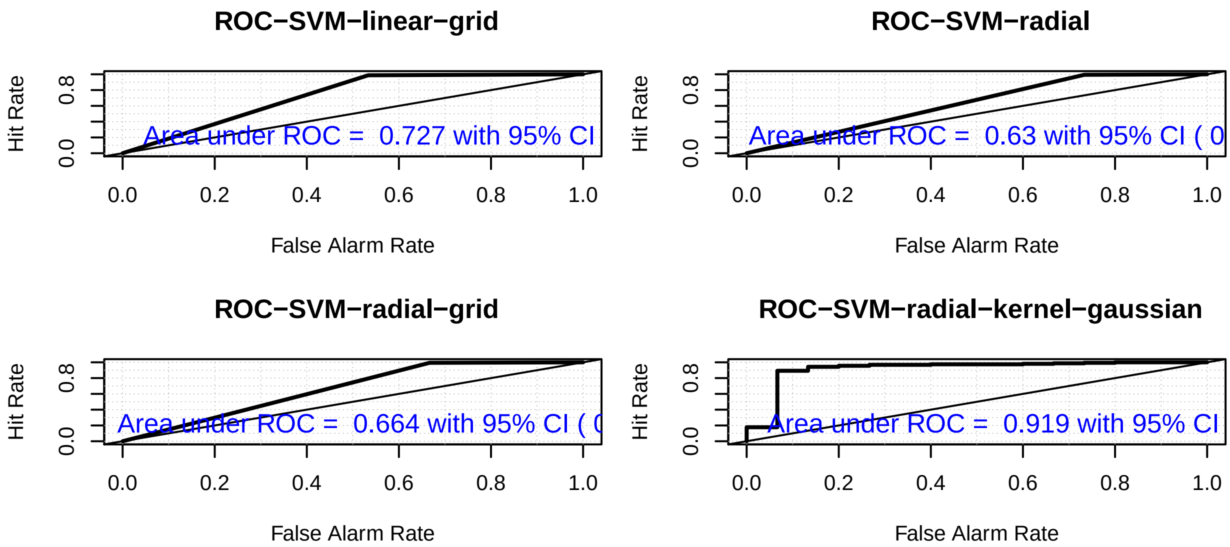
Remarks: ANN single layer gives the most accuracy. At the same time, SVM-Basis-Kernel_Gaussian, logistic regression, ANN with two layers, Random forest perform very well as well.

ROC curve:

```
par(mfrow=c(3, 2))
roc.plot(mod.glm, plot.thres=NULL, main=" ROC-Logistic Regression")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(AUC$cvAUC, digits = 3), "with 95% CI (",
                         auc.ci[1], ",", auc.ci[2], ").", sep = " "), col="blue", cex =1.2)

roc.plot(mod.rf, plot.thres=NULL, main=" ROC-Random Forest")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(rf_AUC$cvAUC, digits = 3), "with 95% CI ("
                         rf_auc.ci[1], ",", rf_auc.ci[2], ").", sep = " "), col="blue", cex =1


roc.plot(mod.net1, plot.thres = NULL, main=" ROC-ANN (1 layer)")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(net_AUC$cvAUC, digits = 3), "with 95% CI ("
                         net_auc.ci[1], ",", net_auc.ci[2], ").", sep = " "), col="blue", cex =

roc.plot(mod.nn, plot.thres = NULL, main=" ROC-ANN (2 layers)")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(nn_AUC$cvAUC, digits = 3), "with 95% CI ("
                         nn_auc.ci[1], ",", nn_auc.ci[2], ").", sep = " "), col="blue", cex =1

roc.plot(mod.nn2, plot.thres = NULL, main=" ROC-ANN (3 layers)")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(nn2_AUC$cvAUC, digits = 3), "with 95% CI ("
                         nn2_auc.ci[1], ",", nn2_auc.ci[2], ").", sep = " "), col="blue", cex =


roc.plot(mod.svm_lin, plot.thres=NULL, main=" ROC-SVM-linear")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_lin_AUC$cvAUC, digits = 3), "with 95% C
                         svm_lin_auc.ci[1], ",", svm_lin_auc.ci[2], ").", sep = " "), col="blue
```




```
roc.plot(mod.svm_lin_g, plot.thres=NULL, main=" ROC-SVM-linear-grid")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_lin_g_AUC$cvAUC, digits = 3), "with 95%
                         svm_lin_g_auc.ci[1], ",", svm_lin_g_auc.ci[2], ").", sep = " "), col="

roc.plot(mod.rad_h, plot.thres = NULL, main=" ROC-SVM-radial")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm_rad_AUC$cvAUC, digits = 3), "with 95% C
                        svm_rad_auc.ci[1], ",", svm_rad_auc.ci[2], ").", sep = " "), col="blue"

roc.plot(mod.svm3, plot.thres = NULL, main=" ROC-SVM-radial-grid")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svm3_AUC$cvAUC, digits = 3), "with 95% CI
                         svm3_auc.ci[1], ",", svm3_auc.ci[2], ").", sep = " "), col="blue", ce

roc.plot(mod.svmprob, plot.thres = NULL, main=" ROC-SVM-radial-kernel-gaussian")
text(x=0.7, y=0.2, paste("Area under ROC = ", round(svmprob_AUC$cvAUC, digits = 3), "with 95% C
                         svmprob_auc.ci[1], ",", svmprob_auc.ci[2], ").", sep = " "), col="blue
```




Remarks: From the ROC curve, we see that ANN single layer gives the most accuracy. At the same time, SVM-Basis-Kernel_Gaussian, Logistic Regression, ANN with two layers, Random Forest perform very well as well.