StatisticalMachineLearning_EndTerm_Assignment

Kaushik Srinivas_20201127

24/04/2021

Setting the working directory where my data file to analyze has been downloaded.

```
setwd("D:/UCD/Spring/Statistical ML/Final assignment/")
```

Loaded the rotten tomatoes review data into a variable. Then we removed the phrase column from the data set as we won't use it for analysis of the model which is a text data. We could see that there are 6874 observations and 80 variables are in this data set.Out of 80 we have 79 as predictor variables and one target variable class upon which different models has to be built to and choose the best model which would classify the sentiment of a movie review as negative or positive perfectly.

```
#loading the data in rtr variable
rtr=read.csv("data_rotten_tomatoes_review.csv")
#removing the column phrase from the data and keeping the numerical features
and target variable
rtr=rtr[,-1]
#checking the first three rows after removing the phrase column
head(rtr,n=3)
```

```
##   n_uq_chars n_commas   n_digits n_exclaims   n_lowers  n_lowersp     n_ca
ps
## 1  0.1207968 0.442474 -0.2134511 -0.1038426  1.2433068  1.0037988 -0.68750
73
## 2 -0.2323971 0.442474 -0.2134511 -0.1038426 -0.1098001  0.5575211 -0.68750
73
## 3  0.5988981 1.263669 -0.2134511 -0.1038426  0.1403369 -0.3472745  1.08066
78
##     n_puncts    n_capsp sent_afinn  sent_bing sent_syuzhet    n_polite
## 1 -0.9758369 -0.8996543  1.5706396 1.09057493    0.9417869 -0.02466707
## 2 -0.9758369 -0.4499548  1.0582413 1.55176149    1.1477516 -1.62230602
## 3  0.1222689  0.3183715 -0.9471978 0.01082872   -0.4175802 -0.02466707
##   n_first_person n_first_personp n_second_person n_second_personp     n_to
be
## 1     -0.4233958      -0.2264772      -0.3044738       -0.7361605  0.82735
98
## 2      2.1035981      -0.2264772      -0.3044738       -0.7361605  0.82735
98
## 3      3.5817948      -0.2264772      -0.3044738       -0.7361605 -0.76260
05
##   n_prepositions          w2         w4         w5         w6         w7
## 1      1.2520726 -0.32416637 -0.3091345 -0.4633992 -0.1949826 -0.1730423
## 2     -1.7362446 -0.02619313  0.9995526 -0.4633992 -0.4831844 -0.4552968
## 3      0.3035993 -0.21620505 -0.4561780  3.6878830 -0.2189994 -0.4552968
##              w8          w9         w15        w20         w21        w22
## 1 -0.4594595  3.6179931 -0.03693163  0.3342344 -0.30362721 -0.4461182
## 2  0.0241688 -0.4608782 -0.39030573 -0.4519702 -0.43065388  0.4355356
## 3  0.3465877 -0.4608782 -0.39030573 -0.4519702  0.03511058 -0.2012144
##              w23         w24         w26        w28         w29         w30
## 1  0.97380539  0.2118083  0.6770700 -0.4351941  0.14038133  0.06463583
## 2 -0.01294634  0.4217432 -0.4510137 -0.4351941  0.03458482 -0.45393465
## 3  0.03404183 -0.4442382 -0.4510137 -0.4351941 -0.46417019  0.49677791
##            w32        w33         w34        w37         w38        w39
## 1 -0.4488587 -0.4536693 -0.05755862  0.1559071 -0.32996275 -0.2778467
## 2  0.4511433 -0.4536693 -0.44459002 -0.4580608 -0.01456843 -0.4192329
## 3 -0.1988581 -0.1963339 -0.44459002 -0.4580608 -0.46709072 -0.4192329
##            w40        w42         w43        w44        w45        w46
w47
## 1  0.1139157 -0.4605785 -0.17786317 -0.1503605 -0.4584601 -0.1665023 -0.43
37444
## 2  0.9296220 -0.4605785 -0.42809690 -0.4291018  1.9986485 -0.4428809 -0.43
37444
## 3 -0.4391056  1.1551485  0.03066494 -0.4291018 -0.4584601 -0.4428809 -0.43
37444
##            w48         w50        w52         w53        w54        w55
w57
## 1 -0.4614204  0.32940877 -0.4710700 1.395010736  0.4979265  0.4839783 -0.4
785
## 2 -0.4614204  0.05517922 -0.4710700 0.001487591  0.4456493 -0.4643193 -0.4
785
## 3 -0.4614204 -0.47714872 -0.1991198 0.049240721 -0.1773211 -0.4643193 -0.4
785
##              w58         w60         w62         w63        w65        w66
## 1 -0.192389985 -0.03449159  0.03677083 -0.32936447 -0.4683988 -0.4340807
## 2 -0.036272138  0.01001945 -0.46936184 -0.01735229 -0.4683988  0.3735885
## 3  0.007761101 -0.47960200 -0.46936184  0.52979952  0.8195627 -0.2097282
##             w69        w72        w78        w80        w81          w83
## 1 -0.13172790 1.43899120 -0.3091702 -0.1998829 -0.3431334 -0.328448615
## 2  0.05135335 0.01854405 -0.4411493 -0.4618189 -0.4728893  0.005384107
## 3 -0.41339138 1.41458489 -0.4411493  0.0183970  0.9544266 -0.473593276
##            w84        w85        w86         w87        w89        w93
## 1 -0.4295101 -0.4231787  1.1852553 -0.31110048 -0.4138702 -0.2878923
## 2  4.1360499  0.8810776 -0.4324621 -0.03505551 -0.4138702 -0.4312614
## 3 -0.4295101 -0.4231787 -0.4324621 -0.43112003 -0.4138702  7.4540370
##             w94        w95        w96        w97        w98    class
## 1 -0.41445634 -0.4192596  0.8711155  0.9309612  0.9129291 negative
## 2 -0.41445634 -0.4192596 -0.4591863 -0.4584339 -0.4455004 positive
## 3  0.05849308  1.0249038 -0.4591863 -0.4584339 -0.4455004 negative

#checking the structure and dimensions of the data set
#str(rtr)
dim(rtr)
```

```
## [1] 6874   80
```

We could see that our target variable is character variable so we convert it to factor variable for our analysis.

```
#converting the class variable to factor variable for the analysis
rtr$class<-factor(rtr$class)

#checking the range of the data
range(rtr[,-80])

## [1] -13.46027  41.07558
```

We could see a lot of variation in the data so we scaled the data so that all the numerical variables will be in the same scale.

```
#since the range is varying across the data we can scale the numerical data b
efore   analysing

rtr[,-80]<-scale(rtr[,-80])
#moving it to new data frame for analysis
rtr1<-rtr
```

Library **"caret"** package which provides an extensive and integrated collection of functionalities that we can use for data splitting and model tuning using resampling-based methods has been used for the analysis of the data.

We used **createDataPartition()** function to split the data, here the split is 80% for train data and 20% for test data.

trainControl() function is used to perform k fold cross validation where the "number" parameter 2 indicates the number of k-folds and "repeats" 10 indicates number of replications that is how many times the loop should run. we took train data and test data into separate variables for analysis purpose.

```
library(caret)

## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

## Loading required package: ggplot2

# split
train_val <- createDataPartition(rtr1$class, p = 0.80, list = FALSE)
dat <- rtr1[train_val,]
dat_test <- rtr1[-train_val,]
#implementing 5 fold cross validation with 10 replicates using trainControl f
unction
train_ctrl <- trainControl(method = "repeatedcv", number = 2, repeats = 10)
```

Model Tuning and comparisons:

Implemented three different classification algorithms namely Support Vector Machines, Random Forest and logistic regression. We tuned each model by passing different hyper parameter values and trained using k-fold cross validation with 2 k-folds and replications as 10 and plotted the corresponding accuracies for each models with different hyper parameters passed.

Model 1: SVM

- Support vector machines (SVM) are an effective family of supervised classifier for data with complex non-linear structures.
- We use kernel trick to map original data into high dimensional spaces with less computational cost.
- The performance of support vector machine classifiers can be sensitive to the choice of kernel function and hyper parameters.
- Here we used Gaussian Radial Basis Function as kernel function which takes two parameters Cost C and Sigma.
- Cost function is the penalty term used to controls training error and margins in the data, Sigma is used to control over fitting and under fitting of the data that is it helps to generalize the data well.
- We took different values for C and sigma and trained the SVM model with different combination of values and plotted the validation accuracy across the different parameter values using k-fold cross validation technique.
- The method here used is "svmRadial" which indicates SVM - GRBF model for train function and data here used is train data and the target variable is class.
- trControl takes trainControl parameters which indicates k-fold cross validation and tuneGrid takes values of the expanded grid value combinations of parameters C and sigma.
- Optimal cost and sigma value obtained were sigma = 0.001 and C = 1.

Model 2: Random Forest

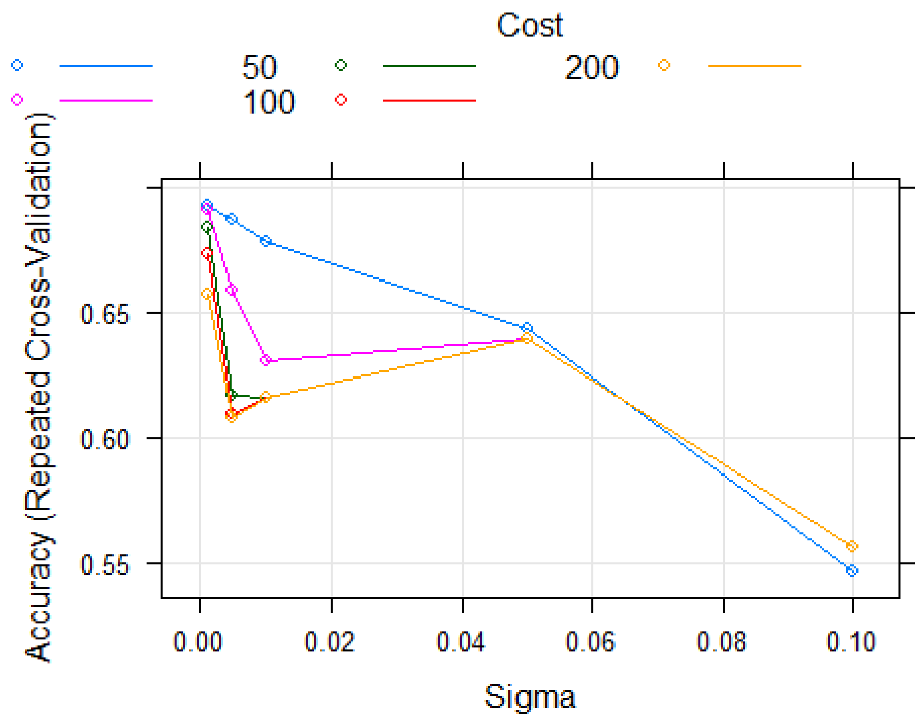
- Random Forest is a powerful and flexible ensemble learning procedures, improves the prediction performance of a classifiers.
- Random forests uses the same machinery of bagging, but a clever tweak that is at each split of the tree use a random subset of the input variables which reduces the correlation between trees and improves the predictive performance.
- Here the hyper parameter for random forest model is mtry which is number of predictors for split.We took here different mtry values starting from 2 to 79 by increasing at each step value as 5 and 79 which is the total number of parameters also included in the list.

- The method used here is ‘rf’ which indicates random forest model for the train function and data we used here is train data and the target variable is “class”.
- trControl takes trainControl parameters which indicates k-fold cross validation and tuneGrid takes values of the expanded grid value combinations of mtry parameters.
- Optimal value for mtry obtained from the models is mtry = 7

Model 3: Logistsic Regression

- Logistic regression is one of the best model when the target variable is binary response variable.
- Glmnet method that fits generalized linear and similar models via penalized maximum likelihood. The regularization path is computed for the lasso or elastic net penalty at a grid of values (on the log scale) for the regularization parameter lambda. The algorithm is extremely fast and can exploit sparsity in the input data.
- Here it takes two parameters alpha and lambda, alpha =0 indicating Ridge Regression which is a default value and alpha = 1 indicating Lasso Regression we took alpha =0 and lambda being penalty coefficient takes different range of values.
- The method used here is ‘glmnet’ which indicates Logistic Regression model for the train function and data we used here is train data and the target variable is “class”.
- trControl takes trainControl parameters which indicates k-fold cross validation and tuneGrid takes values of the expanded grid value combinations of alpha and lambda parameters.
- Optimal value for lambda obtained from the models were alpha = 0 and lambda = 0.175

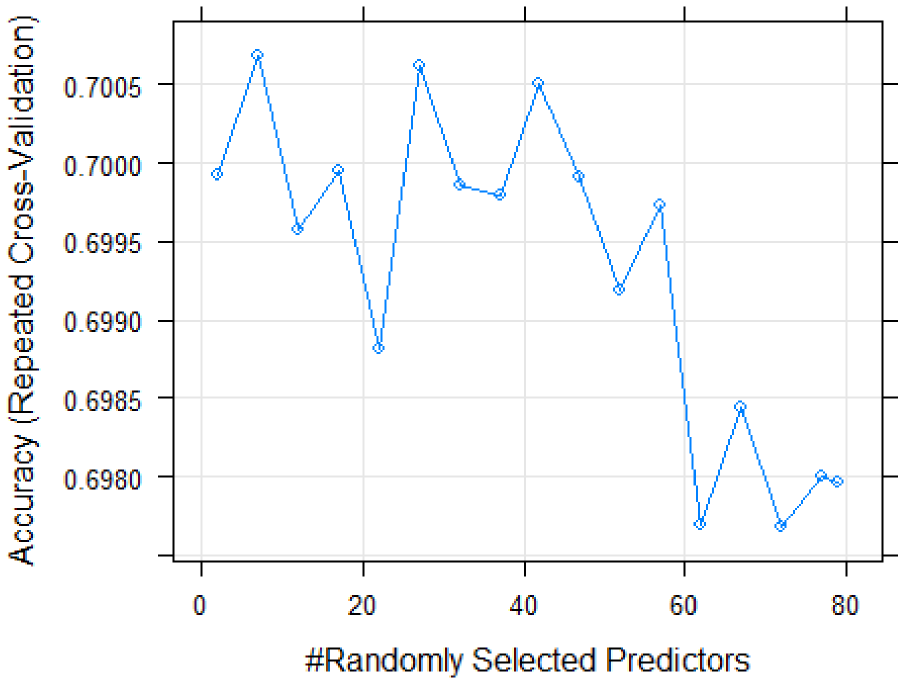
```
library(doParallel)

## Warning: package 'doParallel' was built under R version 4.0.5

## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel

cl <- makeCluster(4)
registerDoParallel(cl)


#sVMModel
tune_grid_svm =expand.grid(C=c(1, 10, 50, 100, 200),
                           sigma = c(0.001, 0.005, 0.01, 0.05, 0.1))
set.seed(20201127)
fit_svm_grbf <- train(class ~ ., data = dat,
method = "svmRadial",
trControl = train_ctrl,
tuneGrid = tune_grid_svm)
fit_svm_grbf

## Support Vector Machines with Radial Basis Function Kernel
##
## 5500 samples
##   79 predictor
##    2 classes: 'negative', 'positive'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 10 times)
## Summary of sample sizes: 2750, 2750, 2750, 2750, 2750, 2750, ...
## Resampling results across tuning parameters:
##
##   C    sigma  Accuracy   Kappa
##     1  0.001  0.6930000  0.38209817
##     1  0.005  0.6873091  0.37189403
##     1  0.010  0.6778727  0.35326346
##     1  0.050  0.6435636  0.27863199
##     1  0.100  0.5464545  0.05160113
##    10  0.001  0.6908182  0.37896060
##    10  0.005  0.6590909  0.31602077
##    10  0.010  0.6308727  0.25966182
##    10  0.050  0.6395091  0.27191265
##    10  0.100  0.5559636  0.07461507
##    50  0.001  0.6839091  0.36551783
##    50  0.005  0.6167273  0.23144672
##    50  0.010  0.6160545  0.23000198
##    50  0.050  0.6394182  0.27182956
##    50  0.100  0.5560545  0.07482301
##   100  0.001  0.6736364  0.34510063
##   100  0.005  0.6095091  0.21709996
##   100  0.010  0.6158000  0.22947305
##   100  0.050  0.6396364  0.27228791
##   100  0.100  0.5560545  0.07482301
##   200  0.001  0.6574000  0.31270212
##   200  0.005  0.6078909  0.21382904
##   200  0.010  0.6160182  0.22991938
##   200  0.050  0.6396364  0.27228791
##   200  0.100  0.5560545  0.07482301
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.001 and C = 1.

plot(fit_svm_grbf)
```



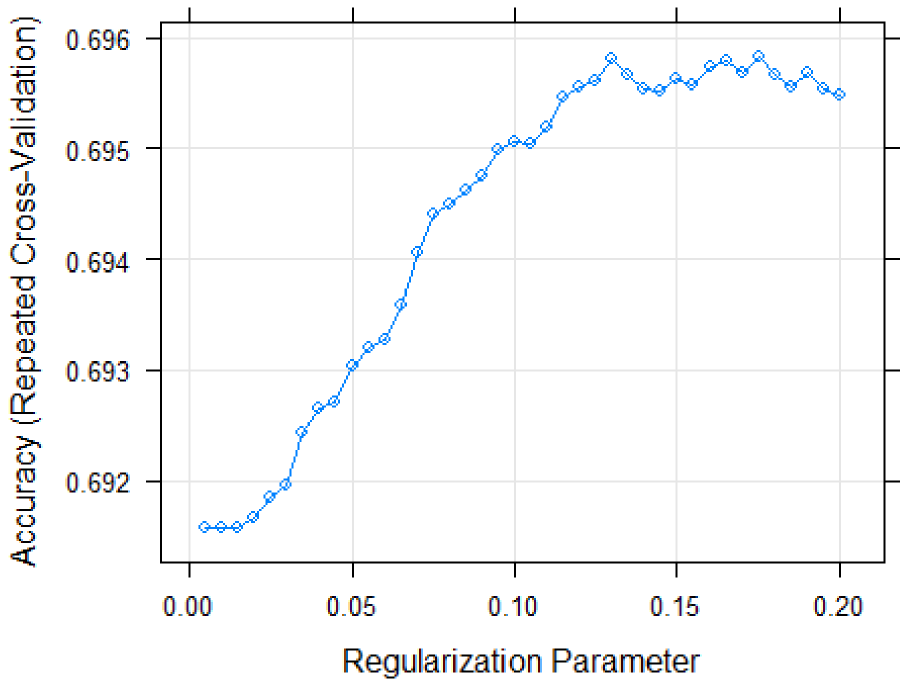
```
# random forest
# set grid
# note that we have 79 predictors
tune_grid_rf <- expand.grid( mtry = c(seq(2,79,5),79))
#
set.seed(20201127)
fit_rf <- train(class ~ ., data = dat,
method = "rf",
trControl = train_ctrl,
tuneGrid = tune_grid_rf)
fit_rf
```

```
## Random Forest
##
## 5500 samples
##   79 predictor
##    2 classes: 'negative', 'positive'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 10 times)
## Summary of sample sizes: 2750, 2750, 2750, 2750, 2750, 2750, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.6999273  0.3964717
##    7    0.7006909  0.3988794
##   12    0.6995818  0.3968438
##   17    0.6999455  0.3976819
##   22    0.6988182  0.3955205
##   27    0.7006182  0.3991605
##   32    0.6998545  0.3976986
##   37    0.6998000  0.3975466
##   42    0.7005091  0.3990495
##   47    0.6999091  0.3979410
##   52    0.6991818  0.3964021
##   57    0.6997273  0.3975256
##   62    0.6976909  0.3934070
##   67    0.6984364  0.3949694
##   72    0.6976727  0.3934097
##   77    0.6980000  0.3941149
##   79    0.6979636  0.3939395
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 7.
```

```
plot(fit_rf)
```



```
#logistic regression Model
tune_grid_lr<-expand.grid(alpha=seq(0,0,40),lambda=seq(0.005,0.200,0.005))
set.seed(20201127)
fit_lr<-train(class ~ ., data = dat,
              method="glmnet",
              trControl=train_ctrl,
              family="binomial",
                tuneGrid = tune_grid_lr)
plot(fit_lr)
```



```
fit_lr

## glmnet
##
## 5500 samples
##   79 predictor
##    2 classes: 'negative', 'positive'
##
## No pre-processing
## Resampling: Cross-Validated (2 fold, repeated 10 times)
## Summary of sample sizes: 2750, 2750, 2750, 2750, 2750, 2750, ...
## Resampling results across tuning parameters:
##
##   lambda  Accuracy   Kappa
##   0.005   0.6915636  0.3802325
##   0.010   0.6915636  0.3802325
##   0.015   0.6915636  0.3802325
##   0.020   0.6916545  0.3804057
##   0.025   0.6918545  0.3807635
##   0.030   0.6919636  0.3809348
##   0.035   0.6924364  0.3818128
##   0.040   0.6926545  0.3822103
##   0.045   0.6927091  0.3822672
##   0.050   0.6930364  0.3828796
##   0.055   0.6932000  0.3831570
##   0.060   0.6932727  0.3832668
##   0.065   0.6935818  0.3838551
##   0.070   0.6940545  0.3847778
##   0.075   0.6944000  0.3854371
##   0.080   0.6944909  0.3855753
##   0.085   0.6946182  0.3857938
##   0.090   0.6947455  0.3860289
##   0.095   0.6949818  0.3864664
##   0.100   0.6950545  0.3865913
##   0.105   0.6950364  0.3865062
##   0.110   0.6951818  0.3867647
##   0.115   0.6954545  0.3872866
##   0.120   0.6955455  0.3874147
##   0.125   0.6956000  0.3874849
##   0.130   0.6958182  0.3878857
##   0.135   0.6956727  0.3875446
##   0.140   0.6955273  0.3872052
##   0.145   0.6955091  0.3871287
##   0.150   0.6956182  0.3873115
##   0.155   0.6955636  0.3871790
##   0.160   0.6957455  0.3875060
##   0.165   0.6958000  0.3875931
##   0.170   0.6956909  0.3873497
##   0.175   0.6958364  0.3876101
##   0.180   0.6956727  0.3872512
##   0.185   0.6955455  0.3869658
##   0.190   0.6956909  0.3872217
##   0.195   0.6955273  0.3868545
##   0.200   0.6954727  0.3867197
##
## Tuning parameter 'alpha' was held constant at a value of 0
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were alpha = 0 and lambda = 0.175.

stopCluster(cl)
```

used the function resamples to easily compare the validation predictive performance of the three models across the folds and replications. summary function provides Accuracy and Kappa values for all the three model we will select the best model from this Accuracy values.

```
comp <- resamples(list(svm_grbf = fit_svm_grbf, rf = fit_rf,lr=fit_lr))
summary(comp)

##
## Call:
## summary.resamples(object = comp)
##
## Models: svm_grbf, rf, lr
## Number of resamples: 20
##
```

```
## Accuracy
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svm_grbf 0.6869091 0.6956364 0.6987273 0.6988545 0.7034545 0.7065455    0
## rf       0.6960000 0.7016364 0.7058182 0.7061818 0.7099091 0.7214545    0
## lr       0.6890909 0.6951818 0.7016364 0.7003091 0.7051818 0.7109091    0
##
## Kappa
##               Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svm_grbf 0.3700135 0.3879491 0.3929301 0.3937588 0.4030436 0.4089852    0
## rf       0.3893995 0.4011286 0.4106067 0.4108066 0.4181312 0.4417788    0
## lr       0.3739199 0.3869448 0.3988553 0.3965174 0.4060139 0.4182234    0
```

We could see from the above Accuracy table for the three different models Max Accuracy for validation set across k-folds is highest for Random forest model which is around 72.1%. Next highest is for the logistic regression with accuracy value as 71 % and we could see SVM has the least accuracy of 70.6 among the three models we used.

Here we choose the best model as Random forest to predict the test dataset.

```
# extract estimated class labels
class_hat <- predict(fit_rf, newdata = dat_test)
# compute metrics
confusionMatrix(class_hat, dat_test$class,positive ="positive")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction negative positive
##   negative      451      202
##   positive      203      518
##
##                Accuracy : 0.7052
##                  95% CI : (0.6803, 0.7293)
##     No Information Rate : 0.524
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.4091
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7194
##             Specificity : 0.6896
##          Pos Pred Value : 0.7184
##          Neg Pred Value : 0.6907
##              Prevalence : 0.5240
##          Detection Rate : 0.3770
##    Detection Prevalence : 0.5247
##       Balanced Accuracy : 0.7045
##
##        'Positive' Class : positive
##
```

- We predicted the test data with the fitted Random forest model and we obtained the test accuracy as 70.52 which seems to be pretty good for the dataset.
- From the confusion Matrix table we could see that from the total 721 positive values 518 has been correctly classified as positive and 203 has been misclassified as negative.
- Similarly for the negative class we could see that total of 653 values in the test data 451 has been correctly classified as negative and 202 has been misclassified at positive.
- Sensitivity/True positive rate is the measure which explains of those who are truly positive, how many were classified as positive and the value obtained is 0.71 which is close to 1 indicating that 71 percent of times model is able to predicate positive value as positive in our model.
- Specificity/True negative rate is the measure which explains of those that are truly negative, how many are classified as negative and the value obtained is 0.68 which is also close to 1 indicating that 68 percent of times model is able to predicate negative value as negative in our model.