# Reading Data:

tweets <- read.csv("training\_data.csv", stringsAsFactors = FALSE)

# Column Names of the data

> names(imdb)

[1] "sentiment" "review"

# Tabulating the target variable

table(imdb$sentiment)

0 1

12500 12500

# Loding the required libraries

library(tm)

library(SnowballC)

library(wordcloud)

library(rpart)  
library(rpart.plot)  
library(randomForest)

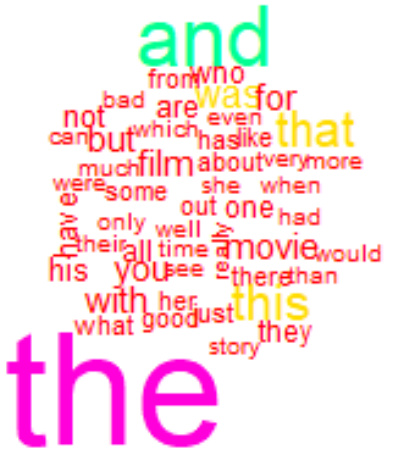
# Creating the corpus

corpus <- Corpus(VectorSource(imdb$review))

<Corpus is analogy to basket where we dump all the data and put them in an order, Corpus comes from tm package>

# Word cloud before any preprocessing:

wordcloud(corpus, colors=rainbow(7), max.words=50)



The top words are and, the, that and for which does not help in a model understanding the true meaning of the textual content. So we do data cleaning or some pre processing

# Data cleaning:

corpus <- tm\_map(corpus, tolower)

#Convert to lower-case: to reduce the error by capital letter and small letter

corpus <- tm\_map(corpus, removePunctuation)

# To remove the Punctuation

stopwords("english")[1:10]

[1] "i" "me" "my" "myself" "we" "our" "ours"

[8] "ourselves" "you" "your"

corpus <- tm\_map(corpus, removeWords, c("and","the", stopwords("english")))

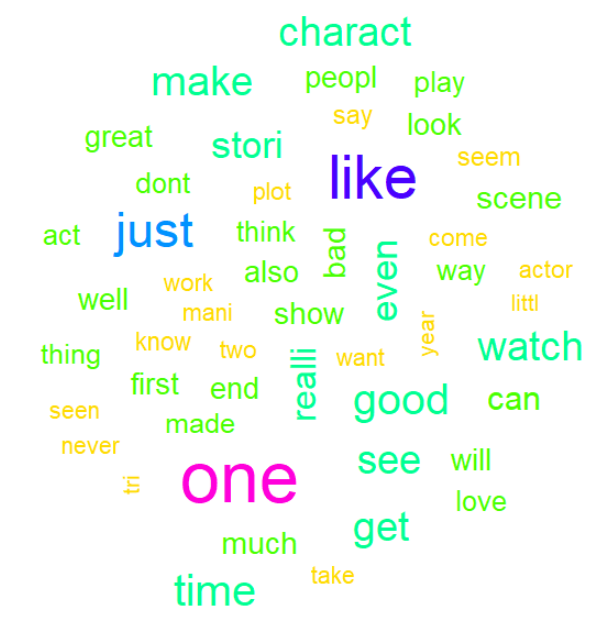
## tn\_map is a function from corpus to remove words

corpus <- tm\_map(corpus, stemDocument)

#To Stem Document: To compress a word from various tenses to a single basic word

# Word cloud after preprocessing:

# wordcloud(corpus, colors=rainbow(7), max.words=50)





Now the top words are “film, movi, like, time”

# Creating the document term matrix:

frequencies <- DocumentTermMatrix(corpus)

# 20 Low frequency words

findFreqTerms(frequencies, lowfreq = 20)

# Inspect the Matrix:

inspect(frequencies[1000:1005, 505:515])

<<DocumentTermMatrix (documents: 6, terms: 11)>>

Non-/sparse entries: 2/64

Sparsity : 97%

Maximal term length: 8

Weighting : term frequency (tf)

Sample :

Terms

Docs 80s trumpet turf unfold uninspir unmitig voic wagner way without

1000 0 0 0 0 0 0 0 0 1 0

# Handling sparse matrix removing words that are having less repetition:

sparse = removeSparseTerms(frequencies, 0.920)

# After Reducing:

#Sparse nrow: 25000

#Sparse ncol: 200

imdbsparse = as.data.frame(as.matrix(sparse))

# Make all variables R friendly: <By adding “X\_ “ in front of all numbers and spl characters>

colnames(imdbsparse) = make.names(colnames(imdbsparse))

# Adding the Dependent variable to the dataset:

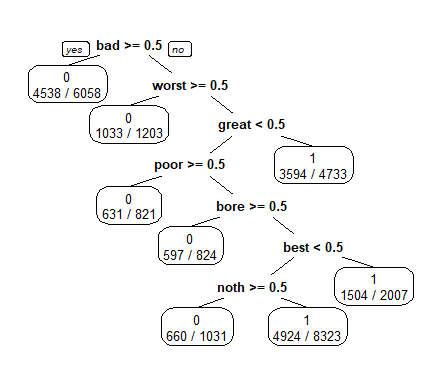
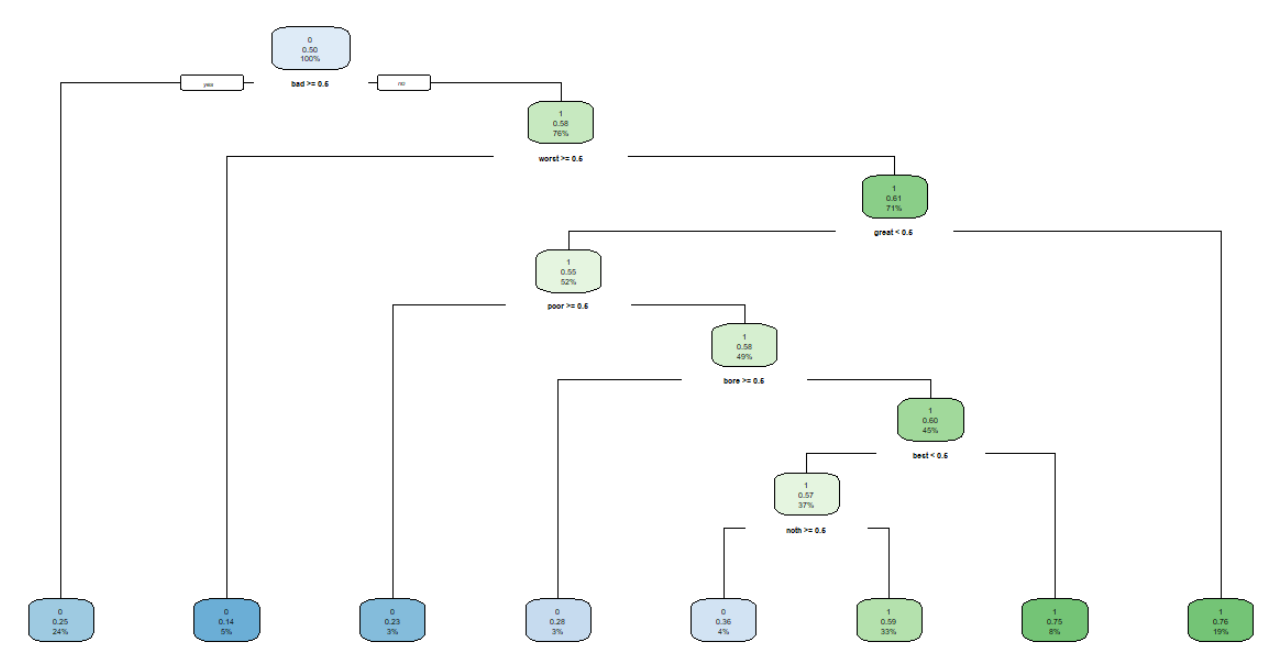
imdbsparse$sentiment <- imdb$sentiment

# Building CART model on sparse matrix :

imdbCART = rpart(sentiment~., data=imdbsparse, method="class")

# Plot of CART:

rpart.plot(imdbCART)



# Inferring CART Model:

Words used in review such as “Bad”, “worst”, “great”, “poor”, “bore”, “best”, “noth” has a major influence in the sentiment.

# Confusion matrix:

> imdbCARTpred <- predict(imdbCART, data=imdbsparse)

> imdbCARTCM <- table("Actual"= imdb$sentiment, "prediction"=imdbCARTpred[,2] > 0.5)

> imdbCARTCM

prediction

Actual FALSE TRUE

0 7459 5041

1 2478 10022

> accuracyCART <- (imdbCARTCM[1]+imdbCARTCM[4])/sum(imdbCARTCM)

> round(accuracyCART \* 100, 2)

[1] 69.92

The model accuracy of CART is 70%

F1 Score calculation:

F1 Score = 2\* (precision \* recall) / (precision + recall)

## F1Score

## [1] 0.7272068

# Building Random Forest with 20 tree limit:

imdbRF <- randomForest(sentiment~., data=imdbsparse,ntree=20)

varImpPlot(imdbRF)

# Confusion Matrix

imdbRFCM

prediction

Actual FALSE TRUE

0 9087 3413

1 3708 8792

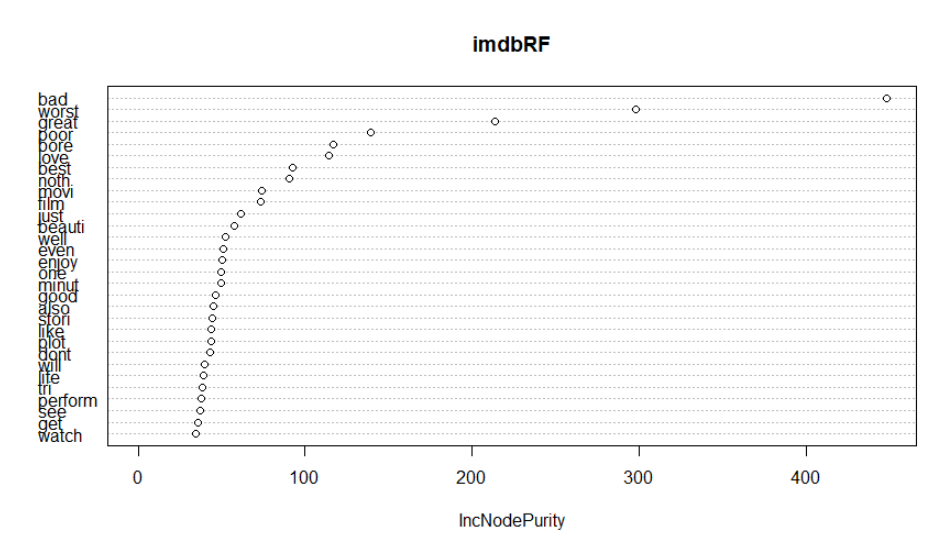
> accuracyRF <- (imdbRFCM[1]+imdbRFCM[4])/sum(imdbRFCM)

> round(accuracyRF \* 100, 2)

[1] 71.52

# Visualizing random forest:

varImpPlot(imdbRF)



# Inference:

Top 5 words that help classify are:

1. “bad”,
2. “worst”
3. ”great”
4. “poor”
5. “bore”

# Building Logistic regression model:

Before building logistic regression we need to reduce the number of columns otherwise we will end up with p>n problem which will need refined logistic regression methods.

We will remove the words that are having less density:

sparse = removeSparseTerms(frequencies, 0.920)

imdbsparse = as.data.frame(as.matrix(sparse))

colnames(imdbsparse) = make.names(colnames(imdbsparse))

> length(names(imdbsparse))

[1] 201

The column count is now reduced.

# Adding the target variable to the dataset

imdbsparse$sentiment <- imdb$sentiment

# Logistic regression model:

imdblr <- glm(sentiment~., data = imdbsparse, family = "binomial")

imdblrCM <- table("Actual" = imdbsparse$sentiment, "Prediction" = imdblr$fitted.values > 0.5)

summary(imdblr)

# Confusion Matrix:

> imdblrCM

Prediction

Actual FALSE TRUE

0 9668 2832

1. 2366 10134

# McFaden R2 to evaluate the model:

> round((1 - (imdblr$deviance/imdblr$null.deviance))\*100, 2)

[1] 34.69

The model explains about 34.69% of variance in the data from the number of data points we have, which is a very good score.

# Summary of the model:

> summary(imdblr)

Call:

glm(formula = sentiment ~ ., family = "binomial", data = imdbsparse)

Deviance Residuals:

Min 1Q Median 3Q Max

-4.1849 -0.7339 0.0000 0.7521 4.7547

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -0.0287809 0.0327204 -0.880 0.379075

actual -0.0386165 0.0365135 -1.058 0.290240

also 0.2529213 0.0278835 9.071 < 2e-16 \*\*\*

anoth -0.1185980 0.0409010 -2.900 0.003736 \*\*

away -0.1208985 0.0519406 -2.328 0.019932 \*

bad -0.7556468 0.0312718 -24.164 < 2e-16 \*\*\*

[ reached getOption("max.print") -- omitted 1 row ]

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 34657 on 24999 degrees of freedom

Residual deviance: 22635 on 24799 degrees of freedom

AIC: 23037

Number of Fisher Scoring iterations: 6

> colnames(words) <- c("pvalue", "zvalue")

> words

pvalue zvalue

also 9.070629 1.183319e-19

anoth -2.899638 3.735941e-03

away -2.327627 1.993191e-02

bad -24.163871 5.337700e-129

bit 6.344694 2.228682e-10

bore -15.707221 1.349663e-55

call -2.988660 2.802040e-03

cours 2.140186 3.233973e-02

differ 5.722361 1.050537e-08

director -4.691603 2.710727e-06

dont -5.358421 8.395258e-08

ever 3.971698 7.136203e-05

feel 4.183491 2.870664e-05

film 2.446408 1.442877e-02

final 2.356675 1.843936e-02

find 3.759276 1.704059e-04

hope -3.368366 7.561519e-04

just -10.087490 6.275389e-24

kid -3.114128 1.844896e-03

know 3.090460 1.998470e-03

let -3.054409 2.255041e-03

line -2.989679 2.792711e-03

lot 3.161289 1.570726e-03

made -2.553148 1.067541e-02

may 5.573670 2.494280e-08

minut -10.278947 8.768073e-25

movi -3.939189 8.175761e-05

one 2.550738 1.074952e-02

origin -6.926231 4.322012e-12

perform 8.167821 3.140082e-16

say -2.058616 3.953099e-02

see 5.811074 6.207334e-09

start -3.271194 1.070943e-03

thing -4.005659 6.184491e-05

think 2.214595 2.678788e-02

tri -9.248043 2.286395e-20

want -3.129812 1.749183e-03

well 10.229366 1.464746e-24

whole -2.750777 5.945407e-03

work 4.032052 5.529189e-05

enjoy 13.357642 1.068966e-40

entertain 5.666654 1.456133e-08

found -2.037295 4.162051e-02

great 20.068627 1.387856e-89

look -7.154728 8.383891e-13

world 7.073750 1.508024e-12

year 3.855451 1.155163e-04

appear -3.689001 2.251363e-04

better -6.883815 5.827053e-12

good 6.814654 9.449114e-12

high 4.767285 1.867252e-06

man 3.517637 4.354080e-04

mani 2.874543 4.046128e-03

much -3.238587 1.201234e-03

quit 3.081538 2.059340e-03

real 2.075692 3.792245e-02

run -3.273698 1.061501e-03

scene -2.248789 2.452595e-02

seem -6.555388 5.549751e-11

stori 4.574979 4.762678e-06

though 5.042183 4.602507e-07

actor -5.153145 2.561539e-07

anyth -4.929310 8.252071e-07

cant -3.539694 4.005917e-04

didnt -6.436117 1.225688e-10

especi 6.662773 2.687087e-11

idea -6.663221 2.678904e-11

play 2.240793 2.503948e-02

read -2.980646 2.876409e-03

seen 6.958569 3.437465e-12

still 8.002493 1.219254e-15

way 3.293566 9.892514e-04

without 2.304425 2.119882e-02

doesnt -4.019277 5.837690e-05

even -10.179457 2.449233e-24

friend 2.765941 5.675873e-03

fun 8.873141 7.111143e-19

least -8.138838 3.990907e-16

life 6.431322 1.264987e-10

littl 2.112077 3.467980e-02

plot -8.622943 6.525470e-18

reason -6.884874 5.783858e-12

script -10.193348 2.123245e-24

there -2.102904 3.547419e-02

time 2.959920 3.077186e-03

action 5.052751 4.354920e-07

pretti -2.182230 2.909258e-02

act -7.243262 4.380196e-13

complet -6.574589 4.878758e-11

keep 4.732935 2.212967e-06

woman -3.148009 1.643864e-03

show 4.032906 5.509133e-05

audienc -2.933947 3.346813e-03

name -3.011172 2.602417e-03

worst -22.933604 2.147901e-116

young 2.786552 5.327215e-03

although 5.034256 4.797081e-07

alway 6.658130 2.773343e-11

far -3.804403 1.421464e-04

recommend 7.701079 1.349218e-14

wonder 6.058282 1.375830e-09

best 14.582984 3.604137e-48

love 13.129173 2.240891e-39

mean -2.245184 2.475631e-02

move 6.251691 4.060324e-10

point -3.275233 1.055748e-03

role 2.554725 1.062718e-02

saw 2.676455 7.440555e-03

surpris 7.619587 2.544894e-14

will 5.101660 3.366871e-07

enough -4.758918 1.946337e-06

noth -13.898752 6.445498e-44

right 3.935429 8.304812e-05

old -3.048271 2.301621e-03

someth -4.642525 3.441771e-06

first 2.357301 1.840833e-02

interest -3.508487 4.506636e-04

poor -17.746171 1.845146e-70

day 3.779609 1.570750e-04

new 3.346205 8.192574e-04

that -2.476057 1.328425e-02

worth 3.451334 5.578228e-04

person 2.909484 3.620258e-03

beauti 11.495312 1.392763e-30

kill -2.852707 4.334855e-03

live 2.795045 5.189243e-03

yet 2.358708 1.833868e-02

help 2.196027 2.809000e-02

might -4.841595 1.288012e-06

From comparing from the other 2 models there are few more words playing an important role:

“also”, “bit”, “ever”, “find”, “hope”, “just”

has made an impact in the sentiment.

# Accuracy of the model:

> accuracylr <- (imdblrCM[1]+imdblrCM[4])/sum(imdblrCM)

> round(accuracylr \* 100, 2)

[1] 79.21

The Accuracy is 79.21% which is quite better than the CART and RF model.

# Accuracy of the model:

> round(c("CART" = accuracyCART, "RF" = accuracyRF, "GLM" = accuracylr) \* 100, 2)

CART RF GLM

69.92 71.52 79.21

# F1 Scores of the Models:

> round(c("F1score\_CART" = F1Score\_CART, "F1score\_RF" = F1Score\_RF, "F1score\_GLM" = F1Score\_glm) \* 100, 2)

F1score\_CART F1score\_RF F1score\_GLM

72.72 71.18 79.59

# Finding Sentiment in the Test data using CART, RF, GLM:

CART Model:

CART Model for test\_data in .csv:



imdb\_test <- read.csv("testdata.csv", stringsAsFactors = FALSE)

imdbtest\_CART <- predict(imdbCART, data=imdb\_test)

imdbtest\_CART$sentiment <- imdbtest\_CART[,2] > 0.5

write.csv(imdbtest\_CART$sentiment, file = " cartmodel.csv")

Random Forest Model:

Random Forest Model for test\_data in .csv:



imdbtest\_RF <- predict(imdbRF, data=imdb\_test)

imdbtest\_RF$sentiment <- imdbtest\_RF>0.5

write.csv(imdbtest\_RF$sentiment, file = "RFmodel.csv")

Generalized Linear Model:

Generalized Linear Model for test\_data in .csv:



imdbtest\_lr$sentiment <- imdbtest\_lr > 0.5

write.csv(imdbtest\_lr$sentiment, file = "C:/Users/sarveshwaran/Desktop/New folder/all/lrmodel.csv")