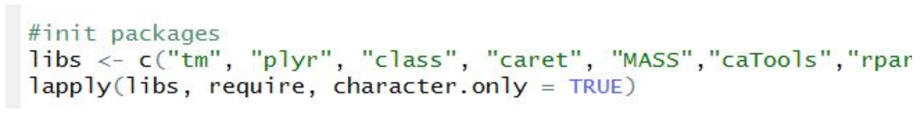
Tejal Bhangale

R : Hands-on exercises

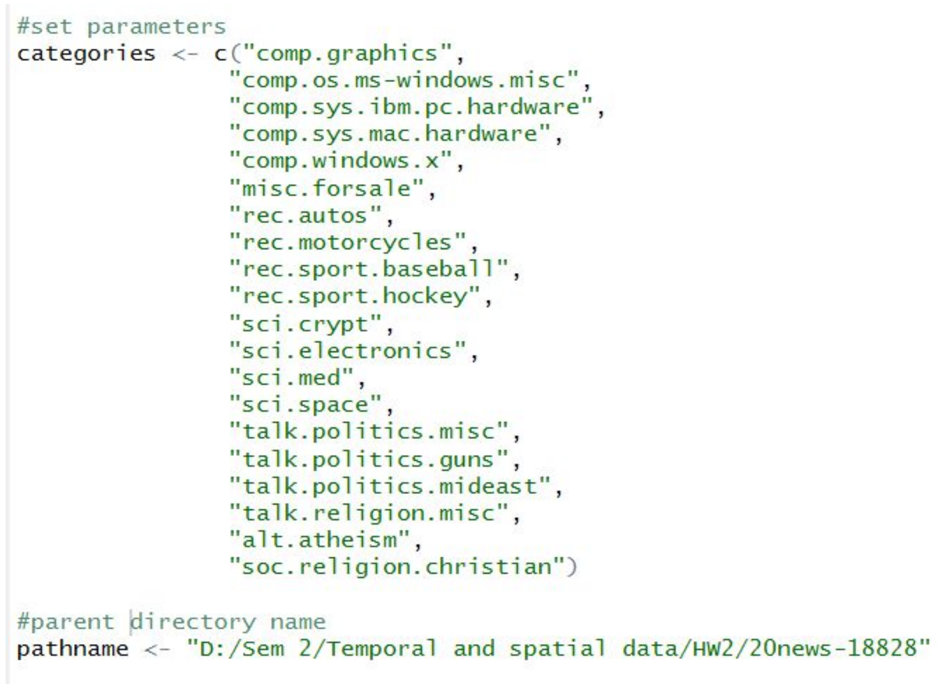
1. Use the same data matrix as in Similarity repository

Packages and libraries :



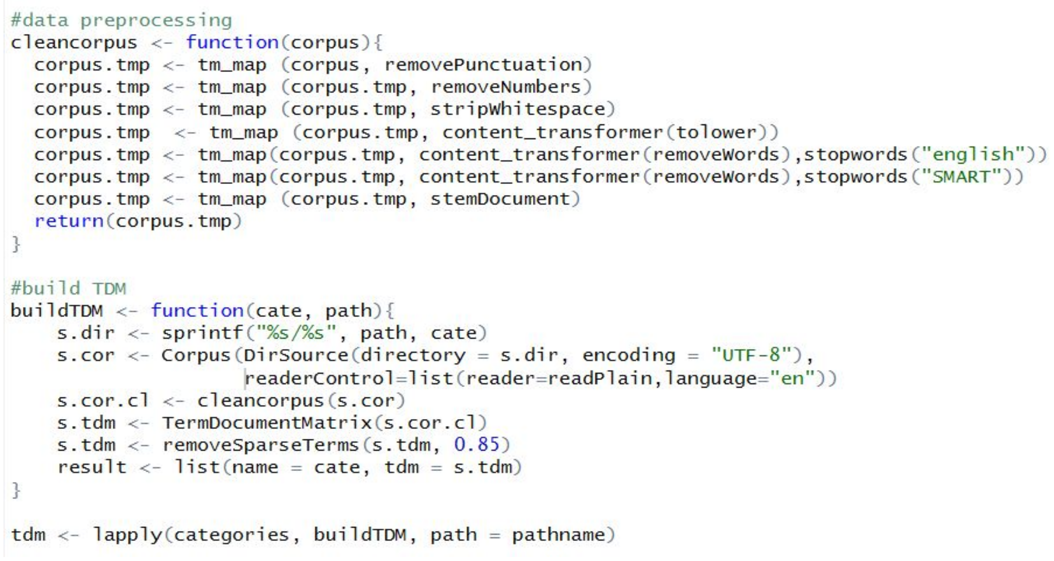
```
#init packages
libs <- c("tm", "plyr", "class", "caret", "MASS","caTools","rpar
lapply(libs, require, character.only = TRUE)
```

Set categories and directory name:



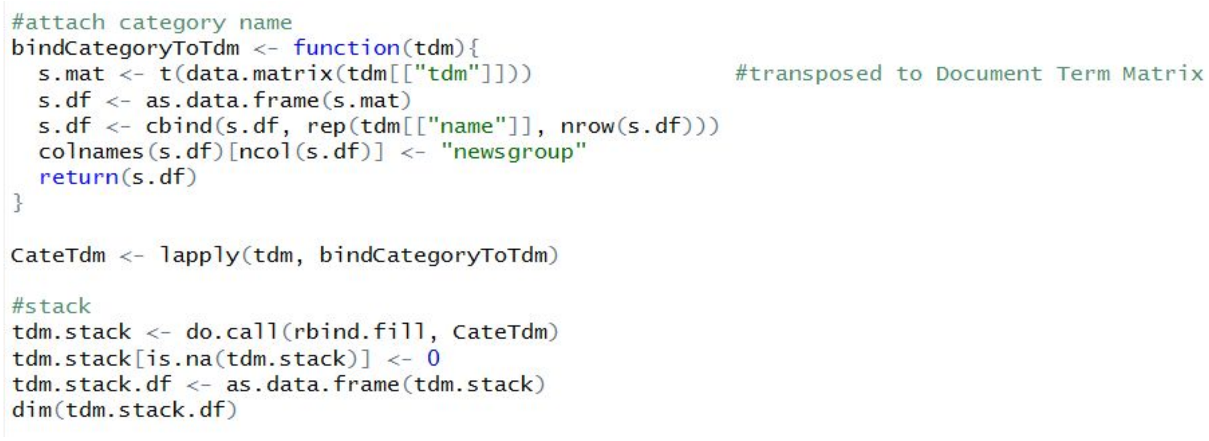
```
#set parameters
categories <- c("comp.graphics",
                "comp.os.ms-windows.misc",
                "comp.sys.ibm.pc.hardware",
                "comp.sys.mac.hardware",
                "comp.windows.x",
                "misc.forsale",
                "rec.autos",
                "rec.motorcycles",
                "rec.sport.baseball",
                "rec.sport.hockey",
                "sci.crypt",
                "sci.electronics",
                "sci.med",
                "sci.space",
                "talk.politics.misc",
                "talk.politics.guns",
                "talk.politics.mideast",
                "talk.religion.misc",
                "alt.atheism",
                "soc.religion.christian")

#parent directory name
pathname <- "D:/Sem 2/Temporal and spatial data/HW2/20news-18828"
```

Data pre-processing and building a Document-Term-Matrix:
(Sparsity factor : 0.85)



```
#data preprocessing
cleancorpus <- function(corpus){
  corpus.tmp <- tm_map (corpus, removePunctuation)
  corpus.tmp <- tm_map (corpus.tmp, removeNumbers)
  corpus.tmp <- tm_map (corpus.tmp, stripWhitespace)
  corpus.tmp  <- tm_map (corpus.tmp, content_transformer(tolower))
  corpus.tmp <- tm_map(corpus.tmp, content_transformer(removeWords),stopwords("english"))
  corpus.tmp <- tm_map(corpus.tmp, content_transformer(removeWords),stopwords("SMART"))
  corpus.tmp <- tm_map (corpus.tmp, stemDocument)
  return(corpus.tmp)
}

#build TDM
buildTDM <- function(cate, path){
    s.dir <- sprintf("%s/%s", path, cate)
    s.cor <- Corpus(DirSource(directory = s.dir, encoding = "UTF-8"),
                    readerControl=list(reader=readPlain,language="en"))
    s.cor.cl <- cleancorpus(s.cor)
    s.tdm <- TermDocumentMatrix(s.cor.cl)
    s.tdm <- removeSparseTerms(s.tdm, 0.85)
    result <- list(name = cate, tdm = s.tdm)
}

tdm <- lapply(categories, buildTDM, path = pathname)
```

Attach newsgroup category name to each article (each row):

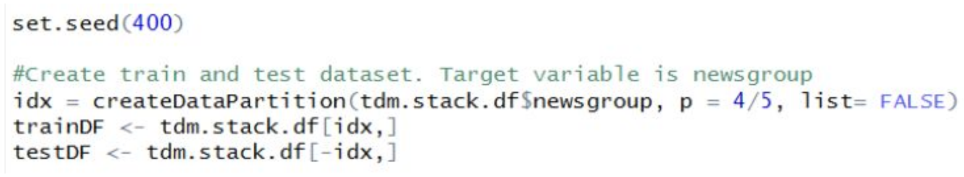


```
#attach category name
bindCategoryToTdm <- function(tdm){
  s.mat <- t(data.matrix(tdm[["tdm"]]))          #transposed to Document Term Matrix
  s.df <- as.data.frame(s.mat)
  s.df <- cbind(s.df, rep(tdm[["name"]], nrow(s.df)))
  colnames(s.df)[ncol(s.df)] <- "newsgroup"
  return(s.df)
}

CateTdm <- lapply(tdm, bindCategoryToTdm)

#stack
tdm.stack <- do.call(rbind.fill, CateTdm)
tdm.stack[is.na(tdm.stack)] <- 0
tdm.stack.df <- as.data.frame(tdm.stack)
dim(tdm.stack.df)
```

2. Run kNN and Decision Tree with 5-fold cross validation

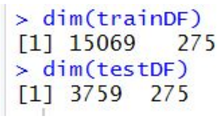
Create data partitions and create test and train datasets. Train dataset contains ⅘ th of the rows (which is part1,part2,part3,part4) and Test dataset contains ⅕ th of the rows (which is part5).

Similarly, to apply testing on this part1, part2, part3, part4, tarinControl function implements 5-fold cross validation. Then, train function implements knn model on training dataset.



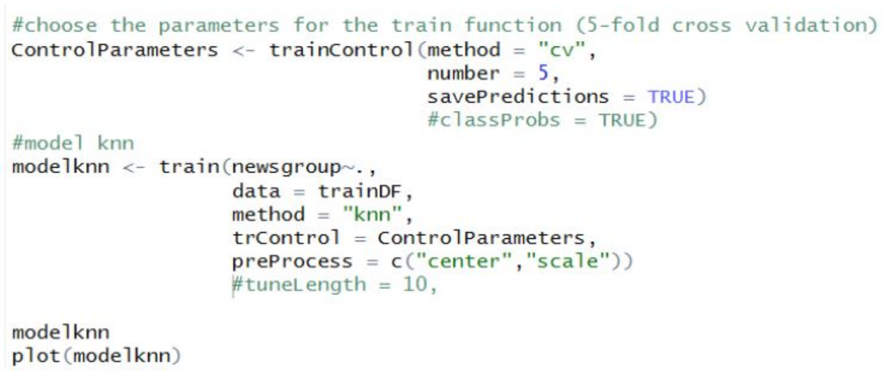
```
set.seed(400)

#Create train and test dataset. Target variable is newsgroup
idx = createDataPartition(tdm.stack.df$newsgroup, p = 4/5, list= FALSE)
trainDF <- tdm.stack.df[idx,]
testDF <- tdm.stack.df[-idx,]
```

Dimensions of train and test dataset:



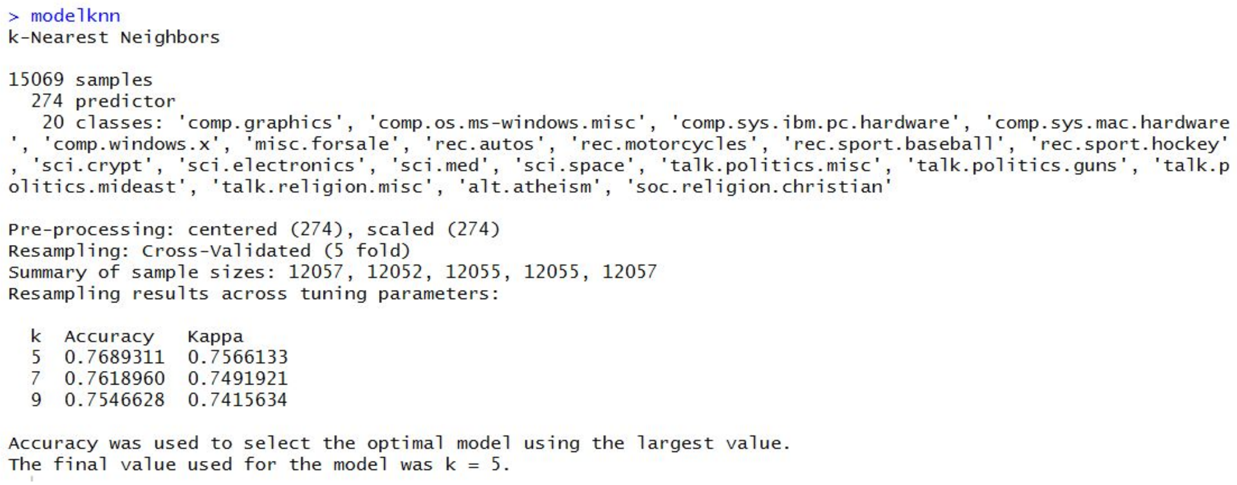
```
> dim(trainDF)
[1] 15069   275
> dim(testDF)
[1] 3759  275
```

3.a. Implement knn model with 5-fold cross validation before feature selection



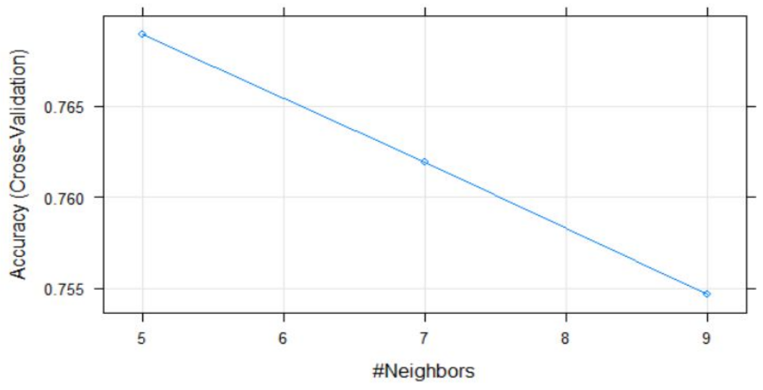
```
#choose the parameters for the train function (5-fold cross validation)
ControlParameters <- trainControl(method = "cv",
                                  number = 5,
                                  savePredictions = TRUE)
                                  #classProbs = TRUE)
#model knn
modelknn <- train(newsgroup~.,
                  data = trainDF,
                  method = "knn",
                  trControl = ControlParameters,
                  preProcess = c("center","scale"))
                  #tuneLength = 10,

modelknn
plot(modelknn)
```

Output : Accuracy for different values of k. The best value of **k = 5**

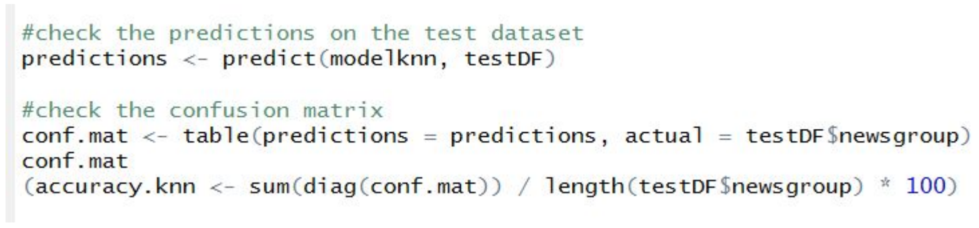


```
> modelknn
k-Nearest Neighbors

15069 samples
  274 predictor
   20 classes: 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware
', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey'
, 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'talk.politics.misc', 'talk.politics.guns', 'talk.p
olitics.mideast', 'talk.religion.misc', 'alt.atheism', 'soc.religion.christian'

Pre-processing: centered (274), scaled (274)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12057, 12052, 12055, 12055, 12057
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.7689311  0.7566133
  7  0.7618960  0.7491921
  9  0.7546628  0.7415634

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

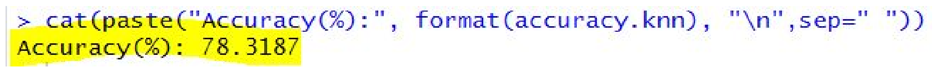
Plot for different k with accuracy: k=5 gives the highest accuracy



Check Predictions on the test dataset:

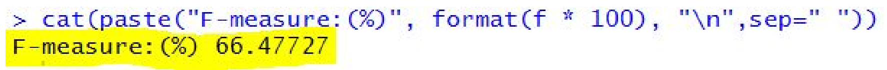


```
#check the predictions on the test dataset
predictions <- predict(modelknn, testDF)

#check the confusion matrix
conf.mat <- table(predictions = predictions, actual = testDF$newsgroup)
conf.mat
(accuracy.knn <- sum(diag(conf.mat)) / length(testDF$newsgroup) * 100)
```

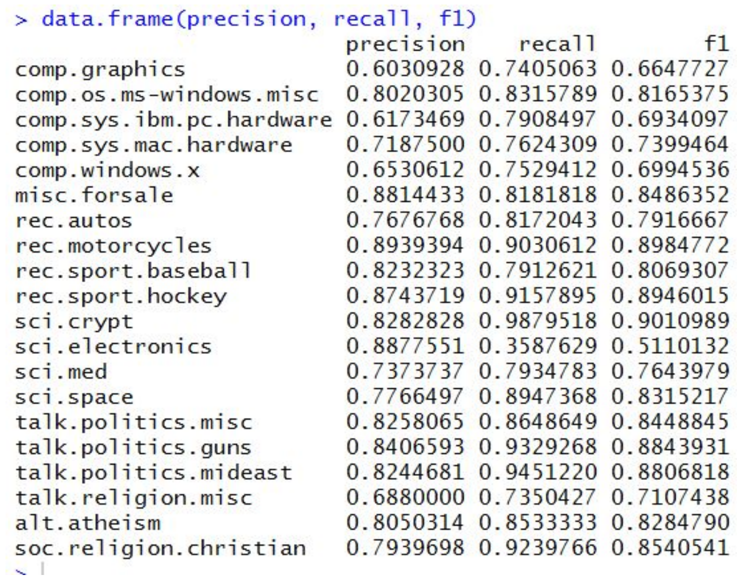


```
> cat(paste("Accuracy(%):", format(accuracy.knn), "\n",sep=" "))
Accuracy(%): 78.3187
```

Overall f-measure and f-measures for each article category:

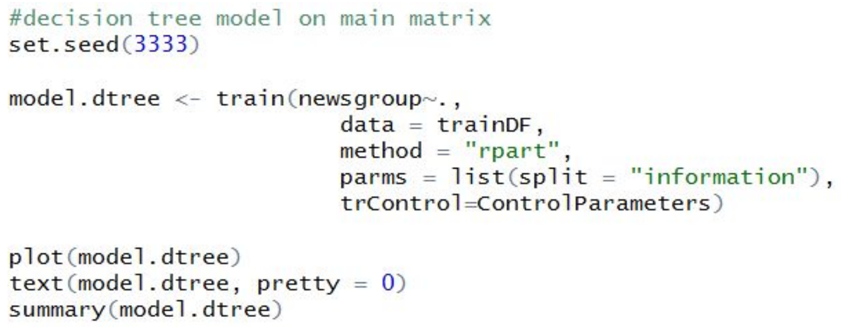


```
> cat(paste("F-measure:(%)", format(f * 100), "\n",sep=" "))
F-measure:(%) 66.47727
```



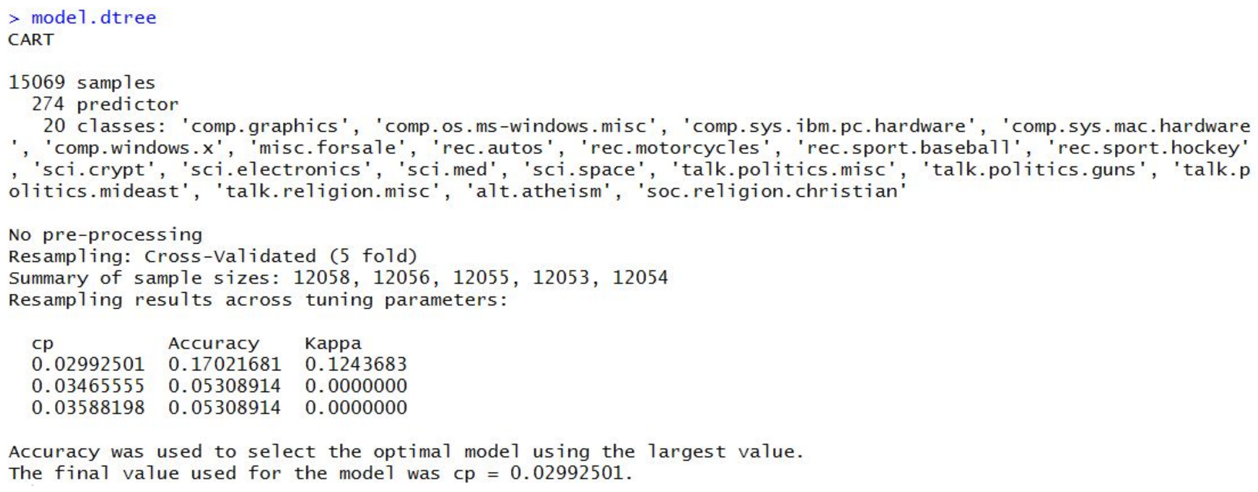
```
> data.frame(precision, recall, f1)
                         precision    recall        f1
comp.graphics            0.6030928 0.7405063 0.6647727
comp.os.ms-windows.misc  0.8020305 0.8315789 0.8165375
comp.sys.ibm.pc.hardware 0.6173469 0.7908497 0.6934097
comp.sys.mac.hardware    0.7187500 0.7624309 0.7399464
comp.windows.x           0.6530612 0.7529412 0.6994536
misc.forsale             0.8814433 0.8181818 0.8486352
rec.autos                0.7676768 0.8172043 0.7916667
rec.motorcycles          0.8939394 0.9030612 0.8984772
rec.sport.baseball       0.8232323 0.7912621 0.8069307
rec.sport.hockey         0.8743719 0.9157895 0.8946015
sci.crypt                0.8282828 0.9879518 0.9010989
sci.electronics          0.8877551 0.3587629 0.5110132
sci.med                  0.7373737 0.7934783 0.7643979
sci.space                0.7766497 0.8947368 0.8315217
talk.politics.misc       0.8258065 0.8648649 0.8448845
talk.politics.guns       0.8406593 0.9329268 0.8843931
talk.politics.mideast    0.8244681 0.9451220 0.8806818
talk.religion.misc       0.6880000 0.7350427 0.7107438
alt.atheism              0.8050314 0.8533333 0.8284790
soc.religion.christian   0.7939698 0.9239766 0.8540541
>
```

Implement decision tree model with 5-fold cross validation before feature selection:

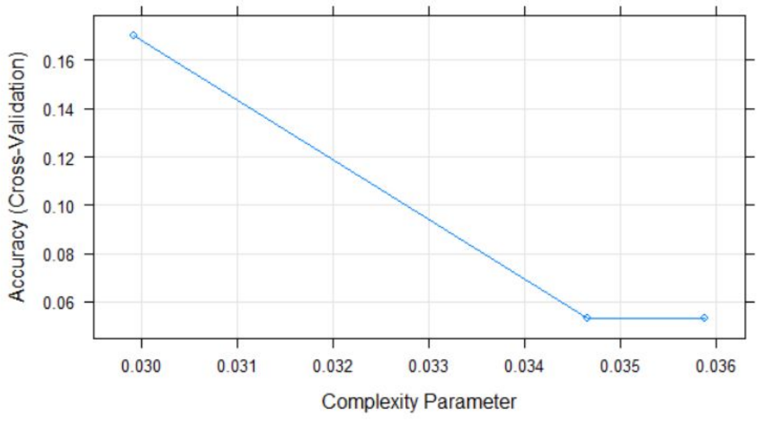


```
#decision tree model on main matrix
set.seed(3333)

model.dtree <- train(newsgroup~.,
                         data = trainDF,
                         method = "rpart",
                         parms = list(split = "information"),
                         trControl=ControlParameters)

plot(model.dtree)
text(model.dtree, pretty = 0)
summary(model.dtree)
```

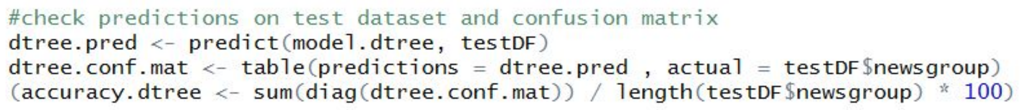
Output: Accuracy with different complexity parameter



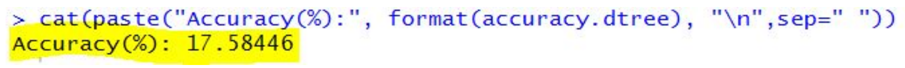
```
> model.dtree
CART

15069 samples
  274 predictor
   20 classes: 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware
', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey'
, 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'talk.politics.misc', 'talk.politics.guns', 'talk.p
olitics.mideast', 'talk.religion.misc', 'alt.atheism', 'soc.religion.christian'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12058, 12056, 12055, 12053, 12054
Resampling results across tuning parameters:

  cp          Accuracy    Kappa
  0.02992501  0.17021681  0.1243683
  0.03465555  0.05308914  0.0000000
  0.03588198  0.05308914  0.0000000

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.02992501.
```



Check Predictions on the test dataset:

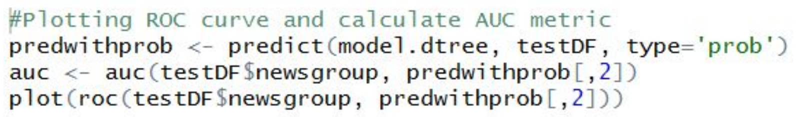


```
#check predictions on test dataset and confusion matrix
dtree.pred <- predict(model.dtree, testDF)
dtree.conf.mat <- table(predictions = dtree.pred , actual = testDF$newsgroup)
(accuracy.dtree <- sum(diag(dtree.conf.mat)) / length(testDF$newsgroup) * 100)
```



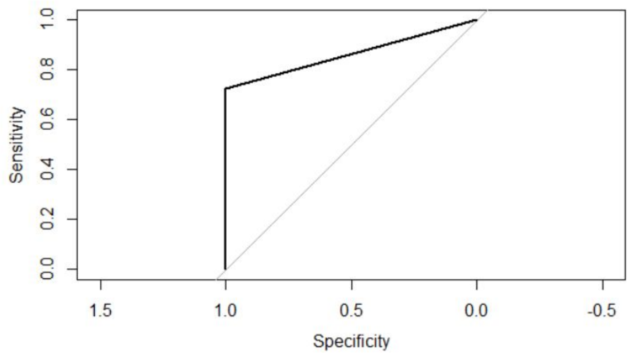
```
> cat(paste("Accuracy(%):", format(accuracy.dtree), "\n",sep=" "))
Accuracy(%): 17.58446
```

Plotting ROC curve:

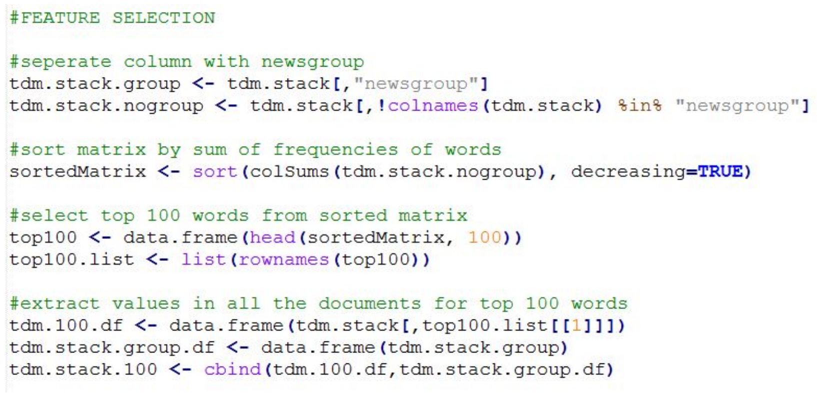


```
#Plotting ROC curve and calculate AUC metric
predwithprob <- predict(model.dtree, testDF, type='prob')
auc <- auc(testDF$newsgroup, predwithprob[,2])
plot(roc(testDF$newsgroup, predwithprob[,2]))
```

The curve with the bold line is far from diagonal line, so this model is behaving better. However,it is not completely towards top-left corner so it is not the best.

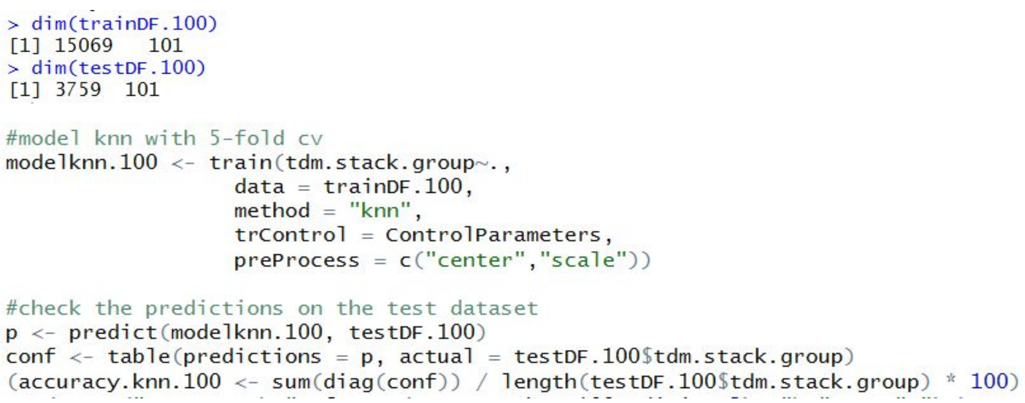


3.b. Implement knn model with 5-fold cross validation after feature selection:



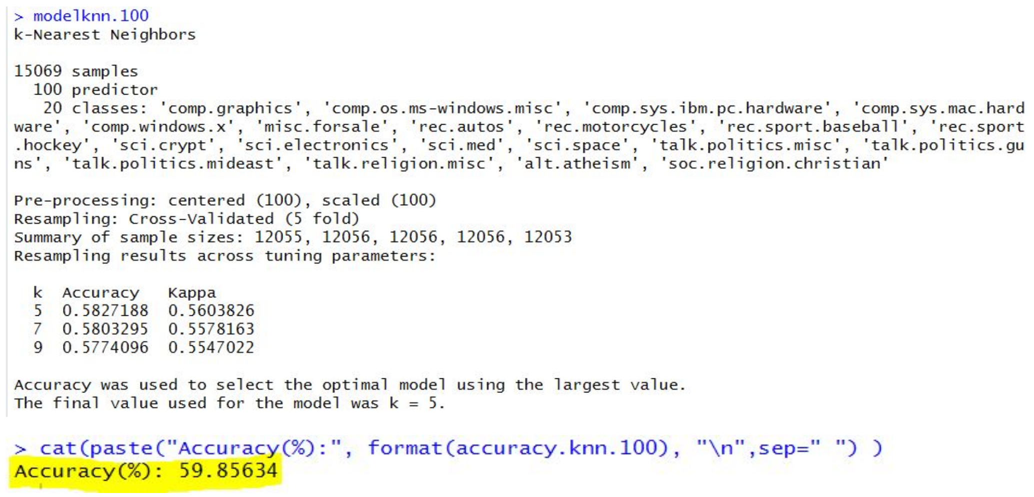
```
#FEATURE SELECTION

#seperate column with newsgroup
tdm.stack.group <- tdm.stack[,"newsgroup"]
tdm.stack.nogroup <- tdm.stack[,!colnames(tdm.stack) %in% "newsgroup"]

#sort matrix by sum of frequencies of words
sortedMatrix <- sort(colSums(tdm.stack.nogroup), decreasing=TRUE)

#select top 100 words from sorted matrix
top100 <- data.frame(head(sortedMatrix, 100))
top100.list <- list(rownames(top100))

#extract values in all the documents for top 100 words
tdm.100.df <- data.frame(tdm.stack[,top100.list[[1]]])
tdm.stack.group.df <- data.frame(tdm.stack.group)
tdm.stack.100 <- cbind(tdm.100.df,tdm.stack.group.df)
```

Dimensions of test and train dataset with less number of features(100):



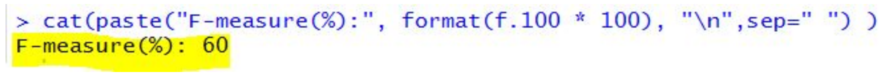
```
> dim(trainDF.100)
[1] 15069   101
> dim(testDF.100)
[1] 3759  101

#model knn with 5-fold cv
modelknn.100 <- train(tdm.stack.group~.,
                     data = trainDF.100,
                     method = "knn",
                     trControl = ControlParameters,
                     preProcess = c("center","scale"))

#check the predictions on the test dataset
p <- predict(modelknn.100, testDF.100)
conf <- table(predictions = p, actual = testDF.100$tdm.stack.group)
(accuracy.knn.100 <- sum(diag(conf)) / length(testDF.100$tdm.stack.group) * 100)
```

Output : Accuracy for different values of k. The best value of **k = 5**

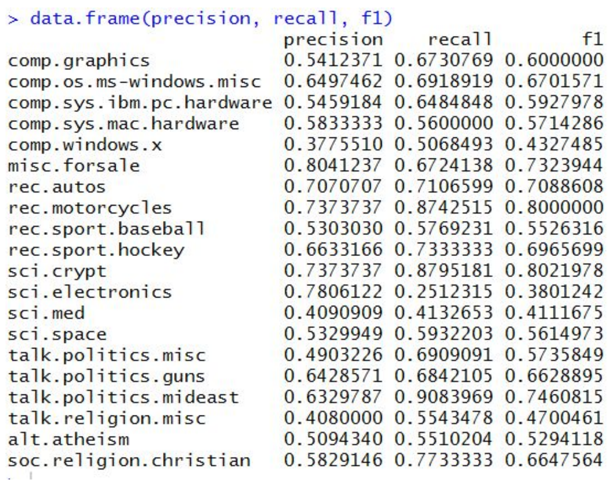


```
> modelknn.100
k-Nearest Neighbors

15069 samples
  100 predictor
   20 classes: 'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hard
ware', 'comp.windows.x', 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport
.hockey', 'sci.crypt', 'sci.electronics', 'sci.med', 'sci.space', 'talk.politics.misc', 'talk.politics.gu
ns', 'talk.politics.mideast', 'talk.religion.misc', 'alt.atheism', 'soc.religion.christian'

Pre-processing: centered (100), scaled (100)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 12055, 12056, 12056, 12056, 12053
Resampling results across tuning parameters:

  k  Accuracy   Kappa
  5  0.5827188  0.5603826
  7  0.5803295  0.5578163
  9  0.5774096  0.5547022

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.

> cat(paste("Accuracy(%):", format(accuracy.knn.100), "\n",sep=" ") )
Accuracy(%): 59.85634
```

Overall f-measure and f-measures for each article category:

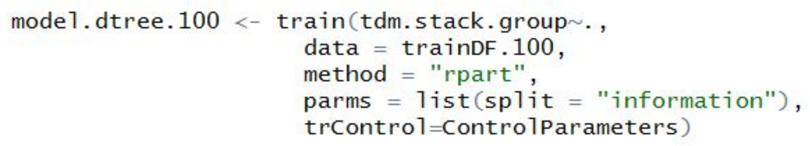


```
> cat(paste("F-measure(%):", format(f.100 * 100), "\n",sep=" ") )
F-measure(%): 60
```

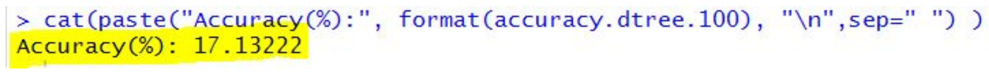


```
> data.frame(precision, recall, f1)
                         precision    recall        f1
comp.graphics            0.5412371 0.6730769 0.6000000
comp.os.ms-windows.misc  0.6497462 0.6918919 0.6701571
comp.sys.ibm.pc.hardware 0.5459184 0.6484848 0.5927978
comp.sys.mac.hardware    0.5833333 0.5600000 0.5714286
comp.windows.x           0.3775510 0.5068493 0.4327485
misc.forsale             0.8041237 0.6724138 0.7323944
rec.autos                0.7070707 0.7106599 0.7088608
rec.motorcycles          0.7373737 0.8742515 0.8000000
rec.sport.baseball       0.5303030 0.5769231 0.5526316
rec.sport.hockey         0.6633166 0.7333333 0.6965699
sci.crypt                0.7373737 0.8795181 0.8021978
sci.electronics          0.7806122 0.2512315 0.3801242
sci.med                  0.4090909 0.4132653 0.4111675
sci.space                0.5329949 0.5932203 0.5614973
talk.politics.misc       0.4903226 0.6909091 0.5735849
talk.politics.guns       0.6428571 0.6842105 0.6628895
talk.politics.mideast    0.6329787 0.9083969 0.7460815
talk.religion.misc       0.4080000 0.5543478 0.4700461
alt.atheism              0.5094340 0.5510204 0.5294118
soc.religion.christian   0.5829146 0.7733333 0.6647564
```

Implement decision tree model with 5-fold cross validation after feature selection:



```
model.dtree.100 <- train(tdm.stack.group~.,
                         data = trainDF.100,
                         method = "rpart",
                         parms = list(split = "information"),
                         trControl=ControlParameters)
```

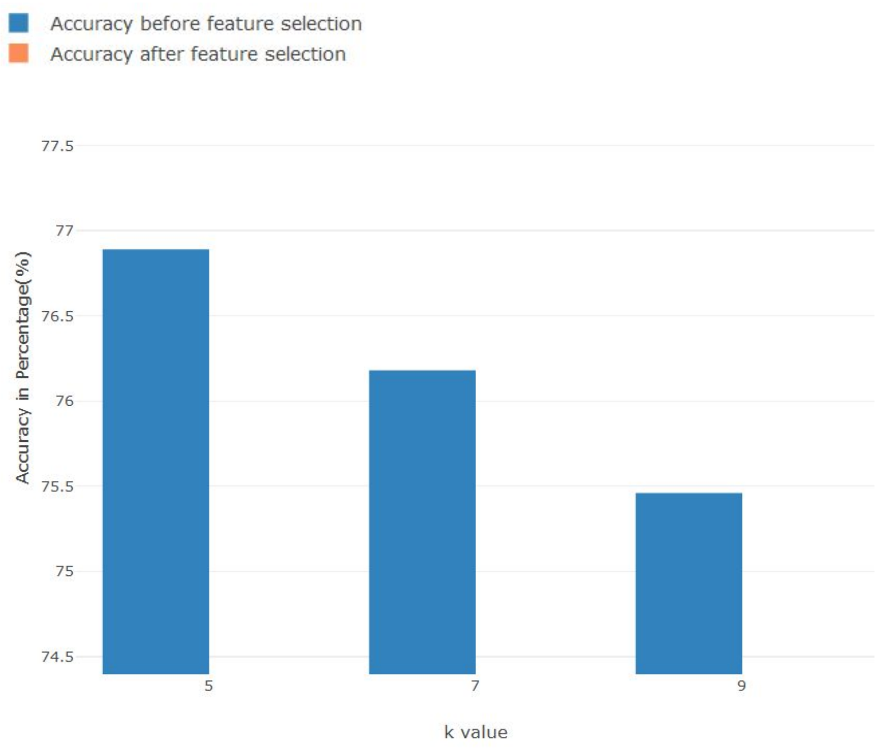


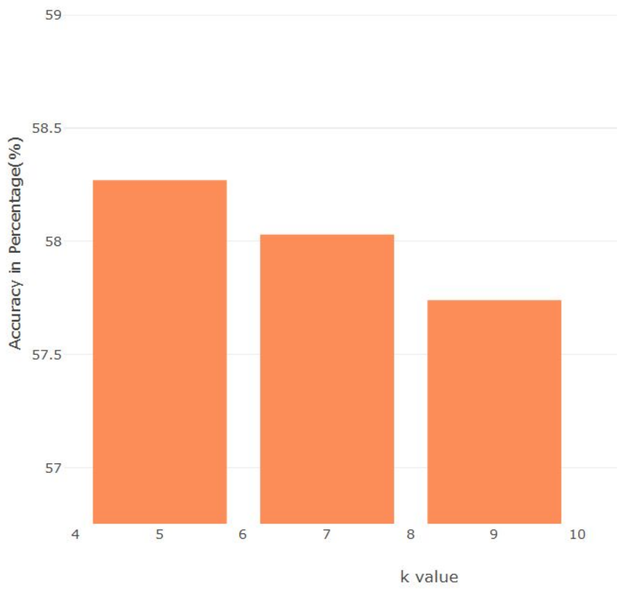
```
> cat(paste("Accuracy(%):", format(accuracy.dtree.100), "\n",sep=" ") )
Accuracy(%): 17.13222
```

Conclusion:

Thus, observing overall accuracies and f-measures before and after feature selection, we can see that feature selection doesn't help improve the accuracies of classifiers. Accuracies of knn and decision tree classifiers decrease after feature selection.

4. Evaluate how K impacts the overall accuracy of kNN on the dataset.

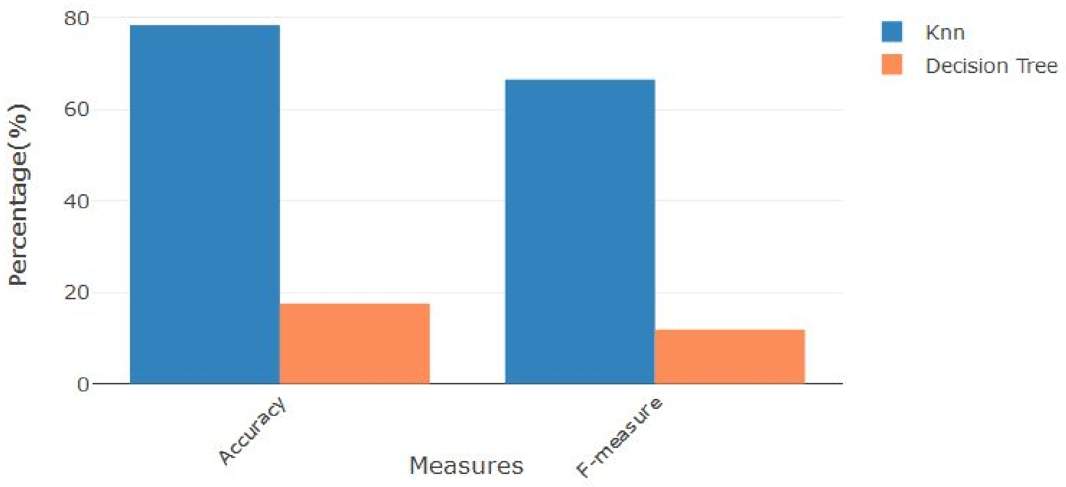




From these plots, we can see overall accuracy of knn on the dataset with different values of k. In both the cases, before feature selection and after feature selection, the best value of k is 5 as it gives high accuracy. As k tends to increase, accuracy of knn tends to decrease.

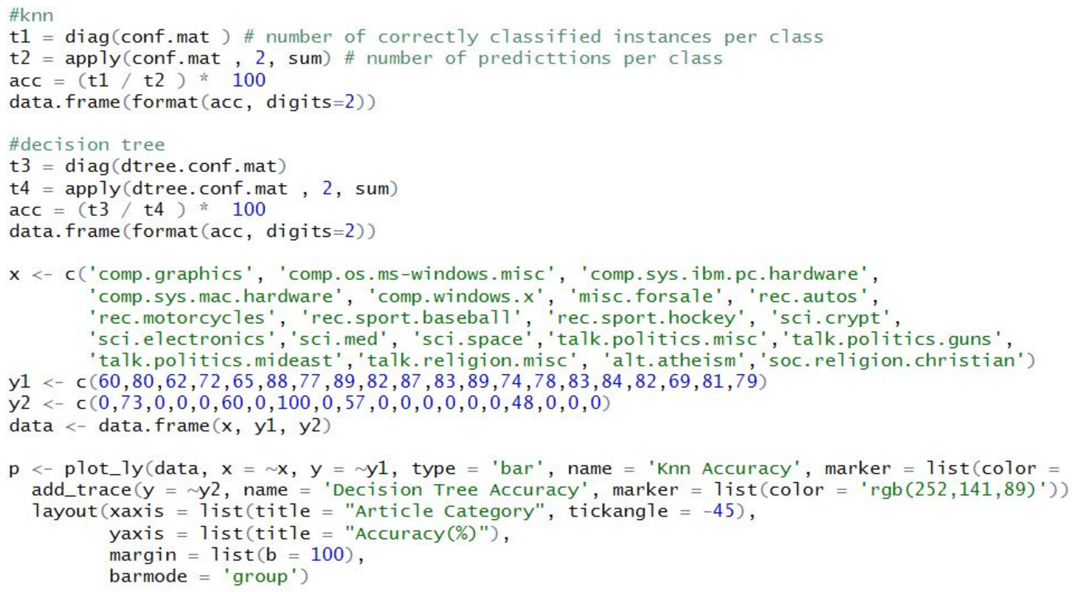
5. Compare the overall accuracies and f-measures of kNN with best K and decision trees.

From the overall accuracies (before feature selection) highlighted above and with computed f-measures (as per code attached) for both the classifiers, below is the plot showing their comparison. For knn, k = 5.

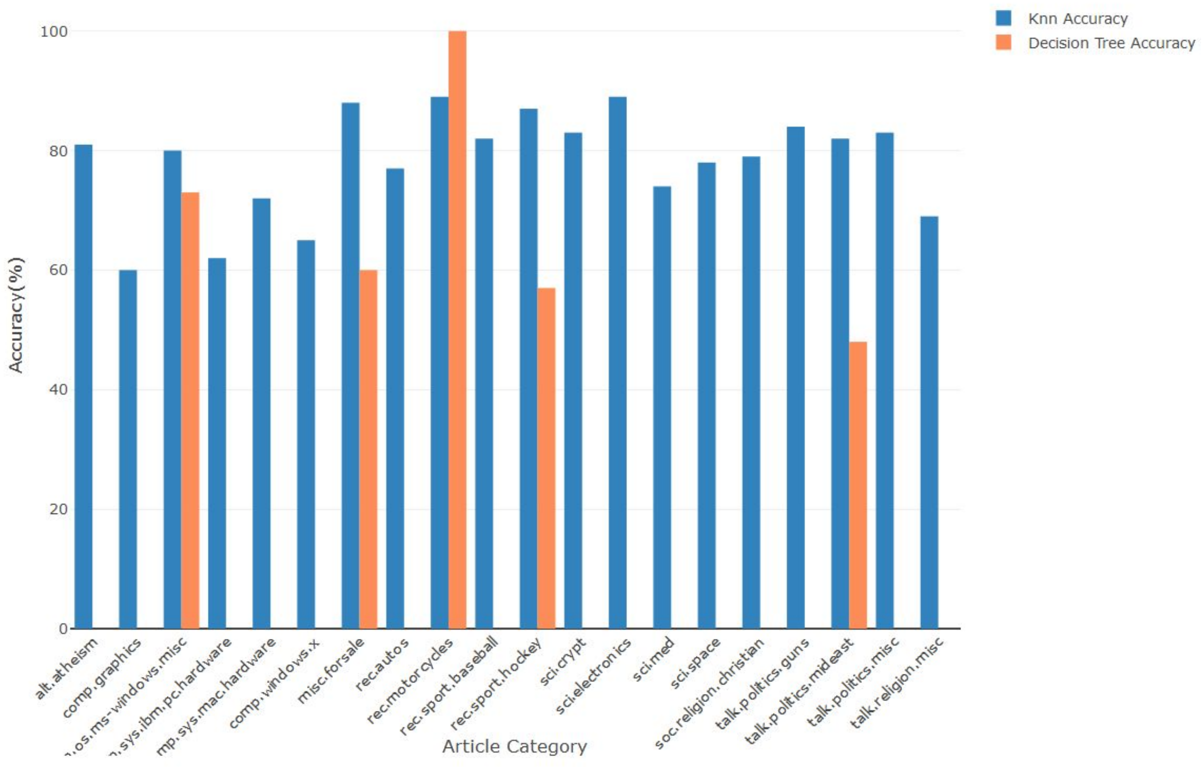


We can see that knn gives high accuracy and f-measure percentage. Hence, knn is better compared to decision tree classifier as it can classify the news article categories well based on the frequency of words in the documents.

6. Compare the accuracies of each article category of kNN with best K and decision trees.



```
#knn
t1 = diag(conf.mat ) # number of correctly classified instances per class
t2 = apply(conf.mat , 2, sum) # number of predicttions per class
acc = (t1 / t2 ) *  100
data.frame(format(acc, digits=2))

#decision tree
t3 = diag(dtree.conf.mat)
t4 = apply(dtree.conf.mat , 2, sum)
acc = (t3 / t4 ) *  100
data.frame(format(acc, digits=2))

x <- c('comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware',
       'comp.sys.mac.hardware', 'comp.windows.x', 'misc.forsale', 'rec.autos',
       'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey', 'sci.crypt',
       'sci.electronics','sci.med', 'sci.space','talk.politics.misc','talk.politics.guns',
       'talk.politics.mideast','talk.religion.misc', 'alt.atheism','soc.religion.christian')
y1 <- c(60,80,62,72,65,88,77,89,82,87,83,89,74,78,83,84,82,69,81,79)
y2 <- c(0,73,0,0,0,60,0,100,0,57,0,0,0,0,0,0,48,0,0,0)
data <- data.frame(x, y1, y2)

p <- plot_ly(data, x = ~x, y = ~y1, type = 'bar', name = 'Knn Accuracy', marker = list(color =
  add_trace(y = ~y2, name = 'Decision Tree Accuracy', marker = list(color = 'rgb(252,141,89)'))
  layout(xaxis = list(title = "Article Category", tickangle = -45),
         yaxis = list(title = "Accuracy(%)"),
         margin = list(b = 100),
         barmode = 'group')
```



As per the graph, we can see for a particular article category, knn classifier with best k is better or decision tree classifier is better. For the maximum categories, knn gives high accuracy compared to decision tree.

7. In general, which classifier is better?

From the previous observations, knn (with k=5) classifier gives highest accuracy (78%), while decision tree classifier gives low accuracy (18%). Additionally, when we observe for particular category, knn predicts the best classification. The knn also computes high f-measure(66%). Therefore, knn is better for our dataset as decision tree has high classification error rate.

However, the main difference is about the domain of application. Knn is used for continuous value inputs, unlike Decision Trees that is applicable for continuous and categorical inputs. Also, Decision trees are white boxes in the sense that the acquired knowledge can be expressed in a readable form(tree), while KNN is generally black box means you cannot read the acquired knowledge in a comprehensible way.