Text Mining – Document Classification

Data Mining II

Josef Bušta

Porto, 2013

Contents

Introduction **3**

Preprocessing **4**
- Used techniques 4
- Dataset 5

Classification **6**
- Decision Trees (DT) 6
- Neural Networks (NN) 6
- Naive Bayes (NB) 7
- K-Nearest Neighbours (KNN) 7
- Support Vector Machines (SVM) 7

Results **8**

Conclusions **12**

Appendix **14**

Introduction

The aim of this work is to construct and compare several classifiers that perform document classification into two categories. For the classification are used data from Reuters and the classifiers are constructed using the system *R*.

This text is divided into 4 chapters, excluding this one. Chapter 2 contains a description of chosen datasets including their main characteristics and information about techniques used to pre-process the documents.

The description and some features of used classifiers are briefly presented in Chapter 3.

Chapter 4 includes the results of performed experiments and their comparison. The results are presented in the tables along with a short commentary of interesting values.

Chapter 5 includes the main conclusions of this assignment.

In Appendix is attached used source code for *R*.

Preprocessing

The preprocessing phase of the text classification converts the original textual data into a representation suitable for classification algorithm, where the most significant text-features are subsequently identified. This phase is the most critical and complex process that leads to the representation of each document by a select set of index terms. The main objective of preprocessing is to obtain the key features or key terms from text documents and to enhance the relevancy between word and document and the relevancy between word and category [10]. The text data are usually characterized by high sparsity, hence a reduction of the dimension or feature selection is often necessary to execute learning algorithm considering time complexity [13].

Used techniques

To improve outcomes are used following preprocessing techniques from the library *tm* in the system *R*.

- Strip white spaces
- Convert words to lowercase
- Remove stopwords – frequent words that carry no interesting information (i.e pronouns, preposition, etc.) [5]
- Remove punctuation
- Remove numbers
- Find frequent terms

The last step is performed for several values to see how the accuracy, recall and precision score is changed depending on the number of features and considering also time complexity.

Dataset

In this assignment is used the simplified Reuters-21578 dataset (prepared by J. Knotek). The simplified Reuters-21578 dataset contains seven categories (coffee, crude, grain, interest, money-fx, sugar and wheat) and is organized into folders, where each folder corresponds to one category and each document of the category is in one file.

Since this assignment requires classification into two categories it was necessary to chose exactly two types of documents. The chosen categories for the experiments are *wheat* and *crude*. The category *wheat* contains 208 and the category *crude* contains 334 text documents.

The documents in each category are split into a training set (70%) and into a testing set (30%). The *crude* training and testing sets include 233 and 101 documents, respectively. The *wheat* has 145 training documents and 63 testing documents.

In the following table is shown how the count of terms is changed after particular steps of preprocessing. The sparsity after application each of the algorithm indicated in the table was 99%.

Algorithm	**Number of terms**	**Sparse entries**	**Non-sparse entries**
Before preprocessing	12959	6963012	60766
Strip white spaces	12959	6963012	60766
Convert words to lowercase	12959	6963012	60766
Remove stop words	12131	6530004	44998
Remove punctuation	8512	4572836	40668
Remove numbers	7144	3834160	37888

Classification

The goal of text categorization is to classify documents into a certain number of predefined categories [12]. Text classification has many applications, e.g. spam filtering, document organization and retrieval, or news filtering and organization [11].

The main ideas of algorithms for text classification available in *R* packages and used in this assignment are following.

Decision Trees (DT)

- *R* library *rpart*

The classification tree is built to predict category for input document. It works by recursively splitting the feature space into a set of non-overlapping regions (subsets of values), and by then predicting the most likely value of the dependent variable within each region. A classification tree represents a set of nested logical if-then conditions on the values of the features variables that allows for the prediction of the value of the dependent categorical variable based on the observed values of the feature variables [4].

Neural Networks (NN)

- *R* library *nnet*

The package *nnet* provides methods for using feed-forward neural networks with a single hidden layer, and for multinomial log-linear models [2].

A feed-forward neural network is an artificial neural network where connections between the units do not form a directed cycle. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network [1].

Naive Bayes (NB)

- *R* library *RWeka*

The Naive Bayes classifier is constructed by using the training data to estimate the probability of each class given the document feature values of a new instance [5].

K-Nearest Neighbours (KNN)

- *R* library *class*

For each row of the test set, the K nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the Kth nearest vector, all candidates are included in the vote [3].

Support Vector Machines (SVM)

- *R* library *e1071*

The main principle of SVMs is to determine separators in the search space which can best separate the different classes. The key in such classifiers is to determine the optimal boundaries between the different classes and use them for the purposes of classification [11].

Results

In this chapter are presented the experimental results which were measured using individual algorithms.

For the R command: $info.terms < -findFreqTerms(dtm.mx, 200)$; the count of terms is 20, as shown below.

```
 [1] "barrels" "billion" "bpd" "crude" "dlrs"
 [6] "energy" "gas" "march" "market" "mln"
[11] "oil" "opec" "pct" "price" "prices"
[16] "production" "reuter" "the" "tonnes" "wheat"
```

Furthermore the decision tree for these terms is displayed.

```
n= 378

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 378 145 crude (0,61640212 0,38359788)
 2) wheat.t< 0,5 236   5 crude (0,97881356 0,02118644) *
 3) wheat.t>=0,5 142   2 wheat (0,01408451 0,98591549) *
```

If the terms *wheat* and *crude* are removed the F1 score is decreased to 0.9547739 (the original score is 0,9853659, if all of the terms are taken into account), but the richer tree is obtained. The tree shows more terms, which are also important to distinguish chosen categories.

```
n= 378

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 378 145 crude (0.61640212 0.38359788)
 2) oil.t>=0.5 222   9 crude (0.95945946 0.04054054) *
 3) oil.t< 0.5 156  20 wheat (0.12820513 0.87179487)
  6) reuter.t< 0.5 30  13 crude (0.56666667 0.43333333)
   12) prices.t>=0.5 8   0 crude (1.00000000 0.00000000) *
   13) prices.t< 0.5 22   9 wheat (0.40909091 0.59090909)
    26) mln.t>=0.5 7   2 crude (0.71428571 0.28571429) *
    27) mln.t< 0.5 15   4 wheat (0.26666667 0.73333333) *
  7) reuter.t>=0.5 126   3 wheat (0.02380952 0.97619048) *
```

The following table displays results for previously mentioned 20 attributes. The best results are achieved using Decision Tree algorithm, the Neural Networks achieve also very high score, while the worst results achieve Naive Bayes. The difference between DT and NN is that NN have the best possible precision, while recall obtains worser value, but almost exactly opposite values are achieved using Decision Tree.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
NB	0,8658537	0,9759036	0,8019802	0,8804348
SVM	0,9512195	0,9428571	0,980198	0,961165
KNN	0,9512195	0,960396	0,960396	0,960396
NN	0,9817073	1	0,970297	0,9849246
DT	0,9817073	0,9711538	1	**0,9853659**

For the R command: $info.terms < -findFreqTerms(dtm.mx, 150)$; the count of terms is 38.

Increasing the number of attributes are improved precision of NB and recall of SVM, while precision of SVM has worse score.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
NB	0,8780488	1	0,8019802	0,8901099
SVM	0,945122	0,9181818	1	0,957346
KNN	0,9695122	0,98	0,970297	0,9751244
NN	0,9756098	1	0,960396	0,979798
DT	0,9817073	0,9711538	1	**0,9853659**

For the R command: $info.terms < -findFreqTerms(dtm.mx, 100)$; the count of terms is 72.

Next increasing of terms used in the classification improves score of NN, especially the recall and consequently also the accuracy and F1 score. Furthermore the score of NB is slightly improved and the precision and recall of KNN decline in comparison with previous table. The values of DT remains the same.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
NB	0,8963415	1	0,8316832	0,9081081
SVM	0,9268293	0,8938053	1	0,9439252
KNN	0,9573171	0,97	0,960396	0,9651741
NN	0,9939024	1	0,990099	**0,9950249**
DT	0,9817073	0,9711538	1	0,9853659

For the R command: $info.terms < -findFreqTerms(dtm.mx, 40)$; the count of terms is 265.

The following table displays a significant increase of recall of NB and greater decline of precision of SVM.

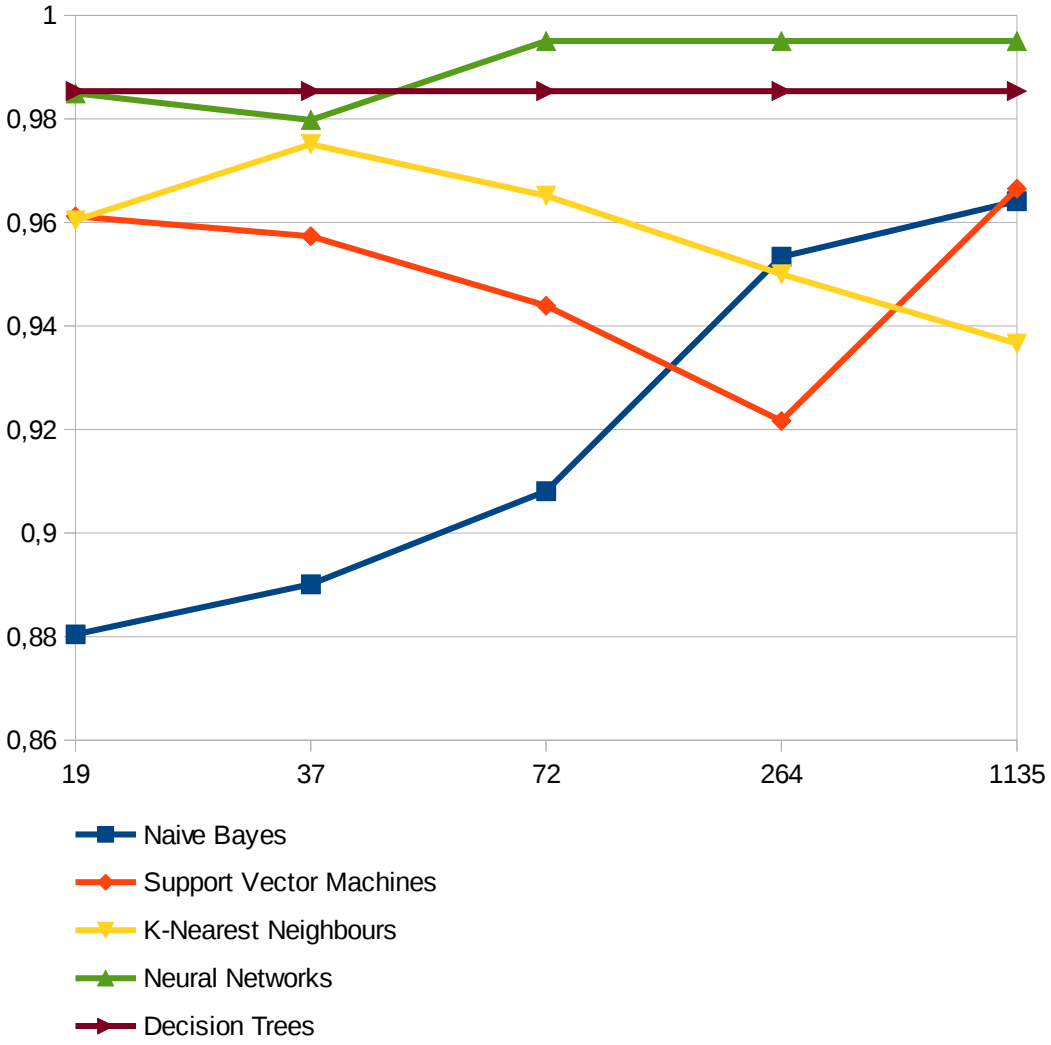
Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
NB	0,945122	1	0,9108911	0,9533679
SVM	0,8963415	0,862069	0,990099	0,921659
KNN	0,9390244	0,959596	0,9405941	0,95
NN	0,9939024	1	0,990099	**0,9950249**
DT	0,9817073	0,9711538	1	0,9853659

For the R command: $info.terms < -findFreqTerms(dtm.mx, 10)$; the count of terms is 1136.

In the last table are presented results for 1136 terms. The time taken for training was appreciably longer than in previous experiments. The score of NB and SVM is increased, while the score of KNN is decreased.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
NB	0,9573171	1	0,9306931	0,9641026
SVM	0,9573171	0,9351852	1	0,9665072
KNN	0,9207317	0,9230769	0,950495	0,9365854
NN	0,9939024	1	0,990099	**0,9950249**
DT	0,9817073	0,9711538	1	0,9853659

The graph displayed on the next page compares achieved F1 scores of all tested algorithms depending on the number of terms.


1
0,98
0,96
0,94
0,92
0,9
0,88
0,86
19
37
72
264
1135
Naive Bayes
Support Vector Machines
K-Nearest Neighbours
Neural Networks
Decision Trees

Conclusions

The goal of this work was the construction of several classifiers and their comparison. At first the input data were loaded into the R and consequently were applied the first 5 preprocessing methods and the resulting data were divided into train and test sets. On such prepared data were launched functions to rename terms to avoid conflicts with R commands. Obtained data were several times used for processing by command for identification of frequent terms and by commands for training, testing and evaluation models. After all that processing the presented results were obtained.

The results indicate that Naive Bayes works better with a larger number of attributes and is not able to take advantage from small amount of informative terms, while for K-Nearest Neighbours a larger number of attributes is disadvantage, if there is small amount of informative terms.

Decision Trees are able to detect informative terms in the small or large amount of terms.

Neural Networks behaves very well as with a small number of attributes, so with a large number of attributes, however it seem that too small amount of terms, is not the best choice for this algorithms.

Support Vector Machines achieved the best score in the experiment with the biggest number of attributes although relatively high scores was also with small amount of attributes.

Bibliography

[1] Y. Singh, A. S. Chauhan. *Neural Networks in Data Mining.* Journal of Theoretical and Applied Information Technology, 2005 - 2009.

[2] Brian Ripley. *Feed-forward Neural Networks and Multinomial Log-Linear Models.* July 1, 2013.

[3] Shengqiao Li. *Fast Nearest Neighbor Search Algorithms and Applications.* May 19, 2013.

[4] Breiman L., Friedman J. H., Olshen R. A., and Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[5] K. Aas and A. Eikvi. *Text categorization: A survey.* Technical report, Norwegian Computing Center, 1999.

[6] Y. Zhao. *R and Data Mining: Examples and Case Studies.* Elsevier, December 2012.

[7] P. Brazdil. *Classification of Documents using Text Mining Package "tm".* F. Letras, UP, 2009.

[8] I. Feinerer. *Introduction to the tm Package Text Mining in R.* 2013.

[9] P. Brazdil. *Introduction to Text Mining.* F. Letras, UP, 2009.

[10] V. Srividhya, R. Anitha. *Evaluating Preprocessing Techniques in Text Categorization.* International Journal of Computer Science and Application Issue 2010.

[11] Charu C. Aggarwal, ChengXiang Zhai. *Mining Text Data – A Survey of Text Classification Algorithms.* Springer US, 2012.

[12] Youngjoong Ko, Jungyun Seo. *Automatic Text Categorization by Unsupervised Learning.* 2000.

[13] M. Ikonomakis, S. Kotsiantis, V. Tampakas. *Text Classification Using Machine Learning Techniques.* 2005.

Appendix

R source code:

```
library(tm);

# Get the corpus for "wheat" and "crude" documents
wheat <- Corpus(DirSource("wheat"), readerControl=list(
    reader=readPlain,language="en_US"));
crude <- Corpus(DirSource("crude"), readerControl=list(
    reader=readPlain,language="en_US"));

# Get corresponding train and test sets (70% and 30%
    respectively)
lg <- length(wheat);
wheat.train <- wheat[1:as.integer(lg*0.7)];
wheat.test <- wheat[(as.integer(lg*0.7)+1):lg];
lg <- length(crude);
crude.train <- crude[1:as.integer(lg*0.7)];
crude.test <- crude[(as.integer(lg*0.7)+1):lg];
l1 <- length(wheat.train);
l2 <- length(crude.train);
l3 <- length(wheat.test);
l4 <- length(crude.test);

# Merge corpora into one collection
docs <- c(wheat.train, crude.train, wheat.test, crude.test);

# Pre-processing
docs.p <- docs;
docs.p <- tm_map(docs.p, stripWhitespace);
docs.p <- tm_map(docs.p, tolower);
docs.p <- tm_map(docs.p, removeWords, stopwords("en"));
docs.p <- tm_map(docs.p, removePunctuation);
docs.p <- tm_map(docs.p, removeNumbers);

# Create a Document-Term matrix
dtm.mx <- DocumentTermMatrix(docs.p, control=list(weighting=
    weightTf));

# Convert the Document-Term matrix into a data frame and
    append class information
```

```
dtm <- as.data.frame(inspect(dtm.mx));
rownames(dtm)<- 1:nrow(dtm.mx);
class <- c(rep("wheat",l1), rep("crude",l2), rep("wheat",l3)
    , rep("crude",l4));
dtm <- cbind(dtm, class);
last.col <- length(dtm);

# Prepare data for training and testing the classifier
dtm.tr <- dtm[1:(l1+l2), 1:last.col];
dtm.ts <- dtm[(l1+l2+1):(l1+l2+l3+l4),1:(last.col-1)];

rename.terms.in.dtm <- function(dtm) {
    for (i in 1:length(dtm)) {
        cat("replaced to ", paste(colnames(dtm)[i],".t", sep
            =""), "\n")
        colnames(dtm)[i] <- paste(colnames(dtm)[i],".t", sep
            ="")
    } #end for i
return(dtm)
}

dtm.tr <- rename.terms.in.dtm(dtm.tr);
dtm.ts <- rename.terms.in.dtm(dtm.ts);

# Identification of informative terms
info.terms <- findFreqTerms(dtm.mx, 10);
cat("Number of features:", length(info.terms) - 1, "\n");

rename.terms.in.list <- function(list) {
    for (i in 1:length(list)) {
        #cat("replaced", list[i], "at", i, "with", paste(
            list[i],".t", sep=""), "\n")
            list[i]<- paste(list[i],".t", sep="")
    } #end for i
#   list <- list[list != "crude.t"];
#   list <- list[list != "wheat.t"];
    return(list)
}

# Rename terms to avoid errors in classification
info.terms <- rename.terms.in.list(info.terms);

# Train the classifier
names.tr <- paste(info.terms, collapse='+');
class.formula <- as.formula(paste('class.t', names.tr, sep='
    ~'));

class.ts <- dtm[(l1+l2+1):(l1+l2+l3+l4),last.col];
class.tr <- dtm[1:(l1+l2),last.col];

library(rpart)
dt <- rpart(class.formula, dtm.tr)
#Prediction
preds.dt <- predict(dt, dtm.ts, type="class")
```

```
#Construct a confusion matrix:
conf.mx.dt <- table(class.ts, preds.dt)
#Percentage of prediction errors
error.rate.dt <- (sum(conf.mx.dt) - sum(diag(conf.mx.dt))) /
     sum(conf.mx.dt)
#Evaluation of classifier
tp.dt <- conf.mx.dt[1,1] #(true positive)
fp.dt <- conf.mx.dt[2,1] #(false positive)
tn.dt <- conf.mx.dt[2,2] #(true negative)
fn.dt <- conf.mx.dt[1,2] #(false negative)
recall.dt = tp.dt / (tp.dt + fn.dt)
precision.dt = tp.dt / (tp.dt + fp.dt)
f1.dt = 2 * precision.dt * recall.dt / (precision.dt +
    recall.dt)
accuracy.dt = (tp.dt + tn.dt) / (tp.dt + fp.dt + tn.dt + fn.
    dt)

library(nnet)
nnet.classifier <- nnet(class.formula, data = dtm.tr, size
    =2, rang=0.1,decay=5e-4, maxit=200)
preds.nn <- predict(nnet.classifier, dtm.ts, type="class")
conf.mx.nn <- table(class.ts, preds.nn)
error.rate.nn <- (sum(conf.mx.nn) - sum(diag(conf.mx.nn))) /
     sum(conf.mx.nn)
#Evaluation of classifier
tp.nn <- conf.mx.nn[1,1] #(true positive)
fp.nn <- conf.mx.nn[2,1] #(false positive)
tn.nn <- conf.mx.nn[2,2] #(true negative)
fn.nn <- conf.mx.nn[1,2] #(false negative)
recall.nn = tp.nn / (tp.nn + fn.nn)
precision.nn = tp.nn / (tp.nn + fp.nn)
f1.nn = 2 * precision.nn * recall.nn / (precision.nn +
    recall.nn)
accuracy.nn = (tp.nn + tn.nn) / (tp.nn + fp.nn + tn.nn + fn.
    nn)

library(class)
preds.knn <- knn(dtm.tr[, info.terms], dtm.ts[, info.terms],
     class.tr, k=1)
conf.mx.knn <- table(class.ts, preds.knn)
error.rate.knn <- (sum(conf.mx.knn) - sum(diag(conf.mx.knn))
    ) / sum(conf.mx.knn)
#Evaluation of classifier
tp.knn <- conf.mx.knn[1,1] #(true positive)
fp.knn <- conf.mx.knn[2,1] #(false positive)
tn.knn <- conf.mx.knn[2,2] #(true negative)
fn.knn <- conf.mx.knn[1,2] #(false negative)
recall.knn = tp.knn / (tp.knn + fn.knn)
precision.knn = tp.knn / (tp.knn + fp.knn)
f1.knn = 2 * precision.knn * recall.knn / (precision.knn +
    recall.knn)
accuracy.knn = (tp.knn + tn.knn) / (tp.knn + fp.knn + tn.knn
     + fn.knn)
```

```
library(e1071)
svm.classifier <- svm(class.formula, dtm.tr)
preds.svm <- predict(svm.classifier, dtm.ts)
conf.mx.svm <- table(class.ts, preds.svm)
error.rate.svm <- (sum(conf.mx.svm) - sum(diag(conf.mx.svm))
    ) / sum(conf.mx.svm)
#Evaluation of classifier
tp.svm <- conf.mx.svm[1,1] #(true positive)
fp.svm <- conf.mx.svm[2,1] #(false positive)
tn.svm <- conf.mx.svm[2,2] #(true negative)
fn.svm <- conf.mx.svm[1,2] #(false negative)
recall.svm = tp.svm / (tp.svm + fn.svm)
precision.svm = tp.svm / (tp.svm + fp.svm)
f1.svm = 2 * precision.svm * recall.svm / (precision.svm +
    recall.svm)
accuracy.svm = (tp.svm + tn.svm) / (tp.svm + fp.svm + tn.svm
     + fn.svm)

library(RWeka)
NB<-make_Weka_classifier("weka/classifiers/bayes/NaiveBayes"
    )
nb.classifier<-NB(class.formula, dtm.tr)
preds.nb<-predict(nb.classifier, dtm.ts)
conf.mx.nb<-table(class.ts, preds.nb)
error.rate.nb <- (sum(conf.mx.nb) - sum(diag(conf.mx.nb))) /
     sum(conf.mx.nb)
#Evaluation of classifier
tp.nb <- conf.mx.nb[1,1] #(true positive)
fp.nb <- conf.mx.nb[2,1] #(false positive)
tn.nb <- conf.mx.nb[2,2] #(true negative)
fn.nb <- conf.mx.nb[1,2] #(false negative)
recall.nb = tp.nb / (tp.nb + fn.nb)
precision.nb = tp.nb / (tp.nb + fp.nb)
f1.nb = 2 * precision.nb * recall.nb / (precision.nb +
    recall.nb)
accuracy.nb = (tp.nb + tn.nb) / (tp.nb + fp.nb + tn.nb + fn.
    nb)

cat("\nNB: Conf. Matrix:", conf.mx.nb, "\n", "Error rate: ",
     error.rate.nb, "\n", "Accuracy: ", accuracy.nb, "\n", "
    Precision: ", precision.nb, "\n", "Recall: ", recall.nb,
     "\n", "F1: ", f1.nb, "\n");

cat("\nSVM: Conf. Matrix:", conf.mx.svm, "\n", "Error rate:
    ", error.rate.svm, "\n", "Accuracy: ", accuracy.svm, "\n
    ", "Precision: ", precision.svm, "\n", "Recall: ",
    recall.svm, "\n", "F1: ", f1.svm, "\n");

cat("\nKNN: Conf. Matrix:", conf.mx.knn, "\n", "Error rate:
    ", error.rate.knn, "\n", "Accuracy: ", accuracy.knn, "\n
    ", "Precision: ", precision.knn, "\n", "Recall: ",
    recall.knn, "\n", "F1: ", f1.knn, "\n");
```

```
cat("\nNN: Conf. Matrix:", conf.mx.nn, "\n", "Error rate: ",
     error.rate.nn, "\n", "Accuracy: ", accuracy.nn, "\n", "
    Precision: ", precision.nn, "\n", "Recall: ", recall.nn,
     "\n", "F1: ", f1.nn, "\n");

cat("\nDT: Conf. Matrix:", conf.mx.dt, "\n", "Error rate: ",
     error.rate.dt, "\n", "Accuracy: ", accuracy.dt, "\n", "
    Precision: ", precision.dt, "\n", "Recall: ", recall.dt,
     "\n", "F1: ", f1.dt, "\n");
```