Sms Verification - Spam/Ham

Aravind

January 24, 2018

Loading Libraries

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(e1071)
require(tidyr)
```

```
## Loading required package: tidyr
```

```
library(NLP)
```

Reading data

```
spam_data<-read.csv("smsspam.csv",stringsAsFactors = F)
head(spam_data)
```

```
##     v1
## 1  ham
## 2  ham
## 3 spam
## 4  ham
## 5  ham
## 6 spam
##
v2
## 1                                             Go until jurong point, crazy.. Av
ailable only in bugis n great world la e buffet... Cine there got amore wat...
## 2
Ok lar... Joking wif u oni...
## 3 Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to
87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
## 4
U dun say so early hor... U c already then say...
## 5
Nah I don't think he goes to usf, he lives around here though
## 6        FreeMsg Hey there darling it's been 3 week's now and no word back! I'd
like some fun you up for it still? Tb ok! XxX std chgs to send, <U+00E5><U+00A3>1.5
0 to rcv
##   X X.1 X.2
## 1
## 2
## 3
## 4
## 5
## 6
```

remove the columns 3 to 5, which is not having any data, so can remove it

```
colnames(spam_data)<-c("label","text")
str(spam_data)
```

```
## 'data.frame':    5572 obs. of  5 variables:
##  $ label: chr  "ham" "ham" "spam" "ham" ...
##  $ text : chr  "Go until jurong point, crazy.. Available only in bugis n great w
orld la e buffet... Cine there got amore wat..." "Ok lar... Joking wif u oni..." "F
ree entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121
to receive entry question("| __truncated__ "U dun say so early hor... U c already t
hen say..." ...
##  $ NA   : chr  "" "" "" "" ...
##  $ NA   : chr  "" "" "" "" ...
##  $ NA   : chr  "" "" "" "" ...
```

```
spam_data<-spam_data[ , 1:2]
str(spam_data)
```

```
## 'data.frame':    5572 obs. of  2 variables:
##  $ label: chr  "ham" "ham" "spam" "ham" ...
##  $ text : chr  "Go until jurong point, crazy.. Available only in bugis n great w
orld la e buffet... Cine there got amore wat..." "Ok lar... Joking wif u oni..." "F
ree entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121
to receive entry question("| __truncated__ "U dun say so early hor... U c already t
hen say..." ...
```

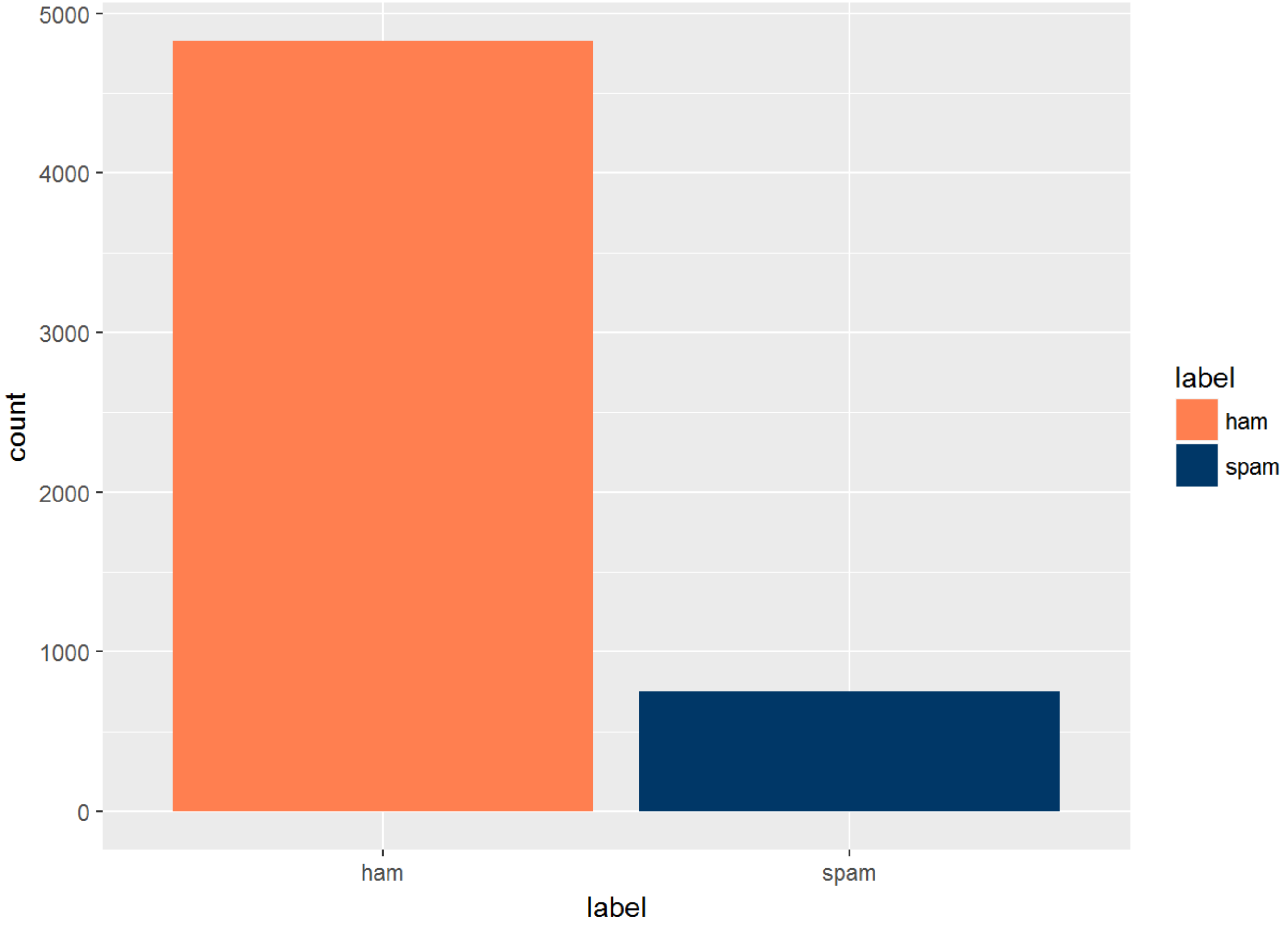
Analysis

Distribution of SMS - Ham / Spam Count

```
spam_data$label<-as.factor(spam_data$label)
prop.table(table(spam_data$label))
```

```
##
##       ham      spam
## 0.8659368 0.1340632
```

```
ggplot(spam_data,aes(x=label,fill=label))+geom_bar(stat="count")+scale_fill_manual(
values=c("#ff7f50","#003767"))+labs("Distribution of SMS")
```

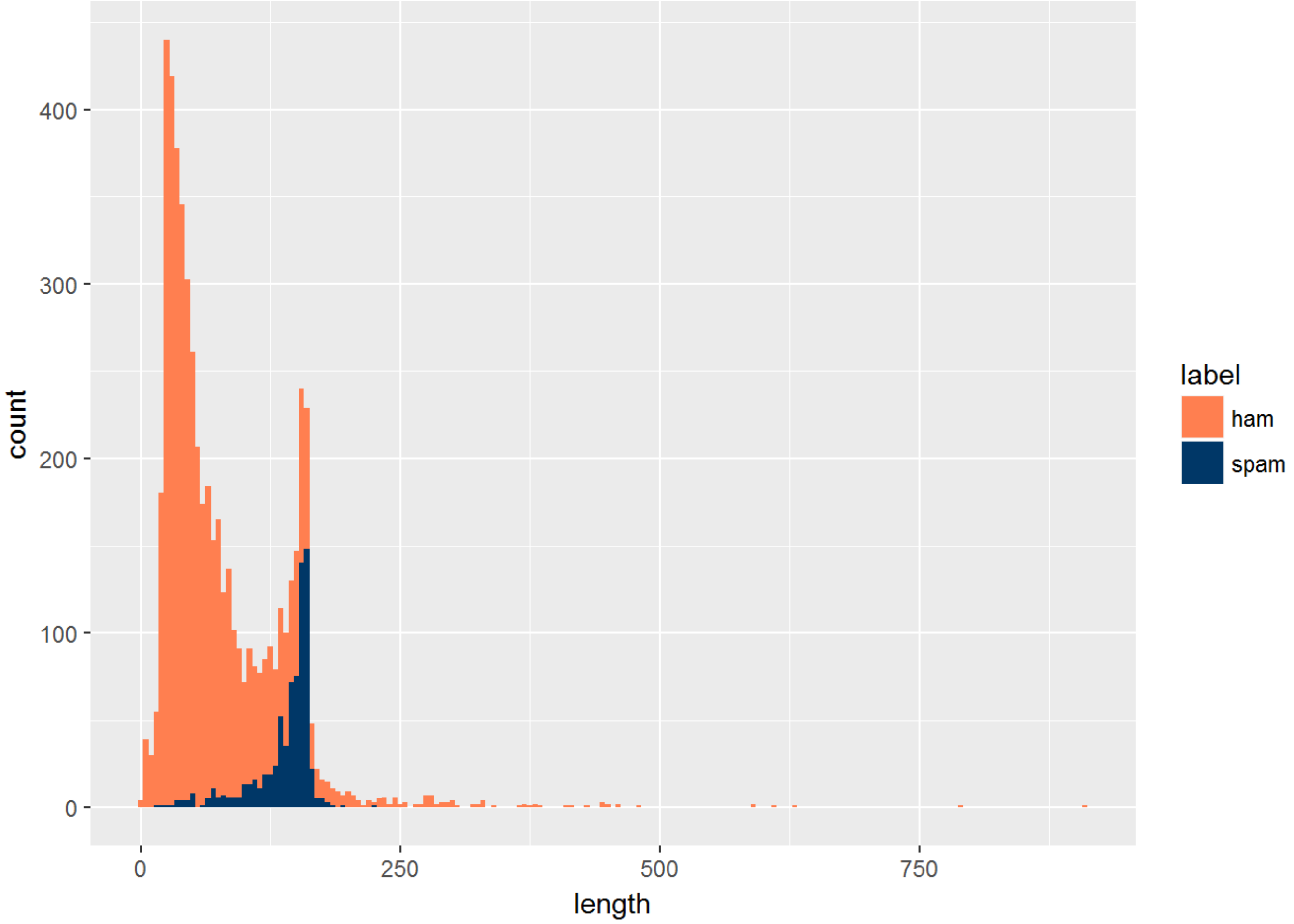


Distribution of SMS -Length

```
spam_data$length<-nchar(spam_data$text)
summary(spam_data$length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   36.00   61.00   80.12  121.00  910.00
```

```
ggplot(spam_data,aes(x=length,fill=label))+geom_histogram(binwidth=5)+scale_fill_ma
nual(values=c("#ff7f50","#003767"))+labs("Distribution of SMS length")
```



Tokenization

In this section, with the function vectorSource and Corpus, split the SMS into words.Each SMS will be considered as a Document,each word in the document as **Token** and each document as vector of features. Dataset has about 5572 SMS, so after tokenization will get 5572 documents

```
corpus <- VCorpus(VectorSource(spam_data$text))
corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5572
```

```
inspect(corpus[1:3])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 3
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 111
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 29
##
## [[3]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 155
```

Document Preprocessing

```
SMS has been converted to tokens , but it may have special characters,sysmbols,punc
tuation, whitespace etc. Here will remove all those unwanted words with tm_map func
tion
```

```
Sys.setlocale("LC_ALL", "C")
```

```
## [1] "C"
```

```
clean_corpus<-tm_map(corpus,removeWords,stopwords(kind="english"))
clean_corpus<-tm_map(corpus,stripWhitespace)
clean_corpus<-tm_map(corpus,content_transformer(tolower))
clean_corpus<-tm_map(corpus,removePunctuation)
clean_corpus<-tm_map(corpus,removeNumbers)
clean_corpus<-tm_map(corpus,stemDocument)
```

Document Term Matrix

Now convert the corpus into Document Term matrix.

```
DocumentTermMatrix(clean_corpus)
```

```
## <<DocumentTermMatrix (documents: 5572, terms: 12047)>>
## Non-/sparse entries: 61294/67064590
## Sparsity           : 100%
## Maximal term length: 57
## Weighting          : term frequency (tf)
```

```
spam_dtm<-DocumentTermMatrix(clean_corpus)
spam_dtm
```

```
## <<DocumentTermMatrix (documents: 5572, terms: 12047)>>
## Non-/sparse entries: 61294/67064590
## Sparsity           : 100%
## Maximal term length: 57
## Weighting          : term frequency (tf)
```

Finding Frequent Terms

```
freq5<-findFreqTerms(spam_dtm,5)
length(freq5)
```

```
## [1] 1767
```

```
freq5[1:10]
```

```
##  [1] "!!!"             "!!''."           "&amp;"
##  [4] "&lt;#&gt;"       "&lt;decimal&gt;" "&lt;time&gt;"
##  [7] "'ok'',"          "(std"            "*grins*"
## [10] "*sighs*"
```

Training/Testing dataset Splitting

```
spam_dtm_train<-spam_dtm[1:4150,]
spam_dtm_test<-spam_dtm[4151:5572,]

corpus_train<-clean_corpus[1:4150]
corpus_test<-clean_corpus[4151:5572]

spam_df_train_label<-spam_data[1:4150,]$label
spam_df_test_label<-spam_data[4151:5572,]$label
prop.table(table(spam_df_train_label))
```

```
## spam_df_train_label
##       ham      spam
## 0.8650602 0.1349398
```

```
prop.table(table(spam_df_test_label))
```

```
## spam_df_test_label
##       ham      spam
## 0.8684951 0.1315049
```

```
dtm_train<- spam_dtm_train[, freq5]

dim(dtm_train)
```

```
## [1] 4150 1767
```

```
dtm_test<- spam_dtm_test[,freq5]

dim(dtm_test)
```

```
## [1] 1422 1767
```

```
convert_count <- function(x) {
  y <- ifelse(x > 0, "yes","no")
    y
}

train<- apply(dtm_train, 2, convert_count)

test <- apply(dtm_test, 2, convert_count)
test[1:10,450:456]
```

```
##       Terms
## Docs   done done. doneut dont door doubl down
##   4151 "no" "no"  "no"   "no" "no" "no"  "no"
##   4152 "no" "no"  "no"   "no" "no" "no"  "no"
##   4153 "no" "no"  "no"   "no" "no" "no"  "no"
##   4154 "no" "no"  "no"   "no" "no" "no"  "no"
##   4155 "no" "no"  "no"   "no" "no" "no"  "no"
##   4156 "no" "no"  "no"   "no" "no" "no"  "no"
##   4157 "no" "no"  "no"   "no" "no" "no"  "no"
##   4158 "no" "no"  "no"   "no" "no" "no"  "no"
##   4159 "no" "no"  "no"   "no" "no" "no"  "no"
##   4160 "no" "no"  "no"   "no" "no" "no"  "no"
```

Training the model with Naive Bayes

```
set.seed(12345)
system.time( classifier <- naiveBayes(train,spam_df_train_label) )
```

```
##    user  system elapsed
##    0.86    0.02    0.87
```

Prediction

```
 pred <- predict(classifier, test)
```

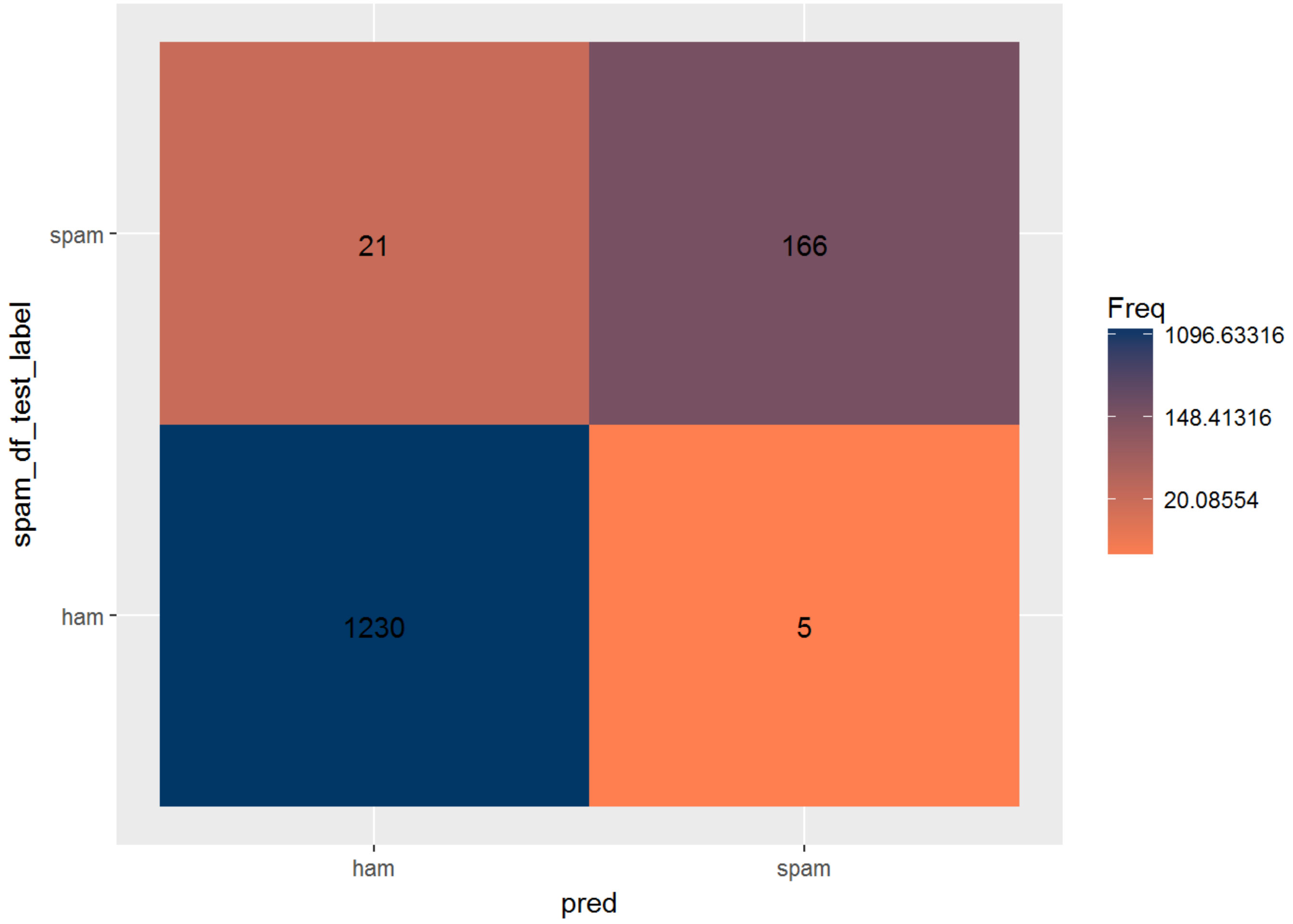
Confusion Matrix

Confusion Matrix

```
conf<- confusionMatrix(pred, spam_df_test_label)
conf
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction  ham spam
##       ham  1230   21
##       spam    5  166
##                                           
##                Accuracy : 0.9817          
##                  95% CI : (0.9733, 0.988) 
##     No Information Rate : 0.8685          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9169          
##  Mcnemar's Test P-Value : 0.003264        
##                                           
##             Sensitivity : 0.9960          
##             Specificity : 0.8877          
##          Pos Pred Value : 0.9832          
##          Neg Pred Value : 0.9708          
##              Prevalence : 0.8685          
##          Detection Rate : 0.8650          
##    Detection Prevalence : 0.8797          
##       Balanced Accuracy : 0.9418          
##                                           
##        'Positive' Class : ham             
## 
```

```
confusion_matrix <- as.data.frame(table(pred, spam_df_test_label))

ggplot(data = confusion_matrix,      aes(x = pred, y = spam_df_test_label)) +
  geom_tile(aes(fill = Freq)) +
  geom_text(aes(label = sprintf("%1.0f", Freq)), vjust = 1) +
  scale_fill_gradient(low = "#ff7f50",
                      high = "#003767",
                      trans = "log")
```



ROC Curve

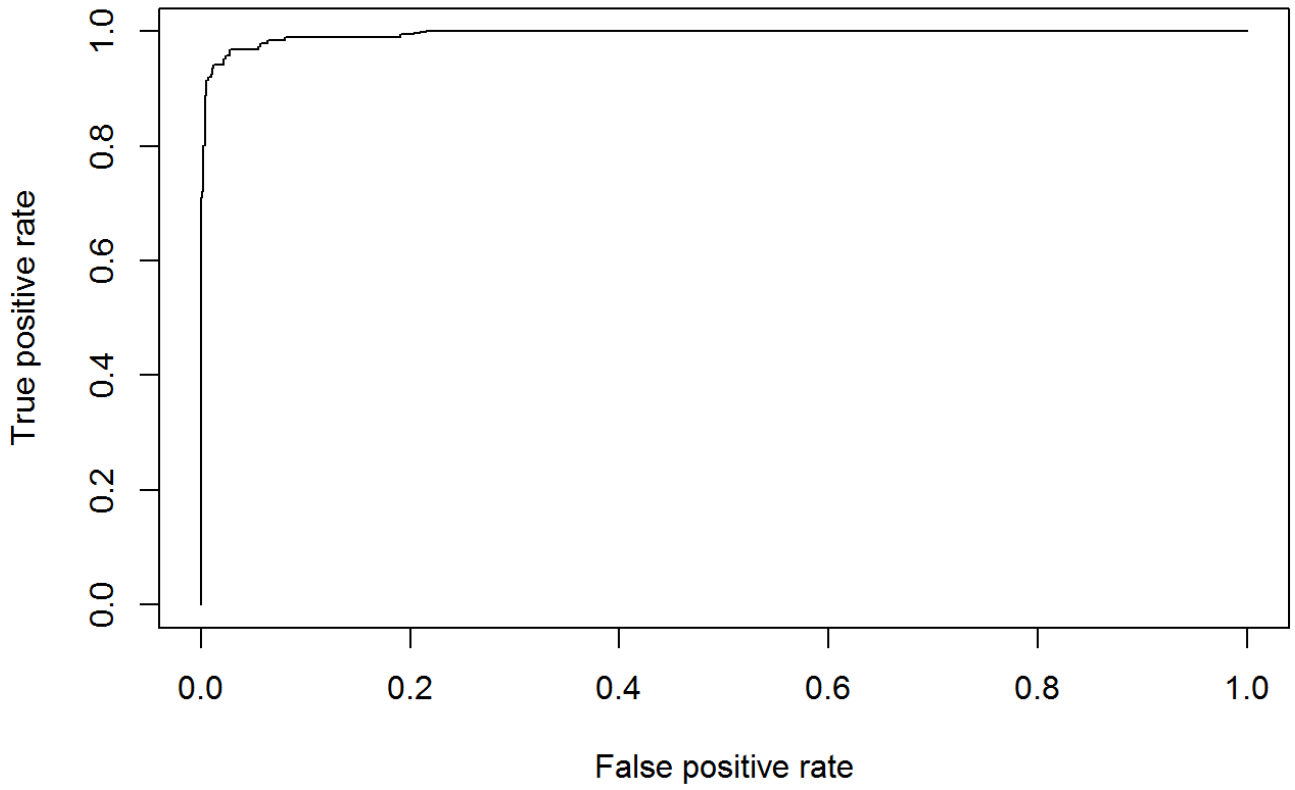
```
probs<-predict(classifier,test,type="raw")

library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred <- prediction(probs[, "spam"], spam_df_test_label)
perf_nb <- performance(pred, measure='tpr', x.measure='fpr')
plot(perf_nb)
```



Conclusion

```
Naive Bayes had classified the SMS with 98 % accuracy.
```