Assignment 5.b

Athul Jose P
11867566

School of Electrical Engineering and Computer Science

Washington State University

CptS 575 Data Science

1

```r
# loading libraries
library(readr)
library(tm)
```

```
Loading required package: NLP


Attaching package: 'NLP'

The following object is masked from 'package:ggplot2':

    annotate
```

```r
library(SnowballC)
library(quanteda)
```

```
Package version: 4.1.0
Unicode version: 15.1
ICU version: 74.1

Parallel computing: 32 of 32 threads used.

See https://quanteda.io for tutorials and examples.


Attaching package: 'quanteda'

The following object is masked from 'package:tm':

    stopwords

The following objects are masked from 'package:NLP':

    meta, meta<-
```

```r
# loading dataset
data <- read_csv("bbc.csv", show_col_types = FALSE)

# preprocessing and creating dtm
corpus <- corpus(data$text)
corpus <- tokens(corpus,
                 what = "word",
                 remove_punct = TRUE,
                 remove_numbers = TRUE) %>%
                 tokens_tolower() %>%
                 tokens_remove(stopwords("english")) %>%
                 tokens_wordstem(language = "en")

dtm <- dfm(corpus)

# top 85% frequency
term_freq <- colSums(as.matrix(dtm))
term_freq <- sort(term_freq, decreasing = TRUE)
threshold <- quantile(term_freq, 0.15)
dtm_thres <- dtm[, term_freq >= threshold]

# 2205th article
article_vector <- as.numeric(dtm_thres[2205, ])
names(article_vector) <- colnames(dtm_thres)
article_vector_filtered <- article_vector[article_vector >= 4]

article_words <- data.frame(
  word = names(article_vector_filtered),
  frequency = article_vector_filtered
)
print(article_words)
```

```
             word frequency
connect   connect         6
user         user         4
project   project         6
say           say         4
use           use         4
inform     inform         4
system     system         4
base         base         4
comput     comput         4
school     school         8
station   station         4
```

```
student   student         5
satellit satellit         5
handheld handheld         4
eduvis     eduvis         5
herren     herren         4
e-slat     e-slat         5
```

The processed output displays a list of key words extracted from the BBC news data, following tokenization and cleaning steps. After removing punctuation, numbers, and stopwords—common but less informative terms—the words were stemmed to their root forms, allowing similar words (like "connected" and "connect") to be grouped together. The remaining terms represent the most relevant words in the document, each with a frequency of 4 or higher, indicating their significance in the content.

The word "school," with the highest frequency, suggests that the article centers around an educational theme, specifically focused on schools. Additional terms like "station," "connect," "satellite," and "base" imply a technical context, possibly describing infrastructure used to facilitate connectivity within the school. Words like "student," "use," "system," and "project" further emphasize the presence of a structured technological initiative benefiting students. Together, these high-frequency terms paint a picture of an article about a school-based technology project, likely involving a satellite or wireless system, designed to enhance student learning.

2

```r
# loading libraries
library(quanteda)
library(caret)
```

```
Loading required package: lattice


Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```r
library(naivebayes)
```

```
naivebayes 1.0.0 loaded

For more information please visit:

https://majkamichal.github.io/naivebayes/
```

```r
library(dplyr)
library(glmnet)
```

```
Loading required package: Matrix


Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-8
```

```r
dtm_df <- as.data.frame(as.matrix(dtm_thres))

# dtm reduction using variance
feature_variances <- apply(dtm_df, 2, var)
variance_threshold <- 0.1
dtm_reduced <- dtm_df[, feature_variances > variance_threshold]

# print dimensions
print(dim(dtm_df))
```

```
[1]  2225 23126
```

```r
print(dim(dtm_reduced))
```

```
[1] 2225 1107
```

```r
# splitting into train and test
data_split <- cbind(category = data$category, dtm_reduced)
set.seed(123)
train_index <- createDataPartition(
  data_split$category,
  p = 0.8,
  list = FALSE
)
train_data <- data_split[train_index, ]
test_data <- data_split[-train_index, ]

# Separate x & y
train_x <- as.matrix(train_data[, -1])
train_y <- as.factor(train_data$category)
test_x <- as.matrix(test_data[, -1])
```

```r
# multinomial naive bayes
nb_model <- multinomial_naive_bayes(train_x, train_y)
nb_predictions <- predict(nb_model, newdata = test_x)
nb_predictions <- factor(nb_predictions, levels = levels(train_y))
test_data$category <- factor(test_data$category, levels = levels(train_y))

# confusion matrix
conf_matrix <- confusionMatrix(nb_predictions, test_data$category)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

               Reference
Prediction      business entertainment politics sport tech
  business            95             2        1     0    2
  entertainment        3            70        0     0    1
  politics             2             3       82     0    1
  sport                0             0        0   102    0
  tech                 2             2        0     0   76

Overall Statistics

               Accuracy : 0.9572
                 95% CI : (0.934, 0.974)
    No Information Rate : 0.2297
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9463

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: business Class: entertainment Class: politics
Sensitivity                   0.9314               0.9091          0.9880
Specificity                   0.9854               0.9891          0.9834
Pos Pred Value                0.9500               0.9459          0.9318
Neg Pred Value                0.9797               0.9811          0.9972
Prevalence                    0.2297               0.1734          0.1869
Detection Rate                0.2140               0.1577          0.1847
Detection Prevalence          0.2252               0.1667          0.1982
Balanced Accuracy             0.9584               0.9491          0.9857
                     Class: sport Class: tech
Sensitivity                1.0000      0.9500
Specificity                1.0000      0.9890
Pos Pred Value             1.0000      0.9500
Neg Pred Value             1.0000      0.9890
Prevalence                 0.2297      0.1802
Detection Rate             0.2297      0.1712
Detection Prevalence       0.2297      0.1802
Balanced Accuracy          1.0000      0.9695
```

```r
# Precision and Recall
conf_table <- conf_matrix$table
precision <- diag(conf_table) / rowSums(conf_table)
```

```r
recall <- diag(conf_table) / colSums(conf_table)
print("Precision by class:")
```

```
[1] "Precision by class:"
```

```r
print(precision)
```

```
     business entertainment      politics         sport          tech 
    0.9500000     0.9459459     0.9318182     1.0000000     0.9500000 
```

```r
print("Recall by class:")
```

```
[1] "Recall by class:"
```

```r
print(recall)
```

```
     business entertainment      politics         sport          tech 
    0.9313725     0.9090909     0.9879518     1.0000000     0.9500000 
```

```r
# multinomial logistic regression
train_y_numeric <- as.numeric(as.factor(train_y)) - 1
log_reg_model <- cv.glmnet(
  as.matrix(train_x),
  train_y_numeric,
  family = "multinomial",
  alpha = 0
)
log_reg_predictions <- predict(
  log_reg_model,
  newx = as.matrix(test_x),
  s = "lambda.min",
  type = "class"
)
log_reg_predictions <- factor(log_reg_predictions, labels = levels(train_y))
test_y <- factor(test_data$category, levels = levels(train_y))

# confusion matrix
conf_matrix_log_reg <- confusionMatrix(log_reg_predictions, test_y)
print(conf_matrix_log_reg)
```

```
Confusion Matrix and Statistics

               Reference
Prediction      business entertainment politics sport tech
  business            97             2        1     0    2
  entertainment        2            74        1     0    1
  politics             2             1       78     0    0
  sport                1             0        2   102    0
  tech                 0             0        1     0   77

Overall Statistics

               Accuracy : 0.964
                 95% CI : (0.9421, 0.9793)
    No Information Rate : 0.2297
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9548

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: business Class: entertainment Class: politics
Sensitivity                   0.9510               0.9610          0.9398
Specificity                   0.9854               0.9891          0.9917
Pos Pred Value                0.9510               0.9487          0.9630
Neg Pred Value                0.9854               0.9918          0.9862
Prevalence                    0.2297               0.1734          0.1869
Detection Rate                0.2185               0.1667          0.1757
Detection Prevalence          0.2297               0.1757          0.1824
Balanced Accuracy             0.9682               0.9751          0.9657
                     Class: sport Class: tech
Sensitivity                1.0000      0.9625
Specificity                0.9912      0.9973
Pos Pred Value             0.9714      0.9872
Neg Pred Value             1.0000      0.9918
Prevalence                 0.2297      0.1802
Detection Rate             0.2297      0.1734
Detection Prevalence       0.2365      0.1757
Balanced Accuracy          0.9956      0.9799
```

```
# Precision and Recall
conf_table_log_reg <- conf_matrix_log_reg$table
precision_log_reg <- diag(conf_table_log_reg) / rowSums(conf_table_log_reg)
```

```r
recall_log_reg <- diag(conf_table_log_reg) / colSums(conf_table_log_reg)
print("Precision by class:")
```

```
[1] "Precision by class:"
```

```r
print(precision_log_reg)
```

```
     business entertainment      politics         sport          tech 
    0.9509804     0.9487179     0.9629630     0.9714286     0.9871795 
```

```r
print("Recall by class:")
```

```
[1] "Recall by class:"
```

```r
print(recall_log_reg)
```

```
     business entertainment      politics         sport          tech 
    0.9509804     0.9610390     0.9397590     1.0000000     0.9625000 
```

The comparison between Multinomial Naïve Bayes and Multinomial Logistic Regression reveals that Logistic Regression provides a marginally better overall fit for this dataset. Logistic Regression achieves a higher accuracy (96.4% vs. 95.7%) and kappa (0.9548 vs. 0.9463) compared to Naïve Bayes, suggesting a stronger agreement with the true classifications. Examining class-specific metrics, Logistic Regression shows higher sensitivity across most categories, particularly for "Business" and "Entertainment," which indicates a greater ability to correctly identify positive instances in these classes. Naïve Bayes, however, slightly outperforms Logistic Regression in the "Politics" category, where it has a sensitivity of 0.9880 compared to 0.9398 for Logistic Regression. In terms of specificity, Logistic Regression consistently surpasses Naïve Bayes, particularly in the "Politics" and "Tech" categories, which suggests that Logistic Regression is better at correctly identifying negative instances for these classes.

Precision and recall scores reinforce these trends. Logistic Regression demonstrates higher precision in "Politics" and "Tech," while Naïve Bayes provides slightly better precision for "Entertainment" and "Sport." Both models achieve excellent recall, with Logistic Regression slightly outperforming Naïve Bayes in the "Entertainment" and "Tech" categories. Both models, however, achieve perfect recall for the "Sport" category, indicating that this class is well-distinguished by both approaches. In summary, Multinomial Logistic Regression provides a slight advantage in terms of accuracy, specificity, and precision, making it a slightly more accurate model for this dataset. Nevertheless, Multinomial Naïve Bayes remains a strong contender and may be preferred when computational efficiency is a priority, as it is less resource-intensive than Logistic Regression.