85

Good job — see comments below

---

title: "Problem_set_2"

output: pdf_document

date: "2025-03-04"

...{r}

rm(list=ls())

install.packages("quanteda")

install.packages("quanteda.textplots")

install.packages("ggplot2")

install.packages("sentimentr")

install.packages("quanteda.textmodels")

install.packages("quanteda.textplots")

install.packages("quanteda.textstats")

install.packages(c("randomForest", "caret", "rpart", "rpart.plot"))

Load necessary libraries

library(quanteda)

library(ggplot2)

library(dplyr)

library(sentimentr)

library(quanteda)

library(quanteda.textmodels)

library(caret)

library(randomForest)

library(rpart)

```
library(rpart.plot)

library(quanteda.textstats)


## Set your working directory

setwd("/Users/arnavsahai/Desktop/Text as Data/R practice/Problem_set_2")


# Load the dataset

base::load("russian_trolls_sample.rdata")



# Create a dictionary for the topic "elections"

election_dict <- dictionary(list(elections = c("election", "vote", "voting", "ballot", "candidate",
"campaign", "poll", "elected", "voter", "elections")))
```

30

...

a) I chose these keywords because they are commonly associated with elections and voting processes. They cover various aspects such as the act of voting, political parties, prominent candidates, and general election-related terms.



...{r}

```
# Create a corpus from the dataset

corpus <- corpus(russian_trolls_sample, text_field = "text")


# Tokenize the corpus

tokens <- tokens(corpus)


# Apply the dictionary

dfm <- dfm(tokens)

dfm_dict <- dfm_lookup(dfm, dictionary = election_dict)
```

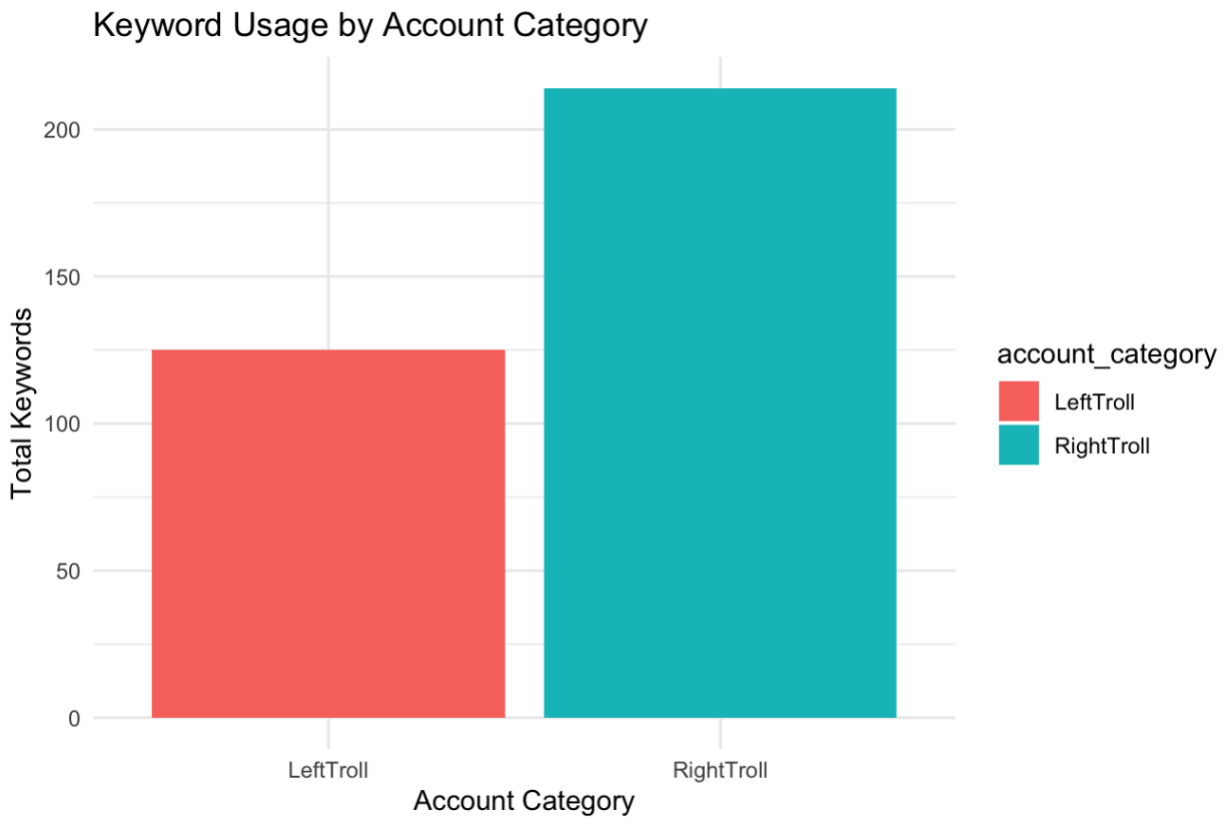
```
#Convert to a data frame
dfm_df <- convert(dfm_dict, to = "data.frame")

# Add the account category to the data frame
dfm_df$account_category <- russian_trolls_sample$account_category

# Summarize the keyword usage by account category
keyword_usage <- dfm_df %>%
  group_by(account_category) %>%
  summarise(total_keywords = sum(elections))

# Plot the results
ggplot(keyword_usage, aes(x = account_category, y = total_keywords, fill =
account_category)) +
  geom_bar(stat = "identity") +
  labs(title = "Keyword Usage by Account Category", x = "Account Category", y = "Total
Keywords") +
  theme_minimal()

...
```



(b) Which group used these keywords more?

The bar plot will show which group (right or left trolls) used the keywords more. This can provide insights into the IRA's strategy.

The RightTrolls used the keywords more which can help us deduce that through tweets they wanted the right political ideology to get more social traction and views and promote that.

... {r}

#Add the date to the data frame

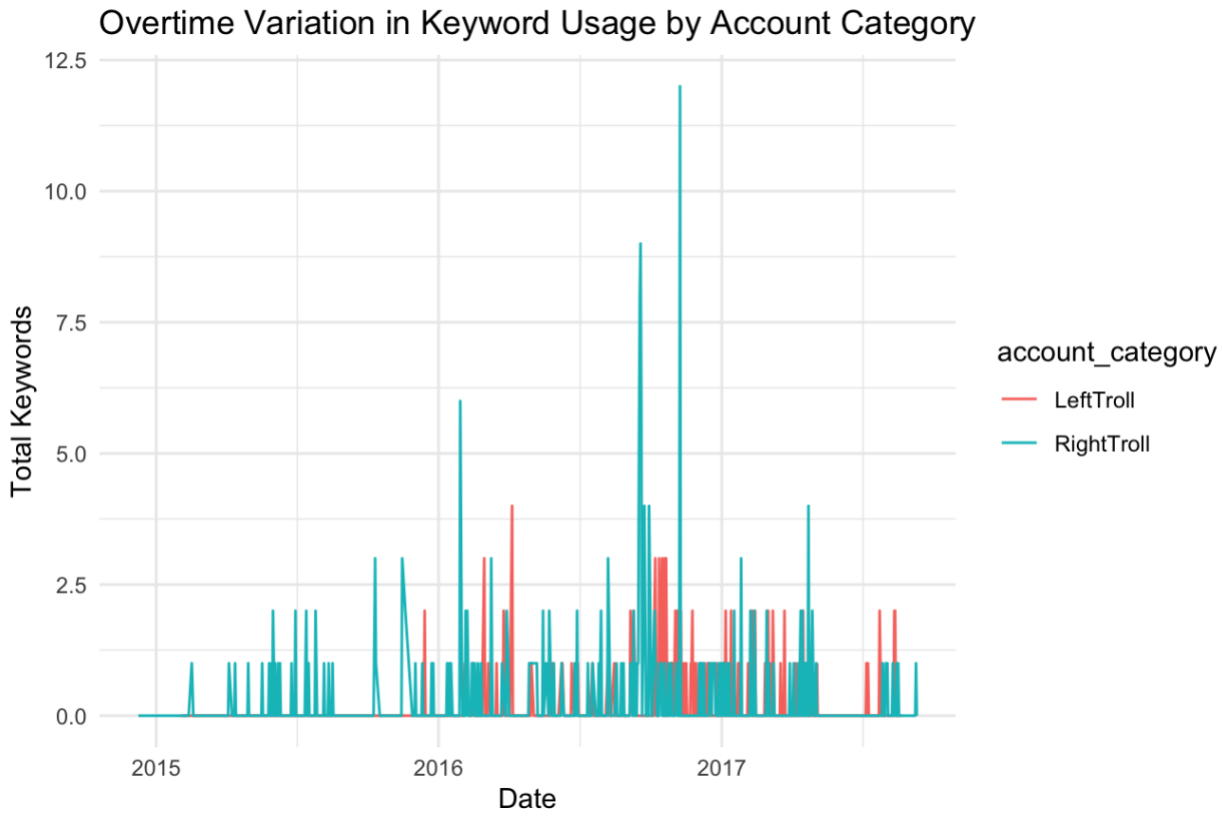
dfm_df$date <- russian_trolls_sample$date

Summarize the keyword usage by date and account category

keyword_usage_overtime <- dfm_df %>%

```
  group_by(date, account_category) %>%
  summarise(total_keywords = sum(elections))

# Plot the results
ggplot(keyword_usage_overtime, aes(x = date, y = total_keywords, color =
account_category)) +
  geom_line() +
  labs(title = "Overtime Variation in Keyword Usage by Account Category", x = "Date", y =
"Total Keywords") +
  theme_minimal()
...
```



(c) we see a spike in late 2016 of right wing trolls which can be attributed to the 2016 presedential elections and promotion of Trump.

```{r}

# Sentiment Analysis

russian_trolls_text <- get_sentences(russian_trolls_sample$text)

russian_trolls_sentiment <- sentiment_by(russian_trolls_text, by =
russian_trolls_sample$account_category)


# Present the results in a table

sentiment_table <- as.data.frame(russian_trolls_sentiment)

print(sentiment_table)


# Plot the sentiment results

plot(russian_trolls_sentiment)


# Profanity Analysis

russian_trolls_profanity <- profanity_by(russian_trolls_text, by =
russian_trolls_sample$account_category)


# Present the results in a table

profanity_table <- as.data.frame(russian_trolls_profanity)

print(profanity_table)


# Emotion Analysis

russian_trolls_emotion <- emotion_by(russian_trolls_text, by =
russian_trolls_sample$account_category)


# Present the results in a table

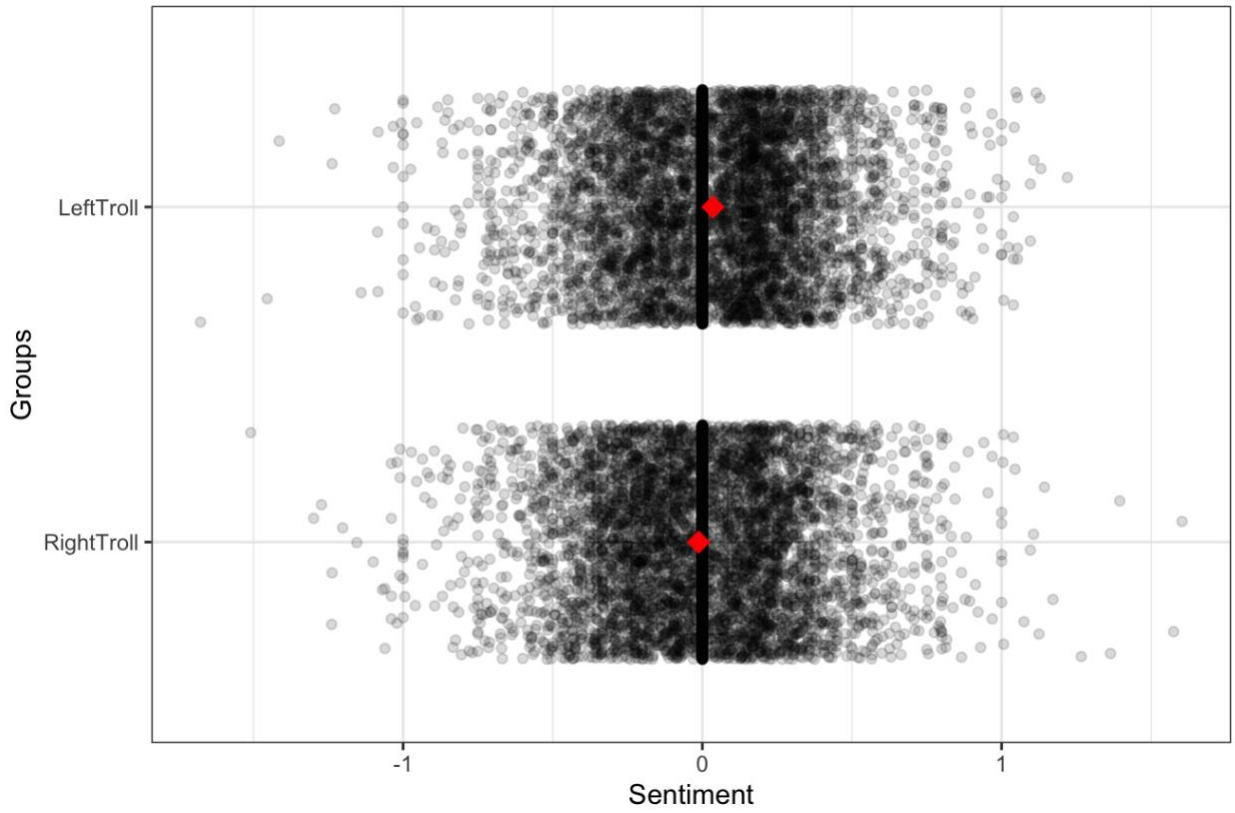
emotion_table <- as.data.frame(russian_trolls_emotion)

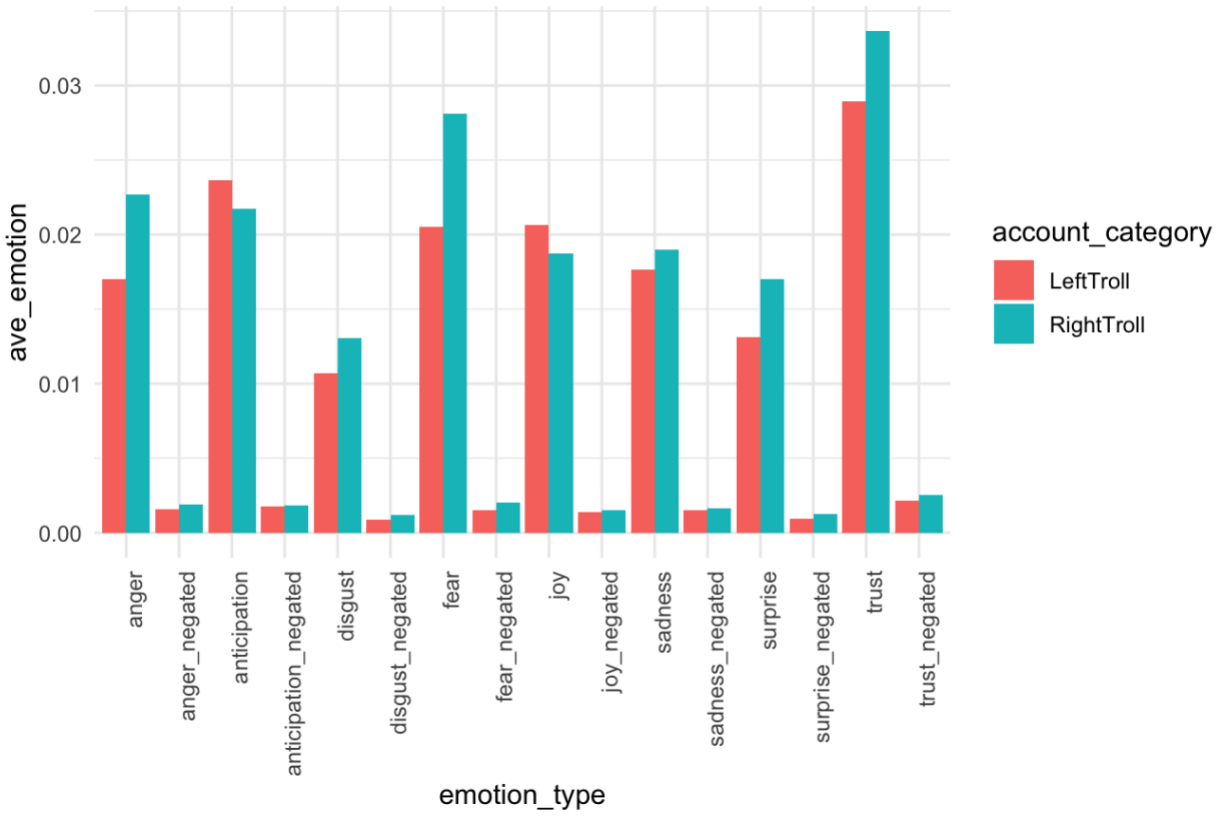
print(emotion_table)


# Plot the emotion results
```

```
ggplot(russian_trolls_emotion, aes(x = emotion_type, y = ave_emotion, fill =
account_category)) +
  geom_bar(position = "dodge", stat = "identity") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

...





sentiment_table

	account_category	word_count	sd	ave_sentiment
1	LeftTroll	94164	0.2548476	0.03550818
2	RightTroll	84731	0.2640546	-0.01253506

lefttroll has higher avg positive sentiment

rightroll has higher avg negative sentiment

profanity_table

	account_category	word_count	profanity_count	sd	ave_profanity
1	LeftTroll	85611	279	0.03315522	0.003258927
2	RightTroll	77099	110	0.01470017	0.001426737

lefttrolls use more profanity

emotion_table

	account_category	emotion_type	word_count	emotion_count	sd	ave_emotion
1	LeftTroll	anger	85611	1455	0.058277862	0.0169954796
2	LeftTroll	anger_negated	85611	137	0.013742110	0.0016002616
3	LeftTroll	anticipation	85611	2027	0.075375206	0.0236768640
4	LeftTroll	anticipation_negated	85611	154	0.015276065	0.0017988343
5	LeftTroll	disgust	85611	914	0.048799031	0.0106761982
6	LeftTroll	disgust_negated	85611	77	0.009858103	0.0008994171
7	LeftTroll	fear	85611	1759	0.061607485	0.0205464251
8	LeftTroll	fear_negated	85611	130	0.015730363	0.0015184965
9	LeftTroll	joy	85611	1769	0.077962673	0.0206632325
10	LeftTroll	joy_negated	85611	120	0.013302916	0.0014016890
11	LeftTroll	sadness	85611	1512	0.051670304	0.0176612818
12	LeftTroll	sadness_negated	85611	132	0.014443094	0.0015418579
13	LeftTroll	surprise	85611	1125	0.057396158	0.0131408347
14	LeftTroll	surprise_negated	85611	82	0.010548230	0.0009578208
15	LeftTroll	trust	85611	2478	0.075822027	0.0289448786
16	LeftTroll	trust_negated	85611	185	0.015489548	0.0021609373
17	RightTroll	anger	77099	1747	0.070967006	0.0226591785
18	RightTroll	anger_negated	77099	144	0.015249002	0.0018677285
19	RightTroll	anticipation	77099	1676	0.065714964	0.0217382845
20	RightTroll	anticipation_negated	77099	142	0.018321423	0.0018417878
21	RightTroll	disgust	77099	1008	0.060385358	0.0130740995
22	RightTroll	disgust_negated	77099	90	0.011025810	0.0011673303
23	RightTroll	fear	77099	2169	0.073739550	0.0281326606
24	RightTroll	fear_negated	77099	158	0.016517336	0.0020493132
25	RightTroll	joy	77099	1442	0.067120918	0.0187032257

26 RightTroll joy_negated 77099 119 0.016136066 0.0015434701

27 RightTroll sadness 77099 1466 0.063701179 0.0190145138

28 RightTroll sadness_negated 77099 129 0.015319074 0.0016731735

29 RightTroll surprise 77099 1312 0.061356291 0.0170170819

30 RightTroll surprise_negated 77099 97 0.012099990 0.0012581227

31 RightTroll trust 77099 2595 0.077558059 0.0336580241

10

32 RightTroll trust_negated 77099 194 0.020214109 0.0025162453

Right troll expresses more anger and fear while left troll expresses more joy

trust fear anticipation and anger are the most prominent emotions expresses in the tweets

... {r}

...existing code...

Set seed for reproducibility

set.seed(123)

Split the data into training and test sets

train_indices <- sample(seq_len(nrow(russian_trolls_sample)), size = 7000)

train_set <- russian_trolls_sample[train_indices,]

test_set <- russian_trolls_sample[-train_indices,]

Create a corpus for each set

corpus_train <- corpus(train_set, text_field = "text")

corpus_test <- corpus(test_set, text_field = "text")

Tokenize the corpora

```
tokens_train <- tokens(corpus_train)
tokens_test <- tokens(corpus_test)

# Create DFMs for each set
dfm_train <- dfm(tokens_train)
dfm_test <- dfm(tokens_test)

# Pre-process the DFMs (e.g., remove stopwords, stemming)
dfm_train <- dfm_remove(dfm_train, stopwords("en"))
dfm_train <- dfm_wordstem(dfm_train)

dfm_test <- dfm_remove(dfm_test, stopwords("en"))
dfm_test <- dfm_wordstem(dfm_test)

# Print the dimensions of the training and test DFMs
cat("Dimensions of training DFM:", dim(dfm_train), "\n")
cat("Dimensions of test DFM:", dim(dfm_test), "\n")

# ...existing code...
...
Dimensions of training DFM: 7000 20717
Dimensions of test DFM: 3000 11186

...{r}
# Trim the DFMs
dfm_train_trimmed <- dfm_trim(dfm_train, min_termfreq = 5, max_termfreq = 400)
dfm_test_trimmed <- dfm_trim(dfm_test, min_termfreq = 5, max_termfreq = 400)
```

Print the dimensions of the trimmed training and test DFMs

cat("Dimensions of trimmed training DFM:", dim(dfm_train_trimmed), "\n")

cat("Dimensions of trimmed test DFM:", dim(dfm_test_trimmed), "\n")

...

Dimensions of trimmed training DFM: 7000 2455

Dimensions of trimmed test DFM: 3000 1204

...{r}

Ensure the test DFM has the same features as the training DFM

dfm_test_trimmed <- dfm_match(dfm_test_trimmed, features = featnames(dfm_train_trimmed))

Get the dimensions of the trimmed training and test DFMs

cat("Dimensions of trimmed training DFM:", dim(dfm_train_trimmed), "\n")

cat("Dimensions of trimmed test DFM:", dim(dfm_test_trimmed), "\n")

Train a Naive Bayes classifier

nb_model <- textmodel_nb(dfm_train_trimmed, y = docvars(dfm_train_trimmed, "account_category"))

Predict account_category in the test set

nb_predictions <- predict(nb_model, newdata = dfm_test_trimmed)

Evaluate the Naive Bayes classifier

nb_confusion_matrix <- confusionMatrix(as.factor(nb_predictions), as.factor(docvars(dfm_test_trimmed, "account_category")))

print(nb_confusion_matrix)

...

Confusion Matrix and Statistics

Prediction	Reference: LeftTroll	Reference: RightTroll
LeftTroll	1152	457
RightTroll	385	1006

Accuracy : 0.7193

95% CI : (0.7029, 0.7354)

No Information Rate : 0.5123

P-Value [Acc > NIR] : < 2e-16

Kappa : 0.4377

Mcnemar's Test P-Value : 0.01441

Sensitivity : 0.7495

Specificity : 0.6876

Pos Pred Value : 0.7160

Neg Pred Value : 0.7232

Prevalence : 0.5123

Detection Rate : 0.3840

Detection Prevalence : 0.5363

Balanced Accuracy : 0.7186

'Positive' Class : LeftTroll

…{r}

```
# Train a classification tree
train_data <- convert(dfm_train_trimmed, to = "data.frame")
train_data$account_category <- docvars(dfm_train_trimmed, "account_category")

# Ensure the column names are valid for rpart and remove any duplicates
colnames(train_data) <- make.names(colnames(train_data), unique = TRUE)

tree_model <- rpart(account_category ~ ., data = train_data)

# Predict account_category in the test set
test_data <- convert(dfm_test_trimmed, to = "data.frame")
test_data$account_category <- docvars(dfm_test_trimmed, "account_category")

# Ensure the column names are valid for rpart and remove any duplicates
colnames(test_data) <- make.names(colnames(test_data), unique = TRUE)

tree_predictions <- predict(tree_model, newdata = test_data, type = "class")

# Evaluate the classification tree
tree_confusion_matrix <- confusionMatrix(as.factor(tree_predictions),
as.factor(test_data$account_category))
print(tree_confusion_matrix)
...
```

Confusion Matrix and Statistics

	Reference	
Prediction	**LeftTroll**	**RightTroll**

LeftTroll 788 751

RightTroll 749 712

Accuracy : 0.5

95% CI : (0.4819, 0.5181)

No Information Rate : 0.5123

P-Value [Acc > NIR] : 0.9146

Kappa : -6e-04

Mcnemar's Test P-Value : 0.9794

Sensitivity : 0.5127

Specificity : 0.4867

Pos Pred Value : 0.5120

Neg Pred Value : 0.4873

Prevalence : 0.5123

Detection Rate : 0.2627

Detection Prevalence : 0.5130

Balanced Accuracy : 0.4997

'Positive' Class : LeftTroll

...{r}

```
# Compare model performance
nb_accuracy <- nb_confusion_matrix$overall["Accuracy"]
tree_accuracy <- tree_confusion_matrix$overall["Accuracy"]
```

```
nb_precision <- nb_confusion_matrix$byClass["Pos Pred Value"]
tree_precision <- tree_confusion_matrix$byClass["Pos Pred Value"]

nb_recall <- nb_confusion_matrix$byClass["Sensitivity"]
tree_recall <- tree_confusion_matrix$byClass["Sensitivity"]

nb_f1 <- 2 * (nb_precision * nb_recall) / (nb_precision + nb_recall)
tree_f1 <- 2 * (tree_precision * tree_recall) / (tree_precision + tree_recall)

# Print model performance statistics
cat("Naive Bayes Model Performance:\n")
cat("Accuracy:", nb_accuracy, "\n")
cat("Precision:", nb_precision, "\n")
cat("Recall:", nb_recall, "\n")
cat("F1 Score:", nb_f1, "\n\n")

cat("Classification Tree Model Performance:\n")
cat("Accuracy:", tree_accuracy, "\n")
cat("Precision:", tree_precision, "\n")
cat("Recall:", tree_recall, "\n")
cat("F1 Score:", tree_f1, "\n")

# Determine which model performed better
if (nb_accuracy > tree_accuracy) {
  cat("The Naive Bayes model performed better based on accuracy.\n")
} else {
  cat("The classification tree model performed better based on accuracy.\n")
```

}

...

Naive Bayes Model Performance:

Accuracy: 0.7193333

Precision: 0.7159727

Recall: 0.749512

F1 Score: 0.7323586

Classification Tree Model Performance:

Accuracy: 0.5

Precision: 0.5120208

Recall: 0.5126871

F1 Score: 0.5123537

The Naive Bayes model performed better based on accuracy.

What about other metrics?