Project 1 - Cyberbullying Detection

2022-06-11

Introduction

In this project, I downloaded data of tweets that are categorized as either cyber bullying or not. This data was downloaded from Kaggle (link: https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification (https://www.kaggle.com/datasets/andrewmvd/cyberbullying-classification)). I decided to explore the data, then create a machine learning model that can classify new text data as either bullying or not with a certain level of accuracy. While this project was very challenging to create for this course, I was happy to finish it as I am very passionate about machine learning.

Imports

First, let's import some libraries that will be useful to us

- *(dplyr, stringr):* Used for data manipulation
- *(ggplot2, plotly):* Used for data visualization
- *(keras, purrr, tensorflow):* Used for classification
- *(shiny):* Used to create the final application

```
library(dplyr)
library(ggplot2)
library(stringr)
library(plotly)
library(keras)
library(purrr)
library(tensorflow)
library(shiny)

tweets= read.csv("cyberbullying_tweets.csv", sep=",", header=TRUE)
```

Adding a new column

The column 'cyberbullying_type' includes 6 categories listed below. I decided to create a new column that describes the text as either bullying or not bullying with outputs as 'yes' or 'no'. This will be useful when training our model later.

```
unique(tweets$cyberbullying_type)
```

```
## [1] "not_cyberbullying"   "gender"             "religion"
## [4] "other_cyberbullying" "age"                "ethnicity"
```

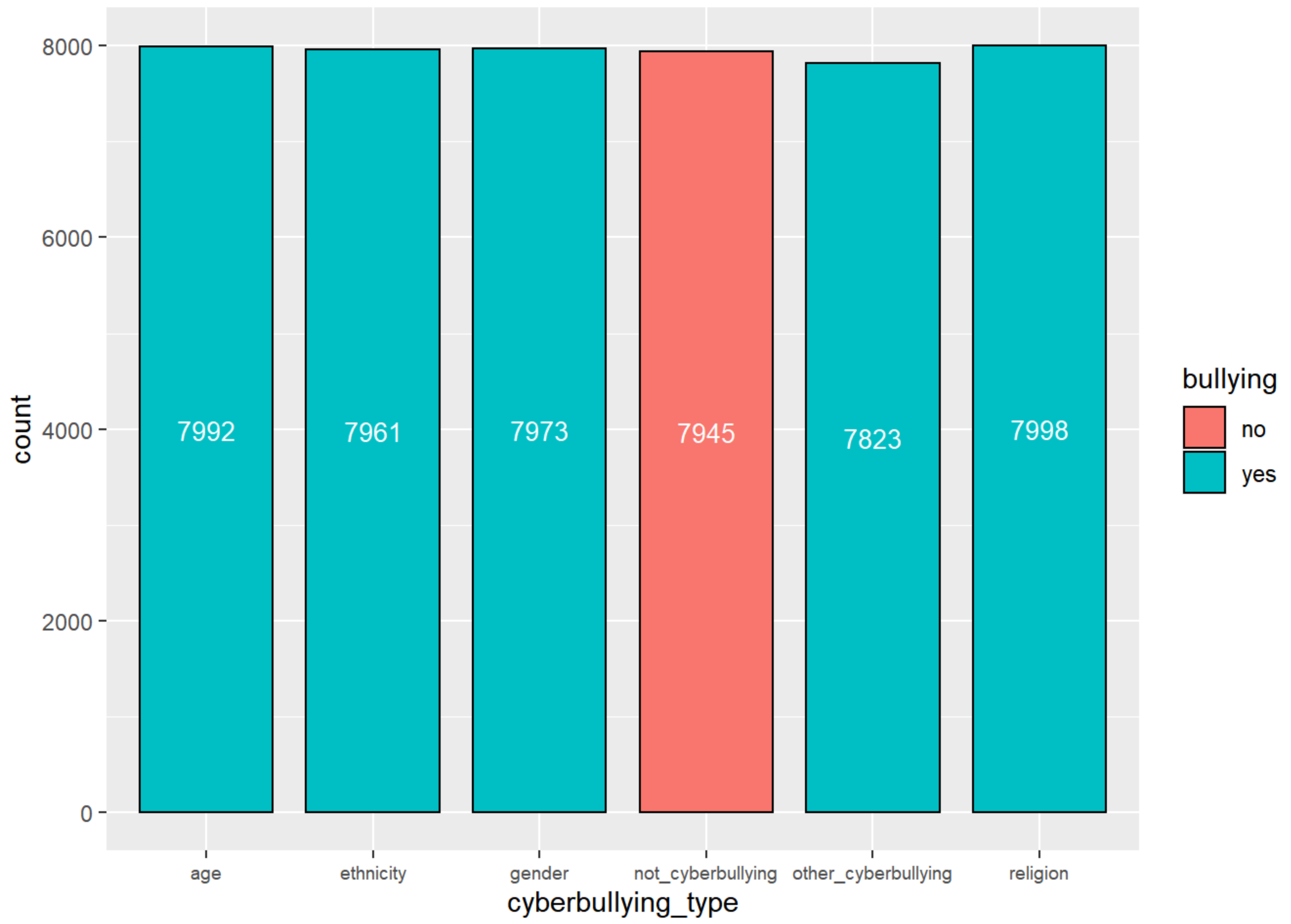
```
tweets = tweets%>%
    mutate(bullying = case_when(cyberbullying_type == "religion" ~ "yes",
                                cyberbullying_type == "age" ~ "yes",
                                cyberbullying_type == "ethnicity" ~ "yes",
                                cyberbullying_type == "other_cyberbullying" ~ "yes",
                                cyberbullying_type == "gender" ~ "yes",
                                cyberbullying_type == "not_cyberbullying" ~ "no",
                                 ))
```

Bar plot!

Below shows the count of tweets per category! As you can see the categories are split evenly. The only issue is that only 1/6 of the tweets are categorized as not bullying. This could cause our training model to not be as accurate as we would like it to be when new tweet inputs are not bullying but may have words that could trigger the model to think it is.

```
fig1 <- ggplot(data=tweets, aes(x= cyberbullying_type, fill = bullying)) +
  geom_bar(colour="black", stat="count", width=0.8, position = position_dodge(width=0.1)) +
  stat_count(geom = "text", colour = "white", size = 3.5,
aes(label = ..count..),position=position_stack(vjust=0.5)) + theme(axis.text.x = element_text
(size = 7))

fig1
```



Adding two more columns!

Here I am just adding the count of each category as we saw above to the data set.

```
type_count <- tweets %>%
  group_by(cyberbullying_type) %>%
  summarize(count = length(unique(tweet_text)))
```

And here, I am using the stringr library to count the number of words in each tweet!

```
tweets <- mutate(tweets,
       word_count = str_count(tweet_text, "\\w+"))
tweets <- merge(tweets, type_count,
                by = "cyberbullying_type")
drop <- c("count.y")
tweets = tweets[,!(names(tweets) %in% drop)]
names(tweets)[names(tweets) == 'count.x'] <- 'count'
colnames(tweets)
```

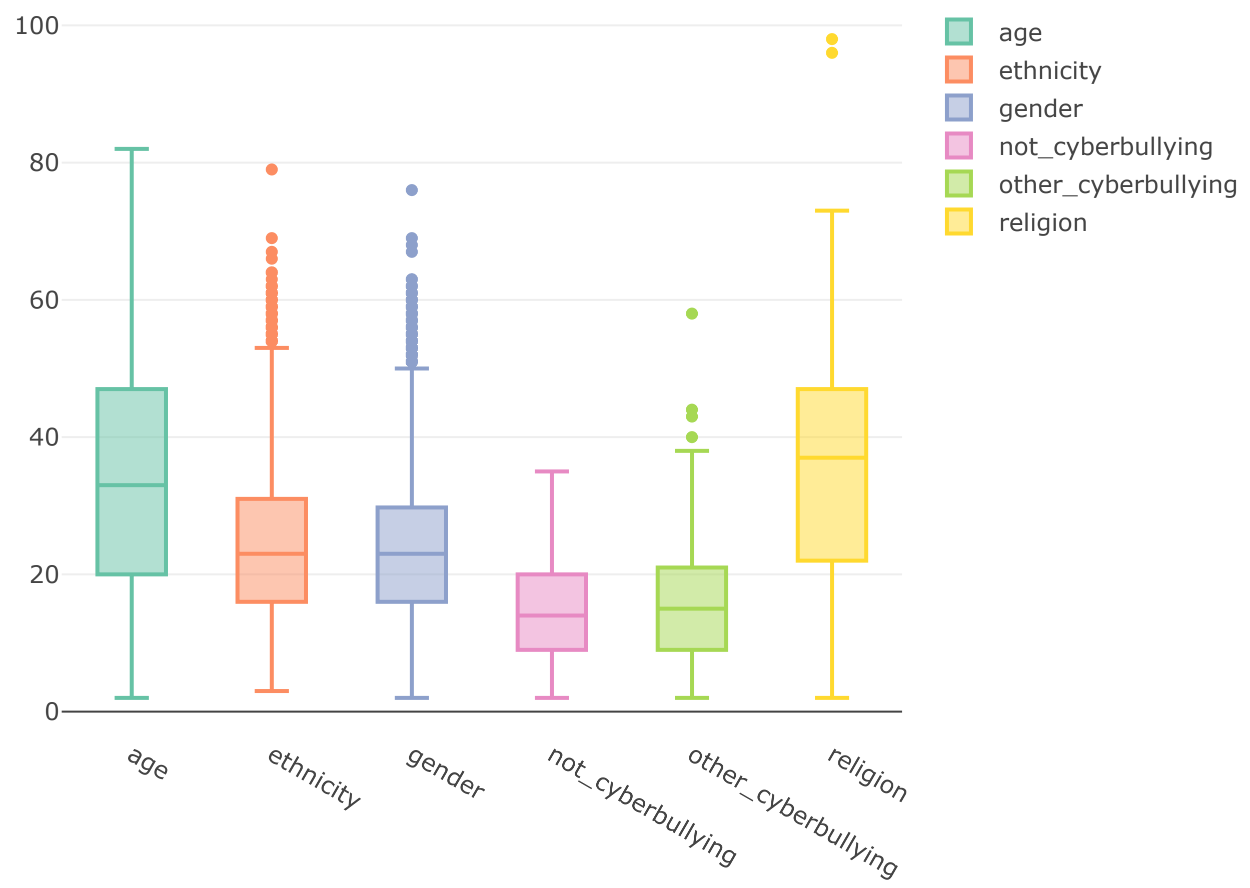
```
## [1] "cyberbullying_type" "tweet_text"         "bullying"
## [4] "word_count"         "count"
```

Box plot!

This box plot will be very useful to understand our classification accuracy later on. I first removed any tweet that is below 1 word or above 100 words. This will make our training model more realistic as there aren't many tweets with 0 or 800 words.

Finally, we can now see that the median value of the tweets fall between 15 to 35 words. Therefore when inputting new text data, It'd be ideal for our word count to be at least 10 for best acuracy.

```
tweets <- tweets[1 < tweets$word_count, ]
tweets <- tweets[tweets$word_count < 100, ]
fig2 <- plot_ly(y = tweets$word_count, type = "box", color = tweets$cyberbullying_type)
fig2
```



Adjusting our data frame for our training model!

Here I created a new data frame that has only two columns, the tweet texts and bullying. These are the only columns we will need to train our model.

```
tweets2 <- select(tweets, tweet_text, bullying)
```

Ideally, we would want to have an equal amount of tweets that fall under each category (yes or no) but this will still work for this project.

```
tweets2 %>% count(bullying)
```

```
##   bullying     n
## 1       no  7902
## 2      yes 39653
```

Training & Testing data sets

Below, we are splitting our data set. 80% of the data will be used for training, and the other 20% will be to test the model for evaluation.

```
training_id <- sample.int(nrow(tweets2), size = nrow(tweets2)*0.8)
training <- tweets2[training_id,]
testing <- tweets2[-training_id,]
```

Words to Tensor

In this chunk of code, text_vectorization will take any text and separate the words into vectors that will become integers to be used for the neural network layers.

```
num_words <- 10000
max_length <- 50
text_vectorization <- layer_text_vectorization(
  max_tokens = num_words,
  output_sequence_length = max_length,
)
```

In the next three chunks of code, text_vectorization will take all the unique words and assign a specific integer for each word.

get_vocabulary will show some of the unique words.

```
text_vectorization %>%
  adapt(tweets2$tweet_text)
```

```
get_vocabulary(text_vectorization)[20:45]
```

```
##  [1] "this"    "was"     "but"     "on"      "like"    "fuck"    "they"
##  [8] "who"     "be"      "with"    "dumb"    "high"    "so"      "all"
## [15] "about"   "have"    "people"  "bullied" "as"      "just"    "your"
## [22] "rt"      "rape"    "if"      "u"       "gay"
```

Here is our first tweet text transformed into a matrix.

```
text_vectorization(matrix(tweets2$tweet_text[1], ncol = 1))
```

```
## tf.Tensor(
## [[ 179   46  385 4655  638   22   18  402    5    6  498    7   18  633
##     10 3152   15  958   23   30   13   25  101  131  754  574  131   14
##     51  379    5   49  131   32    2  119  138 1359   22   18  746 5438
##    395   69  233  230    4  131    0    0]], shape=(1, 50), dtype=int64)
```

Input & Output layer.

The input layer is assigned to be a vector with one column as a string.

The neural network will include our first text_vectorization layer, followed by some advanced layers which include an “activation” layer that is a function of a Sigmoid.

Finally, our model is created.

```
input <- layer_input(shape = c(1), dtype = "string")



output <- input %>%
  text_vectorization() %>%
  layer_embedding(input_dim = num_words + 1, output_dim = 16) %>%
  layer_global_average_pooling_1d() %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dropout(0.5) %>%
  layer_dense(units = 1, activation = "sigmoid")

model <- keras_model(input, output)
```

```
model %>% compile(
  optimizer = 'adam',
  loss = 'binary_crossentropy',
  metrics = list('accuracy')
)
```

Training our model

Here we are training our model to learn which text has the value of bullying as “yes”.

After a few attempts, I found that our accuracy is best at 15 epochs. That is how many times our weights vector change to obtain the lowest number of errors.

```
history <- model %>% fit(
  training$tweet_text,
  as.numeric(training$bullying == "yes"),
  epochs = 15,
  batch_size = 500,
  validation_split = 0.2,
  verbose=2
)
```

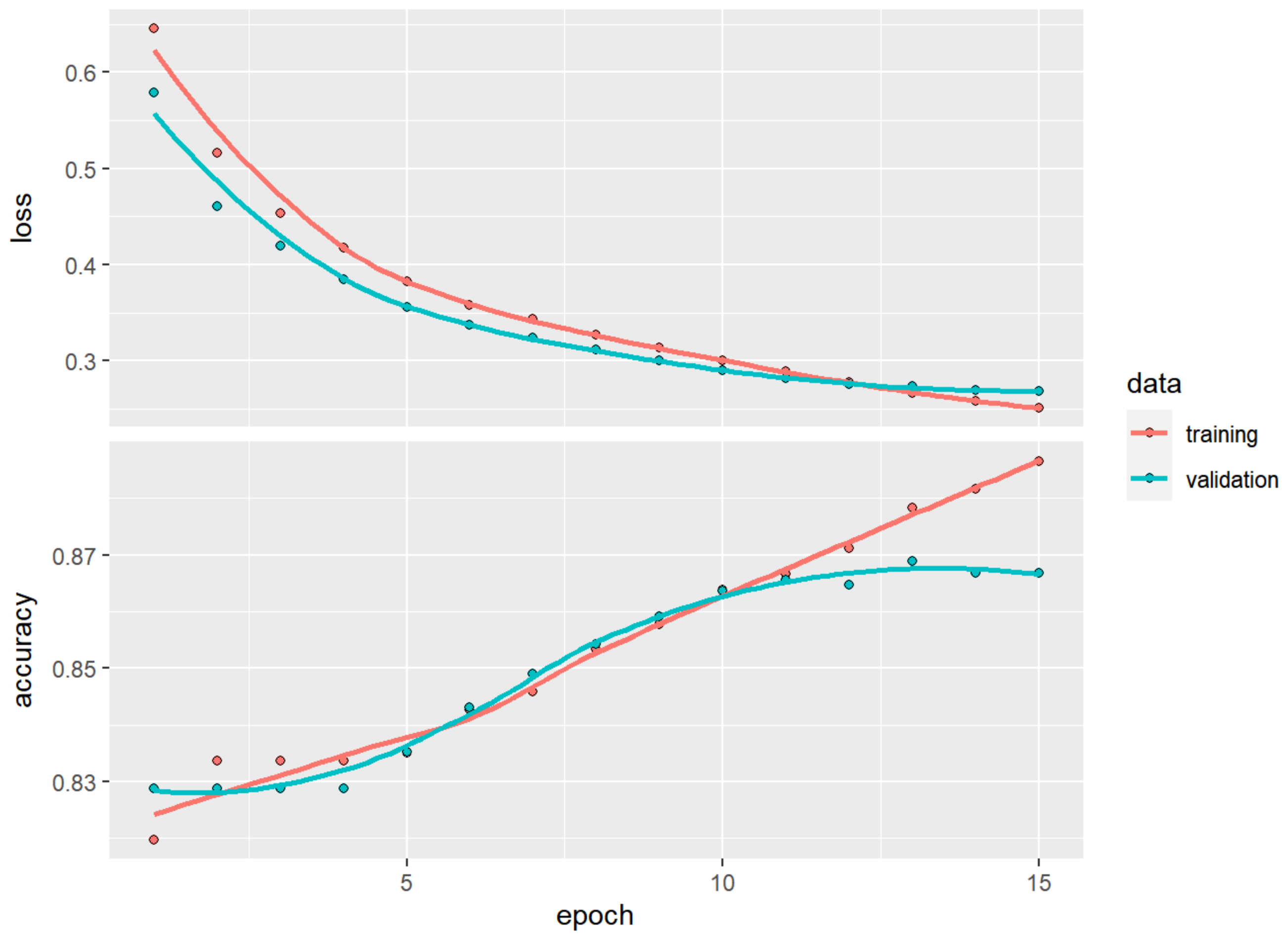
Results of our loss and accuracy

```
results <- model %>% evaluate(testing$tweet_text,
                              as.numeric(testing$bullying == "yes"), verbose = 0)
results
```

```
##      loss  accuracy
## 0.2717188 0.8639470
```

Plot of epochs Vs. Loss & Accuracy.

```
plot(history)
```



Creating an application for our model

Here I created a function that can take in a new input of text, then use the model to output the prediction results for our classification.

```
pred <- function(model,text){
  pred_mod <- round(predict(model,text), 4)
  return <- sprintf("The model is %s percent confident that this text includes bullying", pre
d_mod * 100)
  return(return)
}
```

The Application itself.

```
ui <- fluidPage(
  fluidRow(
    column(6,
           textInput("text_in", label = NULL, value = "", width = 500, placeholder = "Please
enter your text here.")),
    column(5,
verbatimTextOutput('text_out')),
    ))


server <- function(input, output) {
  output$text_out <- renderText({
    pred(model, input$text_in)
    })
}
shinyApp(ui, server)
```

This weather is amazing. Its time to go down the beach..

```
The model is 43.01 percent confident that this text includes bullying
```