Naive Bayes Classification Assignment

Assignment Description

This assignment is designed to test your knowledge of Naive Bayes Classification. It closely mirrors our naive_bayes_penguins.qmd (https://github.com/NSF-ALL-SPICE-Alliance/DS400/blob/main/week7/naive_bayes_penguins.qmd) from lectures 10/1 and 10/3. We reflect back on the true vs fake news dataset from the beginning of the semester and apply the new skills in our bayesian toolbox.

This assignment is worth 16 points and is due by 10:00am on October 15th. Each section has a number of points noted. To turn in this assignment, render this qmd and save it as a pdf, it should look beautiful. If you do not want warning messages and other content in the rendered pdf, you can use `message = FALSE, warning = FALSE` at the top of each code chunk as it appears in the libraries code chunk below.

Load Libraries

```
library(bayesrules)
library(tidyverse)
library(e1071)
library(janitor)
library(skimr)
```

Read in data

```
data(fake_news)
```

Challenge

Exercise 14.7 (https://www.bayesrulesbook.com/chapter-14#exercises-13) **Fake news: three predictors**

Suppose a ***new news article*** is posted online – it has a 15-word title, 6% of its words have negative associations, and its title *doesn't* have an exclamation point. We want to know if it is fake or real

Visualization (Exploratory Data Analysis) - 2 points

Below, insert a code chunk(s) and use `ggplot` to visualize the features of the data we are interested in. This can be one or multiple visualizations

- Type (fake vs real)
- Number of words in the title (numeric value)
- Negative associations (numeric value)
- Exclamation point in the title (true vs false)

```
fake_news_data <- fake_news
```

```
skim(fake_news_data)
```

Data summary

Name	fake_news_data
Number of rows	150

Number of columns	30
_____	
Column type frequency:	
character	4
factor	1
logical	1
numeric	24
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
title	0	1.00	29	136	0	149	0
text	0	1.00	31	31860	0	149	0
url	5	0.97	21	169	0	145	0
authors	27	0.82	6	135	0	84	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
type	0	1	FALSE	2	rea: 90, fak: 60

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
title_has_excl	0	1	0.12	FAL: 132, TRU: 18

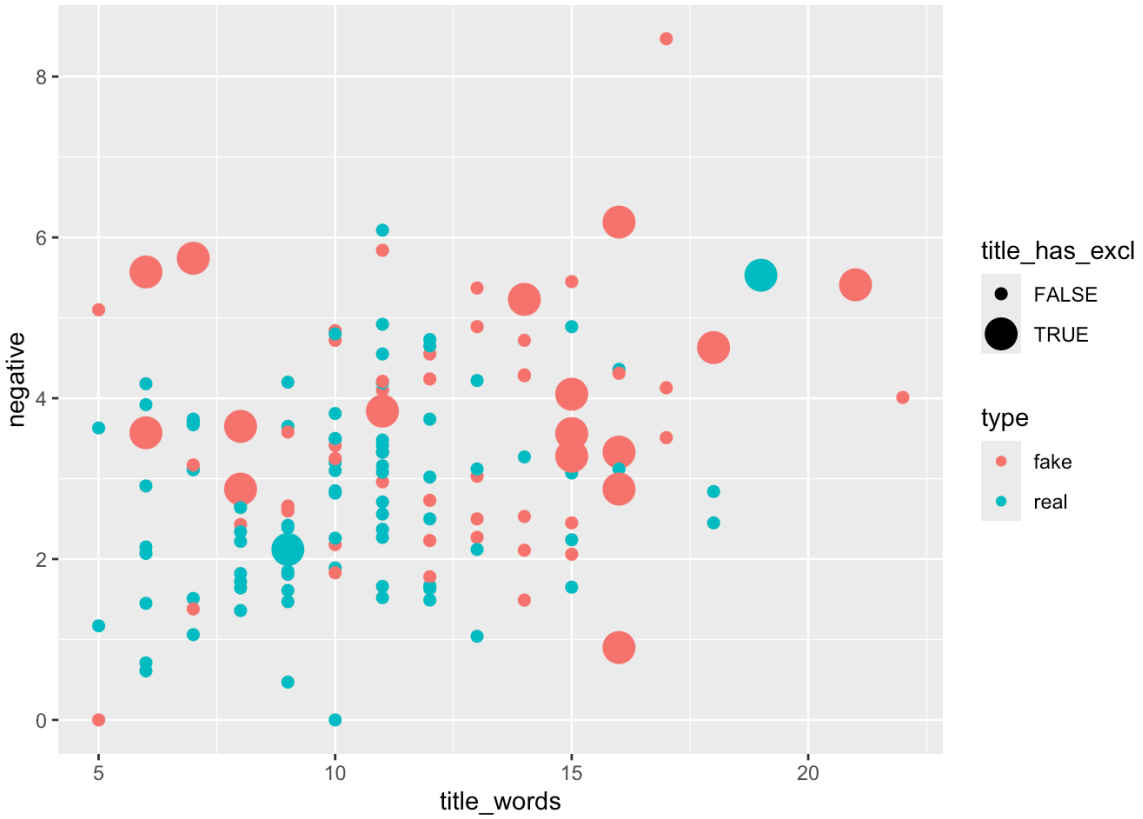
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
title_words	0	1	11.18	3.54	5.00	9.00	11.00	14.00	22.00	
text_words	0	1	533.35	596.44	5.00	253.00	389.00	534.00	5392.00	
title_char	0	1	67.95	20.54	29.00	54.00	65.50	82.75	136.00	
text_char	0	1	3282.56	3604.07	31.00	1527.50	2385.00	3339.25	31860.00	
title_caps	0	1	0.67	1.07	0.00	0.00	0.00	1.00	7.00	
text_caps	0	1	4.77	6.75	0.00	1.00	3.00	5.75	56.00	
title_caps_percent	0	1	5.37	8.34	0.00	0.00	0.00	10.00	43.75	
text_caps_percent	0	1	1.10	1.32	0.00	0.29	0.68	1.52	10.00	
title_excl	0	1	0.15	0.42	0.00	0.00	0.00	0.00	2.00	
text_excl	0	1	0.36	0.88	0.00	0.00	0.00	0.00	8.00	
title_excl_percent	0	1	0.20	0.61	0.00	0.00	0.00	0.00	3.77	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
text_excl_percent	0	1	0.02	0.07	0.00	0.00	0.00	0.00	0.61	▇▁▁▁▁
anger	0	1	1.61	0.92	0.00	0.95	1.52	2.04	4.66	▂▇▇▂▁
anticipation	0	1	1.62	0.71	0.00	1.23	1.56	2.03	4.10	▂▇▇▂▁
disgust	0	1	0.97	0.58	0.00	0.54	0.88	1.39	2.54	▃▇▇▂▁
fear	0	1	1.94	1.21	0.00	0.95	1.66	2.70	5.13	▃▇▆▃▁
joy	0	1	1.07	0.61	0.00	0.61	1.06	1.52	3.12	▃▇▇▂▁
sadness	0	1	1.36	0.80	0.00	0.82	1.29	1.85	4.66	▃▇▅▂▁
surprise	0	1	0.94	0.51	0.00	0.61	0.95	1.16	2.33	▂▇▇▂▁
trust	0	1	3.08	1.75	0.54	2.19	2.89	3.63	20.00	▇▂▁▁▁
negative	0	1	3.13	1.36	0.00	2.21	3.09	4.04	8.47	▂▇▇▂▁
positive	0	1	4.06	1.23	0.00	3.35	4.03	4.55	9.22	▁▃▇▂▁
text_syllables	0	1	912.55	1006.86	10.00	409.50	648.00	945.25	8875.00	▇▁▁▁▁
text_syllables_per_word	0	1	1.72	0.11	1.48	1.63	1.71	1.77	2.12	▁▃▇▂▁

```
ggplot(data = fake_news_data, aes(x = title_words, y = negative, color = type, size = title_has_excl))
+
         geom_point()
```

```
## Warning: Using size for a discrete variable is not advised.
```



Interpretation of Visualization - 2 points

Below, write a few sentences explaining whether or not this ***new news article*** is true or fake solely using your visualization above

Based on the description of the news news article and the visualization created above, I'd say that this new news article is *fake*. It falls around a cluster of points that are fake as well, around 15 words in the title, 6% negative, and it doesn't have an exclamation point.

Perform Naive Bayes Classification - 3 points

Based on these three features (15-word title, 6% of its words have negative associations, and its title *doesn't* have an exclamation point), utilize naive Bayes classification to calculate the posterior probability that the article is real. Do so using `naiveBayes()` with `predict()`.

Below, insert the code chunks and highlight your answer

```
naive_model_hints <- naiveBayes(type ~ title_words + negative + title_has_excl, data = fake_news_data)
```

```
our_article <- data.frame(title_words = 15, negative = 6.00, title_has_excl = FALSE)
```

```
predict(naive_model_hints, newdata = our_article, type = "raw")
```

```
##           fake      real
## [1,] 0.8775779 0.1224221
```

Based on the Naive Bayes model, the article is FAKE.

Break Down the Model - 5 points

Similar to the penguins example, we are going to break down the model we created above. To do this we need to find:

- Probability(15 - word title| article is real) using `dnorm()`
- Probability(6% of words have negative associations | article is real) using `dnorm()`
- Probability(no exclamation point in title | article is real)
    - Multiply these probabilities and save as the object **`probs_real`**
- Probability(15 - word title| article is fake) using `dnorm()`
- Probability(6% of words have negative associations | article is fake) using `dnorm()`
- Probability(no exclamation point in title | article is fake)
    - Multiply these probabilities and save as the object **`probs_fake`**

Lastly divide your **`probs_real`** by the sum of **`probs_real`** and **`probs_fake`** to see if you can reproduce the output from `naiveBayes()` above

```
naive_model_hints
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
## fake real
##  0.4  0.6
##
## Conditional probabilities:
##       title_words
## Y          [,1]     [,2]
##   fake 12.31667 3.743884
##   real 10.42222 3.204554
##
##       negative
## Y          [,1]     [,2]
##   fake 3.606333 1.466429
##   real 2.806556 1.190917
##
##       title_has_excl
## Y           FALSE       TRUE
##   fake 0.73333333 0.26666667
##   real 0.97777778 0.02222222
```

Fake

Title_words

```
dnorm(15, mean = 12.31, sd = 3.74)
```

```
## [1] 0.08235753
```

Negative

```
dnorm(6, mean = 3.61, sd = 1.47)
```

```
## [1] 0.07237491
```

Title_has_excl

```
fake_news_data %>%
  tabyl(type, title_has_excl) %>%
  adorn_percentages("row")
```

```
##  type     FALSE       TRUE
##  fake 0.7333333 0.26666667
##  real 0.9777778 0.02222222
```

```
probs_fake <- (0.08235753*0.07157718*0.73)
probs_fake
```

```
## [1] 0.004303291
```

Real

Title_words

```
dnorm(15, mean = 10.42, sd = 3.20)
```

```
## [1] 0.04476505
```

Negative

```
dnorm(6, mean = 2.81, sd = 1.19)
```

```
## [1] 0.009224487
```

Title_has_excl - 0.98

```
probs_real <- (0.03678032*0.009224487*0.98)
probs_real
```

```
## [1] 0.000332494
```

```
probs_real / (probs_real + probs_fake)
```

```
## [1] 0.07172334
```

Confusion Matrix - 2 points

Calculate a confusion matrix by first mutating a column to fake_news called `predicted_type` . Then, use `tabyl()` to create the matrix

```
fake_news_data <- fake_news_data %>% mutate(predicted_article = predict(naive_model_hints, newdata =
.))
```

```
fake_news_data %>%
  tabyl(type, predicted_article) %>%   adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 2) %>%
  adorn_ns
```

```
##  type        fake        real
##  fake 48.33% (29) 51.67% (31)
##  real 12.22% (11) 87.78% (79)
```

How can our model be improved? - 2 points

Think about the results of the confusion matrix, is the model performing well? Try creating a new model that uses all of the features in the fake_news dataset to make a prediction on type (fake vs true). Then, create a new confusion matrix to see if the model improves.

Based on the confusion matrix, I'd say that the model is performing okay, but can most definitely improve. It struggles with thinking an article is real when it's actually fake, but does well with the real-to-real- articles.

```
naive_model_hints_all_features <- naiveBayes(type ~ ., data = fake_news)
```

```
fake_news_data <- fake_news%>% mutate(predicted_article_2 = predict(naive_model_hints_all_features, ne
wdata = .))
```

```
fake_news_data %>%
  tabyl(type, predicted_article_2) %>%   adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 2) %>%
  adorn_ns
```

```
##  type        fake        real
##  fake 96.67% (58)  3.33%  (2)
##  real  2.22%  (2) 97.78% (88)
```

Considering that we're looking at all the features, the model significantly improved. It would be interesting to see which feature influences the model the most (we could do feature importance and random forest).