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)
```

Read in data

```
data(fake_news)
```

Challenge

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

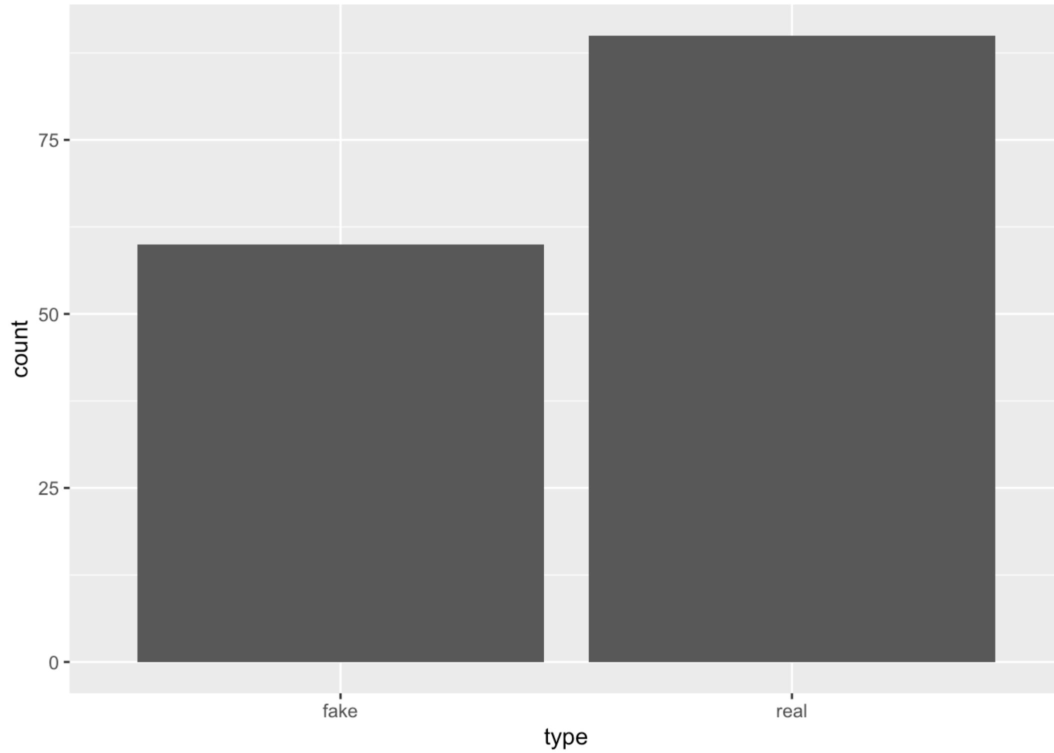
Suppose a ***new news article*** is posted online – it has a 15-word title, 6% of its words have negative associations, and its title *doesn't* have an exclamation point. We want to know if it is fake or real

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)

  ```
  ggplot(data=fake_news, aes(x = type)) +
    geom_bar()
  ```

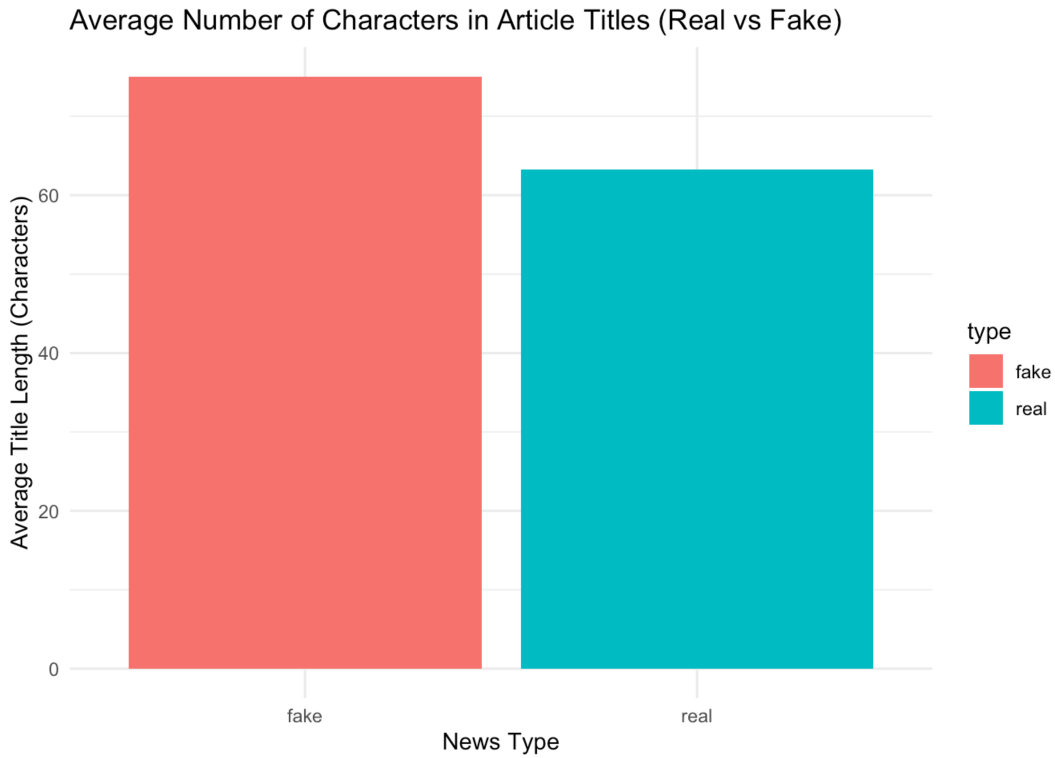


- Number of words in the title (numeric value)

```
fake_news <- fake_news %>%
  mutate(title_length = nchar(title))

average_length <- fake_news %>%
  group_by(type) %>%
  summarise(avg_title_length = mean(title_length))

ggplot(average_length, aes(x = type, y = avg_title_length, fill = type)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Number of Characters in Article Titles (Real vs Fake)",
       x = "News Type",
       y = "Average Title Length (Characters)") +
  theme_minimal()
```



- Negative associations (numeric value)

```
fake_news %>%
  group_by(type) %>%
  summarise(avg_percentage_negative = mean(negative)) %>%
  adorn_totals("row")
```

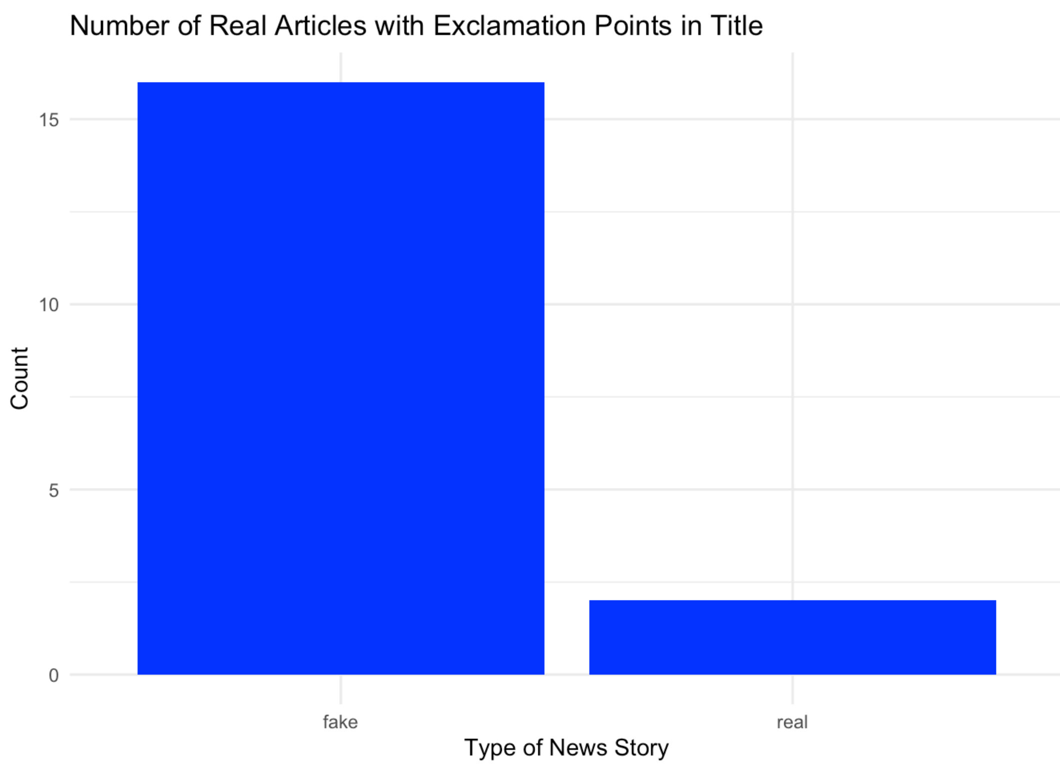
```
##   type avg_percentage_negative
##   fake                3.606333
##   real                2.806556
##  Total                6.412889
```

- Exclamation point in the title (true vs false)

```
fake_news <- fake_news %>%
  mutate(title_has_exclamation = ifelse(grepl("!", title), "Yes", "No"))

exclamation_counts <- fake_news %>%
  filter(title_has_exclamation == "Yes") %>%
  group_by(type) %>%
  summarise(count = n())

ggplot(exclamation_counts, aes(x = type, y = count, fill = type)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Number of Real Articles with Exclamation Points in Title",
       x = "Type of News Story",
       y = "Count") +
  theme_minimal()
```



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

The article is likely fake if over 3% of the words in the article have negative connotation, the title has over 65 characters, or there is an exclamation point in the title

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_article_model <- naiveBayes(type ~ title_char + negative + title_excl, data = fake_news)

sample_article <- data.frame(title_char = 15, negative = 6, title_excl = 0)

predict(naive_article_model, newdata = sample_article, type = "raw")
```

```
##          fake    real
## [1,] 0.44357 0.55643
```

```
fake_news <- fake_news %>%
  mutate(predicted_type = predict(naive_article_model, newdata = .))
```

Based on the three features, the model has predicted a 44% chance of the article being fake and a 56% chance of the article being real.

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()`

```
real_news <- fake_news %>%
  filter(type == "real") %>%
  select(title_char, type, negative, title_excl)

fake_news_ <- fake_news %>%
  filter(type == "fake") %>%
  select(title_char, type, negative, title_excl)

mean(real_news$title_char)
```

```
## [1] 63.23333
```

```
sd(real_news$title_char)
```

```
## [1] 18.794
```

```
dnorm(15, 63.23, 18.8)
```

```
## [1] 0.0007899827
```

- Probability(6% of words have negative associations | article is real) using `dnorm()`

```
mean(real_news$negative)
```

```
## [1] 2.806556
```

```
sd(real_news$negative)
```

```
## [1] 1.190917
```

```
dnorm(6, 2.8, 1.19)
```

```
## [1] 0.009018694
```

- Probability(no exclamation point in title | article is real)

```
mean(real_news$title_excl)
```

```
## [1] 0.03333333
```

```
sd(real_news$title_excl)
```

```
## [1] 0.2346405
```

```
dnorm(0, 0.033, 0.235)
```

```
## [1] 1.680971
```

- Multiply these probabilities and save as the object **`probs_real`**

```
probs_real <- 0.0008 * 0.01 * 1.67
```

- Probability(15 - word title| article is fake) using `dnorm()`

```
mean(fake_news_$title_char)
```

```
## [1] 75.03333
```

```
sd(fake_news_$title_char)
```

```
## [1] 21.16678
```

```
dnorm(15, 75.03, 21.17)
```

```
## [1] 0.0003381964
```

- Probability(6% of words have negative associations | article is fake) using `dnorm()`

```
mean(fake_news_$negative)
```

```
## [1] 3.606333
```

```
sd(fake_news_$negative)
```

```
## [1] 1.466429
```

```
dnorm(6, 3.606, 1.47)
```

```
## [1] 0.07205516
```

- Probability(no exclamation point in title | article is fake)

```
mean(fake_news_$title_excl)
```

```
## [1] 0.3166667
```

```
sd(fake_news_$title_excl)
```

```
## [1] 0.5672314
```

```
dnorm(0, 0.32, 0.57)
```

```
## [1] 0.5978553
```

- Multiply these probabilities and save as the object **`probs_fake`**

```
probs_fake <- 0.0003 * 0.072 * 0.6
```

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

```
sum_probs <- probs_real + probs_fake
probs_real / sum_probs
```

```
## [1] 0.5075988
```

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

```
predicted_type <- fake_news %>%
  tabyl(type, predicted_type) %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting(digits = 2) %>%
  adorn_ns
predicted_type
```

```
##  type        fake        real
##  fake 35.00% (21) 65.00% (39)
##  real  5.56%  (5) 94.44% (85)
```

The model had a difficult time determining if the article was fake, as the confusion matrix shows 65% of fake articles were predicted to be real.

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.

```
fake_news$title_multiple_excl <- ifelse(grepl(".*!.*!", fake_news$title), "yes", "no")

new_naive_article_model <- naiveBayes(type ~ title_char + fear + title_multiple_excl, data = fake_news)

sample_article <- data.frame(title_char = 15, negative = 6, title_multiple_excl = "yes")

predict(new_naive_article_model, newdata = sample_article, type = "raw")
```

```
## Warning in predict.naiveBayes(new_naive_article_model, newdata =
## sample_article, : Type mismatch between training and new data for variable
## 'fear'. Did you use factors with numeric labels for training, and numeric
## values for new data?
```

```
##           fake      real
## [1,] 0.5624097 0.4375903
```

to narrow down the model, I added a column to determine if the article has more than one exclamation point in the title