Exploratory Data Analysis

Table of contents

Title 1
Introduction: 1
Summary 2
Methods and Results 2
Discussion: 16
References 17

Title

Predicting number of shares an article will get based on the number of links, images and videos in the article.

Introduction:

Social media platforms, such as Twitter, have transformed the way in which information is disseminated and consumed on a global scale. These digital arenas enable a vast online community to quickly share and consume personal thoughts and professional insights, threaded through a variety of media forms. With over 500 million active users, Twitter has become particularly noteworthy for pinpointing the workings of information exchange and the features of content that make it most popular López-Goñi and Sánchez-Angulo (2017).

In their research on user activity in Twitter, Oh and Syn (2015) posit that human behavior in everyday life is driven by many factors, and is thus complex and multi-faceted. They uncover a useful lens for examining user activity in casual online environments, while Luo, Freeman, and Stefaniak (2020) report that mobile technology which will exceed 12.3 billion devices by 2022, over 90% of which will be smartphones is increasingly prevalent. This underscores the increasing relevance of social media platforms in the everyday sharing and transmission of information.

This research is fundamentally concerned with discerning the determinants of article popularity, as captured by the number of times each article was shared, to bring to light the many factors that drive user engagement and content diffusion. Using a massive dataset of articles and associated measures over a period of two years from Mashable (Fernandes and Sernadela (2015)) to build models that could more accurately identify the traits that result in the visibility and engagement of content published to social media. By exploring the relationship between shared articles' characteristics and how popular they become, the hope is that this work may yield findings that offer guidance to both enterprise and solo publishers seeking to make their work more prevalent and persuasive in the digital enclave.

```
# Assuming your kernel autoinstalled renv, uncomment the following code
# to install the rest of the packages
# renv::restore()
```

Summary

Our analysis aims to predict the number of shares an article published by Mashable over a period of two years will get based on the number of links, images and videos in the article using knn classification.

Methods and Results

Loading relevant packages.

```
library(kknn)
library(renv)
library(tidymodels)
library(tidyverse)
library(repr)
```

```

Attaching package: 'renv'


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

    embed, update


The following objects are masked from 'package:utils':
    history, upgrade


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

    autoload, load, remove


Attaching packages                      tidymodels 1.1.1

broom        1.0.5      recipes      1.0.10
dials        1.2.0      rsample      1.2.0
dplyr        1.1.4      tibble       3.2.1
ggplot2      3.4.4      tidyr        1.3.0
infer        1.0.6      tune         1.1.2
modeldata    1.3.0      workflows    1.1.4
parsnip      1.2.0      workflowsets 1.0.1
purrr        1.0.2      yardstick    1.3.0

Conflicts                        tidymodels_conflicts()
purrr::discard()  masks scales::discard()
dplyr::filter()   masks stats::filter()
dplyr::lag()      masks stats::lag()
purrr::modify()   masks renv::modify()
recipes::step()   masks stats::step()
recipes::update() masks renv::update(), stats::update()
• Dig deeper into tidy modeling with R at https://www.tmwr.org

Attaching core tidyverse packages              tidyverse 2.0.0
forcats   1.0.0      readr     2.1.4
lubridate 1.9.3      stringr   1.5.1
Conflicts                         tidyverse_conflicts()
readr::col_factor() masks scales::col_factor()
purrr::discard()    masks scales::discard()
dplyr::filter()     masks stats::filter()
stringr::fixed()    masks recipes::fixed()
dplyr::lag()        masks stats::lag()
purrr::modify()     masks renv::modify()
readr::spec()       masks yardstick::spec()
Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

Reading data from dataset.

```r
url <- "https://archive.ics.uci.edu/static/public/332/online+news+popularity.zip"
download.file(url,"articles.zip")
temp_zip <- tempfile()
download.file(url, temp_zip)
# Unzip the file in the data directory
unzip(temp_zip, exdir = "data/")
# Cleanup: Remove the temporary zip file
unlink(temp_zip)
directories <- list.dirs("data/", recursive = FALSE)

# List the files within the first directory, filtering for CSV files
csv_files <- list.files(directories[1], pattern = "\\.csv$", full.names = TRUE)

articles <- read.csv(csv_files, header = TRUE)
```

Creating a boxplot of the variable of interest

```r
ggplot(data = articles) +
    geom_boxplot(aes(y = shares)) +
    labs(title = "Boxplot of Shares") +
    ylab(label = 'Shares') +
    theme(axis.title = element_text(size = 20), axis.text = element_text(size = 15), title =
```

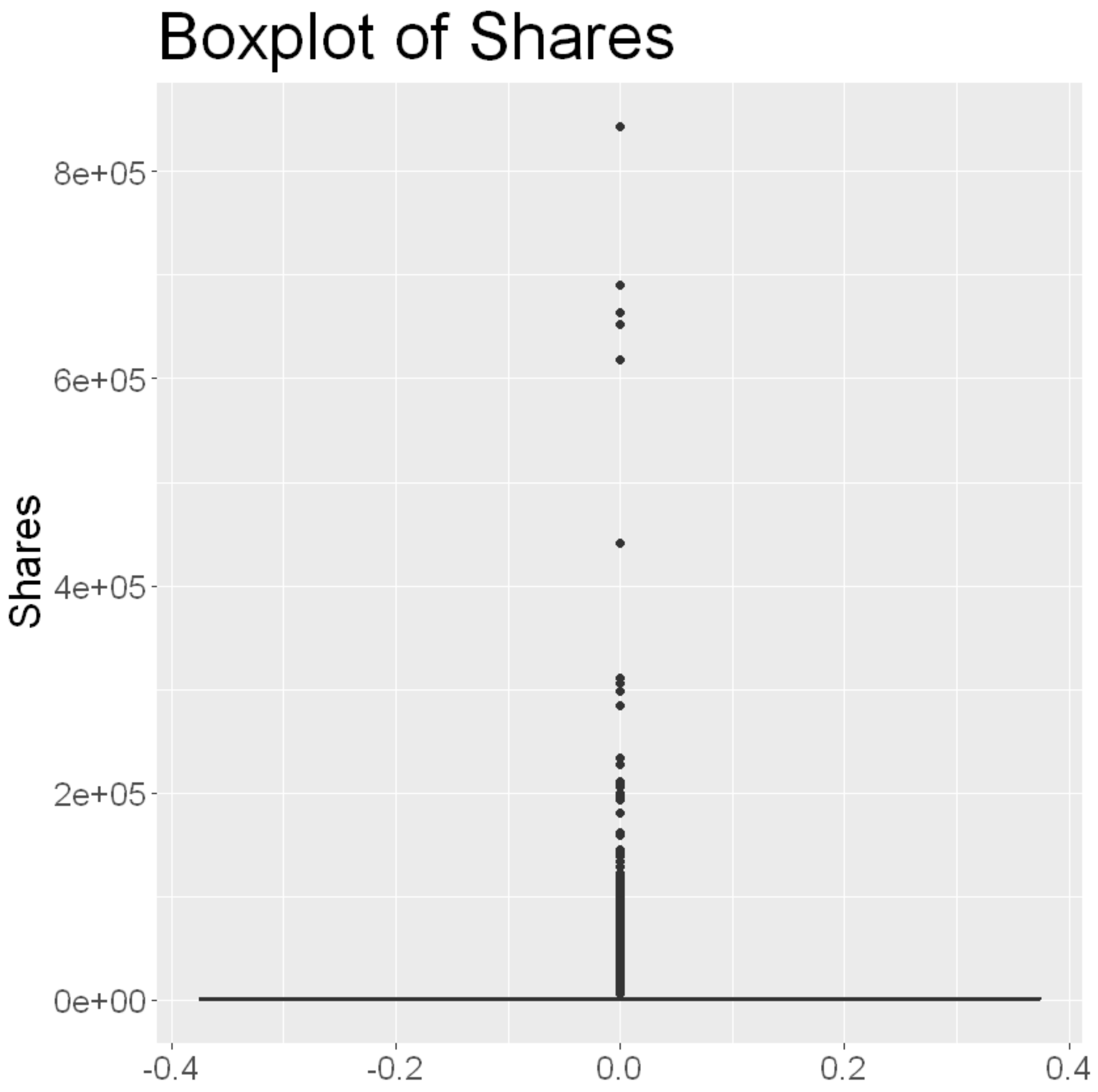

Figure 1. Boxplot of the variable of interest, shares

Obtaining summary statistics about the variable of interest.

```r
summary(articles$shares)
sd(articles$shares)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1     946    1400    3395    2800  843300
```

11626.9507486517

The variable of interest, shares, has a median of 1400 and a mean of 3395. The distribution spread is quite large, with a standard deviation of 11627. We will be using knn classification to determine whether an article will be popular. We will create a new variable called is_popular with two values 1 (popular) and 0 (unpopular)“. An article is classified as popular if it has equal to or more than 1400 shares and unpopular if it has less than 1400 shares. The model aims to predict whether an article will be popular based on the number of links, images, and videos in it. The dataset is publicly available and has 61 attributes out of which we chose 4 including the three predictors and the variable of interest, shares, which refers to the number of shares of the article in social networks.

```r
articles_clean <- articles |>
      select(shares, num_hrefs, num_imgs, num_videos) |>
      mutate(is_popular = ifelse(shares < 1400, 0, 1)) |>
      mutate(is_popular = as.factor(is_popular))

head(articles_clean)
```

A data.frame: 6 × 5

	shares <int>	num_hrefs <dbl>	num_imgs <dbl>	num_videos <dbl>	is_popular <fct>
1	593	4	1	0	0
2	711	3	1	0	0
3	1500	3	1	0	1
4	1200	9	1	0	0
5	505	19	20	0	0
6	855	2	0	0	0

```r
write.csv(articles_clean, 'data/clean_Articles.csv')
```

Since we will be performing classification, we split our data into a training and a testing set with a 60% proportion.

```r
set.seed(2024)

articles_data_split <- initial_split(articles_clean, prop = 0.6, strata = is_popular)
articles_training <- training(articles_data_split)
articles_testing <- testing(articles_data_split)
```

We start our preliminary data analysis process by examining the number of observations we have in the training set for each class.

```r
num_obs_training <- articles_training |>
      group_by(is_popular) |>
      summarize(n = n()) |>
      mutate(percentage = 100*n/nrow(articles_training))

num_obs_training
```

A tibble: 2 × 3

is_popular <fct>	n <int>	percentage <dbl>
0	11094	46.64088
1	12692	53.35912

Table 1. Class distribution of article popularity in training set

We can see that our data is balanced and since we have close percentages of data in both popular and unpopular classes, with popular holding the majority at 53.4%.

Next, we visualize the distributions of our predictor variables: num_hrefs, num_imgs, and num_videos. We use is_popular as the fill argument.

```r
# Histogram 1: Distribution of the number of links in article
mean_hrefs_plot <- ggplot(articles_training, aes(x = num_hrefs, fill = is_popular)) +
  geom_histogram(color = "black") +
  labs(title = "Distribution of number of links",
       x = "Number of links",
       fill = "Popular")+
  scale_fill_discrete(name = "is_popular", labels = c("Unpopular", "Popular"))


# Histogram 1: Distribution of the number of images in article
mean_imgs_plot <- ggplot(articles_training, aes(x = num_imgs, fill = is_popular)) +
  geom_histogram(color = "black") +
  labs(title = "Distribution of number of images",
       x = "Number of images",
       fill = "Popular")+
  scale_fill_discrete(name = "is_popular", labels = c("Unpopular", "Popular"))
```

```r
# Histogram 1: Distribution of the number of videos in article
mean_videos_plot <- ggplot(articles_training, aes(x = num_videos, fill = is_popular)) +
  geom_histogram(color = "black") +
  labs(title = "Distribution of number of videos",
       x = "Number of videos",
       fill = "Popular")  +
  scale_fill_discrete(name = "is_popular", labels = c("Unpopular", "Popular"))
```

```r
mean_hrefs_plot
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

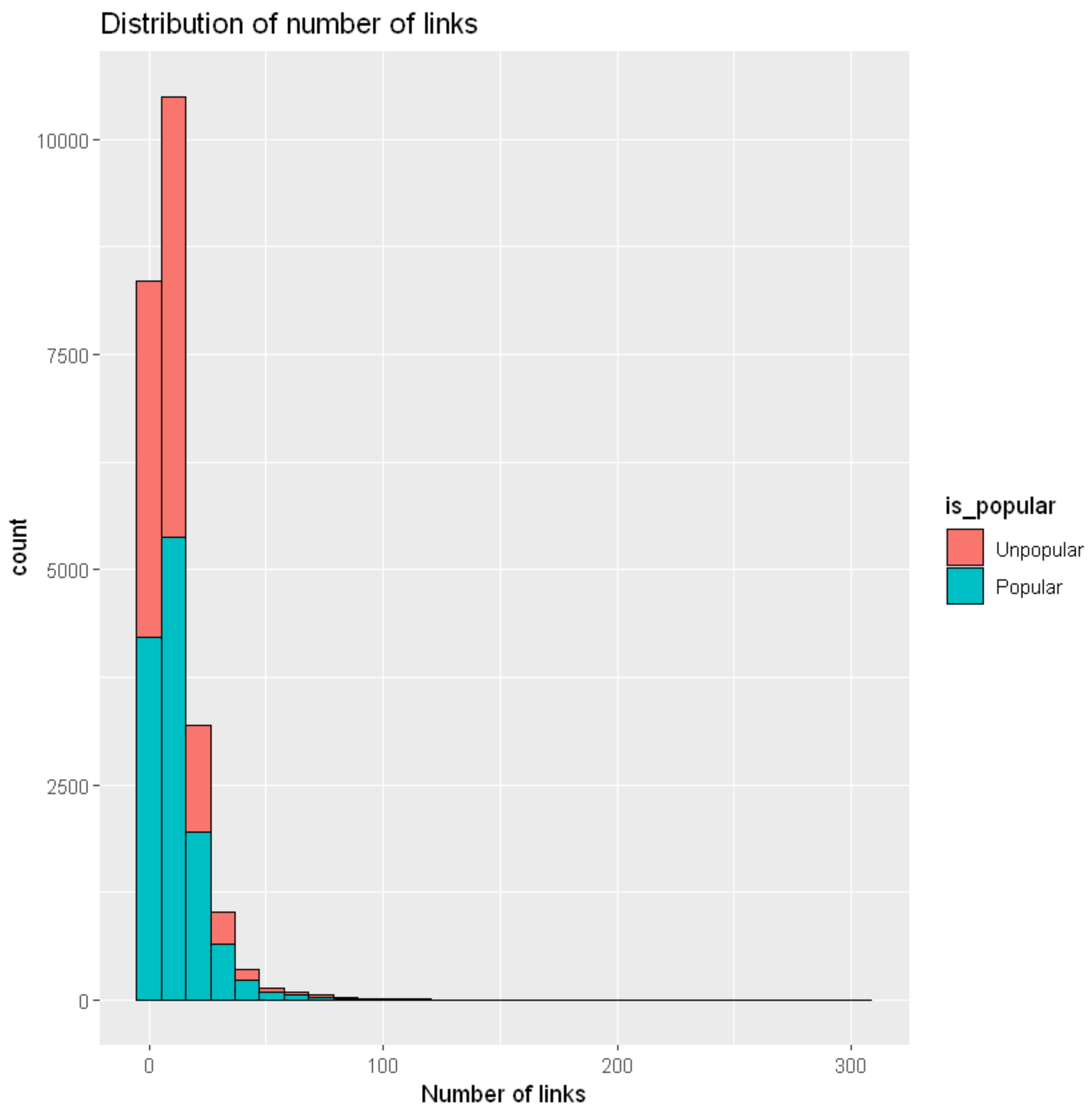

Figure 2. Histogram of number of links in popular and non-Popular articles

```
mean_imgs_plot
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

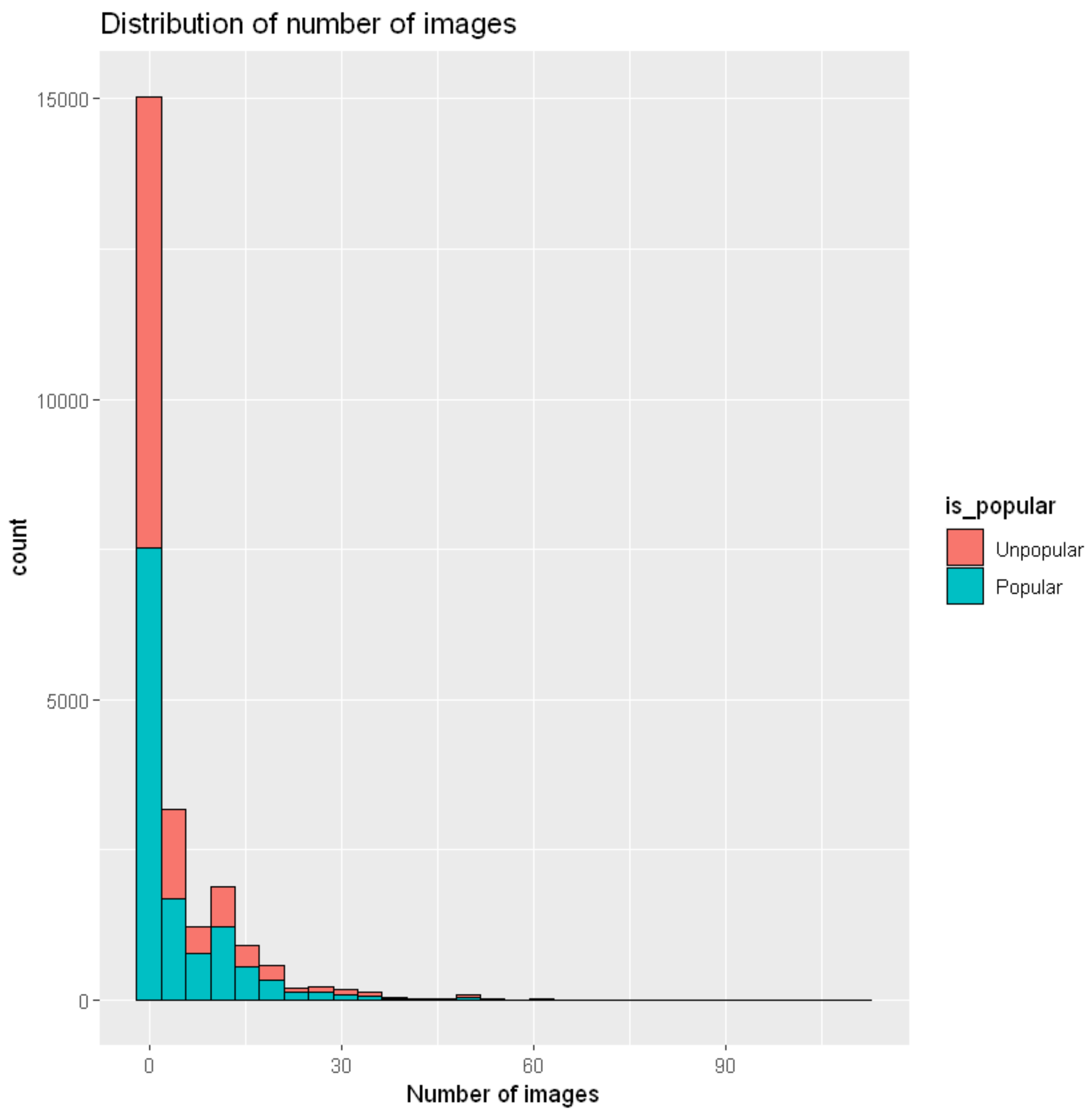

Figure 3. Histogram of number of images in popular and non-Popular articles

```
mean_videos_plot
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

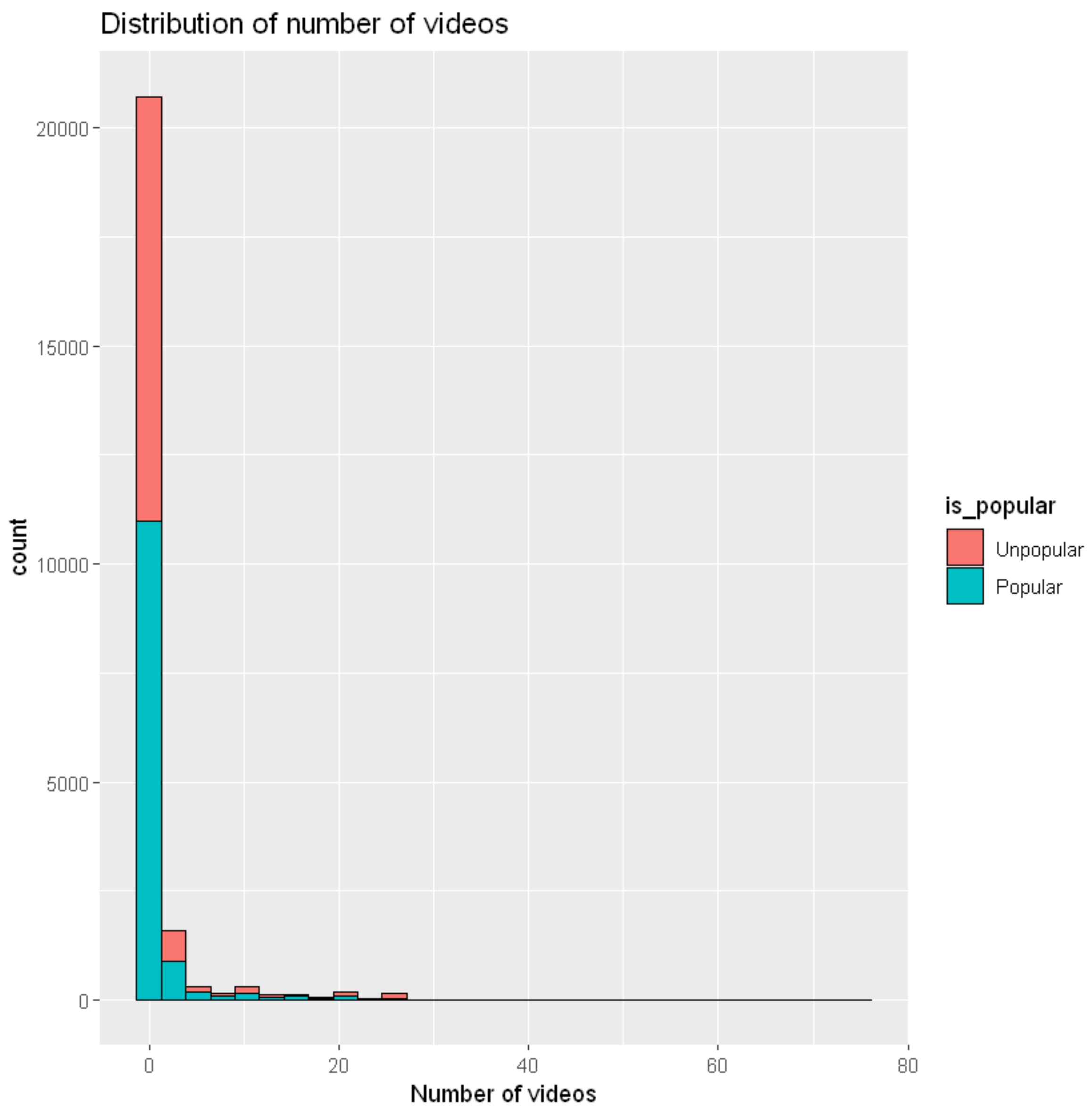

Figure 4. Histogram of number of videos in popular and non-Popular articles

Next, we start building our knn classification model.

We start with making a recipe using the training data.

```
articles_recipe <- recipe(is_popular ~ num_hrefs + num_imgs + num_videos, data = articles_tra
    step_scale(all_predictors()) |>
    step_center(all_predictors())
articles_recipe
```

```
  Recipe


  Inputs

Number of variables by role

outcome:   1
predictor: 3


  Operations

• Scaling for: all_predictors()

• Centering for: all_predictors()
```

Next, we build a tuning model for picking the best value of k.

```r
tune_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = tune()) |>
    set_engine("kknn") |>
    set_mode("classification")
```

Next, we perform cross-validation and create a workflow that calculate the metrics for each of the K values 1, 6, ..., 46, and then return a data frame that shows the accuracy of each K value.

```r
set.seed(2023) # set the seed

# cross-validation
articles_vfold <- vfold_cv(articles_training, v = 5, strata = is_popular)

# create a set of K values
kvals <- tibble(neighbors = seq(from = 1, to = 100, by = 5))

# data analysis workflow
    knn_results <- workflow() |>
    add_recipe(articles_recipe) |>
    add_model(tune_spec) |>
    tune_grid(resamples = articles_vfold, grid = kvals) |>
    collect_metrics()

accuracies <- knn_results |>
    filter(.metric == "accuracy")

print(accuracies)
```

```
# A tibble: 20 × 7
   neighbors .metric  .estimator  mean     n  std_err .config
       <dbl> <chr>    <chr>      <dbl> <int>    <dbl> <chr>
 1         1 accuracy binary     0.495     5 0.00138  Preprocessor1_Model01
 2         6 accuracy binary     0.509     5 0.00347  Preprocessor1_Model02
 3        11 accuracy binary     0.522     5 0.00441  Preprocessor1_Model03
 4        16 accuracy binary     0.526     5 0.00459  Preprocessor1_Model04
 5        21 accuracy binary     0.535     5 0.00166  Preprocessor1_Model05
 6        26 accuracy binary     0.538     5 0.00214  Preprocessor1_Model06
 7        31 accuracy binary     0.538     5 0.00262  Preprocessor1_Model07
 8        36 accuracy binary     0.540     5 0.00197  Preprocessor1_Model08
 9        41 accuracy binary     0.541     5 0.00143  Preprocessor1_Model09
10        46 accuracy binary     0.544     5 0.00182  Preprocessor1_Model10
11        51 accuracy binary     0.545     5 0.00279  Preprocessor1_Model11
12        56 accuracy binary     0.546     5 0.00196  Preprocessor1_Model12
13        61 accuracy binary     0.546     5 0.00285  Preprocessor1_Model13
14        66 accuracy binary     0.544     5 0.00214  Preprocessor1_Model14
15        71 accuracy binary     0.546     5 0.00152  Preprocessor1_Model15
16        76 accuracy binary     0.546     5 0.00182  Preprocessor1_Model16
17        81 accuracy binary     0.545     5 0.00169  Preprocessor1_Model17
18        86 accuracy binary     0.546     5 0.00125  Preprocessor1_Model18
19        91 accuracy binary     0.546     5 0.000648 Preprocessor1_Model19
20        96 accuracy binary     0.548     5 0.00153  Preprocessor1_Model20
```

Table 2. Summary of KNN model accuracy across different K values with 5-fold cross-validation

We now work on making an accuracy vs k plot.

```
best_k_plot <- accuracies |>
    ggplot(aes(x = neighbors, y = mean)) +
    geom_point() +
    geom_line() +
    labs(x = "Number of neighbors", y = "Accuracy Estimate") +
    ggtitle("Accuracy Estimates vs. Number of Neighbors")
best_k_plot
```

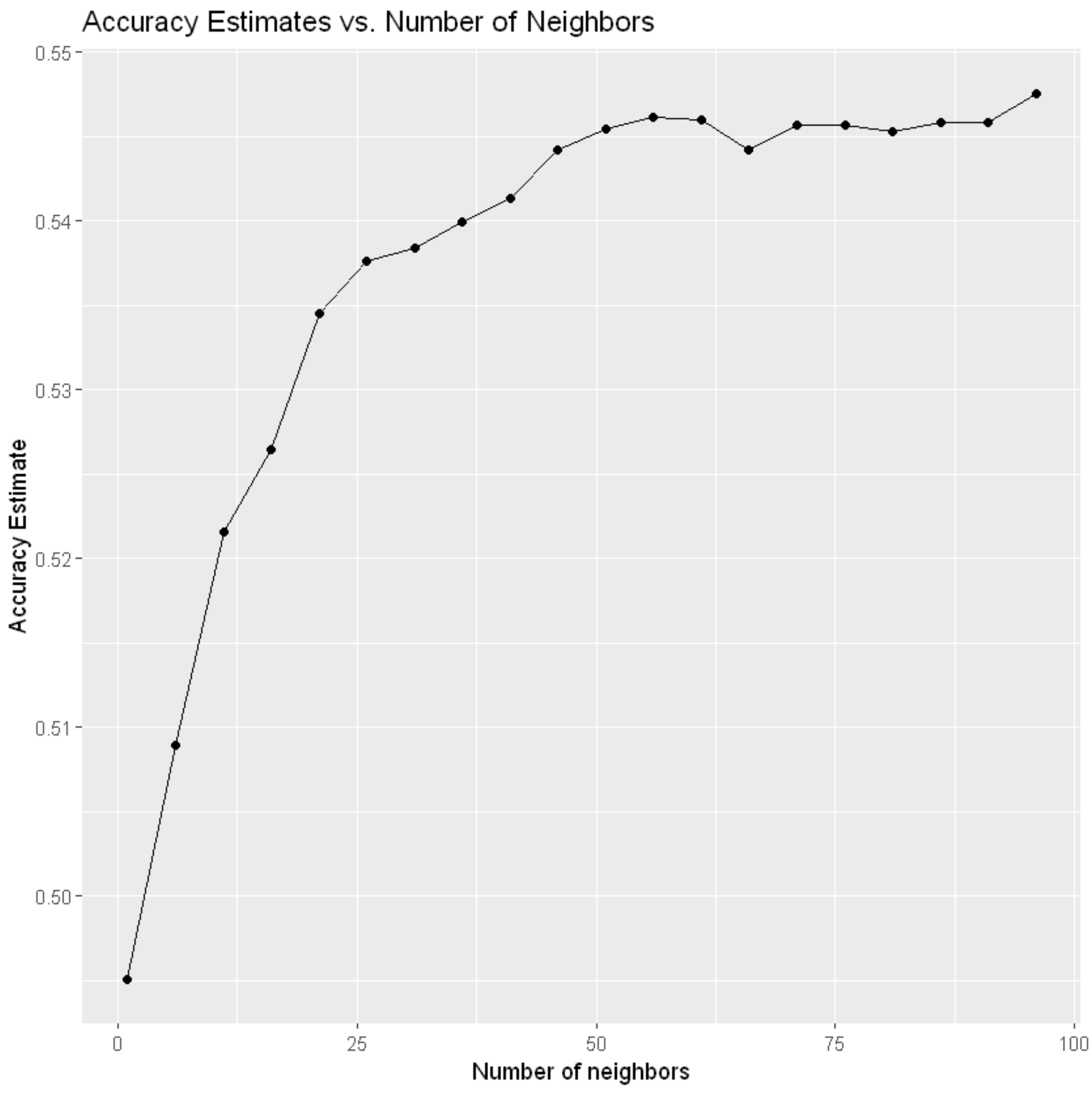

Figure 4. Results from 5-fold cross validation to choose K. (Accuracy was used as the classification metric as K was varied.)

The value of K that generates the highest accuracy is about 50, therefore we choose K = 50 and use that to build the model.

```r
knn_spec <- nearest_neighbor(weight_func = "rectangular", neighbors = 50) |>
    set_engine("kknn") |>
    set_mode("classification")
knn_spec
```

```
K-Nearest Neighbor Model Specification (classification)

Main Arguments:
  neighbors = 50
  weight_func = rectangular

Computational engine: kknn
```

Next, we use a workflow to fit our model using our training set, predict the labels in our testing set, and incorporate the predictions as the new column .pred_class into the testing set.

```r
knn_fit <- workflow() |>
    add_recipe(articles_recipe) |>
    add_model(knn_spec) |>
    fit(data = articles_training)

articles_predict <- predict(knn_fit, articles_testing) |>
    bind_cols(articles_testing)
```

We now look at the accuracy of the classifier using the testing set with the metrics function.

```r
articles_accuracy <- articles_predict |>
    metrics(truth = is_popular, estimate = .pred_class) |>
    filter(.metric == "accuracy")

articles_accuracy
```

A tibble: 1 × 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
accuracy	binary	0.5451507

Table 3. Classifier testing set accuracy

As can be seen, the accuracy of the classifier isn't very high at 54.6%. Below is the confusion matrix of the classifier model.

```
confusion_matrix <- articles_predict |>
    conf_mat(truth = is_popular, estimate = .pred_class)
confusion_matrix
```

```
          Truth
Prediction    0    1
         0 5507 5324
         1 1889 3138
```

Table 4. Confusion matrix of Aarticle popularity predictions on test data

Given that we care more about popular articles with higher shares, we take popular articles as the "positive" observations. The precision is therefore 0.6211 which is slightly higher than the accuracy of 0.546. The recall is 0.3844 which is significantly lower than the accuracy. This suggests that the model is somewhat precise in predicting popular articles but is cautious and misses some actual instances of popular articles in doing so.

Discussion:

Summary: Our analysis aims to predict the number of shares an article published by Mashable over a period of two years will get based on the number of links, images and videos in the article using knn classification. An article is classified as popular if it has equal to or more than 1400 shares and unpopular if it has less than 1400 shares. Our findings suggest that the model is only somwhat reliable with an accuracy of 0.546 with precision higher than accuracy but recall lower than accuracy.

Are the findings as expected: Yes, this is what we expected. Given that a basic KNN model was used for prediction, it's reasonable to assume that this simple model might be slightly underfitting. Consequently, we anticipate achieving improved outcomes by exploring more complex models.

Impact of Findings: This insight has the potential to refine content creation strategies by enriching our comprehension of the elements that contribute to content's engagement and popularity, such as identifying the ideal balance of links, images, and videos within an article. Adopting this strategy could significantly boost the future articles' shareability.

Future Questions from Analysis: This analysis might lead to several Future questions. Firstly, whether there are more model models capable of achieving higher precision and recall. Secondly, the impact of the quality of links, images, and videos within articles on their popularity presents another area for in-depth investigation.

References

Fernandes, Vinagre, Kelwin, and Pedro Sernadela. 2015. "Online News Popularity." UCI Machine Learning Repository.

López-Goñi, Ignacio, and Manuel Sánchez-Angulo. 2017. "Social networks as a tool for science communication and public engagement: focus on Twitter." *FEMS Microbiology Letters* 365 (2): fnx246. https://doi.org/10.1093/femsle/fnx246.

Luo, Tian, Candice Freeman, and Jill Stefaniak. 2020. "'Like, Comment, and Share'—Professional Development Through Social Media in Higher Education: A Systematic Review." *Educational Technology Research and Development* 68 (June). https://doi.org/10.1007/s11423-020-09790-5.

Oh, Sanghee, and Sue Yeon Syn. 2015. "Motivations for Sharing Information and Social Support in Social Media: A Comparative Analysis of Facebook, Twitter, Delicious, YouTube, and Flickr." *Journal of the Association for Information Science and Technology* 66 (10): 2045–60. https://doi.org/https://doi.org/10.1002/asi.23320.