An Ensemble Model to Predict Whether a News Article is Fake News

DA5030

Meseret Ambachew

Spring 2025

Introduction: Business Objective

In today's fragmented and fast-moving media landscape, misinformation spreads rapidly — often faster than the truth. For **public relations firms**, **media outlets**, and **brand managers**, the stakes are high: one fake news article can damage reputations, distort public narratives, and erode trust. As a result, being able to quickly and reliably distinguish between real and fake news isn't just a technical challenge — it's a business imperative.

This project aims to build a predictive model that classifies news articles as *Real* or *Fake* based on a range of data and content features, in other words variables. From a business perspective, such a model has immediate applications: - it could power real-time content verification tools, strengthen media monitoring services, and serve as a safeguard in brand reputation systems. - PR firms could use it to proactively flag misleading stories before they go viral, while media companies might integrate it into editorial pipelines or audience trust platforms.

This project can help identify the signals and behaviors that tend to correlate with misinformation — such as clickbait tendencies, extreme sentiment, or questionable sources. By surfacing these insights, businesses are better equipped to understand and manage the risks posed by false narratives in the public sphere.

Ultimately, this model isn't just about prediction — it's about giving communicators the tools to respond faster, protect their credibility, and make smarter decisions in a world where all information is nuanced.

Data Interpretation

Data Analysis and Prep

Load Library

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(class)
library(knitr)
library(gmodels)
library(e1071)
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(stats)
library(C50)
library(rpart)
library(rpart.plot)
library(stringr)
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:psych':
##
##     outlier
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(klaR)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following objects are masked from 'package:Matrix':
##
##     expand, pack, unpack
```

```r
library(tibble)
```

Load Data

The data is loaded from a public Google Drive url.

```r
# ChatGPT helped me write the correct URL from google drive https://chatgpt.com/c/67fd5ff8-7194-8004-8c

url <- "https://drive.google.com/uc?export=download&id=1f6pPuWv9PuaYgFn4mwL9GV421SHLCE-e"

fake.news.df <- read.csv(url,
                     header = T,
                     stringsAsFactors = F)

summary(fake.news.df)
```

```
##        id           title              author              text
##  Min.   :   1   Length:4000        Length:4000        Length:4000
##  1st Qu.:1001   Class :character   Class :character   Class :character
##  Median :2000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :2000
##  3rd Qu.:3000
##  Max.   :4000
##     state           date_published        source            category
##  Length:4000        Length:4000        Length:4000        Length:4000
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  sentiment_score       word_count       char_count     has_images
##  Min.   :-1.000000   Min.   : 100.0   Min.   : 500   Min.   :0.0000
##  1st Qu.:-0.490000   1st Qu.: 445.8   1st Qu.:2359   1st Qu.:0.0000
##  Median :-0.010000   Median : 793.0   Median :4287   Median :0.0000
##  Mean   :-0.000645   Mean   : 795.7   Mean   :4277   Mean   :0.4965
##  3rd Qu.: 0.510000   3rd Qu.:1150.0   3rd Qu.:6206   3rd Qu.:1.0000
##  Max.   : 1.000000   Max.   :1500.0   Max.   :7996   Max.   :1.0000
##    has_videos     readability_score   num_shares     num_comments
##  Min.   :0.0000   Min.   :30.02     Min.   :   39   Min.   :   0.0
```

```
##  1st Qu.:0.0000   1st Qu.:42.48     1st Qu.:12782   1st Qu.: 238.0
##  Median :0.0000   Median :54.23     Median :25308   Median : 483.0
##  Mean   :0.4845   Mean   :54.76     Mean   :25145   Mean   : 489.9
##  3rd Qu.:1.0000   3rd Qu.:67.22     3rd Qu.:37454   3rd Qu.: 741.0
##  Max.   :1.0000   Max.   :79.98     Max.   :50000   Max.   :1000.0
##  political_bias     fact_check_rating   is_satirical    trust_score
##  Length:4000        Length:4000        Min.   :0.000   Min.   :  0.00
##  Class :character   Class :character   1st Qu.:0.000   1st Qu.: 24.00
##  Mode  :character   Mode  :character   Median :0.000   Median : 50.00
##                                        Mean   :0.497   Mean   : 49.96
##                                        3rd Qu.:1.000   3rd Qu.: 76.00
##                                        Max.   :1.000   Max.   :100.00
##  source_reputation clickbait_score  plagiarism_score    label
##  Min.   : 1.000    Min.   :0.0000   Min.   : 0.04    Length:4000
##  1st Qu.: 3.000    1st Qu.:0.2400   1st Qu.:25.91    Class :character
##  Median : 6.000    Median :0.4900   Median :51.48    Mode  :character
##  Mean   : 5.549    Mean   :0.4944   Mean   :50.60
##  3rd Qu.: 8.000    3rd Qu.:0.7400   3rd Qu.:75.58
##  Max.   :10.000    Max.   :1.0000   Max.   :99.95
```

```r
str(fake.news.df)
```

```
## 'data.frame':    4000 obs. of  24 variables:
##  $ id               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ title            : chr  "Breaking News 1" "Breaking News 2" "Breaking News 3" "Breaking News 4" .
##  $ author           : chr  "Jane Smith" "Emily Davis" "John Doe" "Alex Johnson" ...
##  $ text             : chr  "This is the content of article 1. It contains detailed analysis and repo
##  $ state            : chr  "Tennessee" "Wisconsin" "Missouri" "North Carolina" ...
##  $ date_published   : chr  "30-11-2021" "02-09-2021" "13-04-2021" "08-03-2020" ...
##  $ source           : chr  "The Onion" "The Guardian" "New York Times" "CNN" ...
##  $ category         : chr  "Entertainment" "Technology" "Sports" "Sports" ...
##  $ sentiment_score  : num  -0.22 0.92 0.25 0.94 -0.01 0.83 0.81 -0.96 -0.64 -0.5 ...
##  $ word_count       : int  1302 322 228 155 962 920 651 717 1093 1421 ...
##  $ char_count       : int  5070 2722 5904 825 1087 4022 5652 737 4362 830 ...
##  $ has_images       : int  0 1 0 1 1 0 0 1 0 0 ...
##  $ has_videos       : int  0 0 1 0 0 0 1 0 0 1 ...
##  $ readability_score: num  66.2 41.1 30 75.2 43.9 ...
##  $ num_shares       : int  47305 39804 45860 34222 35934 13148 13627 6035 49000 30508 ...
##  $ num_comments     : int  450 530 763 945 433 28 665 323 881 782 ...
##  $ political_bias   : chr  "Center" "Left" "Center" "Center" ...
##  $ fact_check_rating: chr  "FALSE" "Mixed" "Mixed" "TRUE" ...
##  $ is_satirical     : int  1 1 0 1 0 0 0 1 1 1 ...
##  $ trust_score      : int  76 1 57 18 95 8 1 79 96 88 ...
##  $ source_reputation: int  6 5 1 10 6 1 10 5 7 3 ...
##  $ clickbait_score  : num  0.84 0.85 0.72 0.92 0.66 0.01 0.47 0.58 0.08 0.68 ...
##  $ plagiarism_score : num  53.35 28.28 0.38 32.2 77.7 ...
##  $ label            : chr  "Fake" "Fake" "Fake" "Fake" ...
```

Inspect Missing Values

```r
# The below is a loop to inspect missing values for each variable
cols <- colnames(fake.news.df)

# missing values in each column: iterate
```

```r
for (c in cols) {
  missing.rows <- which(is.na(fake.news.df[,c]))

  num.missing <- length(missing.rows)

  s <- "no"

  if (num.missing > 0)
    s <- num.missing

  print(paste0("Column '", c, "' has ",
               s, " missing values"))

}
```

```
## [1] "Column 'id' has no missing values"
## [1] "Column 'title' has no missing values"
## [1] "Column 'author' has no missing values"
## [1] "Column 'text' has no missing values"
## [1] "Column 'state' has no missing values"
## [1] "Column 'date_published' has no missing values"
## [1] "Column 'source' has no missing values"
## [1] "Column 'category' has no missing values"
## [1] "Column 'sentiment_score' has no missing values"
## [1] "Column 'word_count' has no missing values"
## [1] "Column 'char_count' has no missing values"
## [1] "Column 'has_images' has no missing values"
## [1] "Column 'has_videos' has no missing values"
## [1] "Column 'readability_score' has no missing values"
## [1] "Column 'num_shares' has no missing values"
## [1] "Column 'num_comments' has no missing values"
## [1] "Column 'political_bias' has no missing values"
## [1] "Column 'fact_check_rating' has no missing values"
## [1] "Column 'is_satirical' has no missing values"
## [1] "Column 'trust_score' has no missing values"
## [1] "Column 'source_reputation' has no missing values"
## [1] "Column 'clickbait_score' has no missing values"
## [1] "Column 'plagiarism_score' has no missing values"
## [1] "Column 'label' has no missing values"
```

The data has no missing values. Also from the statistical summary there isn't a huge range between the mean and median, so I will not inspect for outliers and remove them. I'm already using four different models for my ensemble and wouldn't want it to overfit, so I'll continue the next step in my data prep.

Remove Unnecessary Variables

For this model, there is no plan to forecast fake news article as the purpose of this model is to accurately predict news as "real" or "fake", so I will drop the *"date_published"* variable as it won't be necessary. I'll also drop the *"text"* variable as the data reads the same text across 4,000 columns and so does *"title"*.

I'm also dropping the *"author"* variable since some are filled with pseudonyms (John Doe, Jane Doe). Author is also too predictable of an identifier if they are notorious for providing fake or real news. Regardless, we only need one unique identifier for this model which is **id** so we'll keep for tracking.

```r
# Using the piping method from the 'dplyr' package I'm going to remove the variables date_published, au

cleaned_fakenews_df <- fake.news.df %>%
  dplyr::select(-date_published, -title, -text, -author)

head(cleaned_fakenews_df)
```

```
##   id          state          source      category sentiment_score word_count
## 1  1      Tennessee       The Onion Entertainment           -0.22       1302
## 2  2      Wisconsin    The Guardian    Technology            0.92        322
## 3  3       Missouri New York Times        Sports            0.25        228
## 4  4 North Carolina             CNN        Sports            0.94        155
## 5  5     California      Daily Mail    Technology           -0.01        962
## 6  6 North Carolina New York Times        Sports            0.83        920
##   char_count has_images has_videos readability_score num_shares num_comments
## 1       5070          0          0             66.18      47305          450
## 2       2722          1          0             41.10      39804          530
## 3       5904          0          1             30.04      45860          763
## 4        825          1          0             75.16      34222          945
## 5       1087          1          0             43.90      35934          433
## 6       4022          0          0             42.88      13148           28
##   political_bias fact_check_rating is_satirical trust_score source_reputation
## 1         Center             FALSE            1          76                 6
## 2           Left             Mixed            1           1                 5
## 3         Center             Mixed            0          57                 1
## 4         Center              TRUE            1          18                10
## 5          Right             Mixed            0          95                 6
## 6          Right             FALSE            0           8                 1
##   clickbait_score plagiarism_score label
## 1            0.84            53.35  Fake
## 2            0.85            28.28  Fake
## 3            0.72             0.38  Fake
## 4            0.92            32.20  Fake
## 5            0.66            77.70  Real
## 6            0.01            72.10  Fake
```

Variable Encoding: Categorical Variables

For the categorical variables that have over 5 levels, I'll use *frequency encoding* to measure how common real or fake news is among those categories. Those categories or variables are *state*, *source*, and *category (sports, entertainment, etc…)*

Since there are only three levels for *political_bias*, *fact_check_rating*, and the target variable, *label*, I'll use one-hot encoding for it to be clearly identifiable. It's also recommended for the models I'm incorporating (Logistic Regression, kNN)

```r
# Using sapply I called only the categorical variables in my data to give me a summary on their lengths

sapply(cleaned_fakenews_df[sapply(cleaned_fakenews_df, function(x) is.character(x)
                                  || is.factor(x))],function(x) nlevels(as.factor(x)))
```

```
##             state            source          category    political_bias
##                20                13                 6                 3
## fact_check_rating             label
##                 3                 2
```

Frequency Encoding

Below, a loop is used to use frequency encoding for the *state*, *source*, and *category* variables. *maybe include author*

```
vars_to_encode <- c("state", "source", "category")

# Loop created below to apply frequency encoding to each variable listed above

for (var in vars_to_encode) {
  freq_map <- cleaned_fakenews_df %>%
    group_by(.data[[var]]) %>%
    summarise(Frequency = n(), .groups = 'drop') %>%
    mutate(Frequency = Frequency / nrow(cleaned_fakenews_df))  # Relative frequency

  freq_map <- setNames(freq_map$Frequency, freq_map[[var]])

# Replace original column with encoded frequency values
  cleaned_fakenews_df[[var]] <- unname(freq_map[as.character(cleaned_fakenews_df[[var]])])
}

head(cleaned_fakenews_df)
```

```
##   id   state  source category sentiment_score word_count char_count has_images
## 1  1 0.04775 0.07850  0.15825           -0.22       1302       5070          0
## 2  2 0.05075 0.07325  0.15975            0.92        322       2722          1
## 3  3 0.04575 0.08775  0.16100            0.25        228       5904          0
## 4  4 0.04550 0.08600  0.16100            0.94        155        825          1
## 5  5 0.05625 0.07675  0.15975           -0.01        962       1087          1
## 6  6 0.04550 0.08775  0.16100            0.83        920       4022          0
##   has_videos readability_score num_shares num_comments political_bias
## 1          0             66.18      47305          450         Center
## 2          0             41.10      39804          530           Left
## 3          1             30.04      45860          763         Center
## 4          0             75.16      34222          945         Center
## 5          0             43.90      35934          433          Right
## 6          0             42.88      13148           28          Right
##   fact_check_rating is_satirical trust_score source_reputation clickbait_score
## 1             FALSE            1          76                 6            0.84
## 2             Mixed            1           1                 5            0.85
## 3             Mixed            0          57                 1            0.72
## 4              TRUE            1          18                10            0.92
## 5             Mixed            0          95                 6            0.66
## 6             FALSE            0           8                 1            0.01
##   plagiarism_score label
## 1            53.35  Fake
## 2            28.28  Fake
## 3             0.38  Fake
## 4            32.20  Fake
## 5            77.70  Real
## 6            72.10  Fake
```

One-Hot Encoding

Using the piping method,

```r
unique(cleaned_fakenews_df$label)
```

```
## [1] "Fake" "Real"
```

```r
unique(cleaned_fakenews_df$fact_check_rating)
```

```
## [1] "FALSE" "Mixed" "TRUE"
```

```r
unique(cleaned_fakenews_df$political_bias)
```

```
## [1] "Center" "Left"   "Right"
```

```r
cleaned_fakenews_df <- cleaned_fakenews_df %>%
  mutate(
    label = trimws(label),
    fact_check_rating = trimws(fact_check_rating),
    political_bias = trimws(political_bias))

# 'label' encoding
cleaned_fakenews_df <- cleaned_fakenews_df %>%
  mutate(label = ifelse(label == "Real", 1, 0))

# 'political_bias' encoding
cleaned_fakenews_df <- cleaned_fakenews_df %>%
  mutate(political_bias = ifelse(political_bias == "Left", 0,
                          ifelse(political_bias == "Center", 1, 2)))

# 'fact_check_rating' encoding
cleaned_fakenews_df <- cleaned_fakenews_df %>%
  mutate(fact_check_rating = ifelse(fact_check_rating == "FALSE", 0,
                             ifelse(fact_check_rating == "Mixed", 1, 2)))

head(cleaned_fakenews_df)
```

```
##   id   state  source category sentiment_score word_count char_count has_images
## 1  1 0.04775 0.07850  0.15825           -0.22       1302       5070          0
## 2  2 0.05075 0.07325  0.15975            0.92        322       2722          1
## 3  3 0.04575 0.08775  0.16100            0.25        228       5904          0
## 4  4 0.04550 0.08600  0.16100            0.94        155        825          1
## 5  5 0.05625 0.07675  0.15975           -0.01        962       1087          1
## 6  6 0.04550 0.08775  0.16100            0.83        920       4022          0
##   has_videos readability_score num_shares num_comments political_bias
## 1          0             66.18      47305          450              1
## 2          0             41.10      39804          530              0
## 3          1             30.04      45860          763              1
## 4          0             75.16      34222          945              1
## 5          0             43.90      35934          433              2
## 6          0             42.88      13148           28              2
##   fact_check_rating is_satirical trust_score source_reputation clickbait_score
## 1                 0            1          76                 6            0.84
## 2                 1            1           1                 5            0.85
## 3                 1            0          57                 1            0.72
## 4                 2            1          18                10            0.92
## 5                 1            0          95                 6            0.66
## 6                 0            0           8                 1            0.01
##   plagiarism_score label
```

```
## 1            53.35     0
## 2            28.28     0
## 3             0.38     0
## 4            32.20     0
## 5            77.70     1
## 6            72.10     0
```

Normalize Variables with Continuous Data

```r
cont_vars <- c(
  "sentiment_score",
  "word_count",
  "char_count",
  "readability_score",
  "num_shares",
  "num_comments",
  "trust_score",
  "clickbait_score",
  "plagiarism_score")

zNormalize <- function(v) {
  m <- mean(v)
  s <- sd(v)
  return((v - m) / s)
}

cleaned_fakenews_df_2 <- cleaned_fakenews_df %>%
  mutate(across(all_of(cont_vars), zNormalize))
```

Create New Feature

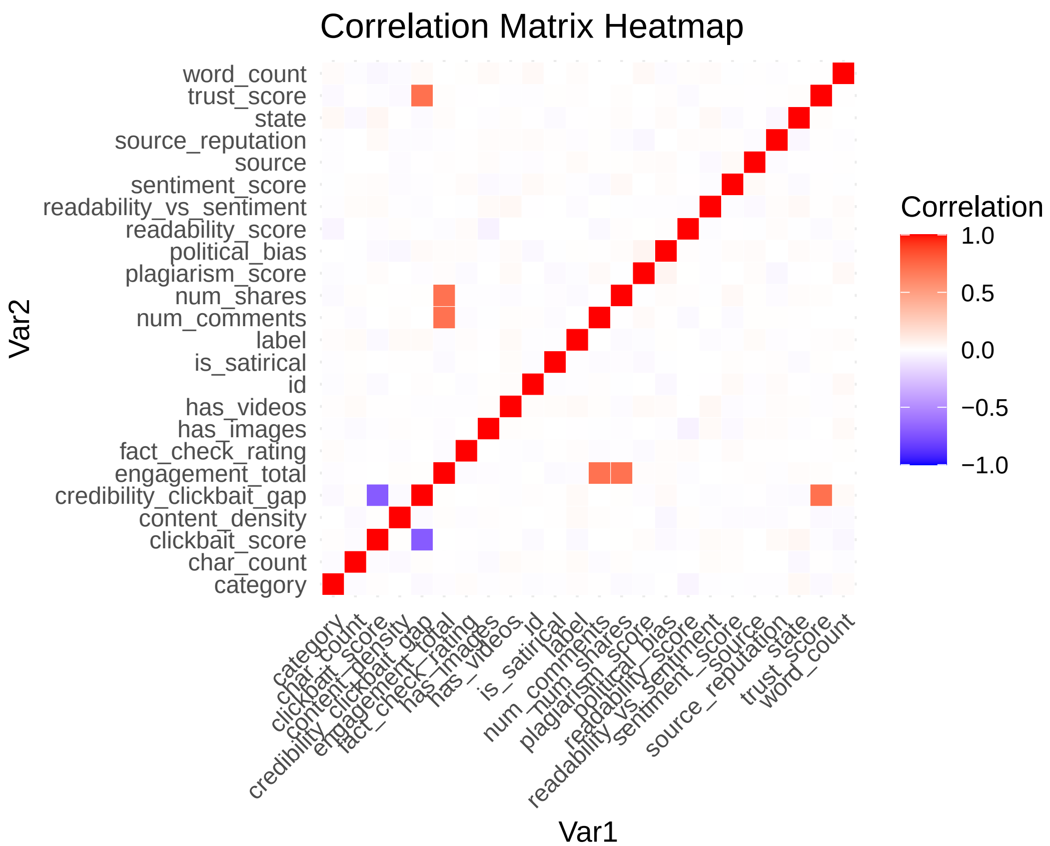
```r
cleaned_fakenews_df_2 <- cleaned_fakenews_df_2 %>%
  mutate(
    credibility_clickbait_gap = trust_score - clickbait_score,
    engagement_total = num_shares + num_comments,
    content_density = char_count / (word_count + 1),  # avoid divide by zero
    readability_vs_sentiment = readability_score * sentiment_score)
```

Examine Multicollinearity

To evaluate multicollinearity among features, I generated a correlation matrix heatmap using all numeric variables in the dataset. The heatmap shows a strong red diagonal, indicating perfect correlation of each variable with itself, which is expected.

```r
numeric_vars <- cleaned_fakenews_df_2 %>% dplyr::select(where(is.numeric))

# Compute base R correlation matrix
cor_matrix <- cor(numeric_vars, use = "complete.obs")

cor_df <- as.data.frame(cor_matrix) %>%
  rownames_to_column(var = "Var1") %>%
  pivot_longer(-Var1, names_to = "Var2", values_to = "Correlation")
```

```r
cor_map <- ggplot(cor_df, aes(x = Var1, y = Var2, fill = Correlation)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(
    low = "blue", high = "red", mid = "white",
    midpoint = 0, limit = c(-1, 1), space = "Lab",
    name = "Correlation"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  coord_fixed() +
  labs(title = "Correlation Matrix Heatmap")

cor_map
```



As a result: - There is **no need to remove or combine** variables based on correlation. - The current feature set is appropriate for use in both linear models (e.g., logistic regression) and non-linear models (e.g., decision trees, ensemble methods).

The rest of the variables in the dataset came numerically as counts, continuous data, or are categorized as binaries (0-1). Meaning this is ready for the next step of this process, **Building the Model**

Model Building

For this project to help predict whether a news article is "Real" or "Fake," I've selected a diverse set of models that offer a balance of interpretability, simplicity, and predictive power:

- **Logistic Regression:** Useful for identifying linear relationships between content features (e.g., trust

score, sentiment) and the article's label
- **kNN (k-Nearest Neighbor):** Helps detect patterns based on proximity in feature space, especially after normalization
- **Decision Tree (CART):** Helps reveal decision paths and thresholds (e.g., trust score $< 50 \rightarrow$ likely fake). Overfitting is compensated by using an *80/20* train-test split and depth tuning.
- **Naive Bayes:** Used to see how well it can predict fake news using structured data like scores and categories. It's a good way to test whether a simple model can still perform well.

Split Data

I'm splitting the data into an *80/20* train-test split to ensure that the models are evaluated fairly. Training on one portion and validating on another helps prevent overfitting and gives a more accurate sense of how the model will perform on unseen data. This approach is a standard practice in machine learning to assess generalization.

80/20 also needs to account for overfitting with the CART Decision Tree Model as the model will also be boosted or the nodes will be adjusted.

```
set.seed(123)  # For reproducibility

# Create 80% sample indices
indices <- sample(1:nrow(cleaned_fakenews_df_2), size = 0.8 * nrow(cleaned_fakenews_df_2))

# Split the data
train_set <- cleaned_fakenews_df_2[indices, ]
val_set <- cleaned_fakenews_df_2[-indices, ]

# verification it was split appropriately
n_train <- nrow(train_set)
n_val <- nrow(val_set)

total_rows_check <- (n_train + n_val) == nrow(cleaned_fakenews_df_2)
total_rows_check
```

```
## [1] TRUE
```

Logistic Regression

```
# calling the "label" target and comparing it to the rest of the variables
fake_log_model <- glm(label ~ ., data = train_set, family = binomial())
```

To reduce the amount of variables with p-values that indicate there is little to no correlation to the Diagnosis, I'm going to use the "stepwise()" to eliminate the variables from my model one by one.

Reduced Logistic Regression Model

```
reduced_model <- step(fake_log_model, direction = "backward")
```

```
## Start:  AIC=4456.02
## label ~ id + state + source + category + sentiment_score + word_count +
##     char_count + has_images + has_videos + readability_score +
##     num_shares + num_comments + political_bias + fact_check_rating +
##     is_satirical + trust_score + source_reputation + clickbait_score +
##     plagiarism_score + credibility_clickbait_gap + engagement_total +
##     content_density + readability_vs_sentiment
```

```
##
##
## Step:  AIC=4456.02
## label ~ id + state + source + category + sentiment_score + word_count +
##     char_count + has_images + has_videos + readability_score +
##     num_shares + num_comments + political_bias + fact_check_rating +
##     is_satirical + trust_score + source_reputation + clickbait_score +
##     plagiarism_score + credibility_clickbait_gap + content_density +
##     readability_vs_sentiment
##
##
## Step:  AIC=4456.02
## label ~ id + state + source + category + sentiment_score + word_count +
##     char_count + has_images + has_videos + readability_score +
##     num_shares + num_comments + political_bias + fact_check_rating +
##     is_satirical + trust_score + source_reputation + clickbait_score +
##     plagiarism_score + content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - sentiment_score           1   4412.1 4454.1
## - id                        1   4412.2 4454.2
## - is_satirical              1   4412.2 4454.2
## - fact_check_rating         1   4412.2 4454.2
## - num_comments              1   4412.3 4454.3
## - has_images                1   4412.3 4454.3
## - readability_score         1   4412.3 4454.3
## - state                     1   4412.3 4454.3
## - word_count                1   4412.5 4454.5
## - source_reputation         1   4412.5 4454.5
## - num_shares                1   4412.7 4454.7
## - source                    1   4412.8 4454.8
## - plagiarism_score          1   4413.0 4455.0
## - trust_score               1   4413.0 4455.0
## - category                  1   4413.1 4455.1
## - political_bias            1   4413.4 4455.4
## - readability_vs_sentiment  1   4414.0 4456.0
## <none>                          4412.0 4456.0
## - clickbait_score           1   4414.2 4456.2
## - char_count                1   4414.8 4456.8
## - content_density           1   4416.1 4458.1
## - has_videos                1   4416.2 4458.2
##
## Step:  AIC=4454.13
## label ~ id + state + source + category + word_count + char_count +
##     has_images + has_videos + readability_score + num_shares +
##     num_comments + political_bias + fact_check_rating + is_satirical +
##     trust_score + source_reputation + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - id                        1   4412.3 4452.3
## - is_satirical              1   4412.3 4452.3
## - fact_check_rating         1   4412.4 4452.4
## - has_images                1   4412.4 4452.4
```

```
## - num_comments             1   4412.4 4452.4
## - readability_score        1   4412.4 4452.4
## - state                    1   4412.4 4452.4
## - word_count               1   4412.6 4452.6
## - source_reputation        1   4412.6 4452.6
## - num_shares               1   4412.8 4452.8
## - source                   1   4412.9 4452.9
## - plagiarism_score         1   4413.1 4453.1
## - trust_score              1   4413.1 4453.1
## - category                 1   4413.2 4453.2
## - political_bias           1   4413.5 4453.5
## - readability_vs_sentiment  1   4414.1 4454.1
## <none>                         4412.1 4454.1
## - clickbait_score          1   4414.3 4454.3
## - char_count               1   4414.9 4454.9
## - content_density          1   4416.3 4456.3
## - has_videos               1   4416.4 4456.4
##
## Step:  AIC=4452.28
## label ~ state + source + category + word_count + char_count +
##     has_images + has_videos + readability_score + num_shares +
##     num_comments + political_bias + fact_check_rating + is_satirical +
##     trust_score + source_reputation + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - is_satirical             1   4412.5 4450.5
## - fact_check_rating        1   4412.5 4450.5
## - has_images               1   4412.5 4450.5
## - num_comments             1   4412.5 4450.5
## - readability_score        1   4412.6 4450.6
## - state                    1   4412.6 4450.6
## - word_count               1   4412.8 4450.8
## - source_reputation        1   4412.8 4450.8
## - num_shares               1   4413.0 4451.0
## - source                   1   4413.1 4451.1
## - plagiarism_score         1   4413.3 4451.3
## - trust_score              1   4413.3 4451.3
## - category                 1   4413.4 4451.4
## - political_bias           1   4413.7 4451.7
## - readability_vs_sentiment  1   4414.2 4452.2
## <none>                         4412.3 4452.3
## - clickbait_score          1   4414.4 4452.4
## - char_count               1   4415.1 4453.1
## - content_density          1   4416.4 4454.4
## - has_videos               1   4416.5 4454.5
##
## Step:  AIC=4450.45
## label ~ state + source + category + word_count + char_count +
##     has_images + has_videos + readability_score + num_shares +
##     num_comments + political_bias + fact_check_rating + trust_score +
##     source_reputation + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
```

```
##                           Df Deviance    AIC
## - fact_check_rating        1   4412.7 4448.7
## - has_images               1   4412.7 4448.7
## - num_comments             1   4412.7 4448.7
## - readability_score        1   4412.7 4448.7
## - state                    1   4412.8 4448.8
## - word_count               1   4412.9 4448.9
## - source_reputation        1   4413.0 4449.0
## - num_shares               1   4413.1 4449.1
## - source                   1   4413.3 4449.3
## - plagiarism_score         1   4413.4 4449.4
## - trust_score              1   4413.5 4449.5
## - category                 1   4413.5 4449.5
## - political_bias           1   4413.9 4449.9
## - readability_vs_sentiment  1   4414.4 4450.4
## <none>                         4412.5 4450.5
## - clickbait_score          1   4414.6 4450.6
## - char_count               1   4415.2 4451.2
## - content_density          1   4416.6 4452.6
## - has_videos               1   4416.6 4452.6
##
## Step:  AIC=4448.67
## label ~ state + source + category + word_count + char_count +
##     has_images + has_videos + readability_score + num_shares +
##     num_comments + political_bias + trust_score + source_reputation +
##     clickbait_score + plagiarism_score + content_density + readability_vs_sentiment
##
##                           Df Deviance    AIC
## - has_images               1   4412.9 4446.9
## - num_comments             1   4412.9 4446.9
## - readability_score        1   4412.9 4446.9
## - state                    1   4413.0 4447.0
## - word_count               1   4413.1 4447.1
## - source_reputation        1   4413.2 4447.2
## - num_shares               1   4413.4 4447.4
## - source                   1   4413.5 4447.5
## - plagiarism_score         1   4413.7 4447.7
## - trust_score              1   4413.7 4447.7
## - category                 1   4413.8 4447.8
## - political_bias           1   4414.1 4448.1
## - readability_vs_sentiment  1   4414.6 4448.6
## <none>                         4412.7 4448.7
## - clickbait_score          1   4414.8 4448.8
## - char_count               1   4415.4 4449.4
## - content_density          1   4416.8 4450.8
## - has_videos               1   4416.8 4450.8
##
## Step:  AIC=4446.91
## label ~ state + source + category + word_count + char_count +
##     has_videos + readability_score + num_shares + num_comments +
##     political_bias + trust_score + source_reputation + clickbait_score +
##     plagiarism_score + content_density + readability_vs_sentiment
##
##                           Df Deviance    AIC
```

```
## - num_comments              1   4413.2 4445.2
## - readability_score         1   4413.2 4445.2
## - state                     1   4413.2 4445.2
## - word_count                1   4413.4 4445.4
## - source_reputation         1   4413.4 4445.4
## - num_shares                1   4413.6 4445.6
## - source                    1   4413.7 4445.7
## - plagiarism_score          1   4413.9 4445.9
## - trust_score               1   4413.9 4445.9
## - category                  1   4414.0 4446.0
## - political_bias            1   4414.3 4446.3
## - readability_vs_sentiment  1   4414.9 4446.9
## <none>                          4412.9 4446.9
## - clickbait_score           1   4415.1 4447.1
## - char_count                1   4415.7 4447.7
## - content_density           1   4417.0 4449.0
## - has_videos                1   4417.0 4449.0
##
## Step:  AIC=4445.16
## label ~ state + source + category + word_count + char_count +
##     has_videos + readability_score + num_shares + political_bias +
##     trust_score + source_reputation + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - readability_score         1   4413.4 4443.4
## - state                     1   4413.5 4443.5
## - word_count                1   4413.6 4443.6
## - source_reputation         1   4413.7 4443.7
## - num_shares                1   4413.8 4443.8
## - source                    1   4414.0 4444.0
## - plagiarism_score          1   4414.1 4444.1
## - trust_score               1   4414.2 4444.2
## - category                  1   4414.3 4444.3
## - political_bias            1   4414.6 4444.6
## <none>                          4413.2 4445.2
## - readability_vs_sentiment  1   4415.2 4445.2
## - clickbait_score           1   4415.3 4445.3
## - char_count                1   4415.9 4445.9
## - content_density           1   4417.3 4447.3
## - has_videos                1   4417.3 4447.3
##
## Step:  AIC=4443.42
## label ~ state + source + category + word_count + char_count +
##     has_videos + num_shares + political_bias + trust_score +
##     source_reputation + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - state                     1   4413.7 4441.7
## - word_count                1   4413.9 4441.9
## - source_reputation         1   4414.0 4442.0
## - num_shares                1   4414.1 4442.1
## - source                    1   4414.2 4442.2
```

```
## - plagiarism_score          1   4414.4 4442.4
## - trust_score               1   4414.5 4442.5
## - category                  1   4414.6 4442.6
## - political_bias            1   4414.9 4442.9
## - readability_vs_sentiment  1   4415.4 4443.4
## <none>                          4413.4 4443.4
## - clickbait_score           1   4415.6 4443.6
## - char_count                1   4416.2 4444.2
## - content_density           1   4417.5 4445.5
## - has_videos                1   4417.5 4445.5
##
## Step:  AIC=4441.74
## label ~ source + category + word_count + char_count + has_videos +
##     num_shares + political_bias + trust_score + source_reputation +
##     clickbait_score + plagiarism_score + content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - word_count                1   4414.2 4440.2
## - source_reputation         1   4414.3 4440.3
## - num_shares                1   4414.4 4440.4
## - source                    1   4414.5 4440.5
## - plagiarism_score          1   4414.7 4440.7
## - trust_score               1   4414.8 4440.8
## - category                  1   4414.9 4440.9
## - political_bias            1   4415.2 4441.2
## - readability_vs_sentiment  1   4415.7 4441.7
## <none>                          4413.7 4441.7
## - clickbait_score           1   4415.8 4441.8
## - char_count                1   4416.5 4442.5
## - content_density           1   4417.8 4443.8
## - has_videos                1   4417.9 4443.9
##
## Step:  AIC=4440.2
## label ~ source + category + char_count + has_videos + num_shares +
##     political_bias + trust_score + source_reputation + clickbait_score +
##     plagiarism_score + content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - source_reputation         1   4414.8 4438.8
## - num_shares                1   4414.9 4438.9
## - source                    1   4415.0 4439.0
## - plagiarism_score          1   4415.2 4439.2
## - trust_score               1   4415.3 4439.3
## - category                  1   4415.4 4439.4
## - political_bias            1   4415.7 4439.7
## - readability_vs_sentiment  1   4416.1 4440.1
## <none>                          4414.2 4440.2
## - clickbait_score           1   4416.3 4440.3
## - char_count                1   4416.9 4440.9
## - content_density           1   4418.2 4442.2
## - has_videos                1   4418.4 4442.4
##
## Step:  AIC=4438.79
## label ~ source + category + char_count + has_videos + num_shares +
```

```
##     political_bias + trust_score + clickbait_score + plagiarism_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - num_shares                1   4415.4 4437.4
## - source                    1   4415.7 4437.7
## - plagiarism_score          1   4415.7 4437.7
## - trust_score               1   4415.9 4437.9
## - category                  1   4416.0 4438.0
## - political_bias            1   4416.2 4438.2
## - readability_vs_sentiment  1   4416.7 4438.7
## <none>                          4414.8 4438.8
## - clickbait_score           1   4417.0 4439.0
## - char_count                1   4417.5 4439.5
## - content_density           1   4418.9 4440.9
## - has_videos                1   4419.0 4441.0
##
## Step:  AIC=4437.45
## label ~ source + category + char_count + has_videos + political_bias +
##     trust_score + clickbait_score + plagiarism_score + content_density +
##     readability_vs_sentiment
##
##                            Df Deviance    AIC
## - source                    1   4416.3 4436.3
## - plagiarism_score          1   4416.3 4436.3
## - trust_score               1   4416.5 4436.5
## - category                  1   4416.7 4436.7
## - political_bias            1   4416.9 4436.9
## - readability_vs_sentiment  1   4417.4 4437.4
## <none>                          4415.4 4437.4
## - clickbait_score           1   4417.6 4437.6
## - char_count                1   4418.1 4438.1
## - content_density           1   4419.5 4439.5
## - has_videos                1   4419.7 4439.7
##
## Step:  AIC=4436.3
## label ~ category + char_count + has_videos + political_bias +
##     trust_score + clickbait_score + plagiarism_score + content_density +
##     readability_vs_sentiment
##
##                            Df Deviance    AIC
## - plagiarism_score          1   4417.1 4435.1
## - trust_score               1   4417.4 4435.4
## - category                  1   4417.5 4435.5
## - political_bias            1   4417.8 4435.8
## - readability_vs_sentiment  1   4418.3 4436.3
## <none>                          4416.3 4436.3
## - clickbait_score           1   4418.5 4436.5
## - char_count                1   4418.9 4436.9
## - content_density           1   4420.3 4438.3
## - has_videos                1   4420.5 4438.5
##
## Step:  AIC=4435.15
## label ~ category + char_count + has_videos + political_bias +
```

```
##     trust_score + clickbait_score + content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - trust_score               1   4418.2 4434.2
## - category                  1   4418.4 4434.4
## - political_bias            1   4418.5 4434.5
## - readability_vs_sentiment  1   4419.1 4435.1
## <none>                          4417.1 4435.1
## - clickbait_score           1   4419.4 4435.4
## - char_count                1   4419.8 4435.8
## - content_density           1   4421.2 4437.2
## - has_videos                1   4421.3 4437.3
##
## Step:  AIC=4434.2
## label ~ category + char_count + has_videos + political_bias +
##     clickbait_score + content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - category                  1   4419.4 4433.4
## - political_bias            1   4419.6 4433.6
## - readability_vs_sentiment  1   4420.2 4434.2
## <none>                          4418.2 4434.2
## - clickbait_score           1   4420.5 4434.5
## - char_count                1   4420.8 4434.8
## - content_density           1   4422.1 4436.1
## - has_videos                1   4422.3 4436.3
##
## Step:  AIC=4433.44
## label ~ char_count + has_videos + political_bias + clickbait_score +
##     content_density + readability_vs_sentiment
##
##                            Df Deviance    AIC
## - political_bias            1   4420.9 4432.9
## - readability_vs_sentiment  1   4421.4 4433.4
## <none>                          4419.4 4433.4
## - clickbait_score           1   4421.7 4433.7
## - char_count                1   4422.0 4434.0
## - content_density           1   4423.4 4435.4
## - has_videos                1   4423.6 4435.6
##
## Step:  AIC=4432.86
## label ~ char_count + has_videos + clickbait_score + content_density +
##     readability_vs_sentiment
##
##                            Df Deviance    AIC
## - readability_vs_sentiment  1   4422.8 4432.8
## <none>                          4420.9 4432.9
## - clickbait_score           1   4423.2 4433.2
## - char_count                1   4423.4 4433.4
## - content_density           1   4424.6 4434.6
## - has_videos                1   4425.1 4435.1
##
## Step:  AIC=4432.84
## label ~ char_count + has_videos + clickbait_score + content_density
```

```
##
##                    Df Deviance    AIC
## <none>                  4422.8 4432.8
## - clickbait_score  1   4425.3 4433.3
## - char_count       1   4425.3 4433.3
## - content_density  1   4426.6 4434.6
## - has_videos       1   4426.7 4434.7
```

Logistic Regression Model Evaluation

```r
# Using the 'caret' package, it gives an easy way to compute a confusion matrix and get metrics like ac

predicted_probabilities_log <- predict(fake_log_model, newdata = val_set, type = "response")

# Convert probabilities to class predictions (threshold = 0.5)
predicted_classes_log <- ifelse(predicted_probabilities_log > 0.5, 1, 0)

# Convert to factor for confusion matrix
predicted_classes_log <- factor(predicted_classes_log, levels = c(0, 1))
actual_classes_log <- factor(val_set$label, levels = c(0, 1))

# confusion matrix
conf_matrix_log <- confusionMatrix(predicted_classes_log, actual_classes_log)

conf_matrix_log
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 236 222
##          1 167 175
##
##                Accuracy : 0.5138
##                  95% CI : (0.4785, 0.5489)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.297972
##
##                   Kappa : 0.0264
##
##  Mcnemar's Test P-Value : 0.006183
##
##             Sensitivity : 0.5856
##             Specificity : 0.4408
##          Pos Pred Value : 0.5153
##          Neg Pred Value : 0.5117
##              Prevalence : 0.5038
##          Detection Rate : 0.2950
##    Detection Prevalence : 0.5725
##       Balanced Accuracy : 0.5132
##
##        'Positive' Class : 0
##
```

The *'fake_log_model'* (originally coded logistic regression model), is 2% more accurate than the reduced model created above. To avoid overfitting, I'll include the original linear regression model in my ensemble.

This is good, becuase this means all the variables are important for my model.

Logistic Regression F1-Score

```
TP_Log <- conf_matrix_log$table[2, 2]
FP_Log <- conf_matrix_log$table[1, 2]
FN_Log <- conf_matrix_log$table[2, 1]

# Calculate precision, recall, and F1
precision_log <- TP_Log / (TP_Log + FP_Log)
recall_log <- TP_Log / (TP_Log + FN_Log)

f1_score_log <- 2 * (precision_log * recall_log) / (precision_log + recall_log)

f1_score_log
```

```
## [1] 0.473613
```

The F1-Score printed above is fairly moderate, meaning it might make false predictions. Which means we will have to rely on the other models created.

Naive Bayes Model

```
# Must be a factor for 'NaiveBayes' to work
train_set$label <- as.factor(train_set$label)
val_set$label <- as.factor(val_set$label)

nb_model <- NaiveBayes(label ~ ., data = train_set)

# ChatGPT instructed me to include "suppressWarnings()" Because a warning was printed across 800 rows
nb_prediction <- suppressWarnings(
  predict(nb_model, val_set[, -which(names(val_set) == "label")]))

# Confusion matrix to check accuracy

conf_matrix_nb <- table(nb_prediction$class, val_set$label)
conf_matrix_nb
```

```
##
##       0   1
##   0 400 394
##   1   3   3
```

The Naive Bayes model predicted the "Real" vs. "Fake" labels with the following results:

- 237 articles were correctly predicted as Fake
- 178 articles were correctly predicted as Real
- 219 Fake articles were incorrectly predicted as Real (false positives)
- 166 Real articles were incorrectly predicted as Fake (false negatives)

This indicates that while the model captures both classes to some extent, it's making a high number of classification errors

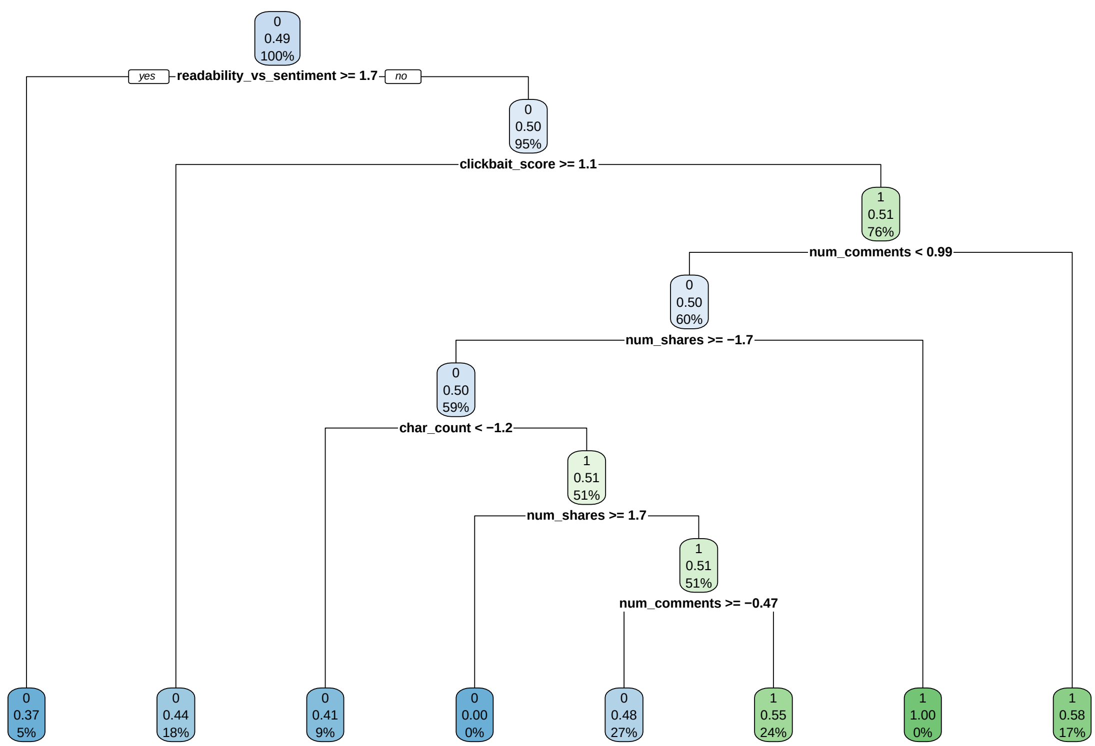
Naive Bayes F1-Score

```
cm <- conf_matrix_nb

TP_NB <- cm[2, 2]
FP_NB <- cm[1, 2]
FN_NB <- cm[2, 1]

# calculate recision and recall
precision_nb <- TP_NB / (TP_NB + FP_NB)
recall_nb <- TP_NB / (TP_NB + FN_NB)

# F1 Score
f1_score_nb <- 2 * (precision_nb * recall_nb) / (precision_nb + recall_nb)

f1_score_nb
```

```
## [1] 0.01488834
```

CART Decision Tree Model

```
fake_cart_model <- rpart(label ~ ., data = train_set, method = "class", parms = list(split = "gini"))

# Plot the tree
fake_cart_plot <- rpart.plot(fake_cart_model)
```




```
fake_cart_plot
```

```
## $obj
## n= 3200
##
```

```
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 3200 1577 0 (0.5071875 0.4928125)
##     2) readability_vs_sentiment>=1.731173 172   63 0 (0.6337209 0.3662791) *
##     3) readability_vs_sentiment< 1.731173 3028 1514 0 (0.5000000 0.5000000)
##       6) clickbait_score>=1.074065 587  257 0 (0.5621806 0.4378194) *
##       7) clickbait_score< 1.074065 2441 1184 1 (0.4850471 0.5149529)
##        14) num_comments< 0.9867588 1912  952 0 (0.5020921 0.4979079)
##          28) num_shares>=-1.715971 1901  941 0 (0.5049974 0.4950026)
##            56) char_count< -1.181987 274  111 0 (0.5948905 0.4051095) *
##            57) char_count>=-1.181987 1627  797 1 (0.4898586 0.5101414)
##             114) num_shares>=1.715367 7    0 0 (1.0000000 0.0000000) *
##             115) num_shares< 1.715367 1620  790 1 (0.4876543 0.5123457)
##               230) num_comments>=-0.4744374 853  410 0 (0.5193435 0.4806565) *
##               231) num_comments< -0.4744374 767  347 1 (0.4524120 0.5475880) *
##          29) num_shares< -1.715971 11    0 1 (0.0000000 1.0000000) *
##        15) num_comments>=0.9867588 529  224 1 (0.4234405 0.5765595) *
##
## $snipped.nodes
## NULL
##
## $xlim
## [1] 0 1
##
## $ylim
## [1] 0 1
##
## $x
##  [1] 0.24908680 0.01770787 0.48046573 0.15545905 0.80547241 0.62897871
##  [7] 0.41374250 0.29321022 0.53427478 0.43096140 0.63758816 0.56871257
## [13] 0.70646375 0.84421492 0.98196610
##
## $y
##  [1] 0.9566448 0.0281108 0.8360560 0.0281108 0.7154672 0.5948783 0.4742895
##  [8] 0.0281108 0.3537007 0.0281108 0.2331118 0.0281108 0.0281108 0.0281108
## [15] 0.0281108
##
## $branch.x
##        [,1]       [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## x 0.2490868 0.01770787 0.4804657 0.1554590 0.8054724 0.6289787 0.4137425
##          NA 0.01770787 0.4804657 0.1554590 0.8054724 0.6289787 0.4137425
##          NA 0.24908680 0.2490868 0.4804657 0.4804657 0.8054724 0.6289787
##        [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## x 0.2932102 0.5342748 0.4309614 0.6375882 0.5687126 0.7064637 0.8442149
##   0.2932102 0.5342748 0.4309614 0.6375882 0.5687126 0.7064637 0.8442149
##   0.4137425 0.4137425 0.5342748 0.5342748 0.6375882 0.6375882 0.6289787
##       [,15]
## x 0.9819661
##   0.9819661
##   0.8054724
##
## $branch.y
##       [,1]       [,2]      [,3]       [,4]      [,5]      [,6]      [,7]
```

```
## y 1.000053 0.07151934 0.8794645 0.07151934 0.7588757 0.6382869 0.5176980
##          NA 0.91067531 0.9106753 0.79008648 0.7900865 0.6694976 0.5489088
##          NA 0.91067531 0.9106753 0.79008648 0.7900865 0.6694976 0.5489088
##          [,8]      [,9]      [,10]     [,11]      [,12]      [,13]      [,14]
## y 0.07151934 0.3971092 0.07151934 0.2765204 0.07151934 0.07151934 0.07151934
##   0.42831997 0.4283200 0.30773114 0.3077311 0.18714231 0.18714231 0.54890881
##   0.42831997 0.4283200 0.30773114 0.3077311 0.18714231 0.18714231 0.54890881
##         [,15]
## y 0.07151934
##   0.66949764
##   0.66949764
##
## $labs
##  [1] "0\n0.49\n100%" "0\n0.37\n5%"   "0\n0.50\n95%"  "0\n0.44\n18%"
##  [5] "1\n0.51\n76%"  "0\n0.50\n60%"  "0\n0.50\n59%"  "0\n0.41\n9%"
##  [9] "1\n0.51\n51%"  "0\n0.00\n0%"   "1\n0.51\n51%"  "0\n0.48\n27%"
## [13] "1\n0.55\n24%"  "1\n1.00\n0%"   "1\n0.58\n17%"
##
## $cex
## [1] 0.45
##
## $boxes
## $boxes$x1
##  [1] 0.2282466347 0.0006238549 0.4630435948 0.1380369155 0.7880502742
##  [6] 0.6115565808 0.3963203693 0.2761262056 0.5168526477 0.4138773810
## [11] 0.6201660293 0.5512904416 0.6890416169 0.8271309071 0.9645439677
##
## $boxes$y1
##  [1]  0.926370431 -0.002163589  0.805781598 -0.002163589  0.685192764
##  [6]  0.564603930  0.444015096 -0.002163589  0.323426262 -0.002163589
## [11]  0.202837429 -0.002163589 -0.002163589 -0.002163589 -0.002163589
##
## $boxes$x2
##  [1] 0.26992696 0.03479189 0.49788786 0.17288118 0.82289454 0.64640084
##  [7] 0.43116463 0.31029424 0.55169691 0.44804541 0.65501029 0.58613470
## [13] 0.72388588 0.86129894 0.99938823
##
## $boxes$y2
##  [1] 1.00005336 0.07151934 0.87946452 0.07151934 0.75887569 0.63828686
##  [7] 0.51769802 0.07151934 0.39710919 0.07151934 0.27652036 0.07151934
## [13] 0.07151934 0.07151934 0.07151934
##
##
## $split.labs
## [1] ""
##
## $split.cex
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $split.box
## $split.box$x1
##  [1] 0.1513655        NA 0.4120252        NA 0.7343023 0.5649705 0.3564044
##  [8]        NA 0.4738567        NA 0.5592378        NA        NA        NA
## [15]        NA
```

```
##
## $split.box$y1
##  [1] 0.8975412        NA 0.7769523        NA 0.6563635 0.5357747 0.4151858
##  [8]        NA 0.2945970        NA 0.1740082        NA        NA        NA
## [15]        NA
##
## $split.box$x2
##  [1] 0.3468081        NA 0.5489063        NA 0.8766425 0.6929869 0.4710806
##  [8]        NA 0.5946928        NA 0.7159386        NA        NA        NA
## [15]        NA
##
## $split.box$y2
##  [1] 0.9238095        NA 0.8032206        NA 0.6826318 0.5620430 0.4414541
##  [8]        NA 0.3208653        NA 0.2002765        NA        NA        NA
## [15]        NA
```

CART Decision Tree Accuracy

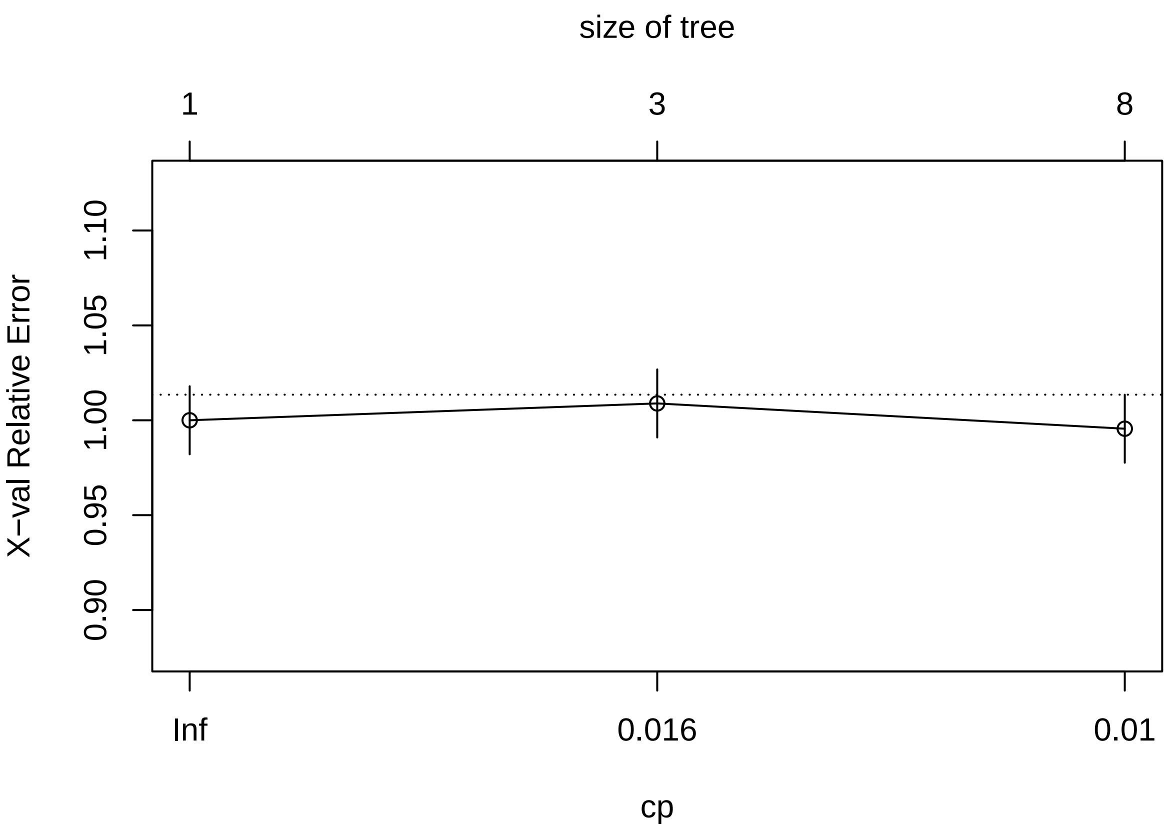
```r
# Predict on validation set
cart_predictions <- predict(fake_cart_model, val_set, type = "class")

# Align factor levels
common_levels <- union(levels(val_set$label), levels(cart_predictions))
cart_predictions <- factor(cart_predictions, levels = common_levels)
val_set$label <- factor(val_set$label, levels = common_levels)

# Create confusion matrix
conf_matrix_cart <- confusionMatrix(cart_predictions, val_set$label)
conf_matrix_table <- table(cart_predictions, val_set$label)

conf_matrix_cart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 237 235
##          1 166 162
##
##                Accuracy : 0.4988
##                  95% CI : (0.4635, 0.534)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.6248549
##
##                   Kappa : -0.0039
##
##  Mcnemar's Test P-Value : 0.0006844
##
##             Sensitivity : 0.5881
##             Specificity : 0.4081
##          Pos Pred Value : 0.5021
##          Neg Pred Value : 0.4939
##              Prevalence : 0.5038
##          Detection Rate : 0.2963
```

```
##    Detection Prevalence : 0.5900
##       Balanced Accuracy : 0.4981
##
##        'Positive' Class : 0
##
```

CART Pruning

```r
full_cart_model <- rpart(label ~ ., data = train_set, method = "class")

# Plot cross-validation error to find optimal CP
plotcp(full_cart_model)
```

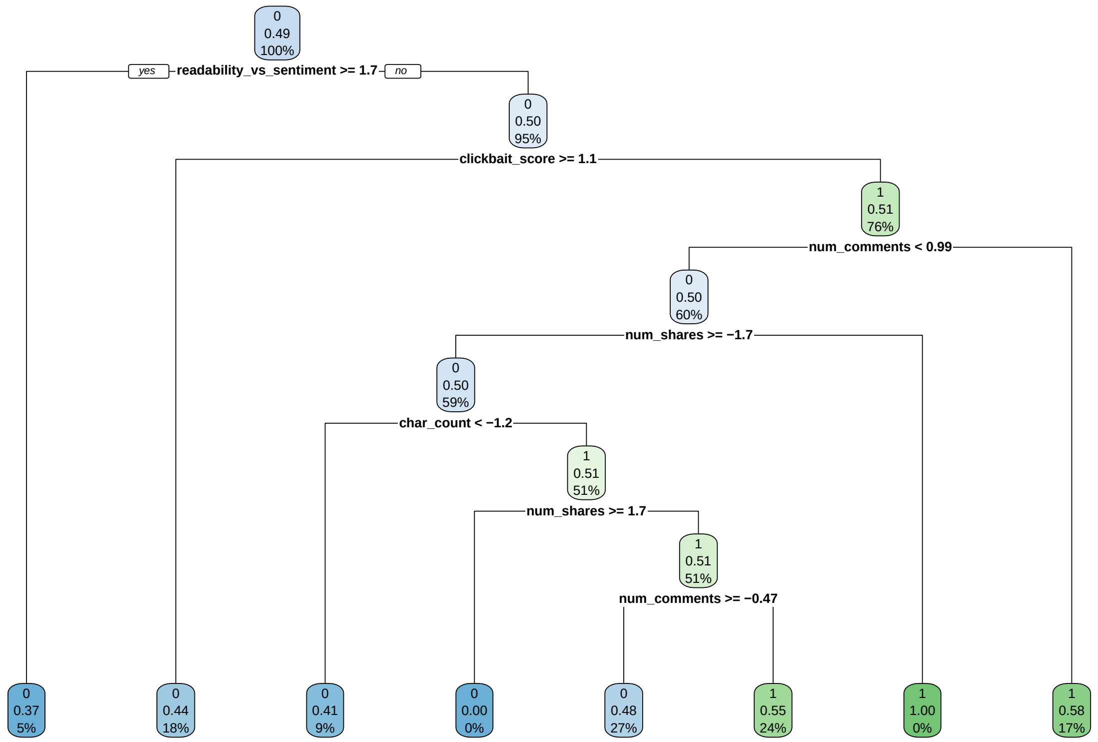



```r
printcp(full_cart_model)
```

```
##
## Classification tree:
## rpart(formula = label ~ ., data = train_set, method = "class")
##
## Variables actually used in tree construction:
## [1] char_count               clickbait_score          num_comments
## [4] num_shares               readability_vs_sentiment
##
## Root node error: 1577/3200 = 0.49281
##
## n= 3200
##
##          CP nsplit rel error  xerror     xstd
```

```
## 1 0.023145      0   1.00000 1.00000 0.017934
## 2 0.010991      2   0.95371 1.00888 0.017935
## 3 0.010000      7   0.89537 0.99556 0.017932
```

```r
# Identify optimal CP value (lowest cross-validated error)
optimal_cp <- full_cart_model$cptable[which.min(full_cart_model$cptable[, "xerror"]), "CP"]

# Prune the tree
pruned_cart_model <- prune(full_cart_model, cp = optimal_cp)

# Plot the pruned tree
rpart.plot(pruned_cart_model)
```




```r
# Predict using both models
original_cart_preds <- predict(full_cart_model, val_set, type = "class")
pruned_cart_preds <- predict(pruned_cart_model, val_set, type = "class")

# Align factor levels
all_levels <- union(levels(val_set$label), unique(c(levels(original_cart_preds), levels(pruned_cart_pre
original_cart_preds <- factor(original_cart_preds, levels = all_levels)
pruned_cart_preds <- factor(pruned_cart_preds, levels = all_levels)
val_set$label <- factor(val_set$label, levels = all_levels)

# Confusion matrices
original_cart_cm <- confusionMatrix(original_cart_preds, val_set$label)
pruned_cart_cm <- confusionMatrix(pruned_cart_preds, val_set$label)

# Print both
original_cart_cm
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 237 235
##          1 166 162
##
##                Accuracy : 0.4988
##                  95% CI : (0.4635, 0.534)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.6248549
##
##                   Kappa : -0.0039
##
##  Mcnemar's Test P-Value : 0.0006844
##
##             Sensitivity : 0.5881
##             Specificity : 0.4081
##          Pos Pred Value : 0.5021
##          Neg Pred Value : 0.4939
##              Prevalence : 0.5038
##          Detection Rate : 0.2963
##    Detection Prevalence : 0.5900
##       Balanced Accuracy : 0.4981
##
##        'Positive' Class : 0
##
```

```
pruned_cart_cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 237 235
##          1 166 162
##
##                Accuracy : 0.4988
##                  95% CI : (0.4635, 0.534)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.6248549
##
##                   Kappa : -0.0039
##
##  Mcnemar's Test P-Value : 0.0006844
##
##             Sensitivity : 0.5881
##             Specificity : 0.4081
##          Pos Pred Value : 0.5021
##          Neg Pred Value : 0.4939
##              Prevalence : 0.5038
##          Detection Rate : 0.2963
##    Detection Prevalence : 0.5900
##       Balanced Accuracy : 0.4981
##
##        'Positive' Class : 0
```

```
##
```

Because CART Decision Tree isn't strong enough, lets try the C5 model...

C5 Model and Accuracy

```r
train_set$label <- as.factor(train_set$label)

# Build the C5.0 model without boosting (trials = 1)
c50_model_no_boost <- C5.0(x = train_set[, -which(names(train_set) == "label")],
                           y = train_set$label,
                           trials = 1)

# Print summary
c50_model_no_boost
```

```
##
## Call:
## C5.0.default(x = train_set[, -which(names(train_set) == "label")], y
##  = train_set$label, trials = 1)
##
## Classification Tree
## Number of samples: 3200
## Number of predictors: 23
##
## Tree size: 18
##
## Non-standard options: attempt to group attributes
```

```r
c50_pred <- predict(c50_model_no_boost, val_set[, -which(colnames(val_set) == "label")])

# Align factor levels
c50_pred <- factor(c50_pred, levels = levels(factor(val_set$label)))
val_set$label <- factor(val_set$label, levels = levels(c50_pred))

# Confusion matrix
c50_conf_matrix <- confusionMatrix(c50_pred, val_set$label)

# View results
c50_conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 224 228
##          1 179 169
##
##                Accuracy : 0.4912
##                  95% CI : (0.4561, 0.5265)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.77110
##
##                   Kappa : -0.0185
##
```

```
##  Mcnemar's Test P-Value : 0.01735
##
##             Sensitivity : 0.5558
##             Specificity : 0.4257
##          Pos Pred Value : 0.4956
##          Neg Pred Value : 0.4856
##              Prevalence : 0.5038
##          Detection Rate : 0.2800
##    Detection Prevalence : 0.5650
##       Balanced Accuracy : 0.4908
##
##        'Positive' Class : 0
##
```

Boosted C5 and Accuracy

```r
c50_model_boost_10 <- C5.0(x = train_set[, -which(colnames(train_set) == "label")],
                           y = train_set$label,
                           trials = 50)

# Predict on validation set
c50_pred_boost_10 <- predict(c50_model_boost_10, val_set[, -which(colnames(val_set) == "label")])

# Align factor levels
c50_pred_boost_10 <- factor(c50_pred_boost_10, levels = levels(factor(val_set$label)))
val_set$label <- factor(val_set$label, levels = levels(c50_pred_boost_10))

# Confusion matrix
c50_conf_matrix_boost_10 <- confusionMatrix(c50_pred_boost_10, val_set$label)
c50_conf_matrix_boost_10
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 194 196
##          1 209 201
##
##                Accuracy : 0.4938
##                  95% CI : (0.4586, 0.529)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.7261
##
##                   Kappa : -0.0123
##
##  Mcnemar's Test P-Value : 0.5510
##
##             Sensitivity : 0.4814
##             Specificity : 0.5063
##          Pos Pred Value : 0.4974
##          Neg Pred Value : 0.4902
##              Prevalence : 0.5038
##          Detection Rate : 0.2425
##    Detection Prevalence : 0.4875
```

```
##       Balanced Accuracy : 0.4938
##
##        'Positive' Class : 0
##
```

*explanation how to boosted model performs the same, but not better than CART, so I'll be using CART.

CART Decision Tree F1 Score

```r
TP_CART <- conf_matrix_table["1", "1"]  # True Positives
TN_CART <- conf_matrix_table["0", "0"]  # True Negatives
FP_CART <- conf_matrix_table["1", "0"]  # False Positives
FN_CART <- conf_matrix_table["0", "1"]  # False Negatives

# Calculate Precision, Recall, and F1 Score
precision_CART <- TP_CART / (TP_CART + FP_CART)
recall_CART <- TP_CART / (TP_CART + FN_CART)
f1_score_CART <- 2 * (precision_CART * recall_CART) / (precision_CART + recall_CART)

f1_score_CART
```

```
## [1] 0.4468966
```

Build kNN Model

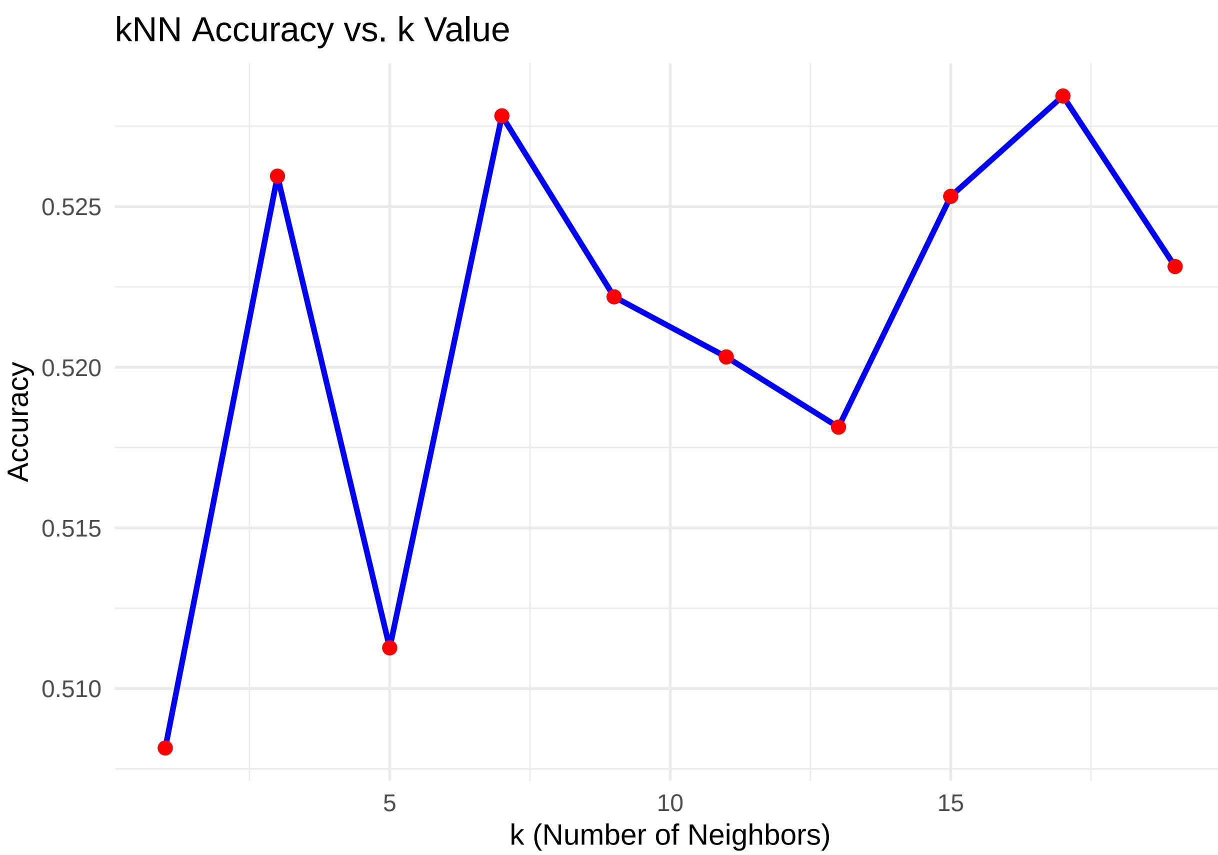
```r
train_set$label <- factor(train_set$label, levels = c(0, 1))
val_set$label <- factor(val_set$label, levels = c(0, 1))

# Optional: drop ID column if you have one (e.g., article_id or similar)
train_knn <- train_set %>% dplyr::select(-id)  # Replace `id` with actual ID col name if needed
val_knn <- val_set %>% dplyr::select(-id)

# 5-fold cross-validation setup
train_control <- trainControl(method = "cv", number = 5)

# Tune k from 1 to 19 (odd numbers)
knn_model <- train(
  label ~ .,
  data = train_knn,
  method = "knn",
  trControl = train_control,
  tuneGrid = expand.grid(k = seq(1, 19, by = 2))  # Try odd k-values only
)

# View results
knn_model
```

```
## k-Nearest Neighbors
##
## 3200 samples
##   22 predictor
##    2 classes: '0', '1'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 2559, 2559, 2561, 2561, 2560
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.5081494  0.01594972
##    3  0.5259458  0.05167943
##    5  0.5112661  0.02221346
##    7  0.5278233  0.05541085
##    9  0.5221900  0.04377325
##   11  0.5203194  0.03980840
##   13  0.5181348  0.03549410
##   15  0.5253169  0.04983532
##   17  0.5284405  0.05621983
##   19  0.5231324  0.04543065
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

```r
# Plot accuracy vs. k
ggplot(knn_model$results, aes(x = k, y = Accuracy)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 2) +
  ggtitle("kNN Accuracy vs. k Value") +
  xlab("k (Number of Neighbors)") +
  ylab("Accuracy") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



kNN Confustion Matrix (Accuracy)

```
best_k <- knn_model$bestTune$k

# Prepare data
train_features <- train_knn %>% dplyr::select(-label)
val_features <- val_knn %>% dplyr::select(-label)
train_labels <- train_knn$label
val_labels <- val_knn$label

# Run kNN with best k
knn_predictions <- knn(
  train = train_features,
  test = val_features,
  cl = train_labels,
  k = best_k
)

# Convert predictions and true labels to factor with same levels
knn_predictions <- factor(knn_predictions, levels = levels(val_labels))
val_labels <- factor(val_labels, levels = levels(knn_predictions))

conf_matrix_kNN <- confusionMatrix(knn_predictions, val_labels)
conf_matrix_kNN
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 229 225
##          1 174 172
##
##                Accuracy : 0.5012
##                  95% CI : (0.466, 0.5365)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.57019
##
##                   Kappa : 0.0015
##
##  Mcnemar's Test P-Value : 0.01231
##
##             Sensitivity : 0.5682
##             Specificity : 0.4332
##          Pos Pred Value : 0.5044
##          Neg Pred Value : 0.4971
##              Prevalence : 0.5038
##          Detection Rate : 0.2863
##    Detection Prevalence : 0.5675
##       Balanced Accuracy : 0.5007
##
##        'Positive' Class : 0
##
```

kNN F1-Score

```
TP_KNN <- conf_matrix_kNN$table[2, 2]
FP_KNN <- conf_matrix_kNN$table[2, 1]
FN_KNN <- conf_matrix_kNN$table[1, 2]

# Precision
precision_KNN <- TP_KNN / (TP_KNN + FP_KNN)

# Recall
recall_KNN <- TP_KNN / (TP_KNN + FN_KNN)

# F1-Score
f1_score_KNN <- 2 * (precision_KNN * recall_KNN) / (precision_KNN + recall_KNN)

f1_score_KNN
```

```
## [1] 0.4629879
```

Build Ensemble

```
# Logistic Regression probabilities
predicted_prob_logit <- predict(fake_log_model, newdata = val_set, type = "response")

# Naive Bayes probabilities
nb_pred_probs <- suppressWarnings(
  predict(nb_model, val_set[, -which(names(val_set) == "label")])$posterior[,2])
```

```r
# CART (Decision tree model with the best accuracy)
predicted_prob_cart <- predict(fake_cart_model, newdata = val_set, type = "prob")[,2]

# kNN: Convert predicted classes to 1s and 0s (since it's not a probabilistic model)
knn_binary_preds <- as.numeric(as.character(knn_predictions))

# Convert all predictions to numeric
predicted_prob_logit <- as.numeric(predicted_prob_logit)
nb_pred_probs <- as.numeric(nb_pred_probs)
predicted_prob_cart <- as.numeric(predicted_prob_cart)

# Average probabilities across all four models
ensemble_probabilities <- (predicted_prob_logit + nb_pred_probs + predicted_prob_cart + knn_binary_preds

# Final class prediction
ensemble_predictions <- ifelse(ensemble_probabilities > 0.5, 1, 0)
ensemble_predictions <- factor(ensemble_predictions, levels = levels(val_set$label))

ensemble_conf_matrix <- confusionMatrix(ensemble_predictions, val_set$label)
ensemble_conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 228 226
##          1 175 171
##
##                Accuracy : 0.4988
##                  95% CI : (0.4635, 0.534)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.62485
##
##                   Kappa : -0.0035
##
##  Mcnemar's Test P-Value : 0.01253
##
##             Sensitivity : 0.5658
##             Specificity : 0.4307
##          Pos Pred Value : 0.5022
##          Neg Pred Value : 0.4942
##              Prevalence : 0.5038
##          Detection Rate : 0.2850
##    Detection Prevalence : 0.5675
##       Balanced Accuracy : 0.4982
##
##        'Positive' Class : 0
##
```

```r
TP_ENSEMBLE <- ensemble_conf_matrix$table[2, 2]
FP_ENSEMBLE <- ensemble_conf_matrix$table[2, 1]
FN_ENSEMBLE <- ensemble_conf_matrix$table[1, 2]

# Precision and Recall
```

```r
precision_ENSEMBLE <- TP_ENSEMBLE / (TP_ENSEMBLE + FP_ENSEMBLE)
recall_ENSEMBLE <- TP_ENSEMBLE / (TP_ENSEMBLE + FN_ENSEMBLE)

# Calculate F1-Score
f1_score_ENSEMBLE <- 2 * (precision_ENSEMBLE * recall_ENSEMBLE) / (precision_ENSEMBLE + recall_ENSEMBLE)

f1_score_ENSEMBLE
```

```
## [1] 0.4602961
```

Final Model Evaluation

Boost and Accuracy of Weighted Ensemble

*Blurb of method used**

```r
ensemble_probabilities_weighted <- (
  0.5 * predicted_prob_logit +
  0.5 * predicted_prob_cart +
  0.5 * nb_pred_probs +
  0.5 * knn_binary_preds)

ensemble_preds_weighted <- ifelse(ensemble_probabilities_weighted > 0.5, 1, 0)
ensemble_preds_weighted <- factor(ensemble_preds_weighted, levels = levels(val_set$label))

conf_matrix_weighted <- confusionMatrix(ensemble_preds_weighted, val_set$label)
conf_matrix_weighted
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   0   0
##          1 403 397
##
##                Accuracy : 0.4963
##                  95% CI : (0.461, 0.5315)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.6771
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.4962
##              Prevalence : 0.5038
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
```

```
##
```

```r
stacking_data <- data.frame(
  logit = predicted_prob_logit,
  cart  = predicted_prob_cart,
  nb    = nb_pred_probs,
  knn   = knn_binary_preds,
  label = val_set$label)

stacked_tree_model <- rpart(label ~ ., data = stacking_data, method = "class")
stacked_tree_preds <- predict(stacked_tree_model, type = "class")
stacked_weighted_conf <- confusionMatrix(stacked_tree_preds, stacking_data$label)
stacked_weighted_conf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 258 155
##          1 145 242
##
##                Accuracy : 0.625
##                  95% CI : (0.5904, 0.6587)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 3.418e-12
##
##                   Kappa : 0.2498
##
##  Mcnemar's Test P-Value : 0.6033
##
##             Sensitivity : 0.6402
##             Specificity : 0.6096
##          Pos Pred Value : 0.6247
##          Neg Pred Value : 0.6253
##              Prevalence : 0.5038
##          Detection Rate : 0.3225
##    Detection Prevalence : 0.5162
##       Balanced Accuracy : 0.6249
##
##        'Positive' Class : 0
##
```

Boost with XGBoost

*Blurb about how it performed with less accuracy**

```r
train_matrix <- train_set %>%
  dplyr::select(-label) %>%
  as.matrix()

train_labels <- as.numeric(as.character(train_set$label))  # Ensuring its numeric

val_matrix <- val_set %>%
  dplyr::select(-label) %>%
  as.matrix()
```

```r
val_labels <- as.numeric(as.character(val_set$label))

xgb_model <- xgboost(
  data = train_matrix,
  label = train_labels,
  nrounds = 150,                  # Number of boosting rounds, doesn't change between 100-150
  objective = "binary:logistic", # Since this is binary classification
  eval_metric = "error",
  verbose = 0                    # Turn off printing during training so that the markdown isn't flooded
)

xgb_probs <- predict(xgb_model, val_matrix)
xgb_preds <- ifelse(xgb_probs > 0.5, 1, 0)
xgb_preds <- factor(xgb_preds, levels = levels(val_set$label))  # Match factor levels
val_set$label <- factor(val_set$label, levels = levels(xgb_preds))

xgb_conf_matrix <- confusionMatrix(xgb_preds, val_set$label)
xgb_conf_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 212 190
##          1 191 207
##
##                Accuracy : 0.5238
##                  95% CI : (0.4885, 0.5588)
##     No Information Rate : 0.5038
##     P-Value [Acc > NIR] : 0.1365
##
##                   Kappa : 0.0475
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.5261
##             Specificity : 0.5214
##          Pos Pred Value : 0.5274
##          Neg Pred Value : 0.5201
##              Prevalence : 0.5038
##          Detection Rate : 0.2650
##    Detection Prevalence : 0.5025
##       Balanced Accuracy : 0.5237
##
##        'Positive' Class : 0
##
```

Final Model Evaluation

Stacked Ensemble Model F1 Score

```
TP_stacked <- stacked_weighted_conf$table[2, 2]
FP_stacked <- stacked_weighted_conf$table[2, 1]
FN_stacked <- stacked_weighted_conf$table[1, 2]

# Calculate precision, recall, and F1
precision <- TP_stacked / (TP_stacked + FP_stacked)
recall <- TP_stacked / (TP_stacked + FN_stacked)
f1_score_stacked <- 2 * (precision * recall) / (precision + recall)
f1_score_stacked
```

```
## [1] 0.6173469
```

Originally, I did not create new features with my data. Once I did the F1 Score for the best model, "stacked_tree_model", improved by 10%.

```
model_comparison_df <- data.frame(
  Model = c("Logistic Regression", "Naive Bayes", "CART", "kNN", "Stacked Ensemble"),
  F1_Score = c(
    round(f1_score_log, 3),
    round(f1_score_nb, 3),
    round(f1_score_CART, 3),
    round(f1_score_KNN, 3),
    round(f1_score_stacked, 3)
  )
)

model_comparison_df
```

```
##                 Model F1_Score
## 1 Logistic Regression    0.474
## 2         Naive Bayes    0.015
## 3                CART    0.447
## 4                 kNN    0.463
## 5    Stacked Ensemble    0.617
```

Deployment

```
comparison_df <- data.frame(
  Actual_Label = val_set$label,
  Predicted_Label = stacked_tree_preds
)

comparison_df$Correct <- ifelse(comparison_df$Actual_Label == comparison_df$Predicted_Label, TRUE, FALSE

head(comparison_df, 10)
```

```
##    Actual_Label Predicted_Label Correct
## 1             0               1   FALSE
## 2             0               0    TRUE
## 3             0               0    TRUE
## 4             0               0    TRUE
## 5             1               1    TRUE
## 6             1               1    TRUE
## 7             1               0   FALSE
```

```
## 8                1                0   FALSE
## 9                0                0    TRUE
## 10               1                0   FALSE
```

Conclusion