Fake News Classification Using Word Frequency Features

Jong Wook Choe
Department of Mathematics, Statistical Data Science
San Francisco State University
jchoe3@sfsu.edu

1 Introduction

Have you ever questioned whether the news you see is real or not? Since the internet became widespread, fake news has increasingly been used to manipulate public opinion. One of the most well-known examples is the Nayirah testimony. On October 10, 1990, a 15-year-old Kuwaiti girl gave false testimony before the United States Congressional Human Rights Caucus. She claimed to be a volunteer nurse at a Kuwaiti hospital during the Iraqi invasion. In her testimony, she said she witnessed Iraqi soldiers removing premature babies from incubators, stealing the equipment, and leaving the babies to die on the floor. This emotional account played a significant role in shaping public support and helped President George H. W. Bush justify military action against Iraq.

However,

"was shown to be almost certainly false by an ABC reporter, John Martin, in March 1991" (The New York Times)

In January 1992, it was revealed that she had never been a nurse and was, in fact, the daughter of Saud Nasser Al-Saud Al-Sabah, the Kuwaiti ambassador to the United States at the time of her testimony. This raises an important question: What should we believe, and what should we not? In an age where misinformation can spread quickly, it's becoming increasingly difficult to know what is true and what is not.

This project aims to build a machine learning model capable of predicting whether a news article is real or fake based solely on the frequency of certain keywords in the article's body text.

2 Data Description

The dataset consists of two CSV files from BuzzFeed:

- `BuzzFeed_real_news_content.csv`
- `BuzzFeed_fake_news_content.csv`

Each dataset includes columns such as `id`, `text`, `title`, `url`, `authors`, and `publish_date`. We combined the datasets and added a binary variable `real_fake` (0 = fake, 1 = real).

After tokenizing the article texts using the `tidytext` package in R, we extracted the top 30 most frequent words with more than 2 syllables, such as "hillary", "president", "debate", "obama", etc. We then counted the number of times each of these words appeared in each article and created new columns such as `word_hillary`, `word_president`, etc.

3 Data Analysis

3.1 Data Preprocessing

To prepare the data for modeling, we performed several preprocessing steps to clean and simplify the dataset. Irrelevant columns such as `title`, `source`, and `id` were removed, as they did not contribute meaningfully to the classification task. We also filtered out rows with missing or uninformative `authors` entries to ensure data quality. The final dataset consisted of the target variable `real_fake`, the `authors` column, and frequency counts for the top 30 most informative words used as predictive features.

	authors <chr>	real_fake <dbl>	word_hillary <int>	word_president <int>	word_debate <int>	word_obama <int>	word_police <int>	word_because <int>	word_american <int>	wo
1	View All Posts,Leonora Cravotta	0	1	1	0	0	2	0	0	
2	More Candace,Adam Kelsey,Abc News,More Adam	0	0	1	0	0	5	1	11	
3	Cassy Fiano	0	0	0	0	9	0	1	1	
4	Jack Shafer,Erick Trickey,Zachary Karabell	0	3	1	1	1	0	0	5	
5	John Parkinson,More John,Abc News,More Alexander	0	1	7	0	5	0	0	0	
6	Cassy Fiano	0	1	0	0	0	4	0	1	

Figure 1: Cleaned Dataframe

3.2 Data Modeling

We built a logistic regression model to classify news articles as real or fake based on the frequency of specific keywords, where each keyword's count was used as a feature.

$$\begin{aligned}\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 &+ \beta_1 \cdot \texttt{word\_hillary} + \beta_2 \cdot \texttt{word\_because} \\ &+ \beta_3 \cdot \texttt{word\_foundation} + \beta_4 \cdot \texttt{word\_september} \\ &+ \beta_5 \cdot \texttt{word\_candidate} + \beta_6 \cdot \texttt{word\_washington} \\ &+ \beta_7 \cdot \texttt{word\_nominee} + \cdots + \beta_{30} \cdot \texttt{word\_federal}\end{aligned}$$

Estimated coefficients determine which words were significant predictors. **Positive coefficients** indicated words associated with **real news**, while **negative coefficients** suggested a link to **fake news**. Statistical significance was assessed using p-values, allowing us to identify the most informative features for the classification task.

```
Call:
glm(formula = real_fake ~ ., family = binomial, data = df_drop %>%
    select(real_fake, starts_with("word_")))

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.440460   0.327169   1.346  0.17821
word_hillary      0.266105   0.089689   2.967  0.00301 **
word_president   -0.064753   0.157755  -0.410  0.68147
word_debate      -0.166217   0.121317  -1.370  0.17065
word_obama        0.017436   0.103830   0.168  0.86664
word_police      -0.038328   0.135362  -0.283  0.77706
word_because      0.489868   0.243744   2.010  0.04446 *
word_american    -0.116989   0.358935  -0.326  0.74447
word_presidential -0.248344   0.365377  -0.680  0.49670
word_before      -0.080731   0.243101  -0.332  0.73982
word_america      0.009173   0.188798   0.049  0.96125
word_election    -0.149069   0.195227  -0.764  0.44512
word_republican   0.222320   0.158252   1.405  0.16007
word_foundation   0.194467   0.124447   1.563  0.11813
word_according   -0.118416   0.254539  -0.465  0.64177
word_every       -0.159901   0.208705  -0.766  0.44358
word_september   -0.277283   0.226894  -1.222  0.22168
word_united      -0.147672   0.224656  -0.657  0.51097
word_another     -0.141246   0.316099  -0.447  0.65499
word_political   -0.006208   0.322731  -0.019  0.98465
word_candidate   -0.210932   0.285561  -0.739  0.46011
word_americans    0.206327   0.377521   0.547  0.58470
word_national    -0.148936   0.238394  -0.625  0.53214
word_charlotte    0.012825   0.149919   0.086  0.93183
word_democratic   0.384992   0.342935   1.123  0.26159
word_policy      -0.170747   0.424574  -0.402  0.68757
word_something    0.237028   0.393315   0.603  0.54675
word_washington  -0.554039   0.376453  -1.472  0.14109
word_nominee     -0.913919   0.450750  -2.028  0.04261 *
word_security    -0.094380   0.171247  -0.551  0.58154
word_federal     -0.095249   0.118152  -0.806  0.42015
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 252.31  on 181  degrees of freedom
Residual deviance: 195.77  on 151  degrees of freedom
AIC: 257.77

Number of Fisher Scoring iterations: 6
```

Figure 2: Full model summary result

The logistic regression results show that many predictors have large p-values, indicating that they may not contribute significantly to the classification of real versus fake news. For instance, variables such as `word_president`, `word_obama`, and `word_america` have p-values well above common significance thresholds, suggesting limited predictive value.

3.2.1 Model Selection

To address potential overfitting and improve model interpretation, we applied stepwise selection using the Akaike Information Criterion (AIC). This method helps identify a more parsimonious model by balancing goodness-of-fit with model complexity, retaining only variables that meaningfully contribute to the prediction.

```
Call:
glm(formula = real_fake ~ word_hillary + word_because + word_foundation +
    word_september + word_candidate + word_washington + word_nominee,
    family = binomial, data = df_drop %>% select(real_fake, starts_with("word_")))

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     0.002078   0.232369   0.009  0.99287
word_hillary    0.245275   0.079848   3.072  0.00213 **
word_because    0.419138   0.205564   2.039  0.04145 *
word_foundation 0.204499   0.122965   1.663  0.09630 .
word_september -0.378342   0.199557  -1.896  0.05797 .
word_candidate -0.491847   0.208352  -2.361  0.01824 *
word_washington -0.469565   0.312426  -1.503  0.13285
word_nominee   -0.774788   0.347108  -2.232  0.02561 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 252.31  on 181  degrees of freedom
Residual deviance: 207.09  on 174  degrees of freedom
AIC: 223.09

Number of Fisher Scoring iterations: 5
```

Figure 3: Best model selection using AIC

Important words with strong coefficients (positive or negative) included

- **Positive (real news indicators)**: `hillary`, `because`, `foundation`
- **Negative (fake news indicators)**: `candidate`, `nominee`, `september`, `washington`

The best logistic regression model includes seven predictors identified as informative for distinguishing real from fake news. Akaike Information Criterion (AIC) is 223.09. The lowest AIC value suggests a better trade-off between model fit and complexity. Compared to the null model (AIC = 252.31), the selected model shows a substantial improvement, indicating that the included predictors contribute meaningfully to explaining the variation in the outcome.

3.2.2 Goodness of fit tests

A anova: 8 × 5

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
	<int>	<dbl>	<int>	<dbl>	<dbl>
NULL	NA	NA	181	252.3056	NA
word_hillary	1	7.96336248	180	244.3422	4.773366e-03
word_because	1	0.01252443	179	244.3297	9.108927e-01
word_foundation	1	2.94322715	178	241.3865	8.623828e-02
word_september	1	5.84732907	177	235.5391	1.560071e-02
word_candidate	1	18.95119001	176	216.5879	1.341056e-05
word_washington	1	4.00889588	175	212.5790	4.526078e-02
word_nominee	1	5.49137346	174	207.0877	1.911052e-02

Figure 4: Likelihood ratio test

We performed a Likelihood Ratio Test (LRT) to assess each predictor's individual contribution in the logistic regression model. The results show that `word_hillary`, `word_september`,

`word_candidate`, `word_washington`, and `word_nominee` significantly reduce the deviance when added to the model, with p-values less than 0.05. In particular, `word_candidate` has the most substantial effect ($p < 0.001$), suggesting it is a strong indicator in predicting real vs. fake news. On the other hand, `word_because` and `word_foundation` do not contribute significantly on their own ($p > 0.05$), although they may still be useful in combination with other variables.

3.3 Model Testing

So far, we have tested which variables contribute meaningfully to the model. Now, evaluating the model's performance, which is essential to assess its accuracy and reliability.

3.3.1 Confusion Matrices

Following confusion matrix summarizes the performance of the logistic regression classifier at a cutoff of 0.5.

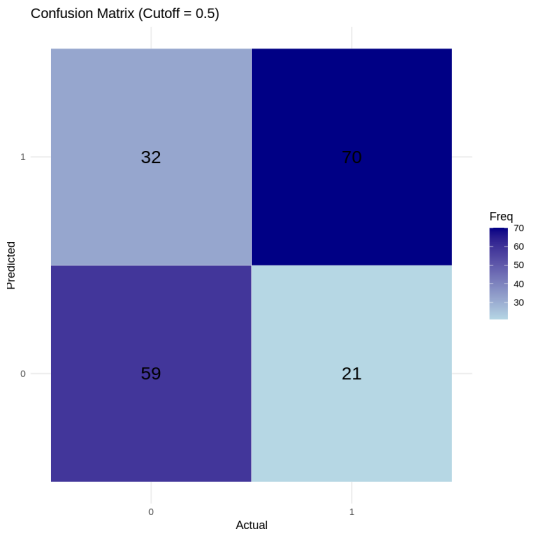

Figure 5: Confusion matrix at cutoff of 0.5

- **True Negatives (TN):** 59 — The model correctly predicted class 0.
- **False Negatives (FN):** 21 — The model predicted 0, but the actual was 1.
- **False Positives (FP):** 32 — The model predicted 1, but the actual was 0.
- **True Positives (TP):** 70 — The model correctly predicted class 1.

From this confusion matrix, we can compute some basic evaluation metrics

- **Accuracy:**
$$\frac{59+70}{59+21+32+70} = \frac{129}{182} \approx 0.7099$$
- **Sensitivity (Recall for class 1):**
$$\frac{70}{70+21} \approx 0.7692$$
- **Specificity (Recall for class 0):**
$$\frac{59}{59+32} \approx 0.6484$$

This suggests that the model performs **better at identifying class 1 (real)** than class 0 (fake), but it still misclassifies a notable number of both.

Table 1: Classification metrics across different cutoff thresholds

Cutoff	Specificity	Sensitivity	Accuracy
0.25	0.2527	0.9560	0.6044
0.30	0.3187	0.9560	0.6374
0.35	0.3736	0.9231	0.6484
0.40	0.4505	0.8352	0.6429
0.45	0.5824	0.8242	0.7033
0.50	0.6484	0.7692	0.7088
0.55	0.7802	0.5604	0.6703
0.60	0.8022	0.5055	0.6538
0.65	0.8901	0.3846	0.6374
0.70	0.9231	0.3187	0.6209
0.75	0.9560	0.2637	0.6099
0.80	0.9780	0.2088	0.5934
0.85	1.0000	0.1429	0.5714
0.90	1.0000	0.1209	0.5604

To improve the trade-off between false positives and false negatives, one potential strategy is to **adjust the decision cutoff threshold**.

This table is the result of how model performance changes with different classification cutoff thresholds. As the cutoff increases from 0.25 to 0.90

- **Specificity** steadily increases meaning the model becomes better at correctly identifying class 0 (fake).
- **Sensitivity** decreases indicating the model misses more class 1 (real) cases as the cutoff increases.
- **Accuracy** peaks around cutoff 0.50 (at 70.88%) and declines afterwards.

This trade-off illustrates a common phenomenon in binary classification: **increasing specificity often comes at the cost of reduced sensitivity**, and vice versa.

Choosing an appropriate cutoff depends on the context. For example:

- If **false negatives are costly** (e.g., missing real news), a lower cutoff (e.g., 0.3 or 0.4) may be preferred to maintain high sensitivity.
- If **false positives are more harmful** (e.g., falsely labeling fake news as real), a higher cutoff (e.g., 0.7 or 0.8) may be more appropriate to increase specificity.

In this case, a balanced cutoff around **0.48** seems optimal, offering a good trade-off between sensitivity, specificity, and overall accuracy.

3.3.2 ROC curve

Another way to evaluate classification models is to see the Receiver Operating Characteristic (ROC) curve and check Area Under the Curve (AUC) which is the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative one.

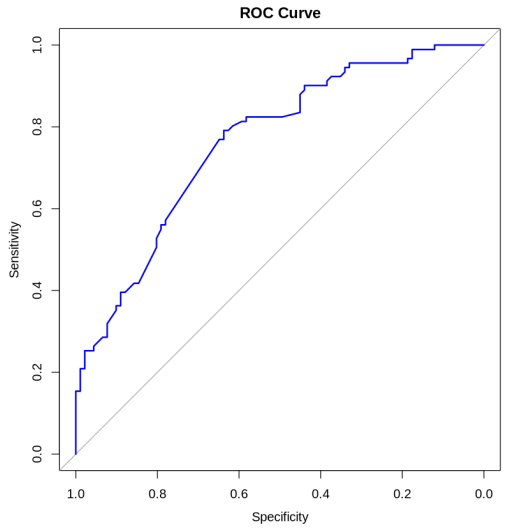

Figure 6: Receiver Operating Characteristic (ROC) curve

The AUC of 0.7628 indicates that the model has a good ability to distinguish between real and fake instances. This means there's a 76.28% chance that the model will rank a randomly chosen real case higher than a fake one.

While not perfect, an AUC between 0.7 and 0.8 is considered decent, especially when dealing with noisy or imbalanced data. This result suggests the model performs well, but there's still room to improve, possibly by adjusting the cutoff or refining the features.

3.3.3 K-fold Cross Validation

K-fold cross-validation is also another technique used to see the performance of a model by dividing the data into k equal-sized folds.

Table 2: Cross-Validation Error Estimates for Logistic Regression Model

Model	**Raw Error (CV)**	**Bias-Corrected Error (CV)**
Model 1	0.450549	0.445054
Model 2	0.439560	0.435530
Model 3	0.351648	0.353309

Overall, the difference between the raw and bias-corrected errors is small, suggesting that the models are relatively stable and not heavily overfitting.

3.4 Modeling with Probit and Identity Links

In generalized linear models (GLMs), the link function is used to transform the predicted values so that they fall within a specified range. The two link functions, probit and identity, are commonly used for different types of modeling.

First, Probit Link function is used to model binary outcomes (like yes/no or success/failure) and is often applied in probit regression. It is the inverse of the cumulative distribution function (CDF) of the standard normal distribution. Essentially, it transforms probabilities (which range between 0 and 1) into a continuous scale using the normal distribution.

It is commonly used when the data is assumed to follow a normal distribution with a binary outcome.

```
Call:
glm(formula = real_fake ~ word_hillary + word_because + word_foundation +
    word_september + word_candidate + word_washington + word_nominee,
    family = binomial(link = "probit"), data = df_drop)

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.00340    0.14189  -0.024  0.98088
word_hillary    0.14816    0.04570   3.242  0.00119 **
word_because    0.25311    0.11957   2.117  0.03427 *
word_foundation 0.12471    0.06883   1.812  0.06999 .
word_september -0.22512    0.11592  -1.942  0.05212 .
word_candidate -0.30320    0.11782  -2.573  0.01007 *
word_washington -0.27201    0.17433  -1.560  0.11870
word_nominee   -0.45015    0.20456  -2.201  0.02777 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 252.31  on 181  degrees of freedom
Residual deviance: 206.72  on 174  degrees of freedom
AIC: 222.72

Number of Fisher Scoring iterations: 6
```

Figure 7: Probit Link model summary result

Next, is Identity Link which is simply a linear relationship between the predictor and the response variable. It's used when the model assumes that the expected value of the outcome is directly related to the linear predictors.

The identity link is often used for continuous outcomes and is the default for linear regression models.

```
Call:
glm(formula = real_fake ~ word_hillary + word_because + word_foundation +
    word_september + word_candidate + word_washington + word_nominee,
    family = gaussian(link = "identity"), data = df_drop)

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.49937    0.04655  10.729  < 2e-16 ***
word_hillary    0.03529    0.01008   3.501 0.000588 ***
word_because    0.07598    0.03481   2.183 0.030399 *
word_foundation 0.03343    0.01685   1.983 0.048905 *
word_september -0.06526    0.03303  -1.976 0.049739 *
word_candidate -0.06768    0.02741  -2.469 0.014510 *
word_washington -0.07612    0.03786  -2.011 0.045908 *
word_nominee   -0.11600    0.05499  -2.109 0.036349 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.2106017)

    Null deviance: 45.500  on 181  degrees of freedom
Residual deviance: 36.645  on 174  degrees of freedom
AIC: 242.8

Number of Fisher Scoring iterations: 2
```

Figure 8: Identity Link model summary result

Based on both summary results, the probit model has the lowest AIC, even compared to the logit model. A lower AIC suggests that the probit model strikes a better balance between model complexity and goodness-of-fit.

In conclusion, when dealing with binary outcomes, the probit link provides a more fitting approach, especially when the goal is to estimate probabilities within the (0, 1) range, and its lower AIC value suggests that it may offer better predictive accuracy.

4 Conclusion

In this study, we have built a logistic regression model to classify news articles as real or fake based on the frequency of certain keywords. Through preprocessing, feature engineering, and model selection, we identified significant words associated with real and fake news. The final model achieved a reasonable accuracy of 70.9% and provided insights into which words are predictive of real versus fake news.

Further improvements could include

- Using more features like author's name or categorized topic.
- Experimenting with other classifiers such as Random Forests or Support Vector Machines to compare performance.
- Expanding the dataset to improve generalization and performance.

Future work should expend the interpretation of machine learning models in the context of fake news detection and further refine classification for practical use in real-world scenarios.

5 Reference

- Opinion — Remember Nayirah, Witness for Kuwait? (Published 1992), www.nytimes.com/1992/01/06/opinion/remember-nayirah-witness-for-kuwait.html. Accessed 6 May 2025.
- Shu, Kai, et al. "FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media." arXiv.Org, 27 Mar. 2019, arxiv.org/abs/1809.01286.
- Mahudeswaran, Deepak. "FakeNewsNet." Kaggle, 2 Nov. 2018, www.kaggle.com/datasets/mdepak/fakenewsnet/data.

6 Appendix

Listing 1: Notebook Code Excerpt

```
# -*- coding: utf-8 -*-

df_real = read.csv("BuzzFeed_real_news_content.csv")
df_fake = read.csv("BuzzFeed_fake_news_content.csv")

colnames(df_real)

colnames(df_fake)

df_real["real_fake"] = 0
df_fake["real_fake"] = 1

buzzfeed = rbind(df_real, df_fake)

install.packages(c("tidytext", "dplyr", "stringr", "tidyr"))

# Load necessary libraries
library(dplyr)
library(stringr)
library(tidytext)
library(tidyr)

# Create temporary dataset
df <- buzzfeed
```

```r
# Tokenize text and count word frequencies
all_words <- df %>%
  select(text) %>%
  unnest_tokens(word, text) %>%
  count(word, sort = TRUE)

# Function to estimate syllables by counting vowel groups
estimate_syllables <- function(word) {
  str_count(tolower(word), "[aeiouy]+")
}

# Add syllable counts
all_words <- all_words %>%
  mutate(syllables = estimate_syllables(word)) %>%
  filter(syllables > 2)

# Get top 30 words with more than 3 syllables
top50_words <- head(all_words$word, 30)

# Function to count word occurrences in text
count_word <- function(text, word) {
  str_count(tolower(text), fixed(tolower(word)))
}

# Create new columns for each top word
for (w in top50_words) {
  df[[paste0("word_", w)]] <- sapply(df$text, count_word, word = w)
}

# Remove columns by name
df_drop <- df %>%
  select(-title, -source, -publish_date, -id, -text, -url, -top_img, -movies, -images

colnames(df_drop)

print(colSums(is.na(df_drop)))

df_dropna <- df_drop %>%
  filter(
    !is.na(authors),
    trimws(authors) != "",
    trimws(authors) != "View■All■Posts"
  )

buzzfeed_data <- df_dropna
head(buzzfeed_data)


buzzfeed_data$authors

# Extract the first author from the authors column
buzzfeed_data$first_author <- sapply(strsplit(as.character(buzzfeed_data$authors), "
  x <- trimws(x)                    # remove extra spaces
  x <- x[x != "View■All■Posts"]     # remove "View All Posts"
  if (length(x) > 0) x[1] else NA   # return first valid name or NA
})

buzzfeed_data$first_author
```

```r
# Convert categorical variables to factors
buzzfeed_data$first_author <- as.factor(buzzfeed_data$first_author)

nrow(df)
nrow(df_drop)
nrow(df_dropna)
nrow(buzzfeed_data)

# Fit full model
model_full <- glm(real_fake ~ ., data = df_drop %>% select(real_fake, starts_with("w
summary(model_full)


# Variable Selection (Best Subset)
# Perform stepwise selection silently
suppressMessages({
  model_best <- step(model_full, direction = "both", trace = 0)
})

# Show summary of the best model
summary(model_best)

# Likelihood ratio test
anova(model_best, test = "Chisq")

confint(model_best)

# Predict probabilities for the 'real_fake' outcome
pred_probs <- predict(model_best, newdata = df_drop, type = "response")

# Predicted class using cutoff = 0.5
predicted_class <- ifelse(pred_probs > 0.5, 1, 0)

# Confusion matrix
confusion_matrix <- table(Predicted = predicted_class, Actual = df_drop$real_fake)
print(confusion_matrix)

library(ggplot2)

# Create confusion matrix
cm <- table(Predicted = predicted_class, Actual = actual_class)

# Convert to dataframe for ggplot
cm_df <- as.data.frame(cm)
colnames(cm_df) <- c("Predicted", "Actual", "Freq")

# Plot heatmap
ggplot(data = cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 6) +
  scale_fill_gradient(low = "lightblue", high = "darkblue") +
  labs(title = "Confusion Matrix (Cutoff = 0.5)", x = "Actual", y = "Predicted") +
  theme_minimal()

# Function to compute metrics
get_metrics <- function(predictions, actual) {
  cm <- table(Predicted = predictions, Actual = actual)

  # Ensure 2x2 confusion matrix
```

```
  tn <- ifelse("0" %in% rownames(cm) && "0" %in% colnames(cm), cm["0", "0"], 0)
  tp <- ifelse("1" %in% rownames(cm) && "1" %in% colnames(cm), cm["1", "1"], 0)
  fn <- ifelse("0" %in% rownames(cm) && "1" %in% colnames(cm), cm["0", "1"], 0)
  fp <- ifelse("1" %in% rownames(cm) && "0" %in% colnames(cm), cm["1", "0"], 0)

  sensitivity <- tp / (tp + fn)  # True Positive Rate
  specificity <- tn / (tn + fp)  # True Negative Rate
  accuracy <- (tp + tn) / sum(cm)

  return(c(specificity = specificity, sensitivity = sensitivity, accuracy = accuracy
}

# Sequence of cutoff values
cutoffs <- seq(0.25, 0.9, by = 0.05)

# Store results
results <- data.frame(Cutoff = numeric(), Specificity = numeric(), Sensitivity = num

for (cutoff in cutoffs) {
  predictions <- ifelse(pred_probs > cutoff, 1, 0)
  metrics <- get_metrics(predictions, df_drop$real_fake)
  results <- rbind(results, data.frame(Cutoff = cutoff,
                                       Specificity = round(metrics["specificity"], 6
                                       Sensitivity = round(metrics["sensitivity"], 6
                                       Accuracy = round(metrics["accuracy"], 6)))
}

print(results)

# Find best cut-off point (Youden's Index maximization)
best_cutoff <- coords(roc_curve, "best", ret = "threshold")
best_cutoff

install.packages("pROC")
library(pROC)

# Generate ROC curve
roc_curve <- roc(df_drop$real_fake, pred_probs)
plot(roc_curve, main = "ROC Curve", col = "blue")

# Compute AUC
auc(roc_curve)

# Load necessary packages
install.packages("boot")
library(boot)

# Example logistic regression models on df_drop
fit0 <- glm(real_fake ~ word_hillary, family = binomial, data = df_drop)
fit1 <- glm(real_fake ~ word_hillary + word_candidate, family = binomial, data = df_
fit2 <- glm(real_fake ~ word_hillary + word_candidate + word_nominee, family = binom

# Define custom cost function: classify correctly if |prediction - actual| <= 0.5
cost <- function(r, pi = 0) mean(abs(r - pi) > 0.5)

# 10-fold cross-validation
out0 <- cv.glm(df_drop, fit0, cost, K = 10)
out1 <- cv.glm(df_drop, fit1, cost, K = 10)
out2 <- cv.glm(df_drop, fit2, cost, K = 10)
```

```
# Print cross-validation error estimates
out0$delta
out1$delta
out2$delta

# Fit Probit regression model using the probit link
fit_probit <- glm(real_fake ~ word_hillary + word_because + word_foundation +
                    word_september + word_candidate + word_washington + word_nominee

# View summary of the model
summary(fit_probit)

# Fit model with Identity link
fit_identity <- glm(real_fake ~ word_hillary + word_because + word_foundation +
                    word_september + word_candidate + word_washington + word_nominee
                  , family = gaussian(link = "identity"))

# View summary of the model
summary(fit_identity)
```