

Yelp Reviews: Sentiment Analysis

Kaitlyn Chou (group leader), Jensen Harvey, and Emily Friedman

2026-01-30

1. Load and Prepare Data

```
yelp <- read_csv("Yelp Restaurant Reviews.csv")

yelp <- yelp %>%
  clean_names() %>%
  rename(
    url = yelp_url,
    text = review_text
  )

yelp <- yelp %>%
  mutate(
    sentiment = ifelse(rating >= 4, "Positive", "Negative"),
    sentiment = factor(sentiment, levels = c("Negative", "Positive"))
  )

cat("Dataset dimensions:", dim(yelp), "\n")

## Dataset dimensions: 19896 5

cat("\nSentiment distribution:\n")

##
## Sentiment distribution:

print(table(yelp$sentiment))

##
## Negative Positive
##      4566      15330

cat("\nProportions:\n")

##
## Proportions:
```

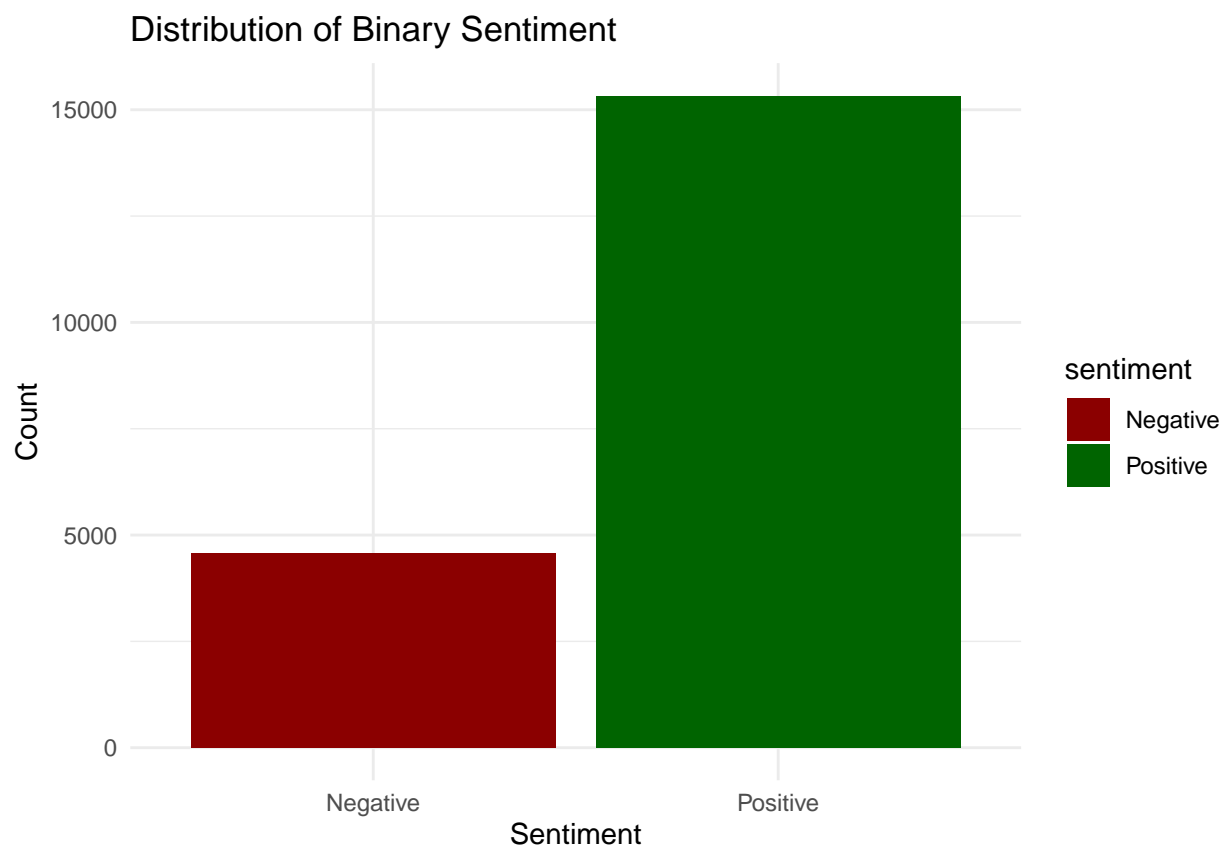
```
print(prop.table(table(yelp$sentiment)))
```

```
##  
## Negative Positive  
## 0.2294934 0.7705066
```

2. Exploratory Data Analysis

Distribution of Binary Sentiment

```
ggplot(yelp, aes(x = sentiment, fill = sentiment)) +  
  geom_bar() +  
  scale_fill_manual(values = c("Negative" = "darkred", "Positive" = "darkgreen")) +  
  labs(title = "Distribution of Binary Sentiment",  
       x = "Sentiment",  
       y = "Count") +  
  theme_minimal()
```



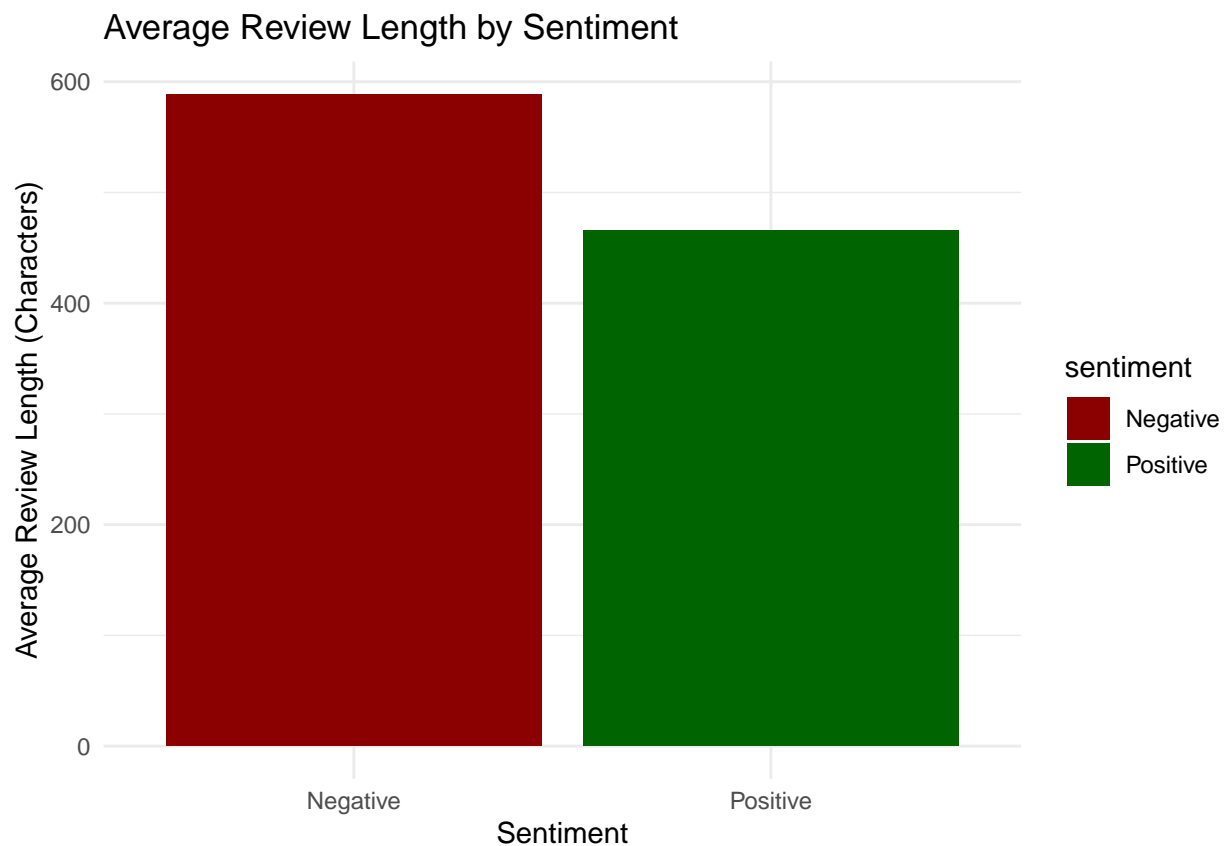
Review Length by Sentiment

```

yelp <- yelp %>%
  mutate(review_length = nchar(text))

yelp %>%
  group_by(sentiment) %>%
  summarise(
    avg_length = mean(review_length),
    median_length = median(review_length)
  ) %>%
  ggplot(aes(x = sentiment, y = avg_length, fill = sentiment)) +
  geom_col() +
  scale_fill_manual(values = c("Negative" = "darkred", "Positive" = "darkgreen")) +
  labs(title = "Average Review Length by Sentiment",
       x = "Sentiment",
       y = "Average Review Length (Characters)") +
  theme_minimal()

```



3. Text Preprocessing and Feature Engineering

```

preprocess_text <- function(text_vector) {
  text_clean <- text_vector %>%
    tolower() %>%
    str_replace_all("http\\S+|www\\S+", "") %>%

```

```

    str_replace_all("\\S+@\\S+", "") %>%
    str_replace_all("[[:punct:]]", " ") %>%
    str_replace_all("[[:digit:]]+", "") %>%
    str_replace_all("\\s+", " ") %>%
    str_trim()

    return(text_clean)
}

yelp <- yelp %>%
  mutate(
    text_clean = preprocess_text(text),
    review_id = row_number()
  )

```

Create TF-IDF Features

```

food_stopwords <- tibble(word = c(
  "ice", "cream", "chocolate", "vanilla", "cookie", "cookies",
  "cake", "cakes", "flavor", "flavors", "bakery", "donut", "donuts"
))

tfidf_data <- yelp %>%
  select(review_id, sentiment, text_clean) %>%
  unnest_tokens(word, text_clean) %>%
  anti_join(stop_words, by = "word") %>%
  anti_join(food_stopwords, by = "word") %>%
  count(review_id, word) %>%
  bind_tf_idf(word, review_id, n)

discriminative_words <- tfidf_data %>%
  left_join(yelp %>% select(review_id, sentiment), by = "review_id") %>%
  group_by(word, sentiment) %>%
  summarise(
    total_tfidf = sum(tf_idf),
    count = n(),
    .groups = "drop"
  ) %>%
  pivot_wider(names_from = sentiment, values_from = c(total_tfidf, count), values_fill = 0) %>%
  mutate(
    total_count = count_Negative + count_Positive,

    discriminative_score = abs(total_tfidf_Positive - total_tfidf_Negative)
  ) %>%
  filter(total_count >= 10) %>%
  arrange(desc(discriminative_score)) %>%
  slice_head(n = 150) %>%
  select(word)

```

```
dtm_wide <- tfidf_data %>%
  semi_join(discriminative_words, by = "word") %>%
  select(review_id, word, tf_idf) %>%
  pivot_wider(names_from = word, values_from = tf_idf, values_fill = 0)

features_final <- yelp %>%
  select(review_id, sentiment, review_length) %>%
  left_join(dtm_wide, by = "review_id")

features_final[is.na(features_final)] <- 0

cat("Features matrix dimensions:", dim(features_final), "\n")
```

```
## Features matrix dimensions: 19896 153
```

```
cat("Number of discriminative features:", ncol(dtm_wide) - 1, "\n")
```

```
## Number of discriminative features: 150
```

4. Train-Test Split

```
set.seed(42)

train_index <- createDataPartition(features_final$sentiment, p = 0.8, list = FALSE)

train_data <- features_final[train_index, ]
test_data <- features_final[-train_index, ]

X_train <- train_data %>% select(-review_id, -sentiment)
y_train <- train_data$sentiment

X_test <- test_data %>% select(-review_id, -sentiment)
y_test <- test_data$sentiment

cat("Training set distribution:\n")
```

```
## Training set distribution:
```

```
print(table(y_train))
```

```
## y_train
## Negative Positive
##      3653      12264
```

```

cat("\nTest set distribution:\n")

##
## Test set distribution:

print(table(y_test))

## y_test
## Negative Positive
##      913      3066

cat("\nNumber of features:", ncol(X_train), "\n")

##
## Number of features: 151

```

5. Model Training

Model 1: Logistic Regression

```

class_weights <- 1 / table(y_train)
weights <- ifelse(y_train == "Negative",
                  class_weights["Negative"],
                  class_weights["Positive"])

weights <- weights / sum(weights) * length(weights)

logistic_model <- glm(
  sentiment ~ .,
  data = train_data %>% select(-review_id),
  family = binomial(link = "logit"),
  weights = weights
)

logistic_prob <- predict(logistic_model, newdata = X_test, type = "response")
logistic_pred <- factor(
  ifelse(logistic_prob > 0.5, "Positive", "Negative"),
  levels = c("Negative", "Positive")
)

logistic_accuracy <- mean(logistic_pred == y_test)
cat("Logistic Regression Test Accuracy:", round(logistic_accuracy * 100, 2), "%\n")

## Logistic Regression Test Accuracy: 74.49 %

```

Model 2: Random Forest

```
y_train <- factor(y_train, levels = c("Negative", "Positive"))
y_test  <- factor(y_test, levels = c("Negative", "Positive"))

neg_count <- sum(y_train == "Negative")
pos_count <- sum(y_train == "Positive")

rf_model <- randomForest(
  x = X_train,
  y = y_train,
  ntree = 200,
  mtry = floor(sqrt(ncol(X_train))),
  sampsize = c("Negative" = neg_count, "Positive" = neg_count), # Balance by sampling
  importance = TRUE
)

rf_pred <- predict(rf_model, newdata = X_test)
rf_pred <- factor(rf_pred, levels = c("Negative", "Positive"))

rf_accuracy <- mean(rf_pred == y_test)
cat("Random Forest Test Accuracy:", round(rf_accuracy * 100, 2), "%\n")
```

```
## Random Forest Test Accuracy: 78.56 %
```

6. Model Evaluation

Confusion Matrices

```
cat("=== Logistic Regression Confusion Matrix ===\n")
```

```
## === Logistic Regression Confusion Matrix ===
```

```
cm_logistic <- confusionMatrix(logistic_pred, y_test, positive = "Positive")
print(cm_logistic$table)
```

```
##           Reference
## Prediction Negative Positive
##   Negative      737      839
##   Positive      176     2227
```

```
cat("\n")
```

```
print(cm_logistic$overall)
```

```
##           Accuracy           Kappa AccuracyLower AccuracyUpper AccuracyNull
## 7.449108e-01 4.251764e-01 7.310640e-01 7.583970e-01 7.705454e-01
## AccuracyPValue McNemarPValue
## 9.999332e-01 6.701875e-96
```

```
cat("\n=== Random Forest Confusion Matrix ===\n")
```

```
##
## === Random Forest Confusion Matrix ===
```

```
cm_rf <- confusionMatrix(rf_pred, y_test, positive = "Positive")
print(cm_rf$table)
```

```
##           Reference
## Prediction Negative Positive
## Negative         634       574
## Positive         279      2492
```

```
cat("\n")
```

```
print(cm_rf$overall)
```

```
##           Accuracy           Kappa AccuracyLower AccuracyUpper AccuracyNull
## 7.856245e-01 4.555218e-01 7.725401e-01 7.982882e-01 7.705454e-01
## AccuracyPValue McNemarPValue
## 1.198866e-02 7.779792e-24
```

Performance Metrics Comparison

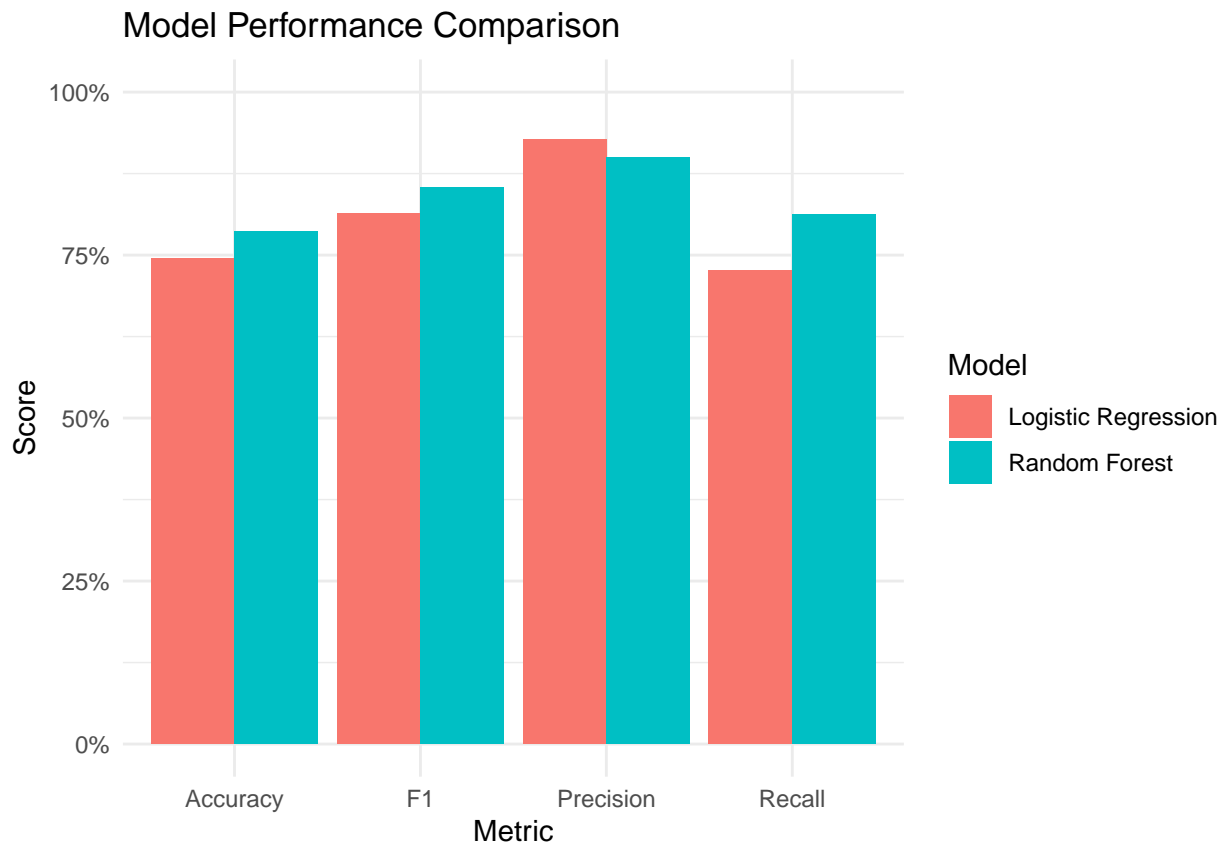
```
metrics_comparison <- data.frame(
  Model = c("Logistic Regression", "Random Forest"),
  Accuracy = c(cm_logistic$overall['Accuracy'], cm_rf$overall['Accuracy']),
  Precision = c(cm_logistic$byClass['Precision'], cm_rf$byClass['Precision']),
  Recall = c(cm_logistic$byClass['Recall'], cm_rf$byClass['Recall']),
  F1 = c(cm_logistic$byClass['F1'], cm_rf$byClass['F1'])
)

print(metrics_comparison)
```

```
##           Model Accuracy Precision Recall      F1
## 1 Logistic Regression 0.7449108 0.9267582 0.7263536 0.8144085
## 2 Random Forest 0.7856245 0.8993143 0.8127854 0.8538633
```



```
metrics_comparison %>%
  pivot_longer(cols = c(Accuracy, Precision, Recall, F1),
               names_to = "Metric",
               values_to = "Score") %>%
  ggplot(aes(x = Metric, y = Score, fill = Model)) +
  geom_col(position = "dodge") +
  scale_y_continuous(limits = c(0, 1), labels = scales::percent) +
  labs(title = "Model Performance Comparison",
       y = "Score") +
  theme_minimal()
```



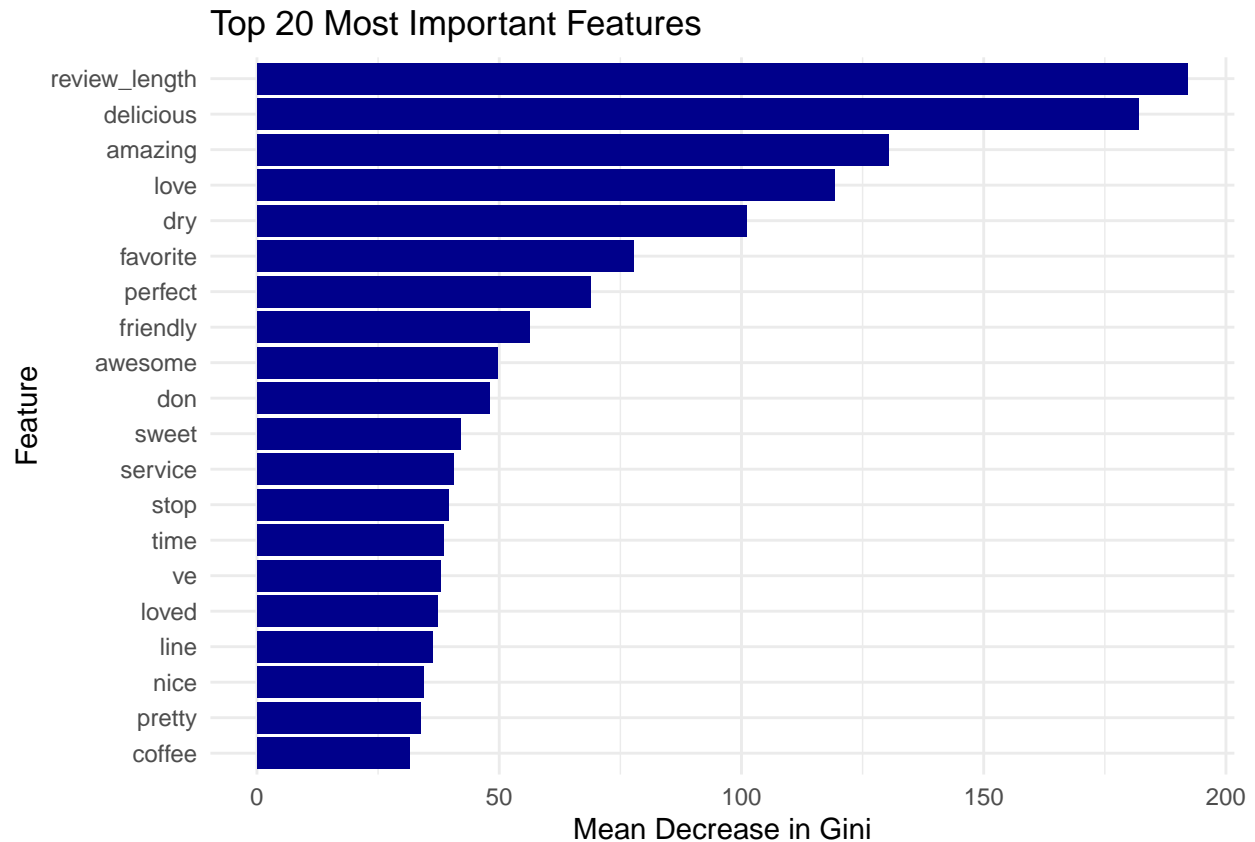
Feature Importance

```
importance_df <- as.data.frame(importance(rf_model))
importance_df$feature <- rownames(importance_df)

top_20_features <- importance_df %>%
  arrange(desc(MeanDecreaseGini)) %>%
  head(20)

ggplot(top_20_features, aes(x = reorder(feature, MeanDecreaseGini), y = MeanDecreaseGini)) +
  geom_col(fill = "darkblue") +
```

```
coord_flip() +
labs(title = "Top 20 Most Important Features",
     x = "Feature",
     y = "Mean Decrease in Gini") +
theme_minimal()
```



7. Cross-Validation

```
train_control <- trainControl(
  method = "cv",
  number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
)

cv_rf <- train(
  x = X_train,
  y = y_train,
  method = "rf",
  trControl = train_control,
  tuneGrid = data.frame(mtry = c(5, 10, 15)),
  ntree = 200,
```

```

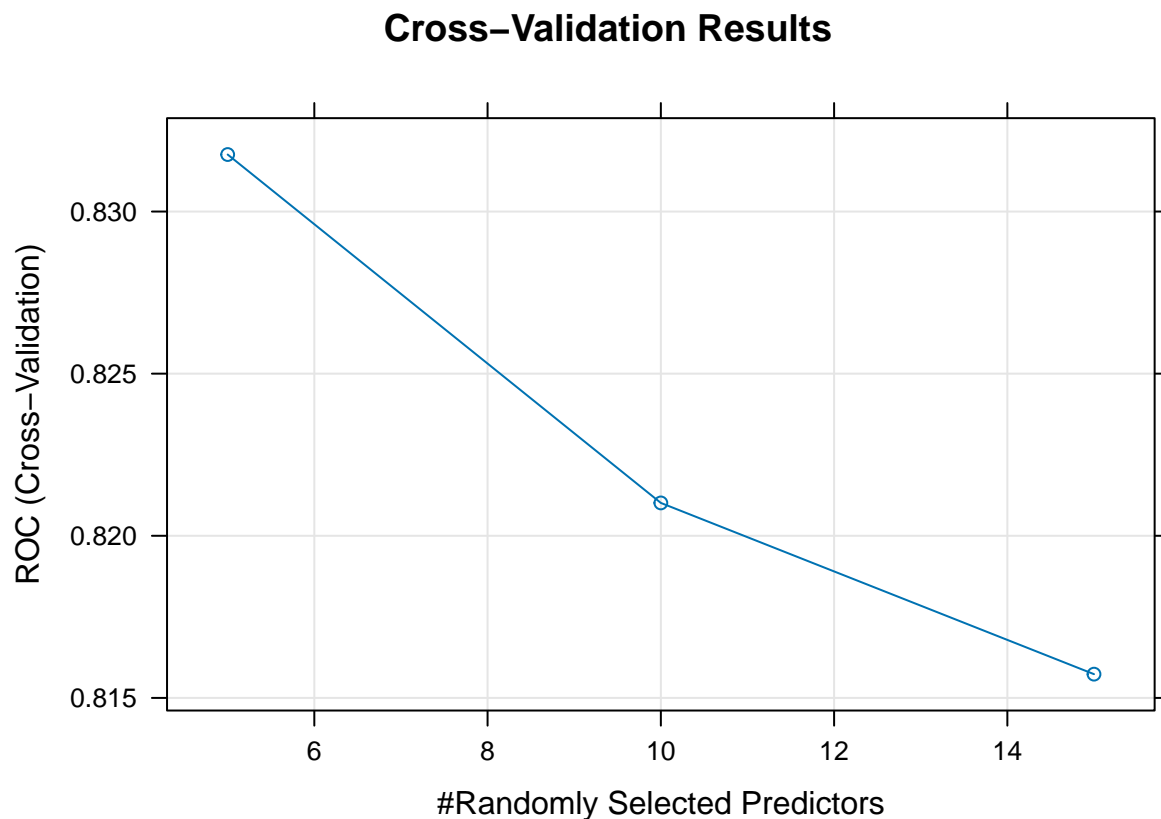
metric = "ROC"
)

print(cv_rf)

## Random Forest
##
## 15917 samples
## 151 predictor
## 2 classes: 'Negative', 'Positive'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 12734, 12733, 12735, 12733, 12733
## Resampling results across tuning parameters:
##
##  mtry  ROC      Sens      Spec
##    5   0.8317581 0.3375249 0.9664872
##   10   0.8210130 0.3739325 0.9526254
##   15   0.8157327 0.3909079 0.9467548
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 5.

plot(cv_rf, main = "Cross-Validation Results")

```



8. Results Summary

Classification Task:

- Binary (Negative vs. Positive)
 - Negative: 1-3 star ratings
 - Positive: 4-5 star ratings

Model Performance:

- Logistic Regression:
 - - Accuracy: 74.49 %
 - - Precision: 92.68 %
 - - Recall: 72.64 %
 - - F1 Score: 81.44 %
- Random Forest:
 - - Accuracy: 78.56 %
 - - Precision: 89.93 %
 - - Recall: 81.28 %
 - - F1 Score: 85.39 %
- Cross-Validated Random Forest:
 - * ROC AUC: 0.832

Top 5 Most Important Features:

1. review_length
2. delicious
3. amazing
4. love
5. dry

```
sessionInfo()
```

```
## R version 4.4.3 (2025-02-28)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib; LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```

##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] caret_7.0-1      lattice_0.22-6    randomForest_4.7-1.2
## [4] janitor_2.2.1    tidytext_0.4.3    lubridate_1.9.4
## [7] forcats_1.0.0    stringr_1.5.1     dplyr_1.1.4
## [10] purrr_1.0.4      readr_2.1.5       tidyr_1.3.1
## [13] tibble_3.2.1     ggplot2_3.5.2     tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
## [1] tidyselect_1.2.1   timeDate_4041.110  farver_2.1.2
## [4] fastmap_1.2.0      janeaustenr_1.0.0  pROC_1.19.0.1
## [7] digest_0.6.37      rpart_4.1.24       timechange_0.3.0
## [10] lifecycle_1.0.4    tokenizers_0.3.0   survival_3.8-3
## [13] magrittr_2.0.3     compiler_4.4.3     rlang_1.1.5
## [16] tools_4.4.3        yaml_2.3.10        data.table_1.17.0
## [19] knitr_1.49         labeling_0.4.3     bit_4.6.0
## [22] plyr_1.8.9         withr_3.0.2        nnet_7.3-20
## [25] grid_4.4.3         stats4_4.4.3       e1071_1.7-16
## [28] colorspace_2.1-1   future_1.67.0      globals_0.18.0
## [31] scales_1.3.0       iterators_1.0.14    MASS_7.3-65
## [34] cli_3.6.4          crayon_1.5.3       rmarkdown_2.29
## [37] generics_0.1.3     rstudioapi_0.17.1  future.apply_1.20.0
## [40] reshape2_1.4.4     tzdb_0.5.0         proxy_0.4-27
## [43] splines_4.4.3      parallel_4.4.3     vctrs_0.6.5
## [46] hardhat_1.4.2      Matrix_1.7-2       hms_1.1.3
## [49] bit64_4.6.0-1      listenv_0.9.1      foreach_1.5.2
## [52] gower_1.0.2        recipes_1.3.1      glue_1.8.0
## [55] parallelly_1.45.1  codetools_0.2-20   stringi_1.8.4
## [58] gtable_0.3.6       munsell_0.5.1      pillar_1.10.1
## [61] htmltools_0.5.8.1  ipred_0.9-15       lava_1.8.1
## [64] R6_2.6.1           vroom_1.6.5        evaluate_1.0.3
## [67] SnowballC_0.7.1    snakecase_0.11.1   class_7.3-23
## [70] Rcpp_1.0.14        nlme_3.1-167       prodlim_2025.04.28
## [73] xfun_0.51          pkgconfig_2.0.3    ModelMetrics_1.2.2.2

```