Fake vs Real News: ML Classification - TBI Technical Test

Khyati Narang

1 Introduction and Approach

The goal of this project is to develop a classification model that distinguishes between fake and authentic news articles using the Fake and Real News Kaggle dataset. This task will develop a baseline binary classification model using text vectorization and XGBoost classifer. Based on the evaluation of the baseline model,advanced feature engineering will be used to improve its performance.

2 Data Processing

The data-sets consists of two CSV files namely fake.csv, and true.csv. Each file contains the headline of the srticles (title), the main article content (text), the category of the article (subject) and date. The code chunk below, combines the two csv files and adds binary labels for articles labeled as fake (0) and articles labeled as real (1). The data is then cleaned and formatted. The missing values are dealt with and the data is shuffled to prepare for machine learning cross validation.

To simplify the evaluation process, as seen in the code chunk below, the subject field was re-coded to the three domains of politics, news and other.

The following show the descriptives of the data set to get an idea. The subject terms does not have any that don't fit either politics or news so the "other" category remains unused.

```
## # A tibble: 2 x 2
##   label     n
##   <chr> <int>
## 1 Fake  22841
## 2 Real  21416
```

```
## # A tibble: 2 x 2
##   domain       n
##   <chr>    <int>
## 1 News     19978
## 2 Politics 24279
```

Since the dataset consists of text data, it is essential to preprocess it for modeling. I did this by removing punctuations, implementing whitespace nromalization, contraction expansion and making sure all text is lowercase.

To ensure unbiased model performance and evaluation, the dataset was randomly split into a 20% test set and 80% training set. 5897 duplicates across the data-set were identified and dealt with to ensure fair evaluation.

3 Feature Engineering

In order to convert words into a matrix of numbers and encode importance the TF-IDF was configured to capture useful patterns. Before implementing this, a word tokenizer was used to remove uninformative words and single-letter tokens.

The cleaned text was transformed into numerical values using the unigram-based TF-IDF vectorization. This captures the importance of the words as seen in the table below.

```
##    clinton      house     people      state      obama     donald republican
##   164.2472   159.5558   157.0732   156.1547   150.7272   146.5808   138.9777
##      white government washington        one        new     united     states
##   133.0706   132.7239   131.7136   127.1823   126.0550   122.1257   122.1245
##     russia      party       told    hillary   election   campaign
##   118.3826   117.2578   117.0941   116.9135   114.2523   113.5712
```

4 Baseline Model

4.1 XGBoost Classifier

In order to create an efficient baseline model, the XGBoost or extreme gradient boosting model was used. Since it is a tree-based algorithm, it is known for its high-performance when it comes to robustness, speed, accuracy on high-dimensional data. This model represents 100 boosted decision trees and was trained on the TF-IDF matrix that consists of more than 105,000 features - corresponding to the unique words that were processed earlier. XGBoost is well-suited for this task of fake news detection because of the following: 1) It processes sparse input without needing dimensionality reduction. 2) It efficiently identifies non-linear interactions that occur between words which helps capture patterns to differentiate the fake news. 3) It also has regularization built in the model which prevents over-fitting and minimizes noise when it comes to the different terms and vocabulary. 4) It is efficient with complex and large datasets - being advantageous for modeling many news articles.

Since it is a binary classification model, it was trained using the logistic loss function with a tree depth of 6, and a 0.3 learning rate in order to balance generalization of the model and its learning capacity.

4.2 Baseline Model Evaluation

The baseline model is evaluated using classification metrices of the confusion matrix, accuracy, precision, recall and F1 score.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3409   35
##          1   65 4163
##
##                Accuracy : 0.987
##                  95% CI : (0.9842, 0.9894)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9737
```

```
##
##  Mcnemar's Test P-Value : 0.003732
##
##             Sensitivity : 0.9917
##             Specificity : 0.9813
##          Pos Pred Value : 0.9846
##          Neg Pred Value : 0.9898
##              Prevalence : 0.5472
##          Detection Rate : 0.5426
##    Detection Prevalence : 0.5511
##       Balanced Accuracy : 0.9865
##
##        'Positive' Class : 1
##
```

The output above shows that the model's accuracy indicates that it performs highly on unseen texts and articles. The precision and recall reflects strong performance when distinguishing real and fake news articles with very few false positives and false negatives. A balance between precision and recall is noticed with the F1 score, suggesting a good model performance that does not disproportionately favor the classes. The results reflect an effective baseline model.

4.3 Baseline Model Evaluation Based on Subject Domains

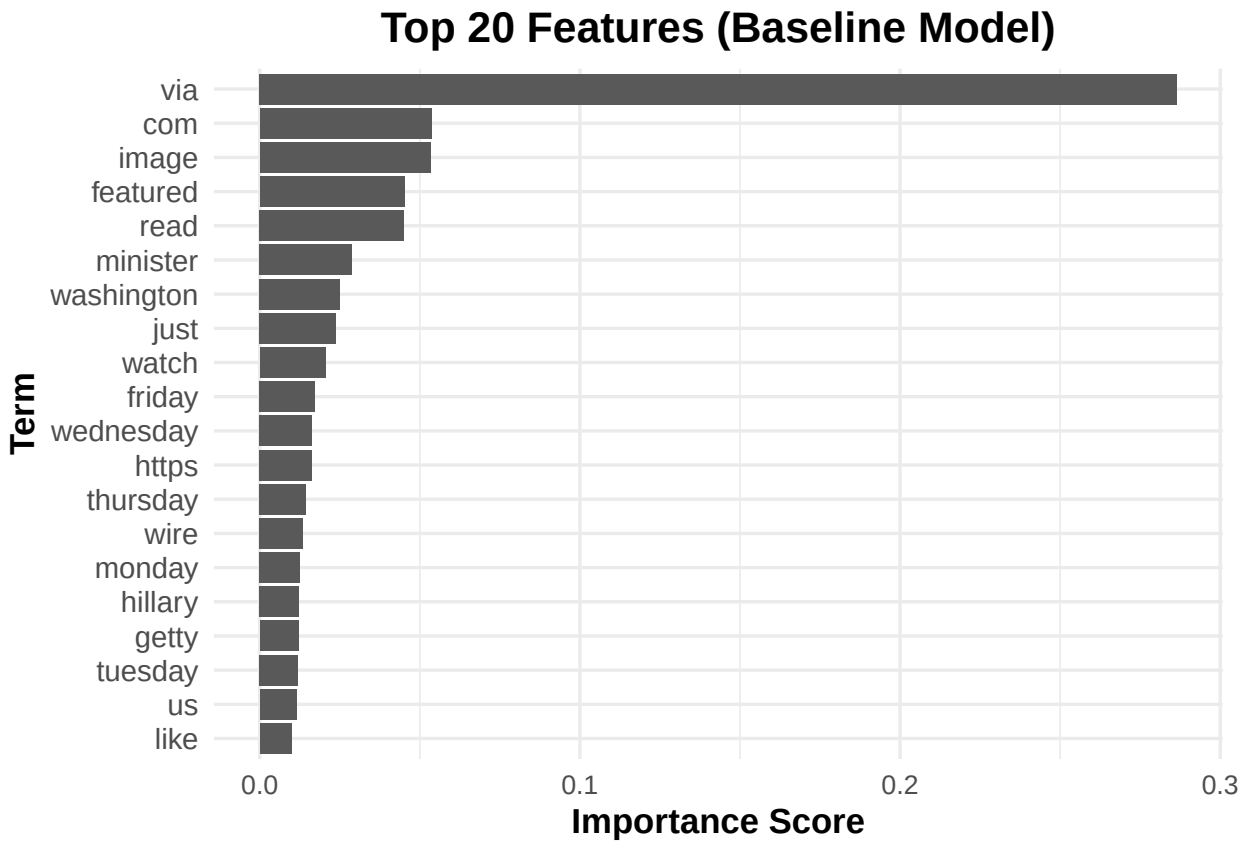
Similar classification metrices were used below for model evaluation of the domains. Since there was no subject that could be classified under "other" domain, just the two domains of news and politics are evaluated.

```
##                Domain  Accuracy Precision    Recall        F1
## Accuracy...1     News 0.9948092 0.9974900 0.9925075 0.9949925
## Accuracy...2 Politics 0.9790521 0.9731664 0.9908925 0.9819495
```

The above table shows that nearly perfect results for the respective domain accuracies, precision, recall and F1. In real-world scenarios, such perfect results are unrealistic, but this may be due to the lack of diversity in the data set and the structured nature of it

4.4 Baseline Model Feature Importance Visual

The following is a visual that illustrates the top most features when it comes to XGBoost importance. It is a very simple visual but provides an understanding of the model's use and output.



As seen above, words like Minister, Washington, Hilary are relevant to the articles. However, the other top 20 features include words that do not carry much semantic meaning. Although XGBoost has strong performance, it still comes with its limitations as it has limited interpretability to understand thematic relationships between texts and cannot leverage semantic meaning. To address, this the model must be improved and the improvement can be done by introducing bigrams in the TF-IDF along with the unigrams. This allows the model to use word sequences and create patterns rather than separate words that don't reflect the content. Here terms like "breaking news" and "sources say" can be reflected more, capturing nuances that the unigram fails to.

5 Improved Model

5.1 Advanced Feature Engineering

To improve the model the TF-IDF vocabulary was re-defined to include n-grams of length 2 - allowing the model to learn sequences of words.

5.2 Improved Model Prediction and Evaluation

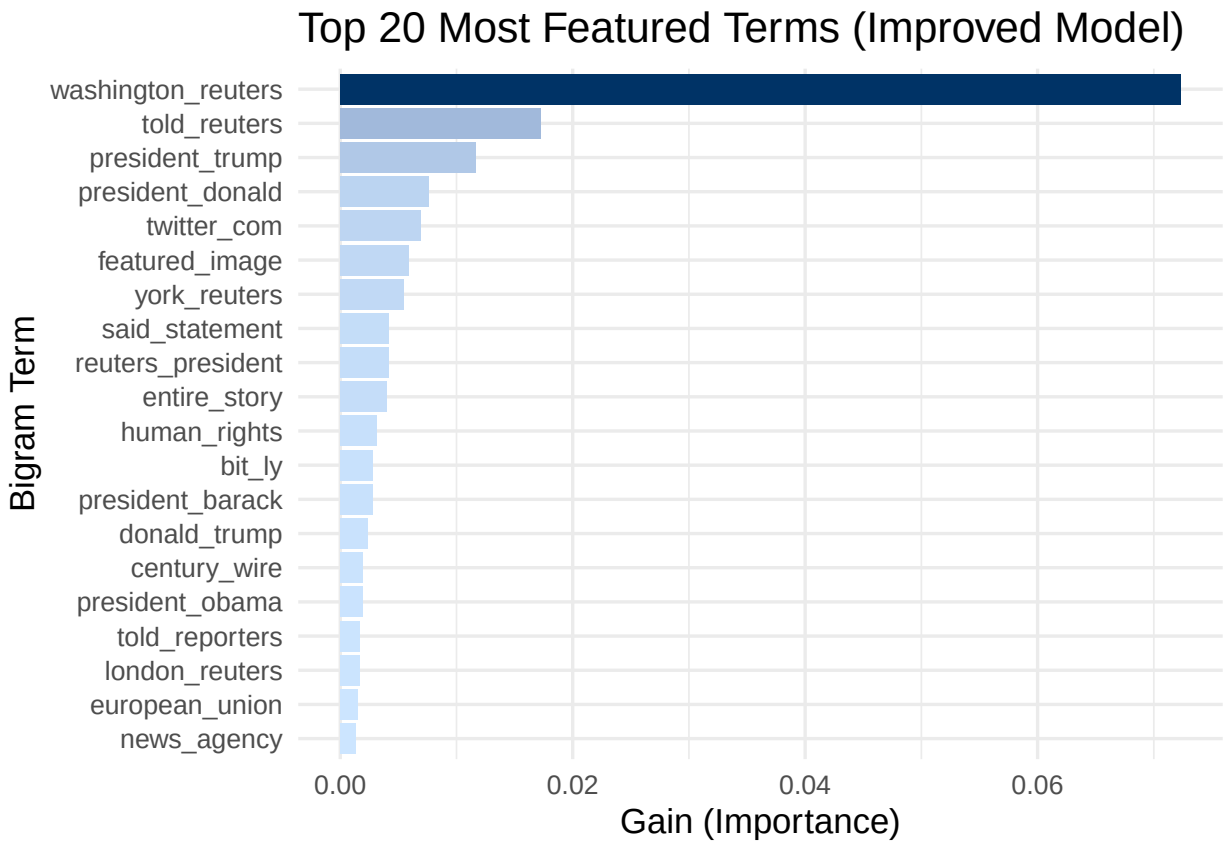
The XGboost model was thereby re-trained, predicted again and evaluated using the confusion matrix

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

```
## Prediction    0    1
##          0 3440   31
##          1   34 4167
##
##                Accuracy : 0.9915
##                  95% CI : (0.9892, 0.9935)
##     No Information Rate : 0.5472
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9829
##
##  Mcnemar's Test P-Value : 0.8041
##
##             Sensitivity : 0.9926
##             Specificity : 0.9902
##          Pos Pred Value : 0.9919
##          Neg Pred Value : 0.9911
##              Prevalence : 0.5472
##          Detection Rate : 0.5431
##    Detection Prevalence : 0.5476
##       Balanced Accuracy : 0.9914
##
##        'Positive' Class : 1
##
```

Upon re-running the model after introducing advanced feature engineering. The overall model performance improved. The accuracy increased from 98.70% to 99.15%, the recall increased from 99.17% to 99.26%, the F1 score also sightly improved from 98.81% to 99.2% reflecting a strong model. The improved model also has lower false positives in comparison to the baseline model. The results indicated that the model is making fewer incorrect predictions and suggests that the predictions are more stable and balanced.

5.3 Improved Model - Most Important Terms Visual



As seen above, the top featured terms provide more insight into the context of the articles. Instead of just report president, it reports president Obama, reflecting which president is being talked about. Furthermore, instead of investigating a unigram that just says human, the bigram suggests human rights providing more information about the article itself - allowing for better judgements as to if the news is fake or real by the model

6 Conclusion

With building a baseline XGBoost model and using TF-IDF features, a model with high performance was created to classify fake vs. real news. Bigrams were added as an advanced feature engineering tool which refined the processes and improved performance of the model which was evaluated using classification metrices. The accuracy was thereby increased to 99.15% and false positives were reduced. The project reflects that feature engineering enhancements can strengthen the classifcation models especially in the context of misinformation detection.

7 Code appendix

```
knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE)
options(reticulate.python = NULL)
library(tidyverse)
```

```r
library(lubridate)
library(textclean)
library(text2vec)
library(stopwords)
library(Matrix)
library(xgboost)
library(stringr)
library(ggplot2)
library(caret)


# Data Processing

# loading both fake and true news datasets and adding a binary label
fake_news <- read_csv("Fake.csv") %>%
  mutate(label = 0)

true_news <- read_csv("True.csv") %>%
  mutate(label = 1)

# combining the two data-sets
news_data <- bind_rows(fake_news, true_news)

# clean data
set.seed(234)
news_data <- news_data %>%
  mutate(date = parse_date_time(date, orders = c("dmy", "ymd", "mdy"))) %>% # parse date
  mutate(date = as.Date(date)) %>%
  sample_frac(1)  %>% # shuffle data
  filter(!is.na(text), !is.na(date)) # deal with missing rows

# classifying subjects into categories
news_data <- news_data %>%
  mutate(domain = case_when(
    subject %in% c("politics", "politicsNews", "left-news", "Government News", "Middle-east") ~ "Politi
    subject %in% c("News", "worldnews", "US_News") ~ "News",
    TRUE ~ "Other"
  ))



# number of real vs fake articles
labels <- news_data %>%
  count(label) %>%
  mutate(label = ifelse(label == 1, "Real", "Fake"))

# distribution of different domains
domains <- news_data %>%
  count(domain)

# View results
labels
domains
```

```r
#clean and normalize the text
news_data <- news_data %>%
  mutate(text = tolower(text),
         text = replace_contraction(text),
         text = str_replace_all(text, "[^a-z\\s]", " "),
         text = str_squish(text))

#splitting data into train and test sets as a pre-process for TF-IDF
set.seed(234)
news_data <- news_data %>% distinct(text, .keep_all = TRUE) # to remove duplicates
split_indices <- sample(nrow(news_data), size = 0.8 * nrow(news_data))
train_set <- news_data[split_indices, ]
test_set <- news_data[-split_indices, ]
# Feature Engineering
# tokenizing the train and test data to create a text iterator
itr_train <- itoken(train_set$text,
                    tokenizer = word_tokenizer,
                    progressbar = TRUE)

itr_test <- itoken(test_set$text,
                   tokenizer = word_tokenizer,
                   progressbar = TRUE)

#removing unnecessary tokens
custom_token <- function(text) {
  tokens <- word_tokenizer(text)
  lapply(tokens, function(x) x[nchar(x) > 1])
}

itr_train <- itoken(train_set$text, tokenizer = custom_token, progressbar = TRUE)
itr_test  <- itoken(test_set$text,  tokenizer = custom_token, progressbar = TRUE)
stop_words <- stopwords("en")
stop_word <- "said"

# building vocabulary using the tokenized text iterations
news_vocab <- create_vocabulary(itr_train, stopwords = stop_words) %>%
  filter(!term %in% stop_word) %>%
  prune_vocabulary(term_count_min = 5, doc_proportion_max = 0.5)
# using the vocab to create a vectorizer
news_vectorizer <- vocab_vectorizer(news_vocab)

#creating a document & term matrix  to train TF-IDF
dtm_train <- create_dtm(itr_train, news_vectorizer)
tf_idf <- TfIdf$new()

#scaling frequency and importance of words
dtm_train_tfidf <- tf_idf$fit_transform(dtm_train)

#applying trained TF-IDF to test set
dtm_test <- create_dtm(itr_test, news_vectorizer)
dtm_test_tfidf <- tf_idf$transform(dtm_test)

col_sums <- colSums(dtm_train_tfidf)
```

```r
head(sort(col_sums, decreasing = TRUE), 20)

# Baseline Model
# XGBoost Classifier
#creating the train and test matrices to prepare for model fitting
train_xgb <- xgb.DMatrix(data = dtm_train_tfidf, label = train_set$label)
test_xgb  <- xgb.DMatrix(data = dtm_test_tfidf, label = test_set$label)

# defining parameters for the baseline classification model
parameters <- list(
  objective = "binary:logistic",
  eval_metric = "logloss",
  max_depth = 6,
  eta = 0.3
)
# Training
baseline_model <- xgb.train(params = parameters, data = train_xgb, nrounds = 100)

# Predicting based on the XGBoost test matrix
prediction_probability <- predict(baseline_model, test_xgb)
prediction_labels <- ifelse(prediction_probability > 0.5, 1, 0)

# Baseline Model Evaluation
#Computing a confusion matrix for evaluation
confusion_matrix <- confusionMatrix(factor(prediction_labels), factor(test_set$label), positive = "1")
print(confusion_matrix)

# Baseline Model Evaluation Based on Subject Domains
#adding the predictions to the test set
test_set$predicted <- prediction_labels

#computing evaluation metrics based on the domain
evaluation_domain <- function(domain) {
  subset <- test_set[test_set$domain == domain, ]
  cm <- confusionMatrix(factor(subset$predicted), factor(subset$label), positive = "1")
  metrics <- data.frame(
    Domain = domain,
    Accuracy = cm$overall["Accuracy"],
    Precision = cm$byClass["Precision"],
    Recall = cm$byClass["Recall"],
    F1 = cm$byClass["F1"]
  )
  return(metrics)
}

#presenting results for both domains
domain_eval_results <- bind_rows(
  evaluation_domain("News"),
  evaluation_domain("Politics"))
print(domain_eval_results)
```

```r
# Baseline Model Feature Importance Visual
importance_matrix_baseline <- xgb.importance(model = baseline_model)
topmost_features <- importance_matrix_baseline[1:20, ]

# Plotting the most important texts/words in ggplot
ggplot(topmost_features, aes(x = reorder(Feature, Gain), y = Gain)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  labs(
    title = "Top 20 Features (Baseline Model)",
    x = "Term",
    y = "Importance Score"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.y = element_text(size = 11),
    axis.title = element_text(face = "bold")
  )

# Improved Model
# Advanced Feature Engineering
# Recreating the vocabulary with bigrams and removing unimportant terms
vocab_ngram <- create_vocabulary(itr_train, stopwords = stop_words, ngram = c(1, 2)) %>%
  prune_vocabulary(term_count_min = 5, doc_proportion_max = 0.5)

#converting vocabulary into a format
vectorizer_ngram <- vocab_vectorizer(vocab_ngram)

#creating a matrix (train data) with rows as documents and columns as bigrams
dtm_train_ngram <- create_dtm(itr_train, vectorizer_ngram)
dtm_train_ngram_tfidf <- tf_idf$fit_transform(dtm_train_ngram)

#creating a matrix (test data) with rows as documents and columns as bigrams
dtm_test_ngram <- create_dtm(itr_test, vectorizer_ngram)
dtm_test_ngram_tfidf <- tf_idf$transform(dtm_test_ngram)

#converts sparce matrices to xgb matrix format
train_matrix_ngram <- xgb.DMatrix(data = dtm_train_ngram_tfidf, label = train_set$label)
test_matrix_ngram  <- xgb.DMatrix(data = dtm_test_ngram_tfidf, label = test_set$label)

# Improved Model Prediction and Evaluation
#training the improved model based on new ngram features
improved_model <- xgb.train(params = parameters, data = train_matrix_ngram, nrounds = 100)

#predicting the improved model
pred_probs_ngram <- predict(improved_model, test_matrix_ngram)
pred_labels_ngram <- ifelse(pred_probs_ngram > 0.5, 1, 0)

#evaluating the improved model
conf_matrix_improved <- confusionMatrix(factor(pred_labels_ngram), factor(test_set$label), positive = "
print(conf_matrix_improved)
```

```r
# Improved Model - Most Important Terms Visual
#formatting the features to prepare for visual modelling
importance_matrix_improved <- xgb.importance(model = improved_model)
top_bigram_features <- importance_matrix_improved %>%
  filter(str_detect(Feature, "_")) %>%
  top_n(20, wt = Gain) %>%
  arrange(Gain)

ggplot(top_bigram_features, aes(x = reorder(Feature, Gain), y = Gain, fill = Gain)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  labs(
    title = "Top 20 Most Featured Terms (Improved Model)",
    x = "Bigram Term",
    y = "Gain (Importance)"
  ) +
  scale_fill_gradient(low = "#cce5ff", high = "#003366") +
  theme_minimal(base_size = 13)
```