Balanced News Recommendation Algorithm

Final Report

Problem Statement

The internet has brought consumers access to a near infinite supply of information. Arguably, the most critical type of information is news. News not only shapes people's perception of the world, but it also can lead to action. Today, most recommendation algorithms curate news media based on user engagement. This tends to polarize individuals to the political extremes and fosters a demand for misinformation and media bias.

Counterintuitively, I sought to design a recommendation algorithm that would feed users news articles they are less than 75% likely to interact with half of the time. The remainder of their news feed being that which the user is likely to interact with. Ideally, this would retain engagement as it also exposes users to a more balanced newsfeed.

Data Wrangling

To build my recommendation algorithm, I selected the Microsoft News Recommendation Dataset (MIND). The raw data frame contained 5 columns and over two million rows. I soon found that a much smaller sample was sufficient to train my algorithm. Thus, I took a random sample of 5,000 rows to work with. For my columns, I was only interested in obtaining three features: User ID, Article ID, and an interaction column. User ID was already provided; however, creating an article ID and interaction column required some additional data manipulation steps. I ended up merging two related columns to create my article ID column and then pulled out the embedded user-interaction information within that column to form my interaction column. The interaction column contained a 1 for a user click on an article and a 0 for no click. All null values were dropped. My final dataset contained 3 columns and 346,635 rows.

Exploratory Data Analysis

One of the first things I wanted to ensure before fitting my data to any model was class balance. For example, if my data consisted of 90% clicks and only 10% non-clicks or vice versa, that could skew predictions and cause misleading model accuracy. The visualization below ensured my class data was relatively balanced with slightly fewer clicks than non-clicks.

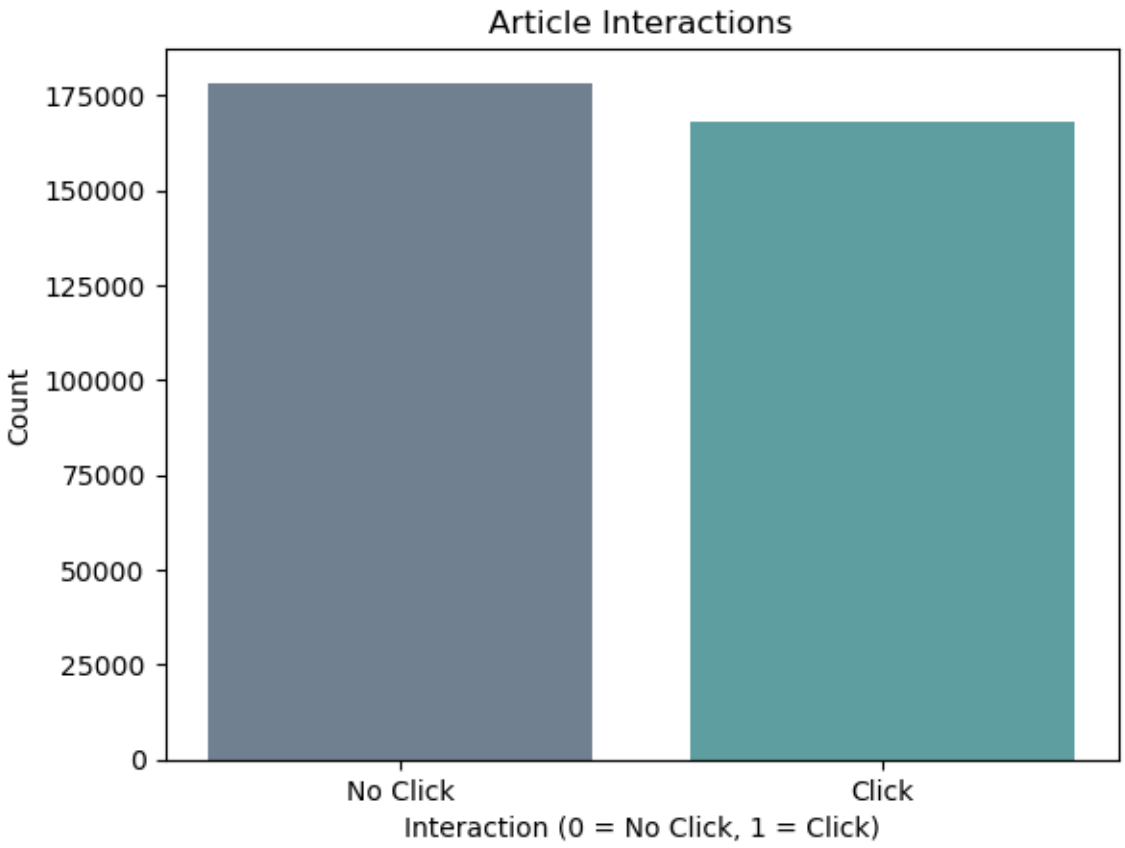

Figure 1: Bar plot showing relative balance of user engagement with news articles from the dataset

The next piece of information I wanted to explore in my dataset was how engaged most users were. I chose a histogram for this visualization as seen in Figure 2. The results showed a positively skewed distribution with a mean of 54 interactions per user within the 6-week period that the dataset was collected.

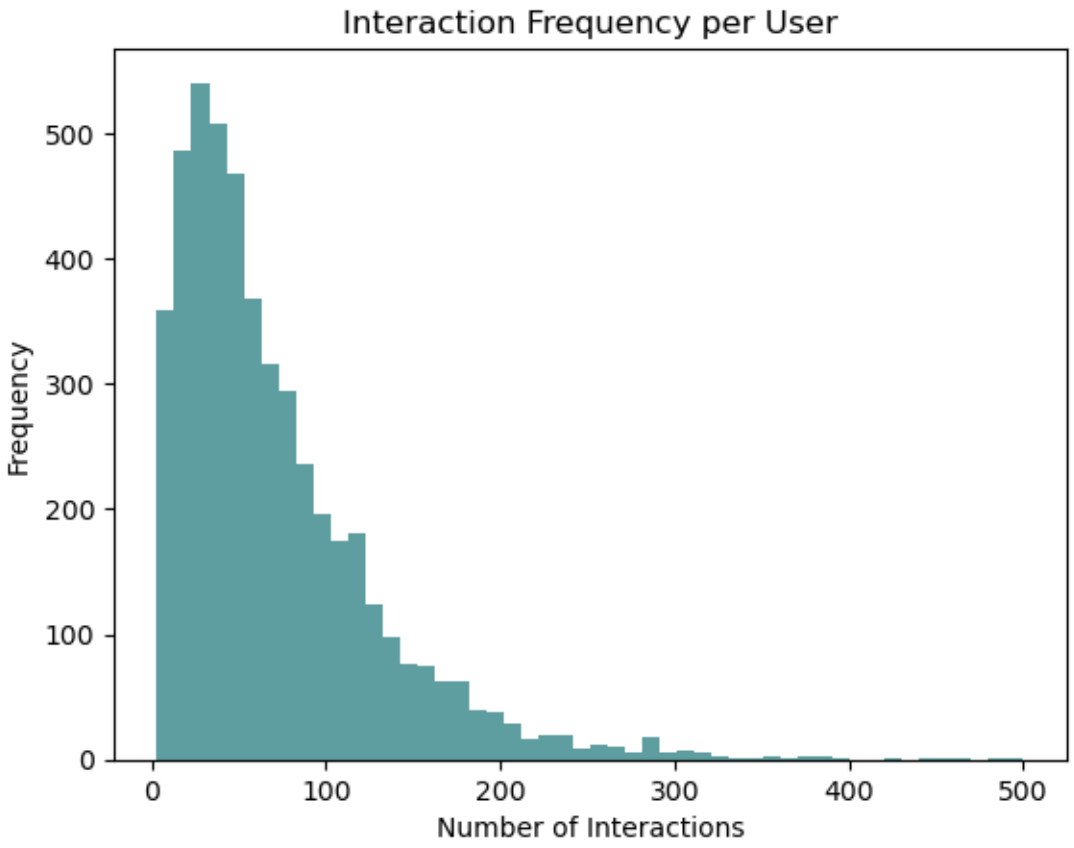

Figure 2: Histogram showing distribution of user engagement with news articles from the dataset

Now that I had confirmed the interaction distribution followed an intuitively correct pattern, I was interested in the distribution of article popularity. After generating a scatter plot of Article ID vs number of clicks, I identified that the mean number of clicks per article was around nine with higher levels of clicks becoming increasingly sparce.

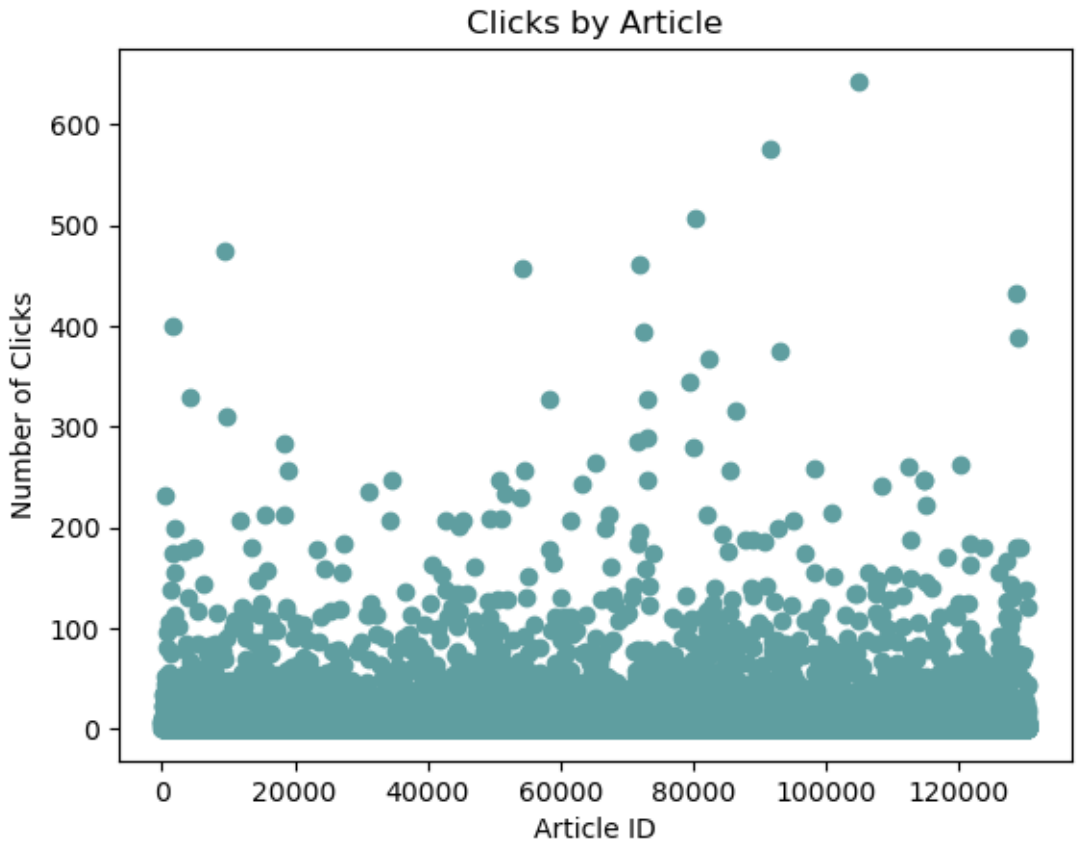

Figure 3: Scatterplot showing distribution of article clicks for all unique articles

I now had a large balanced sampling of users interacting with articles following a confirmed pattern of expected engagement. It was time to prepare for modeling.

Model Building

Before fitting the data to any model, I considered one hot encoding and magnitude standardization. However, neither of these steps were necessary. The Article ID and User ID had extremely high cardinality, so it did not make sense to perform one hot encoding on those columns. Moreover, their values were categorical, so magnitude standardization was unnecessary. On the other hand, the Interaction column was binary, and therefore, one hot encoded—with its magnitude already standardized on 0 and 1.

It was time to split the data into train and test data subsets. I performed this using the train_test_split object from sklearn. I reserved 20% of the data for testing and kept 80% of the data for training.

The first model I chose to fit and test on was a logistic regression model from sklearn. Unfortunately, this produced an accuracy of 0.51 was virtually identical to random guessing. I performed a grid search on the model with different parameters, but did not get

significant improvement. I then decided to try an XG Boosting model and was able to improve my accuracy to 62%. While this algorithm was better than simply random guessing, I wanted an algorithm that gave better scores on performance metrics. The Random Forest Classifier model was the last model I tested the data on. After tuning the model with a Bayesian grid search, I was able to create a model with an accuracy of over 80%. The figure below shows a side-by-side comparison of the tested models.

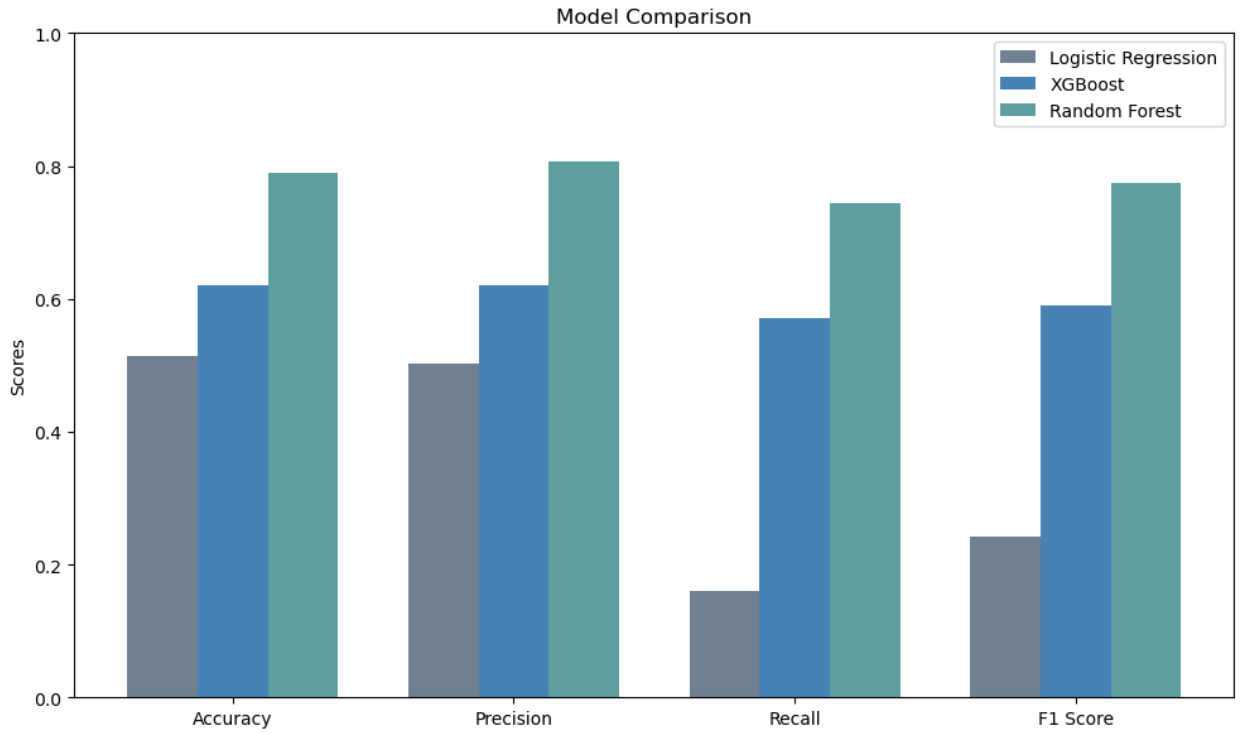

Figure 4: Displays a performance comparison of three models tested on the data

As figure 4 clearly display, the Random Forest Classifier performed best in accuracy, precision, recall, and F1 score. I therefore chose it to be my recommendation algorithm. However, I needed to generate recommendations that limited bias. I used the predict_proba() object on the random forest model to extract the likelihood of a user interacting with an article. I took these probabilities and generated 10 article recommendations for each user with 50% of those recommendations being less than 75% likely to be interacted with and the remaining being higher than 75% likely to be interacted with. Hence, these recommendations would keep users engaged but also provide them with a more balanced newsfeed.

Application & Future Work

This type of recommendation algorithm could be applied to applications such as Google News, Apple News, or Microsoft News. This project does not actually implement

the algorithm into any of these apps, but it does lay the groundwork and demonstrate that a new and more balanced newsfeed is possible. The recommendation algorithm built here follows a rather simple approach to the problem. Future recommendation algorithms seeking to balance news feed but retain user engagement could tweak the predicted interaction cutoff for the recommendation distribution based on a series of experiments.