FINAL PROJECT

2025-04-16

Introduction

In this machine learning project, I will be attempting to generate a natural language processing (NLP) model based on the work conducted for my senior thesis in Barnard's Sociology Department. This NLP project uses hand-curated quotes from a collection of speeches from the former Prime Minister of India, Indira Gandhi, to train a machine learning model that can classify political rhetoric into multiple thematic categories. While the audience for this project is rather targeted to my needs in my political and historical sociology studies, as I hope to be able to more quickly and efficiently code my data to make my analysis more robust and well-supported instead of doing it all by hand, this applied context is also useful for historians and other political sociologists interested in discourse analysis and/or the political trajectory of late 20th-century Indian politics, especially those who are looking to pursue more technologically-advanced forms of data collection. As such, the goal of this project is to be able to automate the coding of historical political text to generate multi-label classification of quotes and enable future large-scale analyses of rhetorical patterns that can be aided using scalable machine learning methods.

XGBoost Model

The most impactful result in trying to generate a machine learning model that can classify historical political dialectic data, split at the quote/paragraph level, was an XGBoost model trained upon a matrix of TF-IDF vectorized quotes. The resulting model maintained somewhat meaningful performance metrics– although, interestingly, it was largely outperformed on all metrics by a logistic regression model– but also provided a clear competency in obtaining semantic insights into the words that most contributing to the assignment of certain codes. The tradeoff in performance metrics in favor of applied semantic meaning was an important decision I made in constructing this model that will be explored in depth below. This model (and the logistic model) entirely failed when applyed to the test set, which indicates that there is a lack of generalizability and a large degree of overfitting that occurred in my model training. While the model cannot be taken into an applied context (just yet), the results still provide meaningful insights into how machine learning algorithms can (potentially) illuminate the relevance of semantically meaningful words in relation to important and lofty social science themes.

Exploratory Data Analysis

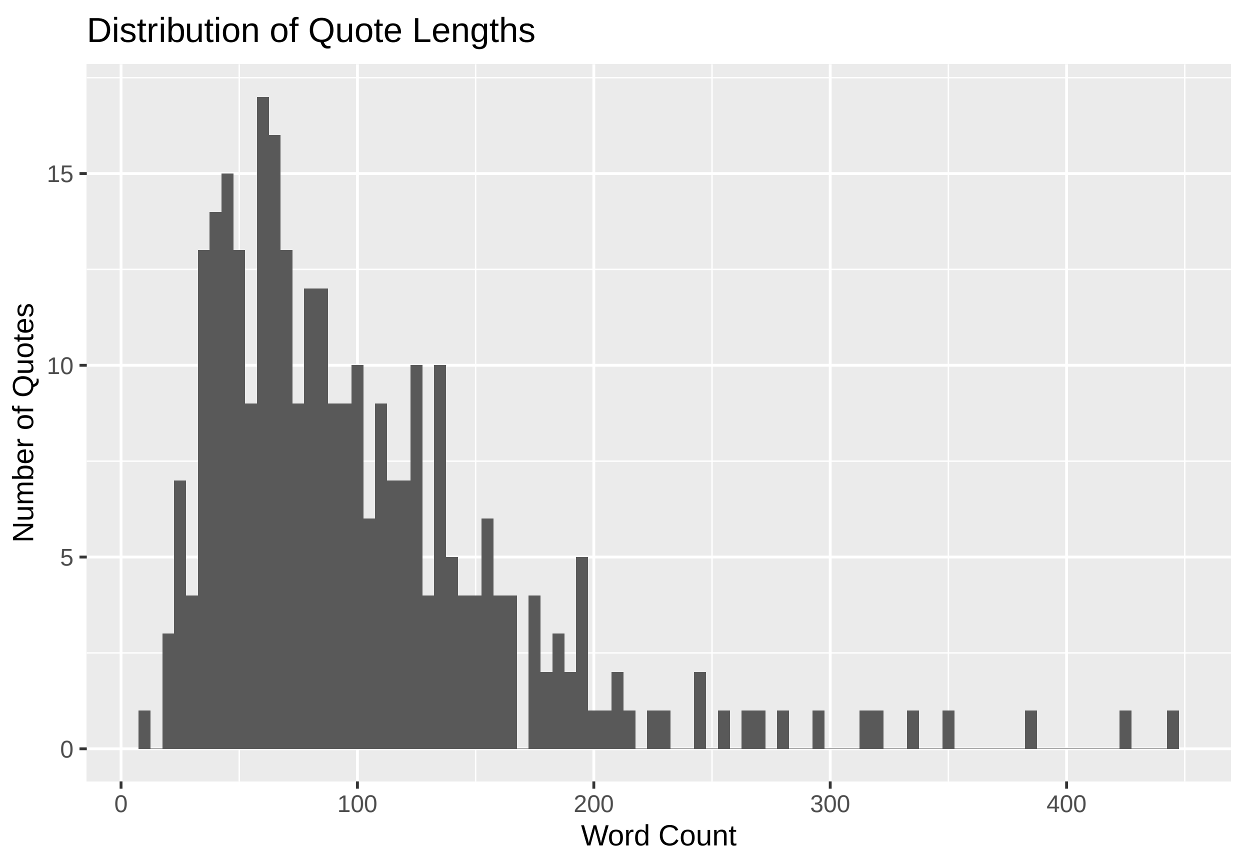
The data used to train this model consists of one single dataset that I manually created from a selection of published resources of Indira Gandhi's speeches. The first set of resources is a catalogue of Independence Day speeches delivered by Indira Gandhi, which totals 15 transcripts, delivered between 1966 to 1975, followed by a gap during which the Janata Party was in power, and concluding with speeches from 1980 to 1984. The other– and more robust– selection of speeches comes from a published book of speeches Indira delivered in Parliament from 1966 to 1984, titled Indira Gandhi, Speeches in Parliament. These roughly 245 speeches were formally curated and published by the Lok Sabha secretariat in 1996. From these resources, I extracted quotes (herein referred to as "documents") that ranged from about two sentences long to thirty sentences long, which can be roughly mapped onto paragraph lengths.

Given the fact that the database was curated by me (and without this original project in mind), I have no doubts that there is surely some element of selection bias in how I chose the quotes to include in the set. A better version of this dataset which could have amounted to a more robust model would've likely come from including all the content of all the resource documents. Importantly, the documents I selected were only ones I deemed significant, and a truly robust model would have benefited from having unimportant documents, as well, as part of the significance of constructing a model such as this one is that the extraction can be done by the model instead of by the researcher only. The reason I still believe this model is meaningful is largely because of the structure of multi classification– technically, all documents are important because they all have an important code associated with them, but not all documents are important relative to each code. Thus, this still allows for a diversity in classifications that is necessary for such a model as all documents are "positives" for some codes, and "negatives" for others (but there are no documents that are all "negatives"). If I had more time in obtaining the data for this project, this is surely an element of the data acquisition I would've pursued.

```
## # A tibble: 6 x 5
##   SOURCE                                       DATE            PAGE QUOTE CODES
##   <chr>                                        <chr>          <dbl> <chr> <chr>
## 1 State Politics & Union-State Relations (IGSIP) March 11, 19~   664 "Man~ "dem~
## 2 India & the World (IGSIP)                    April 7, 1966   755 "But~ "Ind~
## 3 No Confidence Motions (IGSIP)                November 7, ~   272 "I b~ "nat~
## 4 No Confidence Motions (IGSIP)                November 7, ~   274 "A s~ "nat~
## 5 No Confidence Motions (IGSIP)                November 7, ~   275 "Thi~ "emo~
## 6 No Confidence Motions (IGSIP)                November 7, ~   276 "We ~ "Ind~
```

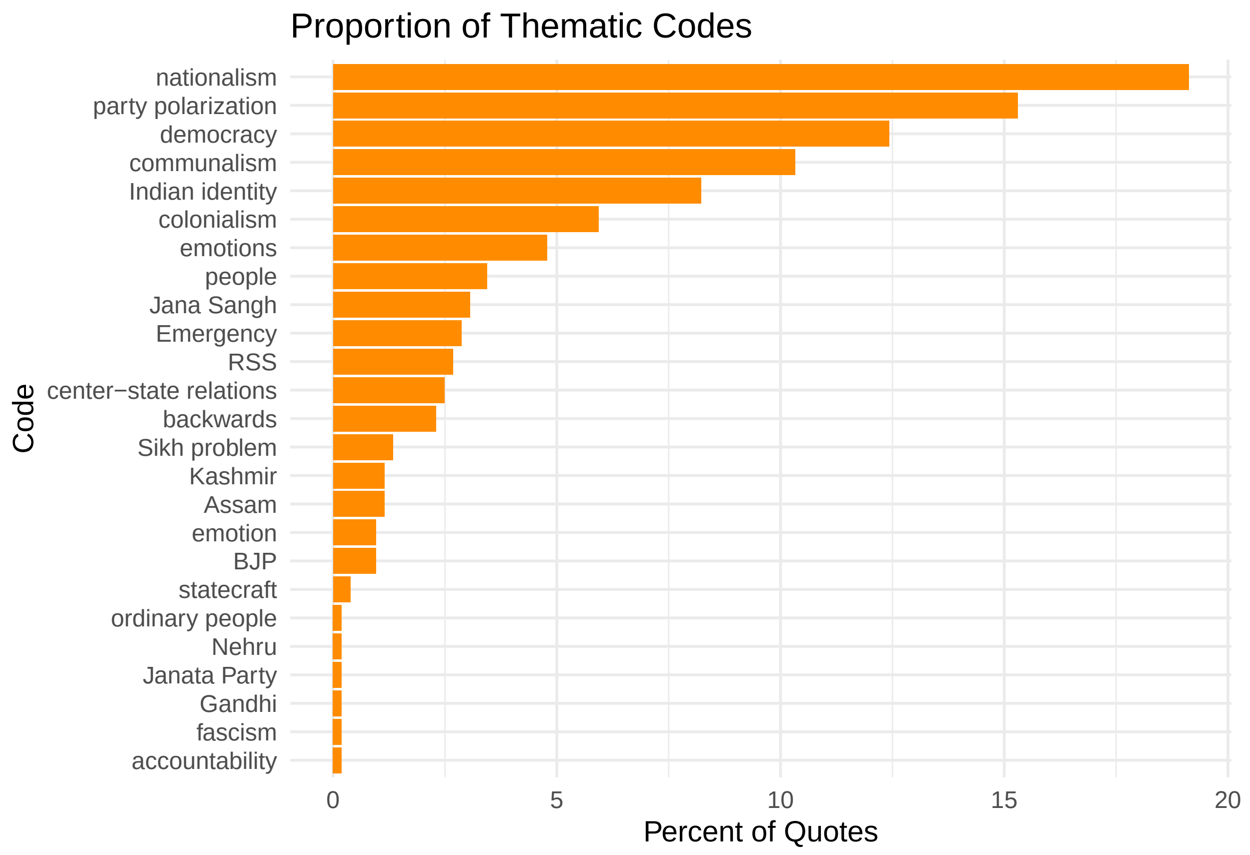
The preprocessed dataset (originally used for my senior thesis), as shown above, included columns for the date the speech was given, the source material (whether that be Independence Day Speeches or a specific chapter of the Indira Gandhi Speeches in Parliament book), the page number(s), the quote itself, plus the code(s) I manually assigned for the quote. The dataset was curated to be in chronological order, and is made of 304 unique documents (n = 304).

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    10.0    57.0    85.0   101.9   130.5   444.0
```



To ensure consistency in document granularity, I visualized the distribution of quote lengths. Most quotes fall between 40–120 words, with a median around 75 words. There is a small number of long outliers (up to 400+ words) and very little extremeley short quotes (under 20 words). There's a strong central cluster in the 50–100 word range. As described earlier, there is a level of irregularity to be wary of in the construction of what constitutes a document, as there isn't a standard form such as the content of a whole speech might be. At the same time, though, the speeches themselves vary tremendously in length as well (and an element of the TF-IDF algorithm that will be used accounts for variation in document length to a certain extent).

```
## # A tibble: 25 x 2
##    code_list          count
##    <chr>              <int>
##  1 nationalism          100
##  2 party polarization    80
##  3 democracy             65
##  4 communalism           54
##  5 Indian identity       43
##  6 colonialism           31
##  7 emotions              25
##  8 people                18
##  9 Jana Sangh            16
## 10 Emergency             15
## # i 15 more rows
```



The distribution of codes– our target variables– ranges from 1 document to 100 documents. Given the size of the dataset, while the p of the data is quite large, the n is rather small at just over 300. As such, there are surely issues with class imbalances and how robust our codes are, which means we must narrow down the codes we will be using to that which is actually trainable for a machine learning model. As such, the top eight codes **(nationalism, party polarization, democracy, communalism, Indian identity, colonialism, emotions, and people)** will be our target "y" variable.

Feature Engineering

In order to generate a trainable dataset, we must preprocess the data and conduct feature engineering. The dataset was first split into a 80-20 testing and training set with a stratified sampling method to ensure all codes were proportionally represented in the training and test sets. I then engineered the features by using a TF-IDF algorithm, which separated each quote into tokens (the features) that mapped onto one word, with capitalization, punctuation, and typical stopwords removed. Rare words (appear < 3 times) and overused words (appear in $> 90\%$ of quotes) were also removed. With the tokens as the columns, the TF-IDF matrix was populated by generating the product of the token's frequency in that quote with the inverse of how frequently it appears across all quotes.

The target or Y matrix, as shown below, was made by doing one-hot encoding for all our target categories, in which columns are codes (like emotions, communalism_caste) and entries are 0 or 1 (whether that code applies to the quote).

```
##      colonialism communalism democracy emotions Indian_identity nationalism
## [1,]           0           0         1        0               0           0
## [2,]           0           0         0        0               1           0
## [3,]           0           0         0        0               0           1
## [4,]           0           0         0        1               0           0
## [5,]           0           0         0        0               1           0
## [6,]           0           0         0        0               0           0
##      party_polarization people RSS
## [1,]                  0      0   0
## [2,]                  0      0   0
## [3,]                  0      0   0
## [4,]                  0      0   0
## [5,]                  0      0   0
## [6,]                  1      0   0
```

Model Selection & Comparison

An essential componenent in selecting a model was in remembering what element to prioritize in the model's performance within this applied social science context. Recall is a very important metric to prioritize, as researchers (myself included) are not meant to use this model as a replacement for conducting actual coding; there is still an expectation that a human eye will be going over the results of the model. As such, accidentally including quotes that may not actually be relevant to the code is a permissable tradeoff in capturing all possible coded documents than being overly conservative in your positive guesses and missing what could've been important data in the first place.

The process of selecting the XGBoost model largely began with generating some form of a baseline model that can be assessed in comparison to further successive, more complex models. That initial model was a multilabel 5-fold cross-validated logistic regression model with an adjusted 0.3 cutoff threshold to generate higher recall performance for minority classes, trained upon a TF-IDF matrix without stopword removal (which is acceptable due to penalization of frequent terms in TF-IDF and regularization in logistic regression). This model performed very well on accuracy, precision, and recall metrics.

After generating this logistic regression model, a 5-fold cross-validated random forest model was generated next, trained upon the stopword removed TF-IDF matrix, also utilizing an adjusted cutoff threshold of 0.3, and computed using upsampling methods to account for major class imbalances in many of the codes (as some codes only have 20-50 positive cases versus 200+ negative ones). This random forest model maintained such poor performance metrics all around the board that it was almost immedietly disregarded in favor of a more nuanced trees-based model via XGBoost, which could also handle imbalanced class data a bit more intuitively. The performance metrics for this random forest model are shown below.

```
## # A tibble: 8 x 4
##   Label              Accuracy Precision Recall
##   <chr>                 <dbl>     <dbl>  <dbl>
## 1 colonialism           0.896    NA      0
## 2 communalism           0.894     1      0.375
## 3 democracy             0.766     0.333  0.1
## 4 emotions              0.917    NA      0
## 5 Indian_identity       0.851    NA      0
## 6 nationalism           0.792     0.733  0.647
## 7 party_polarization    0.708     0.455  0.385
## 8 people                0.938    NA      0
```

> The XGBoost model that I constructed was trained using 5-fold cross validation on TF-IDF features with stopword removal. Initially, it massively underperformed logistic regression across most metrics, but I was able to systematically improve its performance by using upsampling techniques (trainControl(sampling = "up")) to account for the major class imbalances, hyperparameter tuning with a manual grid (e.g. eta, max_depth, min_child_weight), and adjusting the threshold to 0.3 to reduce conservativeness and improve recall.

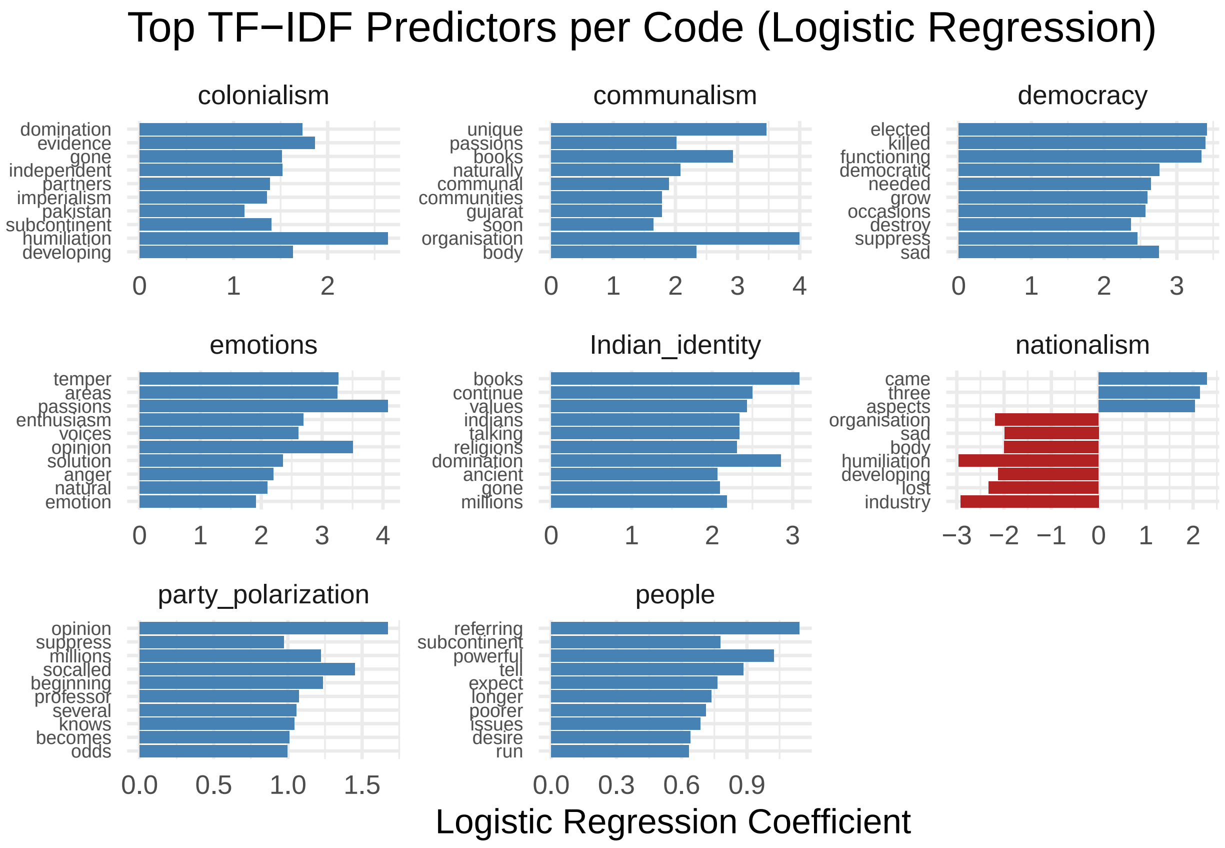
At this point, the decision process behind selecting a model largely came down to what I originally thought was just going to be my "reasonable but stupid" baseline model (logistic regression) and XGBoost. It is worth noting here that in comparing performance metrics across all the models I constructed, with hyperparameter tuning and otherwise, some codes systematically underperformed across the board. Specifically, **colonialism** and **people** maintained very poor precision and recall scores for all models, indicating that there may not be enough data for the models to pick up on meaningful signals for generating labels for those codes.

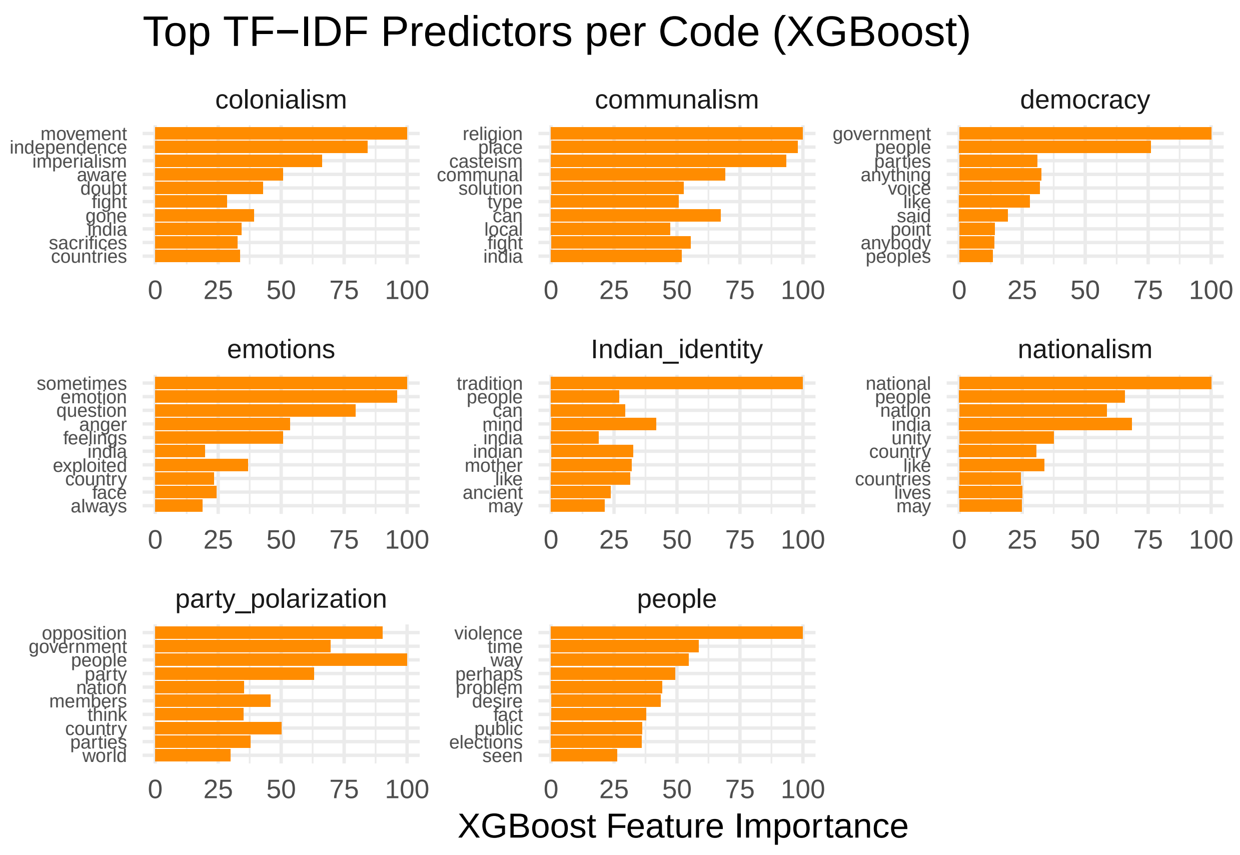
```
## # A tibble: 8 x 4
##   Label              Accuracy Precision Recall
##   <chr>                 <dbl>     <dbl>  <dbl>
## 1 colonialism           0.908     1     0.0435
## 2 communalism           0.987     1     0.930
## 3 democracy             1         1     1
## 4 emotions              0.992     1     0.895
## 5 Indian_identity       0.992     1     0.946
## 6 nationalism           0.987     0.965 1
## 7 party_polarization    0.996     0.985 1
## 8 people                0.937    NA     0
```

```
## # A tibble: 8 x 4
##   Label              Accuracy Precision Recall
##   <chr>                 <dbl>     <dbl>  <dbl>
## 1 colonialism           0.896    NA      0
## 2 communalism           0.766     0.364  0.5
## 3 democracy             0.771     0.471  0.8
## 4 emotions              0.938     1      0.25
## 5 Indian_identity       0.830     0.4    0.286
## 6 nationalism           0.708     0.565  0.765
## 7 party_polarization    0.792     0.579  0.846
## 8 people                0.787     0      0
```

The performance metrics based on averages across the 5 folds of the train set for both the XGBoost model and logistic regression model highlight how **colonialism** and **people** consistently underperform, but also illuminates logistic regression's impressive results. The performance metrics for XGBoost are not as poor as random forest but undoubtedly do not compare to that of logistic regression. I was automatically hesitant to see such perfect performance metrics for the logistic model, which makes me question whether there was a degree of overfitting or something of that sort with the model. But, with three codes maintaining recall scores of above 70%, there is some degree of merit to be found within the XGBoost model that I think could become a lot more workable with additional tuning measures.

Thus, another important componenent that factored into the model selection was interpretability. In generating these performance metrics, I then generated the most important features for each model across all codes. For logistic regression, this was the top ten features with the largest magnitude Beta value for each code, while for XGBoost feature importance was calculated by quantifying the average improvement in the model's function when that feature is used (i.e. the features that contribute the most to improving model performance across all the decision trees in the ensemble when that feature is used to split).





The ability to compare and contrast the important features across the model provided very interesting insights that were particularly meaningful to me as a researcher who has been studying this content for nearly six months now. Both models provided features that were semantically meaningful in the context of the codes they are impacting, but by far XGBoost's important features cohered with my understanding of the dialectical construction of the codes as I manually classified them, capturing essential thematic patterns. For example, XGBoost's terms such as "movement," "independence," and "imperialism" have clear semiotic meaning in the context of **colonialism**, but the terms "beginning" and "dedication" have much looser connections to **colonialism**. Interestingly, even in a code such as **colonialism** which is poorly categorized for all models, XGBoost still is able to pick up on the meaningful connotation of the terms as it relates to the code even if it struggles to generalize and actually classify. Across the board, the important features generated by XGBoost cohere with what my research has concluded in my thesis– for example, a notion of tradition and ancient history is essential for Indira Gandhi's construction of an Indian identity as she tries to move past the pitfalls of her father, the first Prime Minister of India, whose construction of national identity focused only on a modernist progressive future that ignored the reality of India's pre-colonial history which contains the communal identities that are pulling the people's allegience away from allegience to the nation. Thus, "tradition" and "ancient" being top features for **Indian_identity** indicates that this model is accurately generating an understanding of what discriminates certain codes from others, at least from the persepctive of the researcher. Similarly, **communalism** is a central issue for Indira Gandhi's tenure, one that she stringently ties to articulations of religion, region/place, and caste. Additionally, she speaks of communalism almost entirely within the context of seeking solutions to such a problem, all of which are indicated by XGBoost's top features for that code. The top features for **democracy** in XGBoost cohere with conventional operational and infrastructural implementations of democracy.

At this point, I had to generate some sort of decision on these tradeoffs between my two models:

on the one hand, the logistic model was wildly successful in most of its performance metrics, even for recall and precision, but my XGBoost model indicated that it more purposefully generated applicable interpretations of what elements were contributing to the assignment of codes, even if it was not very successful at correctly assigning such codes. As such, I decided it would be important to assess the cases in which logistic regression was correctly classifying a document while XGBoost failed to do so in order to examine if there are systematic issues with the errors XGBoost was making.

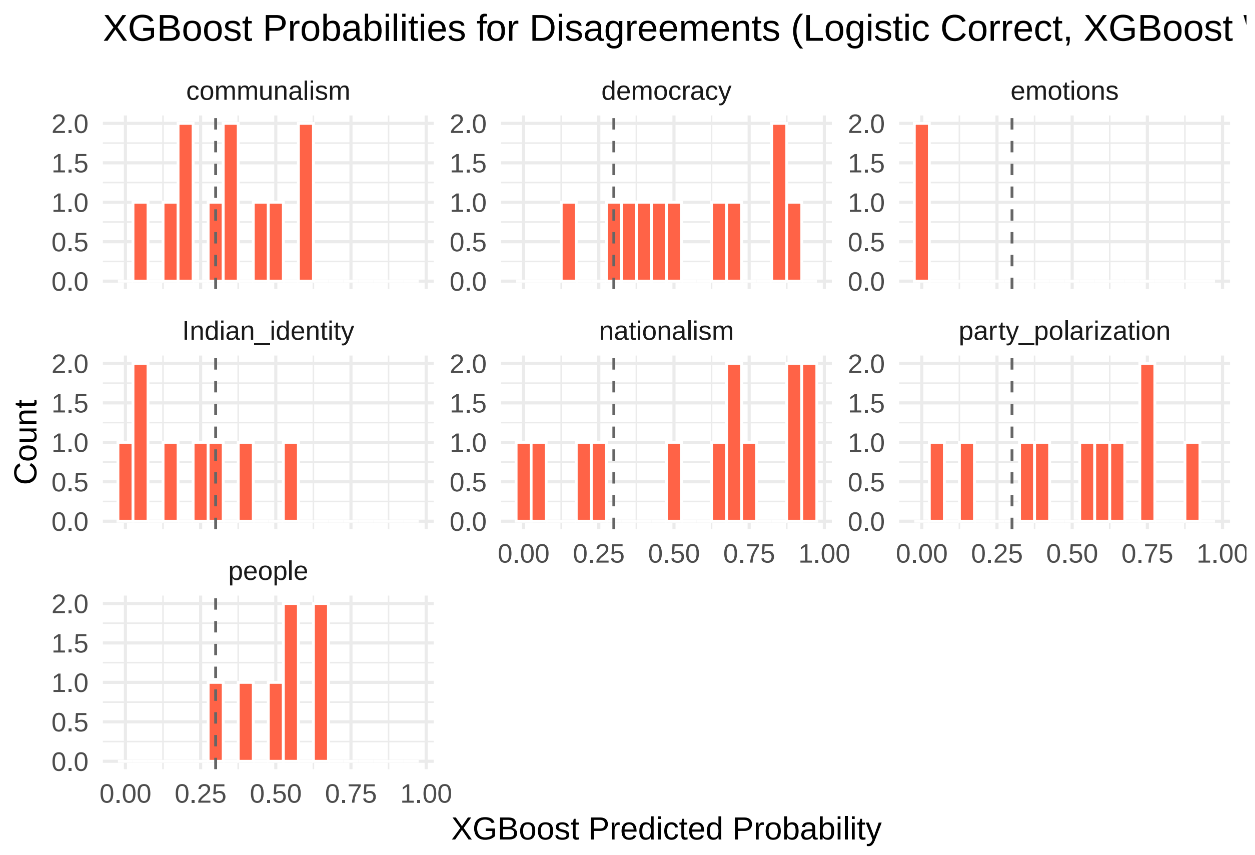
The first avenue I pursued in trying to examine this was by examining the instances in which XGBoost got the label incorrectly while logistic regression was correct. I examined a random selection of examples across the codes.

```
## ----- QUOTE -----
## As the President has rightly said in his Address, it is not the Government alone which has faced all
##
##
## All True Codes:  democracy, party polarization
## Target Label:  Indian_identity
## Logistic Prediction:  0  (Prob: 0.096 )
## XGBoost Prediction:  1  (Prob: 0.324 )
##
## ----- LOGISTIC CONTRIBUTIONS -----
##             term     tfidf        coef contribution
## 1        becomes 0.1154259 -0.72534368   -0.3171255
## 2       category 0.1154259 -0.26437010   -0.3171255
## 3        persist 0.1154259 -0.29507317   -0.3171255
## 4        removed 0.1154259 -0.79442281   -0.3171255
## 5        adopted 0.1079643 -0.06135861   -0.3171255
## 6      challenge 0.1079643 -0.22256394   -0.3171255
## 7  deliberately 0.1079643  0.65096078   -0.3171255
## 8         answer 0.1022006 -0.69105690   -0.3171255
## 9         assert 0.1079643 -0.47947660   -0.3171255
## 10      contrary 0.1079643  0.99676583   -0.3171255
##
## Top Logistic Terms Present:
## Top XGBoost Terms Present: people, india
```

```
## ----- QUOTE -----
## But we must never forget that there is no substitute for hard and determined effort and sacrifice on
##
## All True Codes:  Indian identity
## Target Label:  nationalism
## Logistic Prediction:  0  (Prob: 0.293 )
## XGBoost Prediction:  1  (Prob: 0.401 )
##
## ----- LOGISTIC CONTRIBUTIONS -----
##           term     tfidf        coef contribution
## 1      respect 0.2308518 -0.60709665   -0.1894082
## 2   determined 0.2044012 -0.99931033   -0.1894082
## 3         hard 0.2044012 -0.74064704   -0.1894082
## 4        never 0.2044012 -0.06718337   -0.1894082
## 5        sense 0.1871236 -0.47162758   -0.1894082
## 6      willing 0.1871236  0.53018786   -0.1894082
## 7      forget 0.1871236  0.41672290   -0.1894082
## 8   sacrifice 0.1803100  0.89976974   -0.1894082
## 9      effort 0.3606200 -0.06552980   -0.1894082
## 10    nations 0.1743244  0.43618324   -0.1894082
##
## Top Logistic Terms Present:
## Top XGBoost Terms Present: people
```

```
## ----- QUOTE -----
## They do·. It may disappoint Mr. Masani and Mr. Masani has been saying this practically from the firs
##
## All True Codes:  Indian identity
## Target Label:  party_polarization
## Logistic Prediction:  0  (Prob: 0.204 )
## XGBoost Prediction:  1  (Prob: 0.328 )
##
## ----- LOGISTIC CONTRIBUTIONS -----
##         term     tfidf         coef contribution
## 1  attitudes 0.2436769 -0.127316402   -0.2971173
## 2  emotional 0.2436769 -0.275534598   -0.2971173
## 3   although 0.2279246 -0.120750122   -0.2971173
## 4    created 0.2279246 -0.146753640   -0.2971173
## 5       hand 0.2436769 -0.140142794   -0.2971173
## 6    happens 0.2279246 -0.154649571   -0.2971173
## 7  difficult 0.1975193  0.009679725   -0.2971173
## 8     issues 0.1975193 -0.243039637   -0.2971173
## 9      local 0.1975193 -0.241033697   -0.2971173
## 10    become 0.1840091 -0.213707569   -0.2971173
##
## Top Logistic Terms Present:
## Top XGBoost Terms Present: members, world
```

The logistic regression model appears to have correctly classified many of the quotes by leveraging a broader set of moderately weighted terms which contributed to the final correct decision, but none of which appear in the top 10 terms for predicting the relevant code. XGBoost, in contrast, often has important terms for the code in the document but incorrectly classifies. This could be in part because the presence of such features becomes overpowering in determining the code for the model whereas logistic regression is somehow able to pick up on less deterministic features.



Additionally, the sample of quotes highlights that many of the XGBoost predictions generate probabilities that are close to or around the 0.3 cutoff. As such, the above graphs depict the distribution of probabilities predicted by XGBoost relative to the cutoff. These graphs indicate that for some of these codes in which XGBoost failed but logistic succeeded– namely **Indian_identity**, **communalism**, **democracy**, and **people**– many of XGBoost's predictions for the documents fall close to the 0.3 threshold cut-off. Selecting the correct cut-off is a hard balance to strike, as 0.3 was already a major improvement from the standard 0.5, but there is perhaps further tuning that can be done to make the model even more successful.

Given these illuminations from the above examinations of the two models, it is worthwhile to see how both generalize to the test set.

```
##
## ==============================
## Performance for: COLONIALISM
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic        NA      0    0.869
## 2 XGBoost         NA      0    0.869
##
## ==============================
## Performance for: COMMUNALISM
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic     0.667  0.182    0.836
## 2 XGBoost     NA      0        0.820
##
## ==============================
## Performance for: DEMOCRACY
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic       0.7    0.5    0.836
## 2 XGBoost       NA      0      0.770
##
## ==============================
## Performance for: EMOTIONS
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic        NA      0    0.902
## 2 XGBoost         NA      0    0.902
##
## ==============================
## Performance for: INDIAN_IDENTITY
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic      0.25  0.167    0.869
## 2 XGBoost      NA     0        0.902
##
## ==============================
## Performance for: NATIONALISM
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic     0.522  0.667    0.721
## 2 XGBoost     NA      0        0.705
##
## ==============================
## Performance for: PARTY_POLARIZATION
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic     0.529  0.562    0.754
## 2 XGBoost     NA      0        0.738
##
## ==============================
## Performance for: PEOPLE
## # A tibble: 2 x 4
##   Model    Precision Recall Accuracy
##   <chr>        <dbl>  <dbl>    <dbl>
## 1 Logistic        NA      0    0.951
## 2 XGBoost         NA      0    0.951
```

Despite cross-validated performance on the training set, the XGBoost model somewhat entirely failed to generalize effectively to the test set, as evidenced by the essentially zero performances in precision, recall, and accuracy across all codes. This discrepancy likely stems from a combination of factors: the small size of the overall dataset, substantial class imbalance (with some codes having very few positive examples), and the relatively high variance in document structure and language across quotes. In comparison, the logistic regression model generalized reasonably well on some codes, but also failed in others. Even with upsampling and threshold tuning, the XGBoost model may have overfit to the patterns in the training set, capturing noise or overly specific features that did not translate to the test set. Additionally, the quotes were hand-selected and semantically rich, so the linguistic patterns the models learned to rely on may not hold consistently outside of the training distribution, limiting generalizability. Realistically, despite being what I put forward, the XGBoost model is definitely not in a position to be presented in a formal applied setting, but perhaps there are further means of tuning and additional data collection that can allow for this model to be servicable.

Conclusion

Very broadly, this project sought to generate a natural language processing model using machine learning algorithms that can classify historical dialectical data (i.e. transcripts of speeches, correspondences, etc.) into categories (or "codes" as it is often referred to in the social sciences) for further analysis. In the specific context of my senior sociology thesis, the model I generated is trained to take segments of speeches from Prime Minister of India, Indira Gandhi, and classify them into a range of codes relating to nationalism, Indian identity, and specific political issues that were relevant to her tenure. While this model was curated for my particular needs as a student whose interests lie in historical sociology, discourse analysis, and studies of nationalism, a model such as this one is useful in extracting time-saving techniques for other social science researchers who are interested in language-driven and discourse-driven data. Manual coding– while necessary– is intensely time consuming and many researchers could benefit from having resources redirected. Building a robust model for organizing the bulk of the data instead of being put in a position where all of it must be done by hand could be a valuable addition to conducting social science research.

Given more time, there are numerous ways the infantile groundwork I've laid out with this XGBoost model could be vastly improved. As already mentioned, a larger dataset with more documents (and a wider range of documents– not just "important" ones) would surely vastly improve the performance of the model and likely aid in better generalizability and discriminatory abilities between the different codes. I am confident that additional tuning measures for the XGBoost model could've produced better recall results, as well, an avenue I would've liked to explore more if I had additional time. If there was also more time, continuing more systematically to examine disagreements and misclassifications to refine feature engineering and data labeling would be another worthwhile endeavor to pursue. It also would've been interesting to pursue an examination of the cross-over between codes and documents, as many documents had multiple codes assigned to it– perhaps, the misclassifications of some of the models was, in part, due to the fact that some codes share some sort of relationship with one another.

Even though my model has largely failed in application to the test set, the results of the training and curation of the model has provided meaningful insights into how machine learning algorithms can pick up on semiotics in the context of nuanced and complex dialectic historical data!