Mark Febrizio

DATS 6103 – Summer 2022

June 24, 2022

Final Report

Classifying Federal Register Documents by Type

I. Introduction

The Federal Register is the daily journal of the U.S. government, with a new issue published each business day. Each issue is divided into four sections containing four corresponding document types: Notices, Proposed Rules, Rules, and Presidential Documents. However, some of the data from the 1990s are missing document type labels. When researchers conduct analysis of agency actions, this produces a severe underestimation of the frequency of document types and the content related to specific topic areas. As a solution, I proposed to use the labeled documents to build classifier for document type. After training and testing this classifier on labeled data using supervised learning models, the classifier could be applied to uncategorized data for predicting the correct labels for uncategorized documents.

This report summarizes the process of building the classifier and reports the results of the machine learning models employed. The report is organized as follows: Section II discusses the proposal; Section III describes the dataset and exploratory data analysis; Section IV explains the modeling approaches taken; Section V reports the results of the models; Section VI discusses these results and concludes.

II. Proposal

Generally, Federal Register documents are automatically categorized into the correct type. This is helpful for searching the Federal Register for documents of a particular nature or relevance. Further, advanced users can call the Federal Register REST API to retrieve large quantities of metadata on documents for additional analysis. Given that federal agencies use the Federal Register to officially announce the development and finalization of regulations (also called rules or rulemakings), as well as solicit public input on those regulations, these data are valuable when conducting research on the regulatory process, agency performance, and specific policy areas (e.g., environment, immigration, energy, housing).

One major challenge to research on agency rulemaking is that documents, particularly from the 1990s, remain uncategorized. For some years, this leads to a severe under-counting of the number of rules published. To address this problem, I built a classification model that categorizes Federal Register documents by their type. Although the problem stems from a set of uncategorized documents, a majority of documents are labeled by type, making it possible to train and test a supervised learning model on a target with labels. After achieving a sufficient F1-score, the model can be applied to the uncategorized data to help mitigate the problem of missing document types.

III. Dataset

I will use the Federal Register REST API as my data source. Containing hundreds of thousands of documents (if not millions), the database is sufficiently large to train machine learning algorithms. I plan on restricting the data collected to the 1990s (specifically, 1994-1999) to focus on documents most similar to those with missing labels and to limit the size of dataset to a reasonable amount. The Federal Register API only goes back to 1994, even though the journal was published for several decades prior (see, Govinfo for PDF copies dating back to 1936).

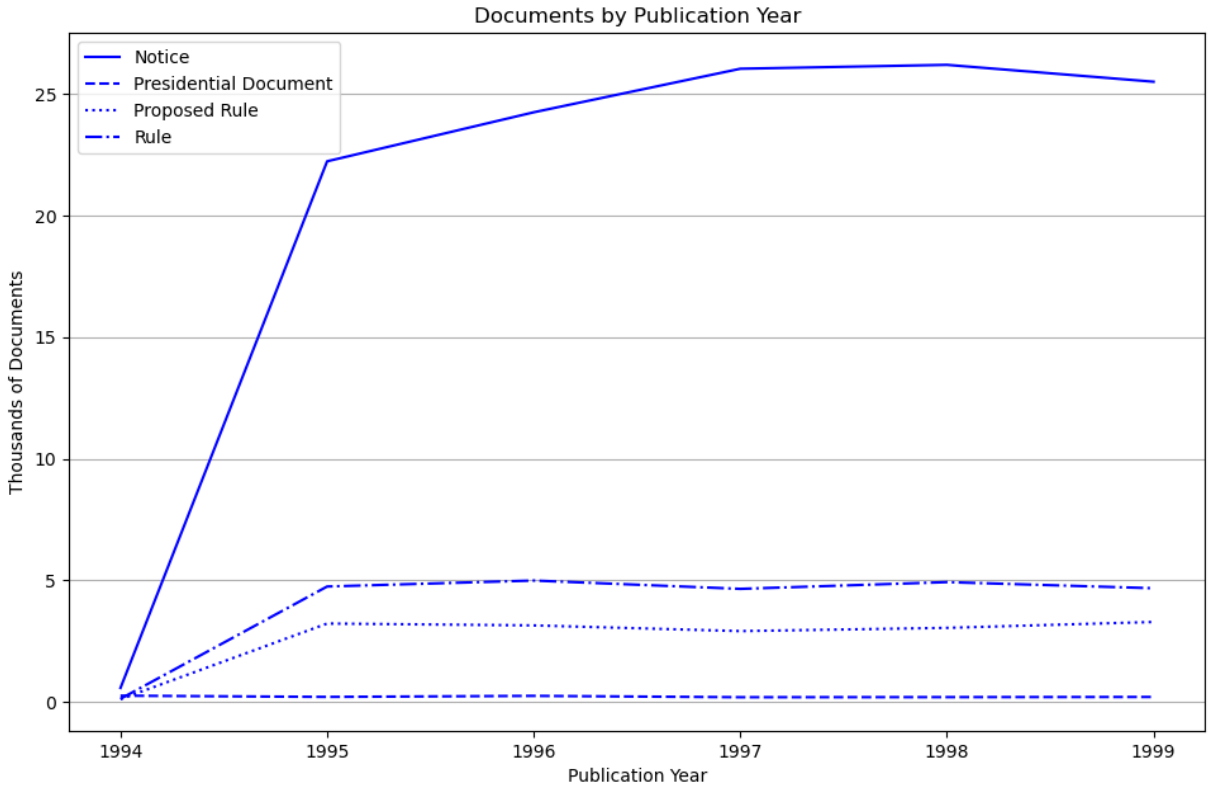
In the available data from the 1990s (1994-1999), more than 80% of 201,591 published documents are categorized – specifically, 166,031 are categorized as one of the four main types and 32,468 are uncategorized. Table 1 describes the number of documents by type, sorted by most to least frequent. Some documents are categorized as a "Correction" or "Sunshine Document," types that are no longer applied to new Federal Register documents. For developing the classifier, I selected only the 166,031 documents that fell into one of the four main categories – Notices, Rules, Proposed Rules, and Presidential Documents.

Table 1: Federal Register Documents by Type, 1994-1999

Document Type	Number of Instances
Notice	124,900
Uncategorized	32,468
Rule	24,114
Proposed Rule	15,721
Correction	2,018
Presidential Document	1,296
Sunshine Document	1,074
Total	**201,591**

Figure 1 shows the annual trend of categorized documents by their type. Notices make up the vast majority of documents, while several thousand proposed and final rules are published most years. Relatively few are presidential documents, making up only 1,296 observations in total. Notably, the 1994 data are primarily made up of uncategorized documents.

Figure 1: Federal Register Documents by Publication Year, 1994-1999



Source: Federal Register API and authors' calculations.

A full description of the document metadata available from the Federal Register can be found in the API's interactive documentation.[1] In addition to unique identifiers like *document_number* and important characteristics like *type*, the metadata includes fields, such as *page_length*, *title*, and the document's *abstract* (i.e., summary). Much of the metadata is in string format, meaning that meaningful features must be extracted from these fields. Table 2 describes the features that were extracted from the metadata.

Table 2: Features Extracted from Federal Register API

Feature Name	Description	Class/Type	Extracted from Fields
page_length	Page length of document	numeric	page_length
agencies_count_uq	Number of unique agencies issuing the document	numeric	agencies
abstract_length	Length of the document's summary/abstract	numeric	abstract

[1] https://www.federalregister.gov/developers/documentation/api/v1

Feature Name	Description	Class/Type	Extracted from Fields
page_views_count	Number of online views of the document	numeric	page_views
RIN_count	Count of "Regulation Identifier Numbers" assigned to the document	numeric	regulation_id_numbers
CFR_ref_count	Number of parts of the Code of Federal Regulations the document references	numeric	cfr_references
sig	Is the document "significant" under Executive Order 12866	categorical	significant
effective_date_exists	Did the document have an effective date?	categorical	effective_on
comments_close_exists	Did the document have a date on which public comments closed?	categorical	comments_close_on
docket_exists	Was the document associated with a Regulations.gov docket?	categorical	regulations_dot_gov_info
eop	Did the Executive Office of the President issue the document?	categorical	agencies
action	Description of the action the document is taking	text	action
abstract	Abstract of the document	text	abstract
title	Title of the document	text	title

I also calculated summary statistics for the features to be used in modeling. Table 3 shows several data points for the numeric features and the binary categorical variables (0 or 1). The mean for binary categorical variables can be interpreted as the percent of instances with value 1

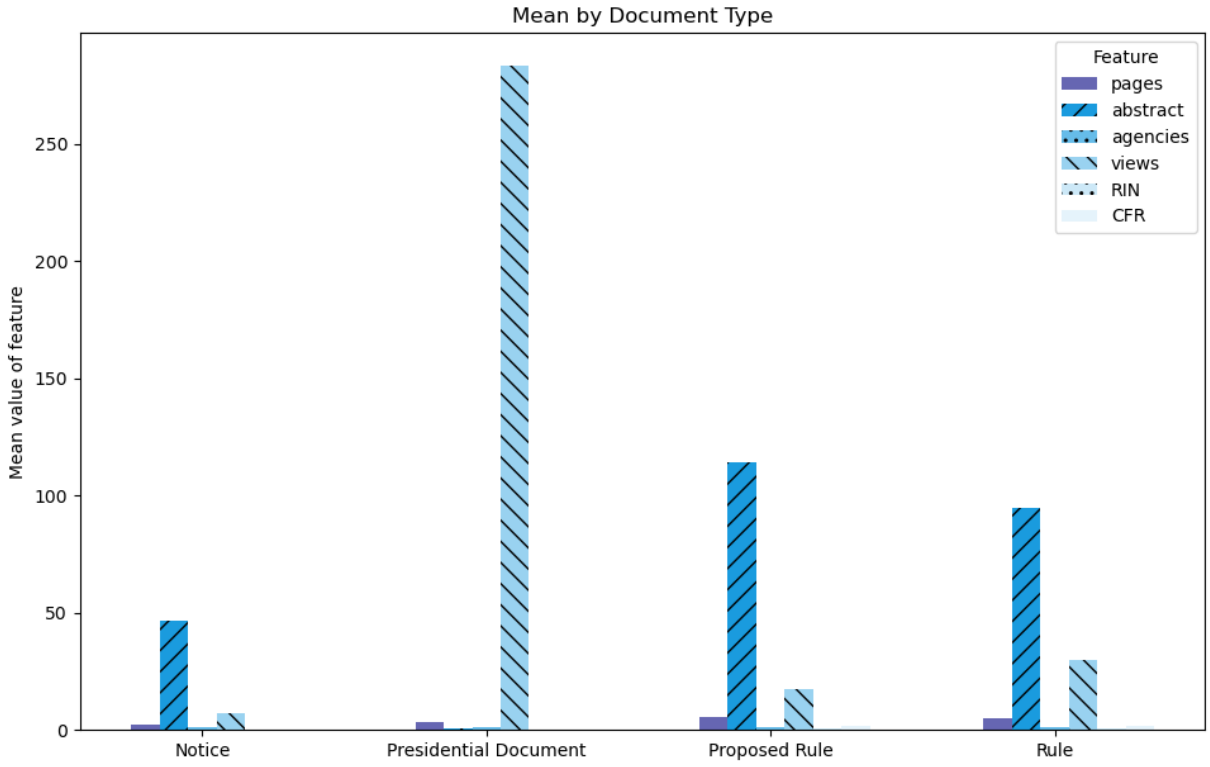
in the dataset – in other words, 20.5% of documents had an effective date, for example. These statistics were calculated after imputing missing values.

Table 3: Summary Statistics of Features

Feature Name	No. Obs	mean	stdev	min	median	max
page_length	166031	2.824	9.955	1	2	2702
agencies_count_uq	166031	1.027	0.262	0	1	31
abstract_length	166031	59.645	89.146	0	38	7757
page_views_count	166031	13.242	628.591	0	2	222295
RIN_count	166031	0.106	0.315	0	0	17
CFR_ref_count	166031	0.339	0.934	0	0	42
sig	166031	n/a	n/a	n/a	n/a	n/a
effective_date_exists	166031	0.208	0.406	0	0	1
comments_close_exists	166031	0.205	0.403	0	0	1
docket_exists	166031	0.001	0.035	0	0	1
eop	166031	0.008	0.089	0	0	1
action	166031	n/a	n/a	n/a	n/a	n/a
abstract	166031	n/a	n/a	n/a	n/a	n/a
title	166031	n/a	n/a	n/a	n/a	n/a

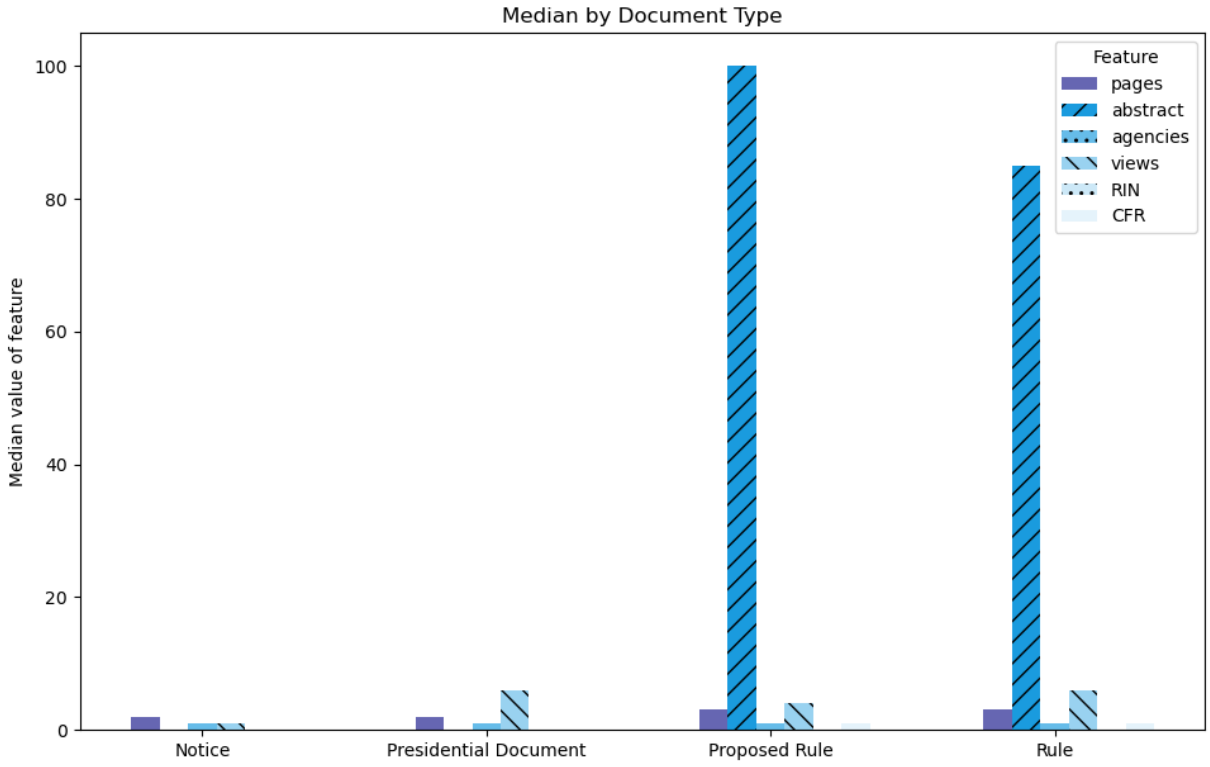
The numeric variables differed depending on the document type. Figure 2 depicts the mean values for the six numeric features in the dataset, and Figure 3 depicts the median values for the same variables.

Figure 2: Mean of Numeric Features by Document Type



Source: Federal Register API and authors' calculations.

Figure 3: Median of Numeric Features by Document Type



Source: Federal Register API and authors' calculations.

IV. Modeling

I used Python and its associated libraries, including sklearn, pandas, and numpy, to implement the machine learning algorithms. More detail on the environment and code I used is contained in the project Github repository.[2] I primarily relied on the Sci-kit Learn User Guide and lecture notes to obtain the necessary background to apply these supervised learning models.[3]

IV.A. Numeric and Categorical Features

I used several types of supervised learning models to build the classifier. First, I trained three models using the categorical and numerical features. As preprocessing steps, I transformed the numeric variables using a min-max scaler, the categorical variables with a one-hot encoder, and the target categories with a label encoder before passing the data to the classifiers.

[2] https://github.com/mfebrizio/data-mining-project.git

[3] scikit-learn 1.1.1, User Guide, https://scikit-learn.org/stable/user_guide.html, revised May 2022; also see, Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

I began with a Naïve Bayes (NB) model because such models are used for document classification in other contexts.[4] Specifically, I used the Complement NB model, rather than the Multinomial NB model, because it is more appropriate for imbalanced data sets.[5] Given that the majority of documents were of one type (Notice), mitigating this imbalance was important.

Still working with the numeric and categorical features, I then used two ensemble methods to build on this initial model. I use the boosting algorithm, AdaBoost, to fit a sequence of weak learners that weight the sampled data so that future models having a higher likelihood of seeing misclassified observations.[6] This approach leveraged 1000 iterations of a Complement NB model. The final classifications were made via a weighted majority vote.

Next, I used a voting classifier algorithm to combine several different types of models and then take a majority vote of the predicted labels.[7] Here, the Complement NB model was combined with logistic regression (logit) and K-Nearest Neighbors (KNN) algorithms. The logit model used balanced class weights, and the KNN model relied on a grid search for the number of neighbors parameter, where "k" equaled 5, 99, or 341.

IV.B. Text Features

Next, I trained one model using one of the text features to classify the documents. Here, I tried a simple Complement NB model again. The text-based feature was extracted using a term frequency times inverse document frequency (tf-idf) vectorizer on the *action* column. This approach permitted the model to account for both the frequency of words appearing in each document as well as in the entire corpus.

V. Experimental Setup

V.A. Code Sequence

To set up the supervised learning models, I implemented the following steps with Python code. All codes are contained in the Github repository. The README file in the repository contains more information on the sequence the codes. Executing the file "main.py" runs the code in sequence.

First, I queried the Federal Register API through the documents endpoint, saving the retrieved data as JSON files. Then, I processed the JSON files using the pandas and numpy packages. This included examining and filtering the data based on the target labels (*type*), cleaning the data, extracting variables from the document metadata, cleaning up the text-based columns, and imputing values for documents that were missing these values. For example, a subset of documents were missing values in the *action* column. For documents that were issued by the Executive Office of the President, I imputed the string "presidential document," and for documents not issued by that office, I imputed the value of the *title* column. This is an

[4] https://scikit-learn.org/stable/modules/naive_bayes.html
[5] https://scikit-learn.org/stable/modules/naive_bayes.html#complement-naive-bayes
[6] https://scikit-learn.org/stable/modules/ensemble.html#adaboost
[7] https://scikit-learn.org/stable/modules/ensemble.html#voting-classifier

appropriate approach because Presidential Documents do not have an *action* field, and for other document types, the title often contains information relevant to the action the agency is taking.

Following preprocessing, I conducted exploratory data analysis of the processed dataset. Then, I developed and ran each set of models. The models were designed to run independently of one another with their results being compared using relevant metrics. I used a train-test split of 70% to 30% for each model. This produced a training set of 116,221 documents vs. a testing set of 49,810 documents.

V.B. Model Evaluation

I used the accuracy score and the F1-score as metrics for evaluating the models. Because the dataset is imbalanced in terms of document types (i.e., the majority of documents are Notices), the F1-score is a better metric to use because it is the harmonic mean of the precision and recall of the model. I also produced and examined a confusion matrix for each model. These matrices helped me get a sense which document types the model is relatively better or worse at classifying.

VI. Results

The target labels have the following encoded classes: (0, Notice), (1, Presidential Document), (2, Proposed Rule), (3, Rule)

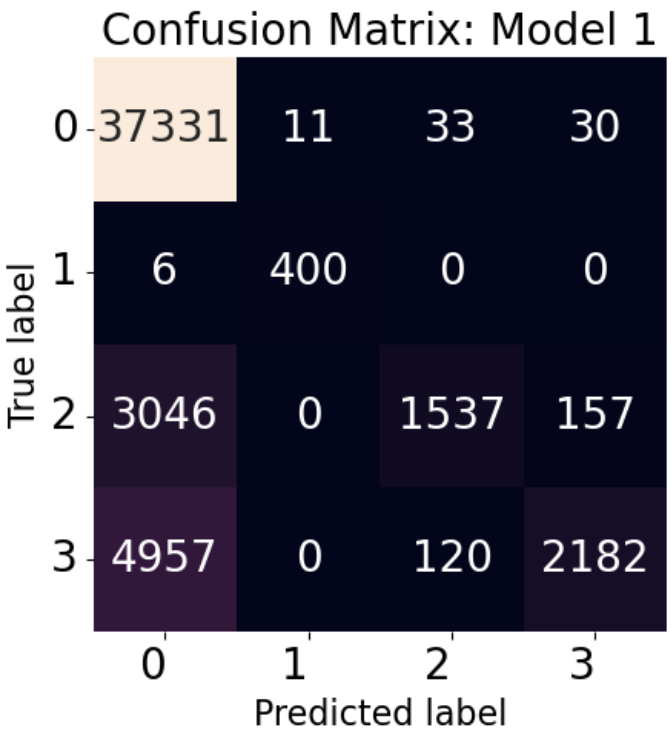
VI.A. Model 1

The first model run was a single Complement NB model, which proved to be a good baseline for interpreting the subsequent models.

The confusion matrix for model 1 demonstrates that the classifier did very well at classifying Notices (0) and Presidential Documents (1), but it struggled at correctly predicting the correct

labels for proposed (2) and final rules (3). Although the model had a relatively good precision for each target label, the recall for proposed and final rules was poor (0.32 and 0.30). In other words, model 1 did fairly well at avoiding false positives but did poorly at finding all the true samples for proposed and final rules.

Model 1 had an overall accuracy rate of 0.832 and a weighted F1 score of 0.797.

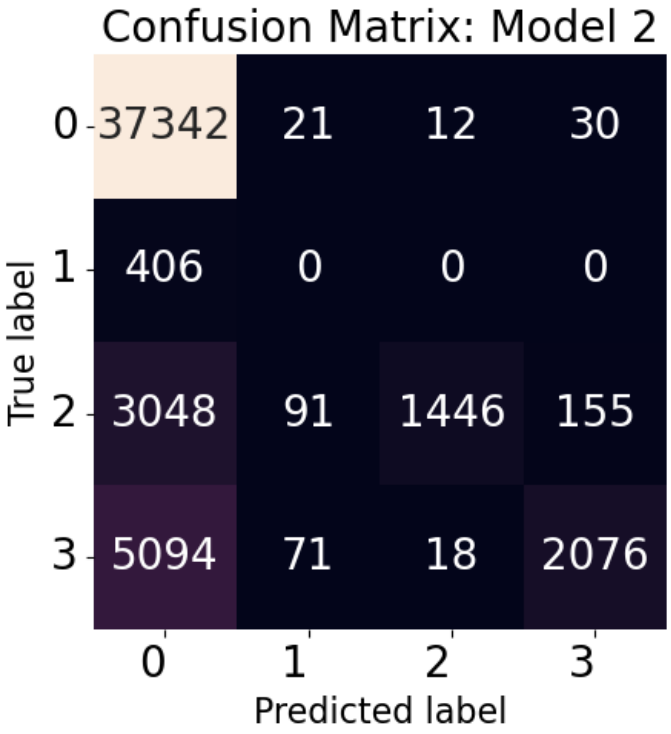


VI.B. Model 2

Model 2, which leveraged 1000 Complement NB estimators in a boosting algorithm, surprisingly performed worse than model 1.

The confusion matrix for model 2 shows that the model did slightly worse at identifying proposed and final rules (with more false negatives overall), and it misclassified every presidential document. Because of the large majority of notices in the sample, it's possible that the 1000 estimators overfit the model to identify notices at the expense of the less common document types.

Model 2 had an overall accuracy rate of 0.820 and a weighted F1 score of 0.781.



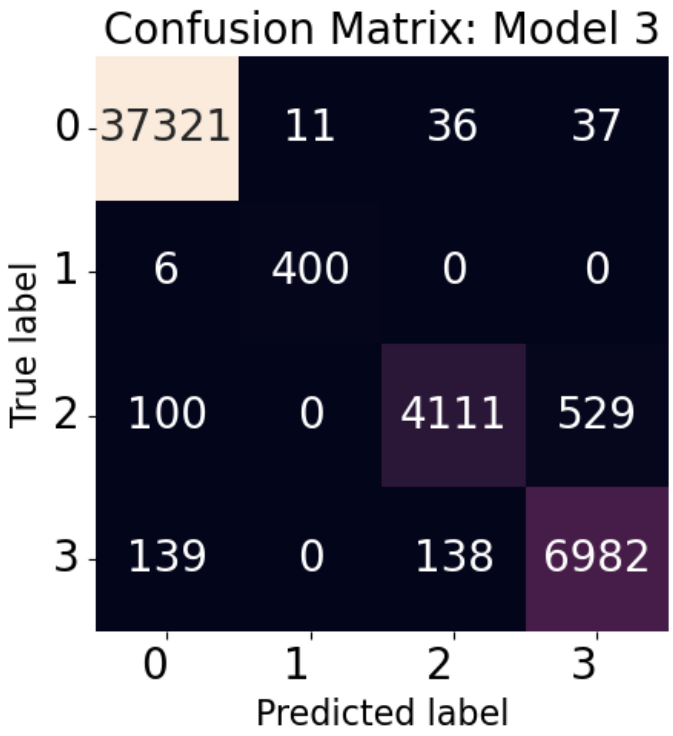
VI.C. Model 3

Model 3 used a voting classifier to identify document types, relying on hard voting across three different types of algorithms (Complement NB, logistic regression, and KNN). This model performed best out of all the models, although it also took the longest time to classify the documents.

This time, the precision for each document type was 0.93 or greater. The recall for notices, presidential documents, and final rules was above 0.96. Even proposed rules, which had the lowest recall at 0.87, performed well.

The voting classifier seemed to avoid the issue of overfitting notices, and it did well at classifying presidential documents again. This model was also the first one to classify the majority of proposed and final rules accurately.

Model 3 had an overall accuracy rate of 0.980 and a weighted F1 score of 0.9798.

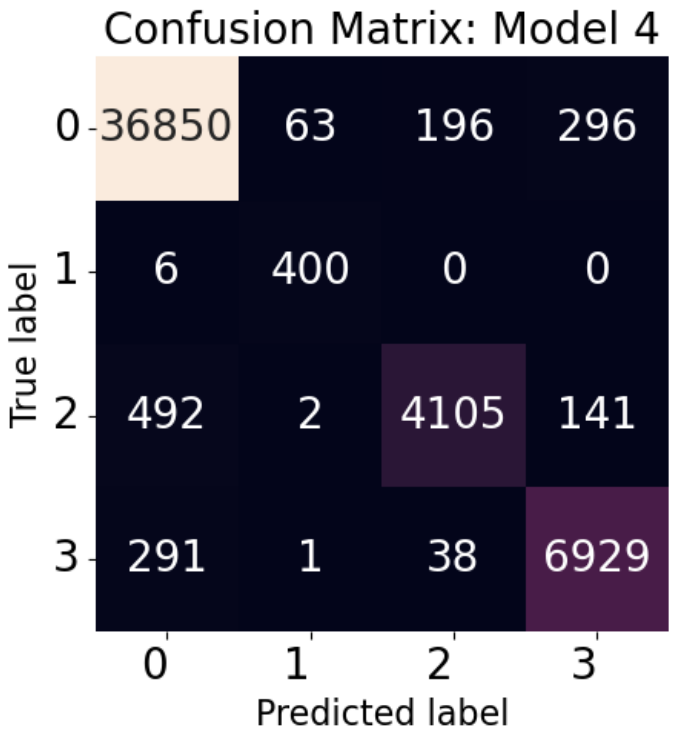


VI.D. Model 4

Model 4 used a simple Complement NB model with a text-based feature – the tf-idf vectorization of the *action* column. This model performed nearly as well as model 3, while also producing the results much more quickly.

Model 4 generally performed slightly worse than model 3, although it had a better precision for final rules. It made fewer false positives of final rules, while having more false positives for notices, presidential documents, and proposed rules.

Model 3 had an overall accuracy rate of 0.9694 and a weighted F1 score of 0.9691.



Discussion and Conclusions

The results indicate that the performance of the models progressively improved from the earlier to the later iterations. Table 4 summarizes the model results from the four models.

Table 4: Summary of Model Results

Model	Classifiers	Features	Accuracy	F1 Score
1	Complement NB	5 categorical 6 numeric	0.832	0.797
2	AdaBoost (1000 Complement NB estimators)	5 categorical 6 numeric	0.820	0.781
3	Hard voting classifier (Complement NB, Logit, KNN)	5 categorical 6 numeric	0.980	0.979
4	Complement NB	1 text (tf-idf)	0.969	0.969

As a whole, developing the classifier was a success. An accurate model can be produced, even when relying on a relatively small assortment of numeric and categorical variables representing features of the documents. Combining different types of machine learning models with a voting classifier produced a model with a 98% accuracy rate and a comparable F1-score. In addition, using text features offers a great possibility for highly accurate document classification. A simple model leveraging a single text-based feature compared favorably to a more sophisticated ensemble model of categorical and numeric features.

Further, integrating the text features with the categorical and numeric features could be an even more effective strategy for classification. One limitation of the *action* column is that it's very concise and has less variety of words than some of the other text-based fields. In the future, analyzing a larger amount of text, such as the full text of documents, with a tf-idf vectorizer could be a worthwhile approach. Another improvement that could be made would be considering different hyper-parameters for the boosting model. For instance, was 1000 estimators too many, leading to overfitting? Using grid search with cross-validation could help identify a better set of hyper-parameters.

In summary, the features available for Federal Register documents can be assembled in a manner that produces an accurate classifier of document types. This model can be applied to uncategorized documents to help estimate the frequency of the different document types across publication years.

References

Federal Register API, https://www.federalregister.gov/reader-aids/developer-resources/rest-api

Govinfo, https://www.govinfo.gov/app/collection/fr

Li, Susan, Multi-Class Text Classification Model Comparison and Selection, Towards Data Science, published Sep. 25, 2018.

Pedregosa et al., Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830, 2011.

scikit-learn 1.1.1, User Guide, revised May 2022.

scikit-learn 1.1.1, API Reference, revised May 2022.

Stackoverflow, Machine Learning Classification using categorical and text data as input, modified Dec. 16, 2020.