Related slide deck here.

Related iPython notebook here.

The Mercatus Center at George Mason University is a university-based economic research center that aims to fill the gap between academic ideas and real-world problems. Mercatus employs over 150 researchers, outreach professional, and support staff, who work together to provide economic research to the federal government and many state governments.

Mercatus has produced hundreds of documents that include full studies, research briefs, editorials, congressional testimonies, and more. These publications are posted publicly on the Mercatus website (www.Mercatus.org) where they are tagged to aid in organizing and presenting similar research. Until now, this tagging has been performed by junior employees who choose from existing tags or create their own, a system that has led to a proliferation of tags and a sometimes-inconsistent approach to document labeling.

The goal of this project is to resolve this problem by creating a predictive tagging system. Such a system should be able to analyze a submitted document and return a list of tags, sorted by predicted probability that the tag correctly describes the document.

This project has several stages:

- *Gathering the data* – The existing publications must be scraped from the Mercatus website, together with tags, authors, and other metadata.
- *Processing the data* – Tags that are obsolete or underutilized should be removed, and text should be cleaned up to aid in model training.

- *Training and testing models* – Machine learning algorithms from the scikit-learn Python library must be trained, tuned, and tested on the source documents.

The end goal of the project is to have a script that can be used to classify new research documents. A web interface is a potential add-on for future development; this project aims simply to produce a script that can classify a document in the working directory from a command-line interface.

The Data

Data for this project comes directly from the Mercatus website, collected with a webscraping script I wrote and ran from a Jupyter Notebook. This script takes a starting “tag page” on the Mercatus site and trawls through the hyperlinks within that page, adding any “tag page” and “publication page” it finds to associative arrays for both types. It then loops through these arrays indefinitely, changing the associative array value of each URL it looks at from 0 to 1.

For each tag page in the associative array, the script will gather all tag page and document page links on the page. For each publication page in the associative array, it will perform the same link-gathering function, as well as saving the publication’s text, together with metadata such as title, date, authors, and publication type, to a text file. The “data” directory in which these text files are stored is organized by subdirectories of tag names, so each document is saved in multiple locations, with one copy in each tag subdirectory that applies to that document.

The resulting data includes 142 tags, which serve as labels for our training documents. Many of these labels are redundant, so consolidation is necessary. Through manual processing, six labels are removed, as documented here.

Further reductions are made after documents and labels are imported to a pandas dataframe for actual model training. The first reduction is of documents themselves, namely documents whose character count falls under a threshold level. Such documents are often the result of PDF conversion errors and simply contain the URL of the document and no other trainable data. Then documents with NaN values are removed – these are exclusively documents whose dates did not save correctly. Further label-level reductions include the elimination of any label that has five or less documents, due to the difficulty of training on such labels, and the removal of labels that have not had a new document added in the past four years, to remove labels that are not currently relevant to Mercatus' mission and research activities. What remains is 90 labels spread over 1,873 unique documents.

The Models

Text features were vectorized using both count and term-frequency/inverse-document-frequency (tf-idf) vectorizations. Models fitted, tuned, and tested include Logistic Regression, Decision Trees, Random Forest, and Multinomial Naïve Bayes.

Hyperparameter tuning is handled by scikit-learn's grid search cross-validation function, with a range of relevant hyperparameters tested for each model. Negative log-loss is used as the

scoring metric, a good choice for multilabel, multiclass classification problems that penalizes models that incorrectly assign extreme probabilities.

First Passes

Model	*Parameters Tested*	*Best Parameters*	*Best Parameters' Score*
Logistic Regression	**C** ∈ [0.5, 1, 5] **dual** ∈ [False, True]	C = 5 dual = False (tf-idf vectorizer)	-3.30943
Decision Tree	**criterion** ∈ ['gini','entropy'] **min_samples_split** ∈ [2,4,6]	criterion = entropy min_samples_split = 6 (count vectorizer)	-30.50875
Random Forest	**criterion** ∈ ['gini','entropy'] **min_samples_split** ∈ [2,4,6]	criterion = entropy min_samples_split = 6 (count vectorizer)	-15.77671
Multinomial Naïve-Bayes	**alpha** ∈ [0.01, 1, 3]	alpha = 3	-4.02127

As can be seen from this table, tf-idf vectorized Logistic Regression with C = 5, dual = False is the winner with a negative log loss of -3.30943. All other models performed dismally in comparison, with the exception of Multinomial Naive-Bayes, which was at least able to return a negative log loss near -4.

Second Passes

Since our Logistic Regression model's optimal tested C value was also the maximum C value tested, it makes sense to test larger values of C while trying some other parameters. Similarly, we will also expand the alpha range on our Multinomial Naive-Bayes.

Model	*Parameters Tested*	*Best Parameters*	*Best Parameters' Score*
tf-idf Logistic Regression	**penalty** ∈ ['l1','l2'] **C** ∈ [1, 5, 10] **tol** ∈ [0.001, 0.0001, 0.00001]	penalty = l1 C = 5 tol = 0.00001	-2.86912
tf-idf Multinomial Naïve-Bayes	**alpha** ∈ [4, 5, 6, 7, 8, 9]	alpha = 8	-3.87078

These results show a significant improvement in both models, with tf-idf Logistic Regression coming out on top with a -2.286912 log-loss. Tf-idf Logistic Regression is the best model for our tag-selection engine.

Appendix

Classes

The following list is a complete collection of the 91 tags used in the classification problem:

Affordable Care Act	Financial Markets	Regulatory Impact Analysis
Balance of Trade	Financial Regulation	Regulatory Institutions and Processes
Bank Regulation	Financial Regulatory Agencies and Process	Regulatory Process Reform
Barriers to Entry	Financial Stability	Regulatory Report Card
Bitcoin	Food and Health Regulations	Retirement Security
Budget Process	Global Financial Markets	Rule of Law
Consumer Finance	Government Accountability	Securities Regulation
Copyright and Intellectual Property	Healthcare	Sharing Economy
Corporate Welfare	Healthcare Favoritism	Social Security Administration
Cybersecurity	Healthcare Innovation	Spectrum Reform
Debt and Deficits	Housing Finance	Spending and Budget Reform
Department of Energy	Immigration Reform	Spending, Deficits, & Debt
Department of Health and Human Services	Income Inequality	State and Local Budget Reform
Development Economics	Innovation	State and Local Policy
Disaster Recovery	Institutional Analysis	State and Local Regulations
Dodd-Frank	Internet Freedom	State and Local Tax Policy
Economic History	Jobs	State-by-State Comparisons
Economic Mobility	Medicaid	Study of American Capitalism
Economics and Public Policy	Medicare	Taxes

Economics of Culture	Midnight Regulations	Technology Policy
Education Policy	Monetary Policy	Telecommunications
Energy and Environmental Regulations	Monetary Rules	Telecommunications and Broadband
Entitlement Reform	NAFTA	Trade
Entitlements	Net Neutrality	Trade and Immigration
Entrepreneurship	Online Privacy	Transportation Financing
Export-Import Bank	Public Choice	Transportation Innovation
Federal Fiscal Policy	Public Sector Pensions	Transportation Policy
Federal Reserve	Puerto Rico	Transportation Regulations
Federalism	Regulation	Workforce Participation
FinTech	Regulatory Accumulation	
Financial Crisis	Regulatory Analysis	

As discussed in the *The Data* section, these 90 tags are a reduction from the original 142 tags gathered from the Mercatus website.