



The New Machine Politics: Identifying Partisans Through Their Browsing History

Citation

Deschler, John. 2019. The New Machine Politics: Identifying Partisans Through Their Browsing History. Bachelor's thesis, Harvard College.

Link

<https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364617>

Terms of use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material (LAA), as set forth at

<https://harvardwiki.atlassian.net/wiki/external/NGY5NDE4ZjgzNTc5NDQzMGIzZWZhMGFIOWI2M2EwYTg>

Accessibility

<https://accessibility.huit.harvard.edu/digital-accessibility-policy>

Share Your Story

The Harvard community has made this article openly available. Please share how this access benefits you. [Submit a story](#)

**The New Machine Politics:
Identifying Partisans Through Their Browsing History**

A thesis presented
by
John W. Deschler

Presented to
The Department of Computer Science
and
The Department of Government
in partial fulfillment of the requirements
for the degree with honors
of Bachelor of Arts.

Harvard College
March 2019

Acknowledgements

This project would not have been possible without the generosity of Professor Latanya Sweeney in sharing her dataset. Thanks go to her, and the two graduate students who helped me get set up with the massive dataset, Jinyan Zang and Ji Su Yoo. To my advisors, Professor Jim Waldo in Computer Science and Professor Stephen Ansolabehere in Government, thank you. It has been an honor and privilege to work with and learn from the two of you, and your guidance and advice was invaluable during this process. Thank you as well to Karen Kaletka and Beth Musser, who handled my many questions as a joint concentrator with smiles, laughter, and subtle digs at each other's department. To Maddie and Molly, thank you for the support, whenever and however much you could give. Thank you to Dunkin, the booths in CGIS, and that table in the back corner of Flour. Most of all, thank you to my parents, for the constant love and support, the place to escape, and the occasional home cooked meal.

Abstract

Political partisanship has been the driving force in American politics virtually since the birth of the nation; being able to detect partisanship has profound consequences for both academics and political professionals. I use a dataset of browsing histories from comScore in order to predict the partisan identification of a computer user. Using both random forest models and a variation on Sweeney's exclusivity indices, I match computer users to voter files in both North Carolina and Florida, demonstrating that both the race and partisan identification of an individual can be determined from his, her, or their browsing history.

Table of Contents

Chapter 1	1
American Elections	2
The Project	8
The Rule of Partisanship	11
The Driving Force of US Politics	14
The Old Machine Politics	18
The Data	22
Minimizing Potential Bias	26
Privacy Concerns	29
Chapter 2	37
Transforming the Data	40
Logistic Regression Models	46
Random Forest Classifiers	49
Race Modeling with Random Forests	50
Partisanship Modeling with Random Forests	56
Exclusivity Indices	65
Race Classification with Exclusivity Indices	68
Modeling Partisanship with Exclusivity Indices	72
The Partisan Internet	75
Chapter 3	78
The Florida Data	80
Modeling Partisanship in Florida	83
Random Forest Modeling	83
Exclusivity Index Modeling	84
Multi-State Modeling	86
North Carolina models, Florida users	87
Florida models, North Carolina users	88
A Combined Model	89
Stratified Florida	90
The Implications of Cross State Modeling	93

The Partisan Internet 2.0	94
Potential Analytic Improvements	94
Better Linking with comScore	96
More Data Collection with Browsing Histories.....	98
More States	98
The Modular Forest.....	100
Unbalanced Decision Tree	101
Conclusions.....	106
Appendix 1: comScore codebook.....	111
Appendix 2: Matrix Transformation Code	113
Appendix 3: Party Imputation Code.....	115
Appendix 4: Random Forest Modeling Code.....	118
Appendix 5: Exclusivity Index Model Code	125
Appendix 6: Cross-state Modeling Code	130
Bibliography	133
Datasets.....	137

List of Figures

Figure 1.1: via Nichter 2008	20
Figure 2.1: Number of domains vs. number of users.....	44
Figure 2.2: Number of domains per user vs. number of users.....	45
Figure 2.3: A Basic Decision Tree, $d=1$	49
Figure 2.4: Census Weights Confusion Matrix.....	51
Figure 2.5: Minority Oversample Confusion Matrix	52
Figure 2.6: Black-White Only Confusion Matrix	53
Figure 2.7: Confusion Matrix for Even Racial Sampling.....	54
Figure 2.8: White vs. Nonwhite Confusion Matrix	56
Figure 2.9: Race vs D_pct_2p	58
Figure 2.10: Age vs D_pct , all races (left) and white only (right).....	59
Figure 2.11: $D_pct > 0.8$	60
Figure 2.12: $D_pct > 0.5$	62
Figure 2.13: $D_pct_2p \geq 0.8$	63
Figure 2.14: $D_pct_2p \geq 0.5$	64
Figure 2.15: Early Exclusivity Index Model.....	69
Figure 2.16: Confusion matrices among classified users, varying t	69
Figure 2.17: Confusion Matrices for Modified Criterion	71
Figure 3.1: Race vs D_pct_2p for Florida	82
Figure 3.2: Florida D_pct_2p by age, all races (left) and white only (right).....	83
Figure 3.3: $D_pct_2p > 0.5$	84
Figure 3.4: $D_pct_2p > 0.5$, $t = 0.9$, Original Criterion.....	85
Figure 3.5: North Carolina models tested on Florida users. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion	88
Figure 3.6: Florida models tested on North Carolina users. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion	89
Figure 3.7: Combined Models. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion	89
Figure 3.8: Stratified Florida.....	90
Figure 3.9: Stratified Florida D_pct_2p , size represents count	91

List of Tables

Table 2.1: Sites by Influence	65
Table 2.2: Classification Details	70
Table 2.3: Classification Details, Modified Criterion	71
Table 2.4: Classification Accuracies for Partisan Identification	73
Table 2.5: Top Exclusivity Indices by Party	76
Table 3.1: Exclusivity Index Model Accuracies, Florida	85
Table 3.2: North Carolina models tested on stratified Florida. $D_{pct_2p} > 0.5$, $t = 0.9$	92
Table 3.3: The Partisan Internet.....	95

Chapter 1

Political parties are central to the function of the American democracy. They organize opinions on issues, and signal to the voting public which candidates are viable choices in elections; in almost every major national or regional American election, there are only two individuals with a chance of winning, each belonging to one of the two major parties. The vast majority of Americans, even those who would not explicitly identify themselves as partisans, fundamentally organize their thoughts and orient their issue opinions along party lines - there are very few true “independents” in American politics. As much as political life is organized through parties, the rest of modern life is organized through the Internet. Social life, shopping, coursework, and entertainment have all moved further and further online as the century has progressed. Along with the Internet has come social media, with Americans updating statuses, tweeting, and posting on their Instagram stories seemingly constantly. As life has moved online, Americans surrender more and more data about themselves to social media companies, creating new data sources for academics and professionals alike. This thesis establishes links between the two dominant facets of modern American life: strong political parties and the Internet. I do not aim to explain partisanship or political identification, but rather simply to predict it. Using the browsing histories of American citizens – simple lists of the websites that they visit – I build two different classes of models that predict both an individual’s race and partisan identification. Practically, the methods and models herein offer the beginnings of a roadmap for how academic researchers, political campaigns, and

other organizations can use the most basic information on Internet activity to predict partisan identification. Furthermore, the results presented in this thesis raise important questions regarding the privacy of individuals in the modern age, and whether or not such techniques can be used to violate the rights of American citizens.

American Elections

American elections have consequences that reverberate throughout the world and send ripples that are felt by history for decades that follow. Electoral battles are fought at massive financial scale, and have impacts both immediate and long lasting. The ways in which two sparring campaigns do battle are of paramount importance to this thesis. We start with a simple fact: all else equal, the addition of a vote to a candidate's total increases their likelihood of winning the election. This fact comes with an important corollary: all else equal, the subtraction of a vote from a candidate's total increases the likelihood of the candidate's opponent winning the election. These two facts are true regardless of the type of election being discussed, from a county level school board seat, to an election for the United States Congress, to a Presidential election within the confines of the Electoral College. A campaign thus has three choices for how to go about achieving victory: *demobilize* their opponent's voters, *persuade* undecided or opposition voters to join their cause, and *turnout* voters who would vote for their candidate, but may not otherwise actually vote. Of the three strategies, persuasion seems to be both the most difficult and least reliable. Research by Kalla and Broockman demonstrates that the resultant change of all

the attempts by campaigns to persuade voters, whether by advertising or direct contact, results in a net change of nothing.¹ Their evidence comes from both a meta-analysis of forty already-finished experiments, as well as the intentional design of new experiments. Though they acknowledge that there exist analyses in contradiction with their own, they also point out that those analyses focus on a time without the hyper-partisan nature of the modern electorate.² The ineffectiveness of persuasion leaves campaigns with two realistic options for maximizing their chances of victory: demobilization and turnout. Importantly, both demobilization of the opponent's electoral base and turnout of one's own electoral base rely on being able to correctly target different portions of the electorate.

The demobilization of the opponent's voters in order to shrink the opposition vote total is commonly attempted via the placement of negative advertisements, a method has been shown to effectively drive down opposition turnout. The results presented by Ansolabehere, et al. demonstrate that demobilization is likely to be a more effective strategy than persuasion.³ Indeed, the notion that demobilization is easier than persuasion seems rather intuitive: there is a lower bar to convincing someone to simply not vote rather than to actually *switch* their vote. That said, however, what seems to be the most effective

¹ Joshua L. Kalla, and David E Broockman. "The Minimal Persuasive Effects of Campaign Contact in General Elections: Evidence from 49 Field Experiments." 112, no. 1 (2018).

² Ibid.

³ Stephen Ansolabehere, et al. "Does Attack Advertising Demobilize the Electorate?" *The American Political Science Review* 88, no. 4 (1994).

of the three campaign strategies is to increase the turnout of a target portion of the electorate. Enos and Fowler showed that campaigns can have a large effect on turnout, up to 7-8 percentage points in states that campaigns explicitly target.⁴ With almost all American presidential elections decided in a tipping point state⁵ within that margin, effective campaigns can clearly tip the balance in a massively important election. However, since persuasion is highly *ineffective*, campaigns must turn to demobilization and turnout. For both the demobilization and turnout strategies to be effective, campaigns must be able to identify the pre-existing vote preference of individuals, and in the modern world, those preferences are tightly linked to an individual's partisanship.⁶ In order to properly execute demobilization, campaigns must identify their opponent's voters, and in order to execute the turnout strategy, they must find those with a predisposition to vote for their candidate.

In his book Hacking the Electorate, Eitan D. Hersh draws an important distinction, between voters and what he calls "perceived voters." According to Hersh, what matters the most in campaign targeting is not how a person actually votes, or has actually voted, but how a political campaign *thinks* that they will vote, with the campaign's opinions being informed by any data that they may

⁴ Ryan Enos and Anthony Fowler. "Aggregate Effects of Large-Scale Campaigns on Voter Turnout." *Political Science Research and Methods* 6, no. 4 (2018).

⁵ A tipping point state is the state that if the winning candidate had won, and also won all states he or she won with a higher margin, he or she would still win the election, but would have lost the election if he or she had lost.

⁶ Alan S. Gerber, Gregory A Huber, and Ebonya Washington. "Party Affiliation, Partisanship, and Political Beliefs: A Field Experiment." *American Political Science Review* 104, no. 4 (2010).

have on an individual.⁷ When campaigns attempt to use data to discern the characteristics of individual voters in an attempt to tailor advertising, or even campaign direct contact visits, they are engaging in microtargeting. Microtargeting is used in far more than just political campaigns in the modern day, and is perhaps best defined as “way to successfully create personalized messages or offers, correctly estimate of their impact (in regards to sub-grouping) and delivery directly to individuals.”⁸ Used by many companies in the corporate and online worlds, successful microtargeting can allow a campaign to properly apply its funds to reach the people most likely to be either mobilized or demobilized by its advertising. Models built to help campaigns microtarget voters already exist in the political sphere – companies like Catalist and NGP VAN have been creating them for Democratic candidates for multiple electoral cycles.⁹ In states where party registration is explicitly available in voter files, campaigns have a much easier time contacting the voters that they wish to activate or demobilize. In states without such information, microtargeting models attempt to shed light on the partisanship of individual voters, so that campaigns can get the correct message in front of the correct people. The better the model, the more effectively their strategies can be applied.¹⁰

⁷ Eitan D. Hersh. *Hacking the Electorate: How Campaigns Perceive Voters*. (New York, NY: Cambridge University Press, 2015), 87.

⁸ Oana Barbu. "Advertising, Microtargeting and Social Media." *Procedia - Social and Behavioral Sciences* 163, no. C (2014).

⁹ Hersh, *Hacking the Electorate*, 96.

¹⁰ *Ibid.*, 113

Though a significant amount of literature exists in the American political science canon surrounding the effects and consequences of partisanship, far less research has been done on the subject of predicting partisan affiliations from various data sources. Some research does exist on the causes of partisanship, or the cause of the modern increase in partisan divides, which is useful for creating a very rough approximation of where partisanship is likely to be strongest. Unfortunately, most of this research takes phenomena that theoretically might cause partisan divide, and finds those phenomena do *not* cause such rifts. Using their DW-Nominate scores, McCarty, Poole, and Rosenthal tested the hypothesis that the gerrymandering of Congressional districts causes increased levels of partisan polarization, with decidedly negative results.¹¹ Similarly, McGhee et. al. test the hypothesis that the nomination system in primary elections causes increases in the partisan divide. Applying the median voter theorem, their idea was that primary electorates have a median voter significantly further from the political center than the general electorate, causing primaries result in politicians, and thus policy and voters, to be dragged away from the center and further to the extremes. The authors, however, find that the degree of openness of the primary, essentially a proxy for the degree of difference between the primary and general electorate, has little effect on the extremism of the resulting politicians.¹²

¹¹ Nolan McCarty, Keith T. Poole, and Howard Rosenthal. "Does Gerrymandering Cause Polarization?" *American Journal of Political Science* 53, no. 3 (2009).

¹² Eric McGhee, et al. "A Primary Cause of Partisanship? Nomination Systems and Legislator Ideology." *American Journal of Political Science* 58, no. 2 (2014).

On the other hand, Seabrook finds that geographic clustering increased greatly as the American political system entered the Obama era in 2008, resulting in significantly more spatial correlation between party votes in the 2008 Presidential election than in 2004.¹³ Based on the results of the 2018 Congressional elections, that trend has continued, with Democrats winning in densely populated urban districts, and Republicans winning in less densely populated, rural districts.¹⁴ Seabrook's result, and the ensuing trend, indicate that a predictive model of partisanship should likely incorporate a geographic factor. Another well known factor that can help play a large role in determining the partisanship of an individual is their race. Indeed, Hersh devotes an entire chapter of Hacking the Electorate to predicting race in states where it is not explicitly listed, pointing out that it can be a very difficult value to predict.¹⁵ Add in the similar splits in partisanship between different income brackets, and even different genders, and one can likely reach a decent approximation of the partisanship of an individual by predicting demographic factors.

At the most basic level, this thesis is about using a data source, in this case browsing history, in order to both improve and then nuance that first approximation. Though individual level partisanship appears to never have been predicted in this manner before, a similar experiment was done using data from

¹³ Nicholas R. Seabrook. "The Obama Effect: Patterns of Geographic Clustering in the 2004 and 2008 Presidential Elections." *The Forum* 7, no. 2 (2009): The Forum, 2009, Vol.7(2).

¹⁴ "U.S. House Election Results 2018." *The New York Times*. January 28, 2019. <https://www.nytimes.com/interactive/2018/11/06/us/elections/results-house-elections.html>

¹⁵ Hersh, *Hacking the Electorate*, 123.

Twitter. Beauchamp took data from over 100 million political tweets, and used them to predict the results of state-level political polls during the 2012 Presidential election, with a fair amount of success. Beauchamp was even able to extend his analysis of the Twitter data to states that were not polled, and believed that it was possible to even further granulate the geographic level of study.¹⁶ Importantly, measuring public opinion, as Beauchamp does, is not the same as measuring individual level partisanship, but is a problem in the same vein; if the Twitter analysis could reach a granularity of one person per geographic unit, then it would essentially be measuring individual level partisanship.

In his literature review, Beauchamp points out that the act of predicting public opinion on broad levels is difficult when using social media data.¹⁷ With success at predicting even broad trends in political opinions difficult to come by, it is little wonder that individual level partisanship predictions are even harder to come by. Predictions based on browsing history can perhaps begin to fill that gap, and provide an effective way to microtarget critical voters for campaigns.

The Project

My goal is rather simple: predict partisanship. I do so using browsing history, a somewhat new source of data for the task, and one that, due to changes in American law, is becoming more and more available. I believe this endeavor has broad implications, for both academics and political professionals alike. Knowing the partisanship, which in most cases maps directly to and from the

¹⁶ Nicholas Beauchamp. "Predicting and Interpolating State-Level Polls Using Twitter Textual Data." *American Journal of Political Science* 61, no. 2 (2017).

¹⁷ Ibid.

party identification, of an individual conveys a great deal of information about who that individual is likely to vote for in an upcoming election. If political professionals gain an edge in predicting the partisanship of voters, particularly using a data source so closely connected to the Internet, then they could reasonably change the ease with which they reach voters – and not just voters, but the *right* voters. An analysis of browsing histories being linked to partisanship has benefits to the academic community as well. As already discussed, partisanship is the driving force behind virtually all of United States politics; a better understanding of partisanship equates quite directly to a better understanding of the American political system. Perhaps an analysis may reveal that the partisan divides that have rent American society in two extend even to online life. It is already known that Americans choose their news sources in accordance with their political views.¹⁸ The same could be true of the chain restaurants they get takeout from, where they read about sports, or even which social media sites they are using. This thesis begins to answer questions about the partisan fabric of the Internet itself.

An important question to ask about any analysis like the one herein is whether it actually reveals something new about partisanship, something that was not earlier measurable, or whether the differing parts of the Internet for each party are merely proxies for the divides that already exist in society. Does browsing history, as a data source, operationalize a new set of variables, or does it

¹⁸ James G. Webster and Thomas B. Ksiazek. "The Dynamics of Audience Fragmentation: Public Attention in an Age of Digital Media." *Journal of Communication* 62, no. 1 (2012): 39-56.

merely capture the same effects that have been observed by other predictive measures of partisanship? Whatever the answer, and future experiments will be needed to be certain, the implications of this analysis for both academia and political professionals remain. If browsing histories reveal some new piece of the partisanship puzzle, then the benefit of this analysis seems obvious, but even if browsing history just functions a proxy for some variable that the political and political science communities have already found a way to measure, the data source still has value, especially with recent changes in American law surrounding how citizens access the Internet.

In late 2017, the Federal Communications Commission voted to repeal a section of the 2015 Open Internet Order, which reclassified Internet service from a public utility to an information service.¹⁹ One of the major results of this decision was a decreased ability of the United States government to regulate Internet service providers (henceforth ISPs), and the fall of a principle known as “net neutrality.” The principle of net neutrality is most succinctly described as the idea that ISPs treat all customers equally, and neither discriminate against nor restrict access to certain parts of the Internet. Before it was repealed, the principle of net neutrality gave the government the ability to protect both Internet users and Internet content producers from discrimination at the hands of ISPs, and now, at least for the immediate future, that protection is gone. An ISP can see the browsing history of all its customers – even if a customer enters “private mode”

¹⁹ Cecilia Kang, “F.C.C. Repeals Net Neutrality Rules,” *The New York Times*. December 14, 2017. <https://www.nytimes.com/2017/12/14/technology/net-neutrality-repeal-vote.html>

on their Internet browser of choice, the ISP has to actually transfer the information packets to the correct places on the world wide web and back, allowing *all* connections to be tracked and logged. In a world without net neutrality, it is easy to imagine a politically active ISP differentially blocking access to sites frequented by people of certain political views, or slowing the load of politically important sites, perhaps voter registration information, to people who are likely members of one party or the other. To political professionals, having a direct channel to microtarget advertisements, as browsing history based partisanship predictions would likely provide, could drastically decrease the cost of some advertisement campaigns, and ensure that they are only activating the correct voters. To academics, having another data source that could verify or even slightly deepen the predictions of partisanship would undeniably be of value. Regardless of whether browsing histories offer an insight into a piece of partisanship that has yet to be observed and operationalized, or whether they function as proxies for already measurable effects, the modern political order begs for an exploration.

The Rule of Partisanship

Any discussion of identifying voter choice in American politics must start with a discussion of the role that partisanship plays, and has played since the late 18th century, in the American political system. When George Washington stepped down from the American Presidency in advance of the 1796 election, he warned

the nation of the rise of political parties within the government.²⁰ Washington was not the first of the founding fathers to warn against the formation of political parties in the nascent republic. James Madison argued in Federalist No. 10 that the system of government enumerated in the Constitution would be effective at safeguarding the Union against the rise of parties, referred to as “faction.”²¹ Indeed, Madison may have been correct that the original formation of the American government under the Constitution should have prevented the formation of political parties, or at least guarded against parties completely overtaking the system, for the Constitution itself does not specify how exactly representatives in the American republic are elected. The single member districts that are the hallmark of American political representation are not enshrined in the Constitution, but are currently written into law by the Apportionment Act of 1911. According to Duverger’s Law, any democracy with single member districts and plurality rule elections, such as first past the post elections, will result in the rise of a bipolar political party system.²² That Law has proved prophetic in America, as the modern American political landscape is utterly dominated by two parties:

²⁰ George Washington. *George Washington Papers, Series 2, Letterbooks -1799: Letterbook 24, April 3, 1793 - March 3, 1797*. 1793. Manuscript/Mixed Material. <https://www.loc.gov/item/mgw2.024/>.

²¹ James Madison, *The Federalist No. 10*, in *The Federalist Papers and the Constitution of the United States: The Principles of the American Government*. (New York, Skyhorse, 2016). 46-53.

²² Though Duverger’s Law has been revised (see Riker 1993) over time in accordance with several counterexamples, its simplest version has proven to hold true throughout American history; there has not been an extended period of a stable three-party system since the birth of the nation, and even so called “independents” in the modern Congress, like Senator Bernie Sanders (I-VT), are treated as if they are members of one of the two major parties.

the Democratic Party and the Republican Party. The two-party system is more or less a direct consequence of laws like the Apportionment Act of 1911, which dictate that “districts shall be equal to the number of Representatives to which such State may be entitled in Congress, no district electing more than one Representative.” The single member districts create the two-party system, and virtually everything in American politics depends on the two-party system. In the United States of America, partisanship rules politics.

There is no single idea more important to American politics than partisanship, and thus no single piece of information more valuable to, or about, any individual voter. Being able to predict an individual’s partisanship has immense power to any political candidate, or indeed any political actor. This thesis concerns itself with just that, using a source of data that is becoming more and more readily available: browsing histories. I claim that an individual’s browsing history conveys information about their partisan affiliation, and thus what party they are likely to cast their ballot for in an upcoming election. Using a dataset of over 80,000 browsing histories from different individuals, I predict both race and partisanship, building a training set for the latter by de-identification to state-level voter files. This chapter outlines the background, from an investigation into partisanship as an electoral force, to why browsing histories are becoming more and more available, to the privacy concerns the release of such data may carry. In Chapter 2, I describe the two classes of models I will use, and fit them using partisanship data from the state of North Carolina. Chapter 3 extends this

analysis to Florida, and undertakes cross-state modeling, as well as a discussion of the expansion of these techniques past the scope of this project.

The Driving Force of US Politics

Partisanship, often measured by party identification, is the starting point for almost any action in the political sphere. In the function and actions of the government, it matters little which individual candidates win elections, but it matters a great deal which party wins the majority of seats. Few could name every Senator, many fewer could name every Representative, but anyone who is even remotely plugged into American political happenings would be able to name the party controlling the votes in each chamber. One could argue that the President is the exception to this rule: individual Presidential candidates, even if members of the same party, certainly matter. That said, only the nominees of the two parties have even a prayer of winning the general Presidential election, and as Cohen, Noel, and Zaller argue in their book The Party Decides, it is by and large the parties themselves who decide who that nominee will be.²³ Even when the Republican party recently saw a non-establishment candidate, Donald Trump, win its nomination, the party itself quickly transformed and became a party of, by, and for Trump.

Partisanship does not just change the actions of the leaders of the political system, but even the opinions of rank-and-file American voters. Take economic conditions, which have long been considered one of the major macro-

²³ Marty Cohen, et al. *The Party Decides: Presidential Nominations Before and After Reform*. (Chicago: University of Chicago Press, 2008), 187.

determinants of American elections.²⁴ A good economy leads to a victory for the incumbent party, and a bad economy a victory for the challengers, goes the conventional wisdom. The next question to ask, of course, is what is a good economy, or perhaps more accurately, how do voters decide if the economy is good? The answer to that question, lies in partisan politics. The tint of a person's partisanship becomes the "lens" through which they see almost all political happenings, including economic conditions.²⁵ In short, partisanship is a key driver of economic perceptions, and thus is the force behind economic conditions impacting elections. Enns, Kellstedt, and McAvoy found this effect on a variety of indicators thought to drive voting behavior. Democrats and Republicans perceived the unemployment rate, inflation rate, and their general confidence in the economy differently depending in large part on which party had control of the White House.²⁶ The modern news cycle is rife with public opinion polls showing stark differences in the ways that Americans of different party identifications view the reality of the nation. The Pew Research Center found that Americans have highly negative views about the members of the other party.²⁷ Regularly

²⁴ Michael B. Mackuen, Robert S. Erikson, and James A. Stimson. "Peasants or Bankers? The American Electorate and the U.S. Economy." *The American Political Science Review* 86, no. 3 (1992).

²⁵ Peter K. Enns, Paul M. Kellstedt, and Gregory E. McAvoy. "The Consequences of Partisanship in Economic Perceptions." *Public Opinion Quarterly* 76, no. 2 (2012).

²⁶ Ibid.

²⁷ "Partisanship and Political Animosity in 2016." Pew Research Center. June 22, 2016. <http://www.people-press.org/2016/06/22/partisanship-and-political-animosity-in-2016/>

released tracking polls show stark divides between members of the two parties regarding Presidential approval and the perceived direction of the country.²⁸

The idea of partisanship being the driving force of American politics is hardly new; the key role of partisanship in the American political system lay at the heart of E.E. Schattschneider's seminal work The Semisovereign People. Schattschneider compellingly argues that the American political system, long assumed to place the power in the hands of the people that it represents, is actually more of a battlefield between two shadowy cabals, the Democratic and Republican parties. Each party co-opts special interest groups, as large as labor unions, and when they lose elections, their response is not to attempt to change their policies or attract more voters, but rather to change the battle lines on which elections are fought.²⁹ Schattschneider's thesis appears to largely circumvent a standing debate in political science over whether party identification causes issue orientation, or vice versa. Instead, according to Schattschneider, the parties determine the very battlefield of political campaigns. In this world, it matters less which issues a party agrees with the majority of the electorate on, because the party can choose – albeit with resistance from the opposition – the size and nature of the conflict in each election. A slightly deeper look shows that Schattschneider's thesis actually brings even greater importance to the aforementioned debate. How can the people make policy if the policy preferences

²⁸ “Daily Survey: Trump Tweets.” Dec 12, 2018. YouGov, Inc. https://d25d2506sfb94s.cloudfront.net/cumulus_uploads/document/88s8hhcgjz/tabs_Trump_Tweets_20181211.pdf

²⁹ E. E. Schattschneider. *The Semisovereign People: A Realist's View of Democracy in America*. (Hinsdale, Ill.: Dryden Press, 1975), 3-7.

of the people are informed by the two parties? With the partisan divide in the electorate deepening, and the frequency of party line voting greatly increasing,³⁰ the importance of whether the parties lead the voters or the voters lead the parties seems more important given the outsize impact that political parties have on the American political system.

One strain of argument on the topic is that the political context drives the direction of the causal arrow. In their analysis of partisanship and issue orientations, Highton and Kam found two distinct periods between 1973 and 1997, in which partisanship led issue orientations until 1982, and the reverse was true between 1982 and 1997. They posit that this switch occurred due to changes in the political context. The pair define the political context as the level of the salience of different issues to individual voters.³¹ As the American electorate has become more and more polarized over the last two decades, the political system appears to have returned to the former of Highton and Kam's two periods, in which the parties lead the people again. One possible driver of such a phenomenon is the increasing partisanship of the media. Over the last several years, the strength of partisan media echo chambers has greatly increased,³² which has the power to drive the two parties further and further apart on issues.³³ Even if

³⁰ Logan Dancey and Geoffrey Sheagley. "Partisanship and Perceptions of Party-Line Voting in Congress." *Political Research Quarterly* 71, no. 1 (2018).

³¹ Benjamin Highton and Cindy D. Kam. "The Long-Term Dynamics of Partisanship and Issue Orientations." *The Journal of Politics* 73, no. 1.

³² Webster and Ksiazek. "The Dynamics of Audience Fragmentation", 2012.

³³ Jason Carmichael, T. Brulle, and Robert Huxster. "The Great Divide: Understanding the Role of Media and Other Drivers of the Partisan Divide in Public Concern over Climate Change in the USA, 2001–2014." *Climatic Change* 141, no. 4 (2017).

the people lead parties on issues, as was true in the second of Highton and Kam's two periods, an increasingly partisan media is disseminating information to those people, and Schattschneider would argue that the media itself has simply become an extension of the parties. A field experiment conducted by Gerber, Huber, and Washington demonstrated that when partisanship changes, so too do issue-oriented attitudes, which they say implies that "partisanship is an active force that causes changes in important political outcomes" and that partisanship could be an "independent source of political decisions and opinions."³⁴ With the partisan media intensification showing no signs of stopping, the American political sphere appears to be solidly situated in a partisanship-leading-opinions phase for at least the next few electoral cycles.

The Old Machine Politics

The political machines of the late nineteenth and early twentieth century, the hallmark of American politics in the urban centers of the North, have caused a significant amount of ink to be put to the page in the canon of American political science. The political machine these days is a dirty phrase, the stuff that makes up negative attack advertisements lobbed during a Presidential campaign.³⁵ Indeed, the inner city machines, were not, or are not, depending on whom you ask, paragons of the democratic process. Local "precinct captains" distributed benefits from the machine infrastructure, anything from postponing evictions to the

³⁴ Gerber, Huber, and Washington. "Party Affiliation, Partisanship, and Political Beliefs," 2010.

³⁵ John McCain. *Political Ad: Chicago Machine (McCain)* (New York: N/a, 2008).

distribution of material goods, to local voters in exchange for their vote in the election.³⁶ The captain's success was "measured by votes which he [secured] for the candidates of his organization."³⁷ This type of vote buying is akin to what many in Western democracies would consider an insidious practice in the modern developing world. An apt synonym for machine politics would be the inherently antidemocratic term "clientelism."³⁸ That being said, the easily told story - of distributing fire protection and educational resources³⁹ - may not have been the true purpose of the classic political machine. An analysis of resource distribution in Chicago, one of the classic bastions of machine politics, found that the distribution of benefits had less to do with which person's vote was needed to win an election, and more to do with a host of other factors, including but not limited to technological advancement and population shifts.⁴⁰ Indeed, the idea of political machines as strict vote buyers carries in at a fatal flaw: the secret ballot. Since the end of the nineteenth century, before the peak in power of the political machines, the United States has used a secret ballot in its electoral processes.⁴¹ Interestingly, the secret ballot, sometimes known as the Australian ballot, actually began to be adopted in an attempt to curtail democratic principles, rather than protect them.

³⁶ Harold Foote Gosnell. *Machine Politics: Chicago Model*. (New York: Greenwood Press, 1968), 72.

³⁷ *Ibid.*, 85.

³⁸ Mariela Laura Szwarcberg. *Mobilizing Poor Voters: Machine Politics, Clientelism, and Social Networks in Argentina*. (New York, 2015), 4

³⁹ Kenneth R. Mladenka. "The Urban Bureaucracy and the Chicago Political Machine: Who Gets What and the Limits to Political Control." *The American Political Science Review* 74, no. 4 (1980): 991-98.

⁴⁰ *Ibid.*

⁴¹ Alan S. Gerber. "The Adoption of the Secret Ballot." Ph.D. dissertation, (Massachusetts Institute of Technology). 1994.

States like Massachusetts and Connecticut had explicit literacy tests at the end of the nineteenth century, and the secret ballot, which one had to be able to read for oneself, became a sort of pseudo-literacy test.⁴² By the time the political machines rose to power, the secret ballot was commonplace,⁴³ so the question remains: how could a machine function as an insidious antidemocratic organization, buying votes from ordinary citizens left and right, if it had no way to ensure the citizens actually voted with the interests of the organization?

Perhaps the picture of the political machine as purely antidemocratic is not quite correct. Indeed, machines in some cases actually worked to expand the

		Party Preference of Recipient vis-à-vis Party Offering Reward	
		Favors Party	Indifferent or Favors Opposition
Reward Recipient Inclined to Vote or Not Vote	Inclined To Vote	“Rewarding Loyalists” <i>Requires No Monitoring</i>	“Vote Buying” <i>Requires Monitoring of Vote Choice</i>
	Inclined Not To Vote	“Turnout Buying” <i>Requires Monitoring of Turnout</i>	“Double Persuasion” <i>Requires Monitoring of Vote Choice and Turnout</i>

Figure 1.1: via Nichter 2008

franchise, as in the case of women’s suffrage in the early 20th century,⁴⁴ though Schattschneider would likely argue that they only did so in order to fundamentally change the battleground after a defeat. In the late 1800s and

early 1900s the answer to the above question likely lay in intimidation, and a leveraging of the information asymmetries inherent in American society.

⁴² Alexander Keyssar. *The Right to Vote: The Contested History of Democracy in the United States*. (Rev. ed. New York: Basic Books, 2009), 115-116.

⁴³ Before its widespread adoption, however, the secret ballot caused partisan battles over its composition (see Keyssar 2009), perhaps precursors to contemporary discussions over ballot design in Florida.

⁴⁴ Keyssar, *The Right to Vote*, 173.

Additionally, both major parties operated under machine politics in the cities, indicating that votes could quite literally be bought with a transfer of services.⁴⁵ Still, though, one cannot simply hand-wave the inherent flaw away, and it becomes all the more apparent in modern society, in which some machines still operate.⁴⁶ So what do, and what did, machines actually do? Rather than buy votes at random, from whoever was selling, machines buy strategically, purchasing the votes from individuals likely to support their politics, but unlikely to vote. A machine can ensure that someone votes, but not who they vote for, so it follows that rather than buy votes, machines and similar organizations actually buy *turnout*.⁴⁷ Figure 1.1, taken from Nichter 2008,⁴⁸ shows the difference between vote buying and turnout buying. A key insight to make is that in order to buy turnout instead of votes, one must be able to identify for whom an individual is likely to vote. This sort of turnout differs slightly from the oft-discussed overall turnout in an election. Higher overall turnout is often considered to be beneficial to Democrats, as many lower-turnout subgroups typically support Democrats, which has led to a significant amount of partisan jockeying surrounding registration laws.⁴⁹ Even if boosting overall turnout helps Democrats, the party

⁴⁵ Gosnell, *Machine Politics*, 91.

⁴⁶ Though this mainly occurs in other countries, namely Argentina (see Szwarcberg 2015), many American cities, such as New York and Chicago, still have machine characteristics.

⁴⁷ Simeon Nichter. "Vote Buying or Turnout Buying? Machine Politics and the Secret Ballot." *American Political Science Review* 102, no. 1 (2008): 19-31

⁴⁸ Ibid.

⁴⁹ Keyssar, *The Right to Vote*, 253-257.

would be relatively better aided by an increase in just Democratic-leaning group turnout, and so the identification of voters remains an important goal.

As the power of the old political machine wanes, the problem of voter identification remains alive for modern political campaigns. Especially in urban centers, campaigns may no longer be able to rely on the old machine infrastructure to identify and turn out voters sympathetic to their cause. This project turns to computer science modeling techniques in order to identify those voters. So as the old machines' powers weaken, the machines of the 21st century rise up to take their place. The question remains one of the most fundamental in a democratic political system: who can one count on to vote in a certain way. Merely the meaning of the word "machine" has changed.

The Data

Thanks to Professor Latanya Sweeney, I was able to procure a dataset of browsing histories from comScore. Initially collected through the Wharton Research Data Services, Professor Sweeney was kind enough to allow me to use the comScore browsing history datasets from 2013, 2015, and 2016. According to information sent to me by Jinyan Zhang, a graduate student working with Professor Sweeney, the dataset that I use is a subset of comScore's larger Media Metrix dataset, which they bill as their "premier audience measurement product."⁵⁰ The dataset is a combination of demographics of computer users and the websites they visited over a yearlong period, organized by machine number –

⁵⁰ "Media Ratings – comScore, Inc." comScore. 2019.
<https://www.comscore.com/Products/Ratings-and-Planning/Media-Ratings>

a unique identifier between different Internet-accessing machines. According to comScore, machines are selected to be in the sample from their larger measured population only if the machines and users have met a number of requirements, including but not limited to reporting sufficient demographic information, and displaying sufficient activity each month.⁵¹ This takes care of eliminating users – which I use as a synonym for machine numbers - from the data that have not been active enough to make predictions about. comScore collected the data by contacting individuals via random digit dialing in order to have them take an enumeration survey that logged their demographics. They searched for only “active internet users,” defined as “persons age 2+ who were active on the Internet in the last 30 days from a home or work computer or shared use computer.”⁵² In between the 2015 and 2016 samples, comScore extended its enumeration survey methodology in order to increase response rates. The company looks at data from machines on a monthly basis, and for a machine to be included in the dataset that I obtained from Professor Sweeney, it must have been active for an entire year. As a result, the larger Media Metrix dataset that comScore creates is much larger, on the order of a million data points instead of the tens of thousands in the dataset I use.

The format of the data that I was given was a large comma separated values file (CSV) in which each row had a machine number, the demographics associated with that machine number, the universal resource locator (URL) of the

⁵¹ ComScore. “comScore Media Metrix Description of Methodology.” comScore, Inc. Januray, 2017. Obtained via email.

⁵² Ibid.

website that the user visited, up to the top level domain.⁵³ Each row also contains the information for how long a user stayed on that site, and how many pages they visited during that one specific visit, as well as when the connection was made. As such, if a user visited a website on Tuesday, and then again on Wednesday, it would show up twice in the dataset, but if they simply navigated to multiple pages on a site, for example going first to “espn.com” and then to “espn.com/boston”, there would only be one row in the CSV, but it would have a value of 2 in the “pages_visited” column. Since the data is organized by machine number, which does not map perfectly to individuals, it is possible, even likely, that in many cases the true browsing history of an individual will not be captured, or the union of the browsing histories of multiple individuals will be. That said, an analysis as described so far using this data is still valuable for the discussed purposes, as each machine will have a primary user, and that is who I will be attempting to predict the partisanship of. The demographic information in the dataset is pertinent to the primary user of each machine. The comScore data contained browsing histories for 46,927 machines in 2013, 54,826 machines in 2015, and 81,408 machines in 2016. The increased size of the 2016 dataset is perhaps due to a change in the comScore sampling methodology that was designed to increase responses to their enumeration survey for the year 2016. Due to the size of the 2016 dataset, and its temporal proximity to the voter files I utilize in Chapters 2 and 3, my analysis focuses solely on the 2016 data. The demographic data available falls into the

⁵³ Top level domains are the “dot-X” pieces of a URL, such as “.com” or “.edu” and so forth.

following categories: racial background, household income, census region, the level of education of the head of the household, the household income, whether or not children are present, and the zip code. For detailed information on how these categories are broken down, see Appendix 1.

Unfortunately, the original format of the dataset is not effective for building predictive models. A better format would have been to have each machine number be a row in a matrix, with each website being a column. The cell of the matrix in the row with machine number M and domain name D would then be the number of times the user M had visited domain D , or the sum of the “pages_viewed” column in all rows of the original dataset with machine number M and domain D . Importantly, this same transformed dataset can be easily used as a dataset where each cell is a binary “visit or did not visit” cell using integer-to-boolean casting in almost any programming language. After first trying to use the python library ‘pandas’ to transform the data, I wrote code to do it by hand, one line at a time. One of the advantages of this method is that it allowed me to make smaller datasets with only a portion of the machines, which took up much less space. These transformed matrices take up $O(m*d)$ space, where d is the number of domains in the dataset or subsample and m is the number of machines. In practice, the actual amount of space that the transformed sample takes up could be decreased by a linear factor if storing only binary values for the cells of the matrix, but the asymptotic space will be the same. Since in the worst case, there could be an infinite number of rows in the original, untransformed dataset, and each row must be processed individually, the algorithmic runtime is not calculable

in terms of d and m . That said, processing each row from the original dataset takes constant time. However, each time a new machine is reached in the larger dataset, a new row must be added, and each time a new website is reached, a new column must be added. With clever coding, and depending on the language, cells can be accessed in constant time, and so the asymptotic runtime of the transformation is $O(r+w+d)$ and since we know with high probability $r \gg w$ and $r \gg d$, we have the asymptotic runtime of $O(r)$. For detailed information on how I transformed the datasets into a usable matrix, see Chapter 2. I am purposefully ignoring the columns that contain information about the time of the visit to a website and the duration of a visit. Though a model could be built to predict partisanship that incorporated these variables, it is outside the scope of this thesis.

Minimizing Potential Bias

With so much of the world being run by computer algorithms in the modern day, it is worth critically questioning the bias and fairness of such algorithms. In 2013, Sweeney found that Google was delivering advertisements that were racially biased against persons of color. She demonstrated that the top Google advertisement was much more likely to be for an arrest record search if one Googled a black sounding name when compared to Googling a white sounding name. The results occurred whether or not a person actually had an arrest record.⁵⁴ Results like this have nontrivial effects on the real world, as many individuals will Google other people and could have their opinion informed by the

⁵⁴ Latanya Sweeney. "Discrimination in Online Ad Delivery." *Queue* 11, no. 3 (2013).

presence of an advertisement for an arrest record search, without actually clicking on the advertisement. Sweeney argues that this bias in the results of the algorithm for online ad delivery amount to racism and racial discrimination on the part of Google.⁵⁵ In their investigative reporting on a system being used in Florida to predict recidivism in petty criminals, ProPublica found that the algorithms at the heart of the system are also biased against black Americans. The authors found that at the same time that black defendants who did not go on to commit more crimes were nearly twice as likely to be misclassified as a risk as white defendants who did not go on to commit more crimes, white defendants who *did* were almost twice as likely to be misclassified as no-risk than black defendants who did. Even when the authors controlled for prior factors, the bias against black defendants remained.⁵⁶ Though Northpointe, the company who had built the prediction algorithm, pointed out that they explicitly did not use race as a factor in their algorithm, many other pieces of information, such as address and even just ZIP code, are effective proxies for race, and can be used as such by a predictive algorithm.⁵⁷ This is not to say that human decision makers are not without the bias that is present in algorithms; a 2016 study found that judges sometimes issue harsher rulings the day after their favorite football team lost a game than the day

⁵⁵ Ibid.

⁵⁶ Jeff Larson, et al. “How We Analyzed the COMPAS Recidivism Algorithm,” ProPublica, May 23, 2016. <https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm>

⁵⁷ Julia Angwin, et al. “Machine Bias,” ProPublica, May 23, 2016. <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>

after said team won.⁵⁸ Humans are not without their own inherent biases, and algorithms can be a piece of the puzzle that helps fix the biases in the judicial system. The judicial system is meant to provide fair treatment to all Americans, and some combination of algorithmic predictions and human predictions by judges is quite possibly the best way to achieve that.⁵⁹ If society moves in such a direction however, it must ensure that the algorithms themselves do not codify the biases that are at present only encoded in human nature.

A natural question to ask is how exactly the algorithms become biased. It seems unlikely that there is a software engineer sitting at Google's headquarters in Mountain View, writing an if statement along the lines of:

```
if sounds_black(name) == TRUE then ad = arrest_record60
```

In any event, the code for modern predictive algorithms, with their deep neural nets and multitude of features, is too complicated to simply read through and check for hard-coded bias. We can only measure the outcomes, and something is biasing the outcomes. One possible source of this algorithmic bias is biased training data. In order to make predictions, algorithms must first have a set of data on which to train themselves, and inform those predictions. If the data being fed into an algorithm is biased, then so too will be the outcomes that the algorithm spits out. An easy way for the training dataset to bias an algorithm is if the

⁵⁸ Ozkan Eren, and Naci Mocan. "Emotional Judges and Unlucky Juveniles." *American Economic Journal: Applied Economics* 10, no. 3 (2018).

⁵⁹ Jon Kleinberg, et al. "Human Decisions and Machine Predictions." *The Quarterly Journal of Economics* 133, no. 1 (2017).

⁶⁰ Presumably any engineer working in Mountain View would not have the redundant boolean, but that is beside the point.

training set is not representative of the set of people the algorithm will eventually be required to make predictions about. Such a phenomenon has been observed and tested in facial recognition technologies, like the ones that are used to open newer versions of the iPhone. The training and testing sets for the technology were composed of primarily white men, and so the technology is significantly better at recognizing the faces of white men than any other demographic group.⁶¹ In collecting their data, comScore takes special care to note that the universe of people that they can sample on is only those with Internet access – a group close enough to the voting eligible population in the United States that it will do for this project. I analyzed the demographic makeup of each of the 2016 dataset I received from Professor Sweeney along the axes present in the data, and compared them to benchmarks from the 2010 United States census. The results indicate that while the demographics of the dataset do not perfectly match those of the population of the United States, there is enough variation that the demographic makeup of a sufficiently large subsample of the data will be close enough to the makeup of the United States at large.

Privacy Concerns

The final stage of this project, linking the demographic information and to information that can identify the partisanship of voters, is a problem of what computer scientists call “re-identification.” Many datasets, such as the comScore dataset that I am using, are allegedly “de-identified” in order to protect the

⁶¹ Joy Buolamwini and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification.” *Proceedings of Machine Learning Research* 81, (2018).

privacy of the individuals whose data is in the dataset – that is, some amount of information has been removed, or in some cases some amount of chaff has been added, in order to make the data anonymous, and thus private. The conception of privacy and anonymity that should be applied to the analysis in this project is that of Latanya Sweeney’s k -anonymity. Sweeney’s defines k -anonymity using the following definitions:

Definition 1: Attributes

Let $B(A_1, A_2, \dots, A_n)$ be a table with a finite number of tuples. The finite set of *attributes* of B are $\{A_1, A_2, \dots, A_n\}$.

Definition 2: Quasi-identifier

Given a population of entities U , an entity-specific table $T(A_1, \dots, A_n)$, $f_c: U \rightarrow T$ and $f_g: T \rightarrow U'$ where $U' \subseteq U$, a *quasi-identifier* of T , written Q_T , is a set of attributes $\{A_i, \dots, A_j\} \subseteq \{A_1, \dots, A_n\}$, where: $\exists p_i \in U, s. t. f_g(f_c(p_i)[Q_T]) = p_i$.

Definition 3: k -anonymity

Let $RT(A_1, \dots, A_n)$ be a table and QI_{RT} be the quasi-identifier associated with it. RT is said to satisfy k -anonymity if and only if each sequence of values in $RT[QI_{RT}]$ appears with at least k occurrences in $RT[QI_{RT}]$.

In plainer English, a quasi-identifier for a dataset is any set of the columns in the dataset, and a dataset is only k -anonymous if each set of values that fill the quasi-identifier columns appear with at least k times. In other words, a dataset is only k -anonymous if there is no set of features exist such that a search of the dataset for those features returns fewer than k individuals – that is, given any set of identifying information, an individual in a k -anonymous dataset is indistinguishable from at least $k - 1$ others.⁶² Note that for the definitions to have

⁶² Latanya Sweeney. "K-anonymity: A Model for Protecting Privacy." *International Journal Of Uncertainty Fuzziness And Knowledge-Based Systems* 10, no. 5 (2002).

meaningful importance, the attributes in a quasi-identifier should be non-sensitive. In many cases, re-identification occurs via what Sweeney refers to as linking, in which k -anonymous datasets containing similar quasi-identifiers are linked together into one larger dataset, that may not respect k -anonymity. One of the classic examples of this linking occurred during the Netflix Prize competition, in which one team was able to identify Netflix users by linking the anonymized Netflix data to their IMDB profiles, which were attached to names, thus identifying the Netflix users.⁶³ In Chapter 2, I undertake a similar process using state-level voter files and the comScore dataset.

I also expect to be helped in re-identification by a lack of l -diversity in the datasets that I am attempting to link. Noting that even k -anonymous datasets are vulnerable to certain types of re-identification attacks, Machanavajjhala et. al. extended the conception of privacy to protect against some of those attacks. We need the following two definitions:

Definition 4: a q^* block.

Given a table T that respects k -anonymity, and a quasi-identifier Q over non-sensitive attributes $\{A_i, \dots, A_j\}$, a q^* -block is a block of the at least k tuples in T with $\{A_i, \dots, A_j\} = q^*$.

Definition 5: l -diversity

A table T with sensitive attribute S is l -diverse if and only if for every q^* -block in T , there are at least l values of S that are well represented.

The concept of l -diversity assumes the existence of a “sensitive attribute” in the dataset, something that an adversary may be attempting to discover about

⁶³ Arvind Narayanan and Vitaly Shmatikov. "How To Break Anonymity of the Netflix Prize Dataset." 2006.

individuals in the dataset.⁶⁴ An example is helpful: imagine a school that releases a 5-anonymous dataset about its students, and an employer that wants to look at the dataset to see if a student applying for a job has ever been failed a class. Say the dataset contains a column that shows the number of classes a student has failed. If the dataset is 5-anonymous, then the student cannot be distinguished from at least four other students in the dataset using any set of identifying information; however, if all five of the students who cannot be distinguished from each other have all failed a class, then the employer knows that the student in question has done so. In short, if the sensitive attributes of a dataset are too homogeneous, then k -anonymity may not be an effective conception of privacy. l -diversity attempts to solve the problem of these “homogeneity attacks.” To have proper l -diversity, the l values in the sensitive column must also be “well represented,” meaning they appear with roughly similar frequencies.⁶⁵ It is important to note that the extension of k -anonymity to l -diversity requires the adversary to be looking for something in particular *about* individuals, rather than simply wanting to identify an individual. In this project, the sensitive attribute that I, the adversary, am looking for is partisanship, or perhaps more accurately, party affiliation. The datasets that I have are likely to be k -anonymous for large values of k – generally speaking many people of similar demographic categories live in the same place, but they are very unlikely to be very l -diverse. With only three

⁶⁴ Ashwin Machanavajjhala, et al. "L-diversity: Privacy beyond K-anonymity." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, no. 1 (2007).

⁶⁵ Ibid.

values of the sensitive column – Democrat, Republican, Independent/Unaffiliated – the highest value of l possible for any return of a search on the datasets is three. Additionally, due to demographic voting blocks, it is likely that in many sets of the demographic searches some of the three values will not be well represented. As such, the datasets, once linked, will likely lack l -diversity, making the task of discovering partisanship from browsing history significantly easier.

In the age of big data, the privacy of individuals may be compromised by their presence in datasets that large corporations collect and release. American laws, such as the Health Insurance Portability and Accountability Act (henceforth HIPAA) and the Federal Education Rights and Privacy Act (henceforth FERPA) are explicitly concerned with the release of data that could reveal sensitive information about the people in question. One can easily imagine a scenario in a world where all health care information is public, where an individual applies for a job but is rejected due to some piece of sensitive health information. It is equally easy to extend the imagination to a situation in which such sensitive health information is used to discriminate against a class protected under American law: if a corporation were to discriminate against people who have prescriptions for medications that only women take, then the corporation would be essentially discriminating against women. Perhaps a corporation discriminates against LGBTQ+ individuals by simply turning people away who have prescriptions for HIV medications. That said, it is also easy to see how sharing personally identifiable health information between hospitals can lead to better outcomes for patients. As such, citizens have competing interests: both in their health data

remaining private, and in their data being available. HIPAA attempts to respect both interests by preventing the release of “individually identifiable health information” to certain “covered entities,” ranging from employers to certain health care entities.⁶⁶ According to the United States Department of Health and Human Services, the privacy regulations present in HIPAA is to “assure that individuals’ health information is properly protected while allowing the flow of health information needed to provide and promote high quality health care and to protect the public's health and well being.”⁶⁷

FERPA is meant to accomplish similar goals in the educational sphere. An added twist in recent years has been the rise of online courses, where data collection is significantly easier for institutions, and the desire to use the data collected to improve educational outcomes via independent analysis. To do this, the data must be publicly released, and in order to follow the law must be released in accordance with FERPA. FERPA allows the release of educational data if “personally identifiable information” has been removed from the dataset. One interpretation of this standard is that the information can be released if it has been de-identified to an acceptable level of k -anonymity. The problem is that in order to reach acceptable levels of k -anonymity, oftentimes just 5-anonymity, the statistical value of the data may be utterly ruined.⁶⁸ Again, the release of the

⁶⁶ The Health Insurance Portability and Accountability Act of 1996 (HIPAA). P.L. No. 104-191, 110 Stat. 1938 (1996)

⁶⁷ “Summary of the HIPAA Privacy Rule.” U.S. Department of Health and Human Services. July 26, 2013. <https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html>

⁶⁸ Jon Daries, et al. "Privacy, Anonymity, and Big Data in the Social Sciences." *Communications of the ACM* 57, no. 9 (2014).

information has two concerns, each running counter to the other in terms of privacy. These two examples are a microcosm of a larger problem: with almost all datasets in the modern day, there may be an interest in protecting privacy, but there could also exist a valid interest in protecting it as little as possible.

Following the Presidential election of 1884, the United States of America adopted the secret ballot en masse, with Kentucky being the final state to adopt the practice in 1891. The secret ballot, of course, prevents a significant amount of intimidation at the polls, significantly decreased vote buying, and is essential to the modern democracy running freely,⁶⁹ though this may not have been the primary goal in its initial adoption, as mentioned above. Experimental evidence has demonstrated that the more people know about the existence of the secret ballot – that is the more people that know that their vote is indeed a secret – the more people vote. This effect is especially prevalent among persons who have not voted before.⁷⁰ If partisanship can be tied tightly to browsing history, however, and one's vote can be tied tightly to partisanship, then how secret is the ballot anymore? If any advertiser, or political group, can get ahold of an individual's browsing history, and use it to predict their partisanship, and then their vote, then the very foundation of democracy – the vote – may be compromised. With the fall of net neutrality, it may even be legal for corporations to discriminate against consumers on the internet by identifying their partisanship through the websites

⁶⁹ Gerber, "The Adoption of the Secret Ballot," 1994.

⁷⁰ Alan S. Gerber, et al. "Do Perceptions of Ballot Secrecy Influence Turnout? Results from a Field Experiment." *American Journal of Political Science* 57, no. 3 (2013): 537-51.

that they visit. This project's success indicates a serious need to rethink the laws surrounding how browsing histories are kept private, and a hard look at how the communications industry uses and sells them in the modern day.

Chapter 2

I begin the analysis portion of this thesis by identifying the outcome variables. At first glance, it might seem obvious that there is only one outcome variable: partisanship. But how should partisanship be measured? Indeed, what does “partisanship” even mean? Partisanship could be taken to mean which party an individual voted for in the most recent election, which party they voted for in some set of the most recent elections, which party they identified as when registering to vote, or even some measure of an amalgamation of a set of political beliefs. The comScore dataset does come with a readily attached set of demographic information, but that demographic information does not include the partisanship of each individual computer user, in any method of measurement. For this study, I define partisanship as an individual’s self-identification to a political party, denoted by the party for which they are registered to vote. This definition comes with the advantage of being explicitly measurable in some states. Many states record a voter file, which contains information on the voters in a given state, and some of those states log the party registration of individual voters in said voter file. By using standard re-identification and linking procedures, I am able to create a measure for the partisanship of an individual, or perhaps more accurately, a measure of the likelihood that a given individual in the comScore dataset identifies as a member of the Democratic party.⁷¹

I also use a secondary outcome variable, race, for a variety of reasons. First, it can be predicted using the same techniques as partisan identification, and

⁷¹ As long as they reside in a state that reports partisanship.

so there is no great increase in the amount of work, programmer time, or computing time required to additionally attempt to predict the race of computer users based on their browsing histories. Second, and unlike partisan identification, race is explicitly reported in the comScore dataset demographic information, in the ‘racial_background’ column of the dataset. Though the comScore data does not include Hispanic or Latino as a racial option, it still breaks users down into the following categories: White, Black, Asian, or Other. Finally, in some cases, race can be a very strong proxy for partisanship. In his 2012 re-election bid, President Barack Obama gained an estimated 93% of the vote from black Americans.⁷² Even without a black candidate at the top of the ticket in 2016, the Democratic party still gained 89% of the black vote for Secretary Hillary Rodham Clinton.⁷³ Though not nearly as drastic, Democrats also won the Asian vote by large margins in both 2012 and 2016. As such, a strong prediction that an individual is black, or merely nonwhite, functions as a strong prediction that that individual is likely to identify as, or vote for, a Democrat. This definition of partisanship, in which race is a proxy for the likelihood to vote for a party, is slightly different than the one I articulate above, in which party identification, and not voting behavior, is the deciding factor. While this is true, one must remember the overall goals of predicting partisanship: for academics, the goal is a deeper understanding of the forces driving or being driven by partisanship, and for professionals, the

⁷² “How Groups Voted in 2012.” Roper Center. Accessed March 18, 2019. <https://ropercenter.cornell.edu/how-groups-voted-2012>

⁷³ “How Groups Voted in 2016.” Roper Center. Accessed March 18, 2019. <https://ropercenter.cornell.edu/how-groups-voted-2016>

goal is being able to accurately microtarget voters for electoral purposes. While the two definitions of partisanship above differ slightly, using either, or indeed using the two in concert, furthers this study as a means to the end goal.

Other possible outcome variables to predict in addition to partisanship for the same reason as race, could include gender, age, and district type – urban, suburban, exurban, rural. Though the comScore dataset does include information on age, there are simply too many categories to predict accurately using the same techniques as for party and race, and even if I were to predict age based on ranges, say 18-25 vs. 25-45 and etc., the race based partisan divide still dominates one based solely on age.⁷⁴ The same can be said for district type – I could conceivably code different ZIP codes, which are present in the demographic data, as rural, urban, exurban, or suburban, but again, the race based effects would dominate. Certainly, though, room for further research exists, potentially attempting to unravel partisanship among white voters, based on using browsing history to predict what type of district a given voter may live in. Gender, meanwhile, is not reported in the comScore dataset, and cannot be estimated from the voter file in the same way as partisanship, giving me no way to predict it. Thus, I am left to predict partisanship, with training data coming from voter files and partisan identification, and race, trained directly from the comScore dataset.

In Hacking the Electorate, Hersh notes that when political campaigns use microtargeting techniques in an attempt to identify voters who will be sympathetic to their candidate, the campaign will only act on an identification if

⁷⁴ See figure 2.10 later in this chapter for an example.

their models predict that an individual has a *greater than 80% chance* of being a co-partisan.⁷⁵ This 80% threshold is the primary goal of my analysis. That said, a secondary goal exists, one that is not quite as strict. Identifying whether or not someone is simply *more likely* to be a Democrat than a Republican, in essence a 50% threshold when looking at data containing only partisans for the major parties in American politics, is also a useful measure for both academics and professionals. Thus, this study is an attempt to predict partisanship, through the proxy of party identification, and race in individuals to the 80% and 50% likelihood thresholds, using browsing histories.

Transforming the Data

The dataset that I received from Professor Latanya Sweeney was organized by triplets of user, domain name, and browsing session. There was one row in the dataset for every time a single user visited one website⁷⁶ (not an individual page on the website), in one browsing session. If a user visited a website twice during the same browsing session, or visited two different pages on the same website, then that was logged in the “pages_viewed” column of that row in the dataset, but if a user visited the same website during two different sessions, the user-site pair would appear more than once. This meant that there were many rows for each individual user in the dataset, and even many rows for the same

⁷⁵ Hersh, *Hacking the Electorate*, 72

⁷⁶ As mentioned in the introductory chapter, websites were reported as Top Level Domains, or TLD. This means that nothing after the slash following .com or the equivalent was reported. It also means that if a user were to visit *twitter.com/Harvard* and then *twitter.com/thecrimson*, it would be reported in the dataset as the user having viewed two pages on *twitter.com*.

user-domain pair in the overall dataset. In addition to the user-domain-session triplet and the information on the number of pages viewed on the domain, each row also contained information on the date and time of the visit, and the duration of the visit to each website. Additionally, each row contained the demographic information for the user that the row referred to.

In order to build my models, I needed to transform the datasets into a matrix where each row represented one specific user, with columns representing specific domain names.⁷⁷ A value in a specific cell of the matrix then could represent either the number of pages on that site a user visited in a year – a sum of the “pages_viewed” column across all different sessions for the same user-domain pair of the original dataset – or a binary value representing whether or not a user had *ever* visited the site during the year. Importantly, the former can easily be turned into the latter by casting all cells to boolean values and then back to integers. Using the python library `pandas`, which is meant to mimic the functions of DataFrames in the R programming language commonly used among political scientists, I first transformed the data into a usable format.

I first attempted to transform the dataset using built in `pandas` functions, as the user-domain-session triplets essentially presented a hierarchical organization structure. Initially, I began by attempting to transform the entire dataset for one year’s worth of browsing histories at once. Doing this required massive memory storage to be able to store and manipulate both the original dataset and the transformed matrix. Even using the Harvard-MIT Research

⁷⁷ For a discussion of time and space complexity, see Chapter 1.

Computing Environment, and procuring more than 200 gigabytes of memory for the jobs, the transformation still resulted in memory errors.⁷⁸ Attempting to make the classic trade-off of space for time, I then wrote code that read the original dataset in line by line, and built the matrix cell by cell. Though this did not require the original dataset to be stored in memory, it took significantly longer, as each line was process individually, instead of taking advantage of fast sum operations from the numpy library. Still, the storage errors persisted. Despite reading the original data file in one line at a time, and forcing python's garbage collector to reclaim the memory when a new line was read in, the transformation program was still running past the memory capacity I had available, even using 2013, the smallest of the datasets in terms of the number of users present. This indicated that the transformed matrix itself was the cause of the memory issues.⁷⁹

The main consequence of the post-transformation data matrix taking up so much memory space is that transformation must be done on a sample by sample basis. First, the modelling program chooses a random sample of the users in the overall dataset - 10,000, for example – and makes a users-by-domains matrix using only that subset of users, and only the domains that those users visited. This decreases the necessary space for the program to run on two fronts: first the number of rows required decreases, from over 80,000 for the full 2016 dataset to just 10,000 for a matrix with only 10,000 users, and second, by decreasing the number of domains visited overall, and thus the number of columns in the dataset.

⁷⁸ When writing the code, I tested it on a smaller dataset containing only 100 browsing histories on my local machine.

⁷⁹ For the code referenced here, see Appendix 2.

Since even in memory efficient languages like C, integers take up a significant amount of space, decreasing the number of columns, which in many cases are filled by mostly zeroes, represents a massive improvement in the amount of space required for the program to function. Figure 2.1 shows the number of domains in a sample as a function of the number of users. The number of domains increase sharply for the first 10,000 or so, before levelling off, but still increases roughly linearly as the number of users increases past 10,000. This is mirrored by a steady, but rather linear decrease in the number of domains per user in the dataset, as the number of users increases (see Figure 2.2). Clearly, the amount of space saved by sampling only 10,000 users, in which there are around 280,000 domains when compared to sampling, compared to using even half of the 2016, in which there would be around 500,000 domains. Back of the envelope math yields upwards of an 80% space saving by sampling. Sampling is also achieved fairly efficiently by first pulling only the demographic information out of the original dataset. Since each individual row in the original dataset for the same user has the same set of demographics included, a massive number of duplicates can be removed, making the demographic dataset significantly easier to handle for things like sampling than the larger dataset as a whole. After the sample is selected, and before any transformation occurs, the rows in the sample are selected from the original dataset and the remainder are dropped in order to save space. This is the principle reason that every program I use requires both a “Sessions” and a

“demographics” argument; the Sessions file is used as the main dataset, while the sample is selected from the demographics file.⁸⁰

Obviously, not using the entire dataset and instead using a sample brings with it modeling consequences. If a model is built using a matrix that may not contain some of the domains visited by a certain user, then one must be careful if making predictions based on that model, watching out for KeyErrors⁸¹ and the like. Using programming tricks and imputing 0s into the necessary columns when testing the data, this can be worked around, and models can be used on data points that were not in their original testing sets. Perhaps more importantly, sampling from the overall dataset, instead of using all the data available, makes models less likely to be accurate, as they are trained on less data.

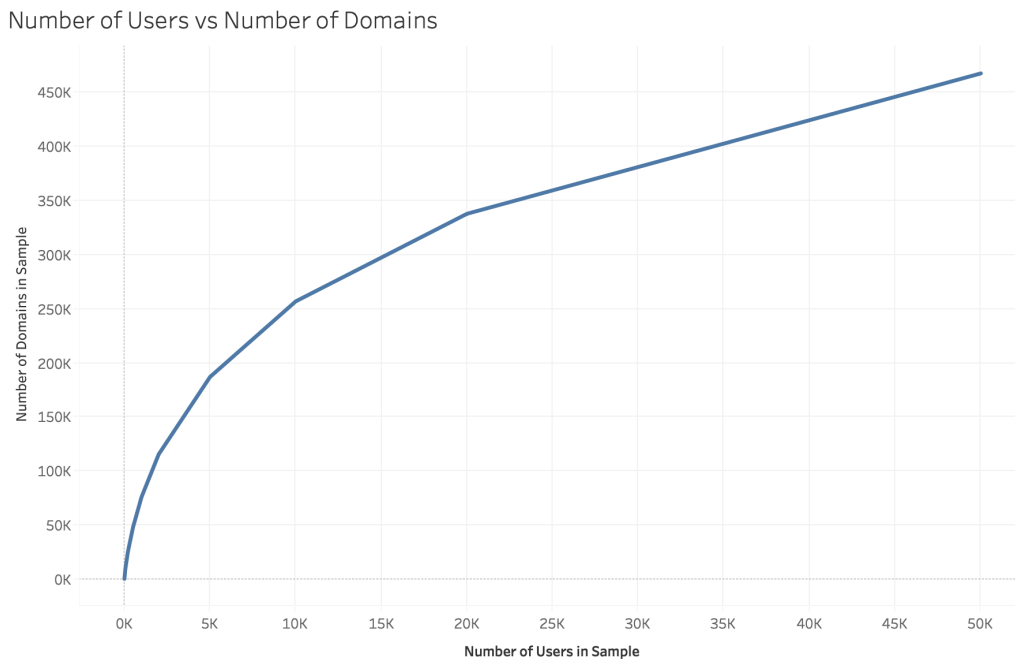


Figure 2.1: Number of domains vs. number of users

⁸⁰ Code can be found in Appendices 2-6.

⁸¹ A KeyError occurs when python attempts in index into a data structure using a certain key, but that key does not exist in the data structure.

Number of Users vs. Number of Domains per User

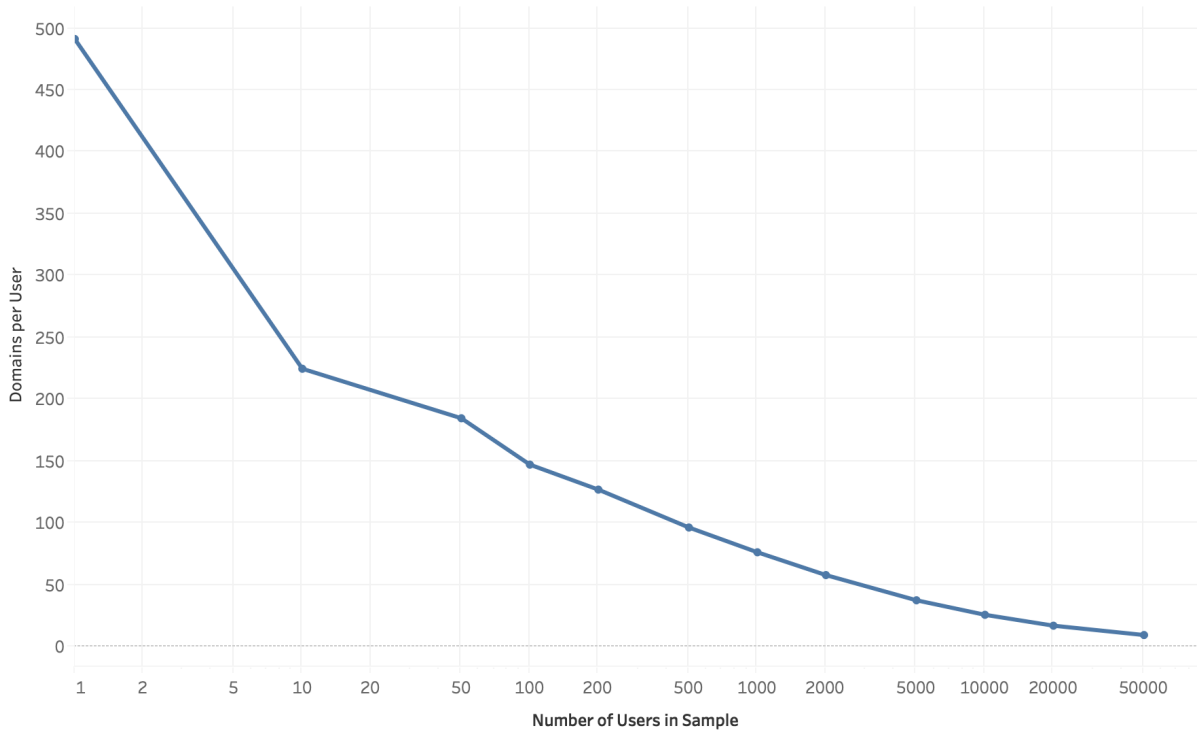


Figure 2.2: Number of domains per user vs. number of users

Generally speaking, more training data – when properly managed – makes models more accurate. In the 2016 case, I threw out roughly 87% of my data every time I made a model using only 10,000 of the users. Unfortunately, this was a necessary price to pay to be able to build any models at all, due to space constraints. One can see how moving all the library functions used in python to a language like C, in which the programmer could potentially store values as shorts and have significantly more power over memory usage, could result in the ability to use more data, but such an undertaking was not feasible for this project.

Sampling the data does have one significant advantage, however. By sampling, I am able to control some of the underlying demographics of the

sample, in order to reduce bias. As mentioned in the introductory chapter, algorithms can become biased when they are trained on training data that is not representative of the population that the algorithm will be used on. Facial recognition algorithms trained on primarily white faces are significantly worse at recognizing non-white faces.⁸² Algorithms meant to predict recidivism in criminals can be racially tinged by biased training data.⁸³ Sampling the overall dataset allowed me to control the underlying demographics, primarily the racial makeup, of the sample. I began my modeling process using samples constructed by weighting each racial group in accordance with their weight in the 2010 United States Census.

Logistic Regression Models

The first class of models I attempted to build – for both race and party identification – were multiclass logistic regression models. This family of models, much like linear regression, attempts to minimize a loss function between predicted values and true values. In my case, I used the `sklearn` library with automatic 5-fold cross validation in the training set, attempting to minimize the standard L2 loss function. Multiclass logistic regression can be used for both the party modeling, in which an individual could either be coded as a Democrat or not, and race modeling, in which case there are four categories in which to place data points from the comScore dataset – a code of 1 for a white individual, 2 for a black individual, 3 for an Asian individual, and 5 for any other category.

⁸² Buolamwini and Gebru, “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,” 2018.

⁸³ Angwin, et al., “Machine Bias,” 2016.

Unfortunately, the original logistic regression models I attempted to fit on the comScore samples that I created as described above – with 80% of the sample being used as a training set, and 20% being used as a testing set - would never converge successfully to a predictive model. Even after increasing the standard number of iterations by a factor of 10, the models still always failed to converge. The lack of convergence was caused by the incredibly high dimensionality of the dataset. As seen in Figure 2.1, a sample of 10,000 computer users results in about 280,000 different domains in the sample. Since each domain name is a predictor variable in the model, the model was attempting to converge over a 280,000-dimensional space – something that could almost never happen. Logistic regression models will fail to converge to a maximum likelihood estimator if a predictor variable, in this case any of the domains, has the same outcome value for any of its values, essentially making it a dummy variable.⁸⁴ With so many domains in the sample – upwards of 25 times the number of users for a sufficiently large sample – it is highly likely that in any given sample, at least some of the predictors are acting as these dummy variables, and causing convergence to fail.

The failure of logistic regression models to reliably converge, or even to converge at all, had two potential solutions. The first was to abandon logistic regression models, and proceed with modeling race and partisanship through other avenues, and the second was to somehow filter the data so that the logistic

⁸⁴ Paul Allison. “Convergence Failures in Logistic Regression.” SAS Global Forum 2008. 360.

regression models converge. I chose to avoid the latter pathway, and instead focus on modeling the quantities of interest through different strategies. The filtering method would have brought with it a host of problems, first and foremost how exactly to filter the data. Filtering out low occurrence domains could remove exactly the predictive information that I was searching for. Additionally, the *absence* of a domain in a users browsing history could be just as predictive as the presence of that domain in another's. Imagine a website that literally *all* Democrats visit, call it *iamademocrat.com*. A filtering method designed to cause logistic regression models to converge might remove that column from the transformed data matrix. However, the absence of *iamademocrat.com* in a given user's browsing history in this contrived example is enough information to code them as a not being a Democrat! Now imagine that *only* Democrats visit *iamademocrat.com*, meaning that not all Democrats may visit the site, but literally *zero* Republicans have the site in their browsing history for the year in question. In this case, the presence of *iamademocrat.com* in a browsing history is sufficient to code the user as a Democrat, and once again, a filtering algorithm designed to ensure convergence may drop the column containing that highly relevant information. This sort of website, which is exclusively visited by one type of person, is what the second kind of model that I *do* use explicitly depends upon. Abandoning multiclass logistic regression models, I instead turned my efforts to two other classes of models: random forest classifiers and a classifier model based off of Latanya Sweeney and Jinyan Zang's concept of exclusivity indices.

Random Forest Classifiers

After abandoning attempts at modeling my quantities of interest through multiclass logistic regression, I turned first to building random forest classifiers.

A random forest is named so because it is essentially an ensemble of decision tree

classifiers, in which data points are

classified by branching at various

decision points based on different

predictors. To continue our previous

example, recall *iamademocrat.com*,

specifically the version which is

visited by only Democrats. A simple

decision tree, of depth 1, may look

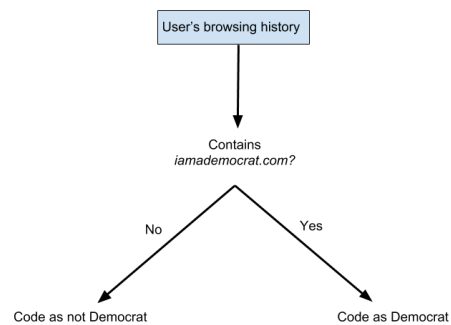


Figure 2.3: A Basic Decision Tree, $d=1$

like the drawing in Figure 2.3. As more decision points are added to the tree, the

depth grows, and then many trees make up the random forest, with the forest

classifier giving the best classification after aggregating the classifications given

by each individual tree. This structure makes random forest classifiers quite

effective at making predictions across a high-dimensionality space, making the

them very well suited for the data space on the order of the hundreds of thousands

of dimensions present in one of my browsing history samples. Like logistic

regression, the library `sklearn` comes with a built in sub-library of functions for

implementing a random forest classifier. Given the computing power at my

disposal, and the relatively low amount of time required to fit a random forest

classifier, I increased the number of trees in the forest from 10 to 256, and set a maximum tree depth of 48. I then began modeling race using random forests.⁸⁵

Race Modeling with Random Forests

I began the modeling process by creating samples of the overall comScore dataset, using the 2016 data both because it was the largest sample available and because it is the most temporally relevant, that had the demographic makeup of 2010 Census – again, the most temporally relevant available dataset. Obtaining the demographic breakdowns from census.gov, I set the original sample at 76.6% white,⁸⁶ 13.4% black, 5.8% Asian, and 4.2% “other.”⁸⁷ I then trained a random forest model on 80% of the data sample, saving 20% to test the model on later. Initial results from the census demographic weights on the surface seemed to be highly successful. Runs of the modelling code repeatedly returned a classification

⁸⁵ Code for random forest modeling can be found in Appendix 4.

⁸⁶ This includes those who identify on the US Census as Hispanic or Latino, as there is no comScore code for that race category.

⁸⁷ “Other” includes Native American, Pacific Islander, and other race categories not explicitly included in the other 3 categories.

accuracy on the testing set in the realm of 75% accurate, well above the 50% threshold and close to the 80% threshold that Hersh uses in Hacking the Electorate. However, a slightly deeper look into the results of the model revealed a troubling picture: instead of accurately classifying 78% of people across different racial groups, the model essentially just predicted nearly everyone to be

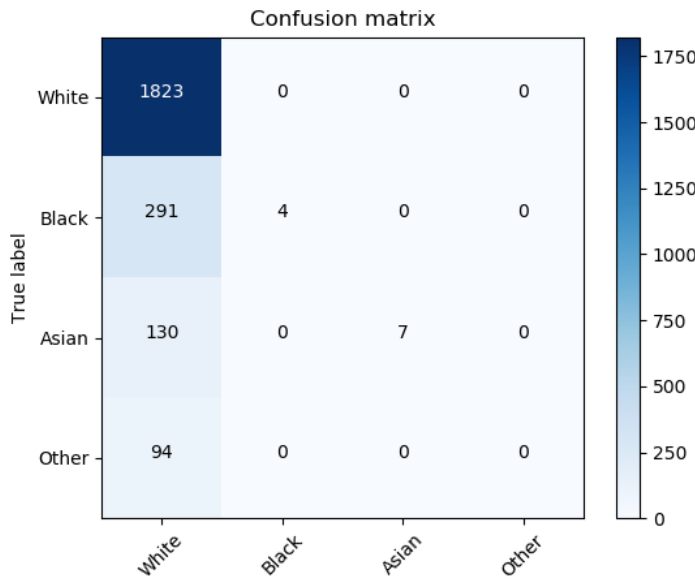


Figure 2.4: Census Weights Confusion Matrix

white, which accounted for the vast majority of the successes, and also the vast majority of the failures. Figure 2.4 shows the confusion matrix for the runs of the random forest

models using samples based off of 2010 Census demographics. The darker the blue color in the square of the matrix, the higher the number of users in the testing set that were classified according to the label on the bottom, but actually belong to the category on the left. Ideally, the square in each row and each column, respectively, with the deepest blue color would be the square along the upper-left to lower-right diagonal. Each box also contains the number of users who were classified as such in the testing set. From the run of the census weight based

model that produced this confusion matrix, only *eleven* of over 2,000 users were classified as non-white.

Since over 70% of the incoming training data was white, the model was simply coding every single user as white. In an attempt to avoid algorithmic bias creeping into my race models by using an accurate demographic makeup in the underlying training data, I in fact did not give the model diverse enough data to train upon. The remedy to this problem was to oversample from the racial minorities in the comScore dataset. I proceeded to use the same process to create another random forest model, but this time with a smaller portion of white individuals in the training set, and more individuals who were black, Asian, or

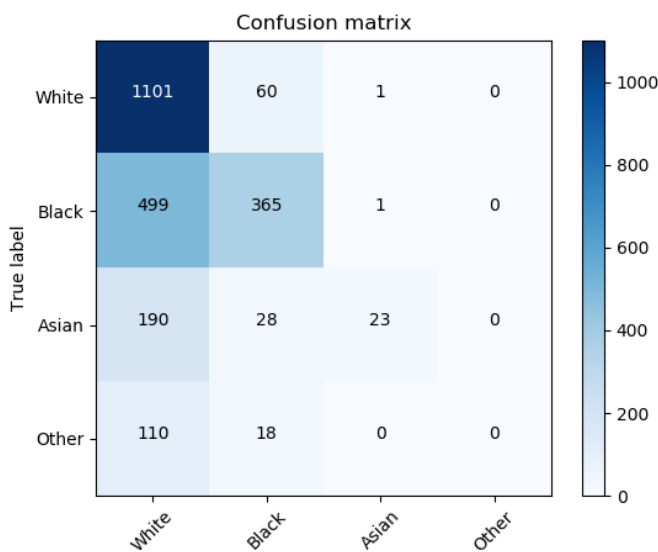


Figure 2.5: Minority Oversample Confusion Matrix

labeled as “Other.” The demographic weights for the new model were: 50% white, 35% black, 10% Asian, and 5% other. The classification accuracy for the newer version of the model dropped

about 10 percentage points, to ~64% overall accuracy, but once again, the confusion matrix tells a different story than the overall classification accuracy of the model. This time the story was slightly *more* encouraging. The confusion matrix for the minority-oversample model, in Figure 2.5, demonstrates that

oversampling minorities leads to better classification accuracies within minority groups. Though exactly zero individuals categorized as “Other” were correctly classified by the model, some Asian individuals were, and nearly half of black individuals in the testing set were correctly classified. Oversampling minority individuals did not entirely solve the problem of differential classification accuracies, but it did help

significantly. To see how far this phenomenon would go, I ran a model using only black users and white users, with 50% of the sample being made up of each. The confusion matrix can be found in Figure 2.6 (only a quarter of the matrix is full

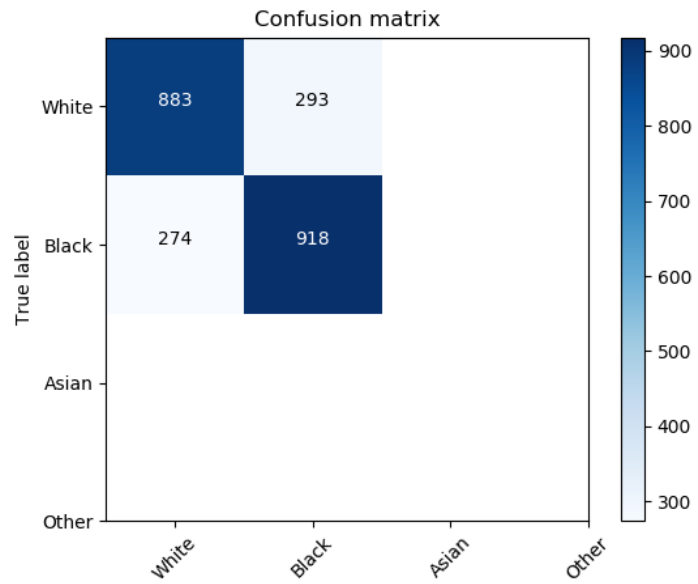


Figure 2.6: Black-White Only Confusion Matrix

because there were no Asian or “Other” individuals in the sample), and the results are extremely encouraging. The black-white only model achieved a classification accuracy of just north of 76%, with virtually equivalent classification accuracies within racial groups. The accuracy for white individuals was just above 75% and the accuracy for black individuals was just above 77%. Given the fact that there were now only two classes in the model, this corresponds to roughly matching specificity and sensitivity values. The fact that the in-group classification accuracies were close to equivalent also indicated that the model was not

inherently biased against either group. Evidently, sampling equivalently from each racial category actually created a model that is was close to unbiased as possible.

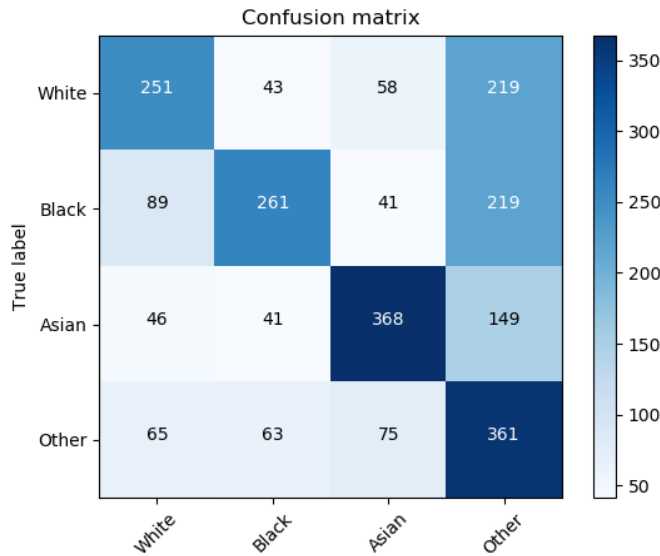


Figure 2.7: Confusion Matrix for Even Racial Sampling

Next, I tested the preceding assumption by creating a model that sampled evenly from each of the four racial categories are included in the comScore dataset,

with 25% of the sample each from white users, black users, Asian users, and users labeled “Other”. Unfortunately, the idea of sampling equally from each individual group did not pan out quite as effectively as I had hoped. The overall classification accuracy of the model dropped to just over 50%, and though the plurality of each individual subgroup was labeled correctly by the model, the model labeled far too many of each category as “other.” As a result, the classification accuracies for both white and black individuals was below 50%, with only the accuracies for Asian and “Other” individuals ticking above even 60%. The confusion matrix for this model, in Figure 2.7, clearly demonstrates its unsuitability for the task of predicting racial background from browsing history.

I was thus faced with the following problem: I could accurately predict whether a user is black or white, but only if no other racial categories were involved. To be able to accurately model all of my data, I decided to tweak the racial coding, at the cost of a bit of precision in the final results. Instead of coding individuals as white, black, Asian, or “Other,” I combined the final three categories, instead coding individuals as either white or nonwhite. Though I lose the ability to code individuals as explicitly black, Asian, or “Other,” I decided this was a relatively small price to pay if it made the model significantly more accurate. According to the Roper Center, even though other nonwhite voters did not support the Democratic candidate in 2016 by the same overwhelming margin as strictly black voters did, both Asians and people falling under the “Other” category supported Secretary Clinton over Donald Trump by at least 20 point margins, and made up less than a third of the portion of the electorate made up by black voters.⁸⁸ I therefore believed that there would be little difference in the effectiveness of each classification, and that difference could be made up by a better racial classification algorithm.⁸⁹ Upon running the model, I was proven correct: the classification of the white vs. nonwhite model jumped back into the 70s – hovering around 72-73% for successive runs – and the model had roughly similar accuracies for both white and nonwhite individuals. The confusion matrix can be seen in Figure 2.8. As a sanity check of the original model parameters, I

⁸⁸ How Groups Voted in 2012.” Roper Center. Accessed March 18, 2019. <https://ropercenter.cornell.edu/how-groups-voted-2016>

⁸⁹ For further discussion, see the Ensemble Modeling section in Chapter 3.

doubled the number of trees in the random forest, and also doubled the maximum depth of the trees, to see if the classification accuracy increased.

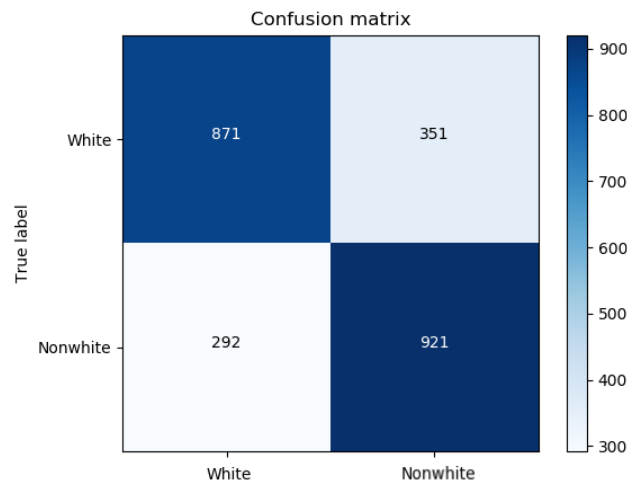


Figure 2.8: White vs. Nonwhite Confusion Matrix

The increases could be described at the absolute best as “marginal,” and so I settled on the original settings, as they took less time to run and train.

Partisanship Modeling with Random Forests

In order to model the partisanship of individuals using random forests, or really any type of model, I needed to somehow impute that information into the comScore dataset’s set of demographics. As mentioned at the outset of this chapter, the quantity I elected to use to measure partisanship was party registration – a piece of information readily available for residents of certain states through the state’s voter file. North Carolina has its voter file freely available online, and explicitly reports party registration, making it the ideal state for analysis. The demographics available to me in the comScore dataset included both age and race, which are also reported for individuals in the North Carolina voter file, allowing me to match users from North Carolina in the comScore dataset back to sets of individuals in the voter file. Then, I recorded four

quantities and added them to the set of recorded demographics for North Carolina individuals in the comScore dataset: vf_k , D_pct , vf_k_2p , and D_pct_2p . The quantity vf_k is the number of individuals in the North Carolina voter file who have the same ZIP code, race, and age as each user in the comScore dataset, and D_pct is the percentage of those users who are registered members of the Democratic party. The quantity vf_k_2p is the number of individuals in the voter file with matching ZIP code, race, and age as each user, but only those who are registered as either Republican or Democratic party members, and D_pct_2p is the percentage of those individuals who are registered as Democrats.⁹⁰ The latter two quantities are the same as the former pair but calculated only amongst the two major parties in American politics, throwing out individuals in the voter file who identify as “Unaffiliated.” Both D_pct and D_pct_2p thus provide a measure of the partisanship of the individual in question: how likely they are to be a Democrat given their ZIP code, age and race. Importantly, D_pct_2p could easily have a sister statistic $R_pct_2p = 1 - D_pct_2p$, in which the operative term is the percentage of individuals who were Republicans instead of Democrats. The code for imputing these quantities into the comScore dataset can be found in Appendix 3.

The 2016 comScore dataset contained over 2,500 individuals from the state of North Carolina, making the North Carolina specific dataset small enough to be able to transform into the usable matrix without having to use subsamples. allowing me to avoid some of the problems I had encountered when modeling

⁹⁰ Note that the following must be true: $vf_k \geq vf_k_2p$, $D_pct_2p \geq D_pct$.

race. Once I imputed the quantities mentioned above, I noticed that many of the pieces of information tracked well with what one would assume about data.

Figure 2.9 shows the D_pct_2p values, separated by race, with the size of each bubble

representing the number of users with identical values for the category. As one would expect, we see a slight rightward (politically) skew among white

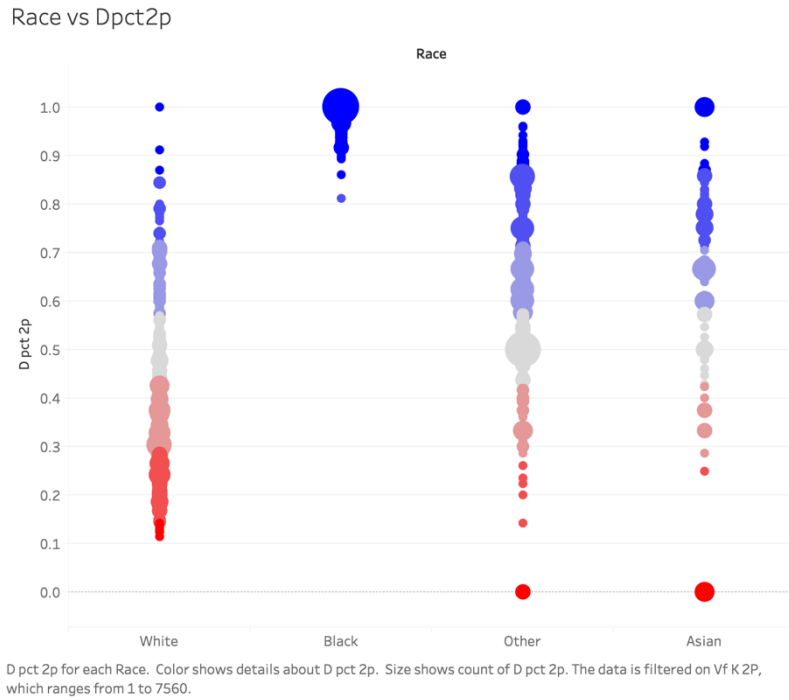


Figure 2.9: Race vs D_pct_2p

individuals – these people have lower values of D_pct_2p , or equivalently would have higher values of R_pct_2p – and a very heavy leftward skew among black individuals. As discussed in the race modeling section, we still see a leftward shift of individuals in the Asian and “Other” categories, but not nearly so distinct as the shift amongst black individuals. Graphing D_pct_2p in similar manner against the age ranges from comScore, seen in the left half of Figure 2.10, again produces results that track well with what one would expect from the general electorate.

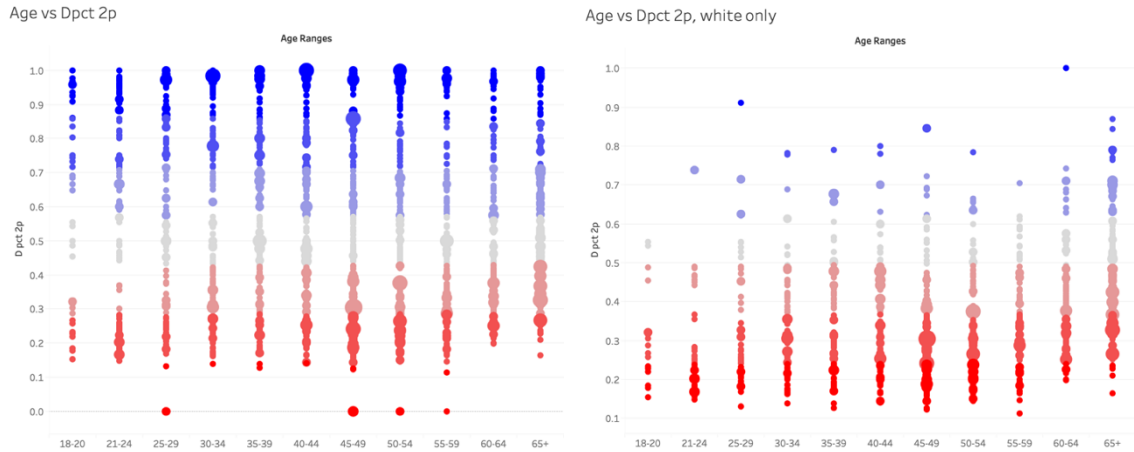


Figure 2.10: Age vs D_pct_2p , all races (left) and white only (right)

There is a significant leftward tilt to the D_pct_2p values for individuals in the 18-20 range and the 21-24 range. Gradually, however, this tilt disappears as individuals get older, with the race base effects from Figure 2.9 perhaps being the only thing keeping the tilt from moving decidedly rightward. Indeed, if one restricts the graph in the left half of Figure 2.10 to only white users from the comScore dataset, the entire distribution shifts significantly to the right, as one would expect. This effect erases any Democratic Party registration advantages among young voters, and lends even more credence to the idea that race can be an extremely powerful factor in predicting partisanship. The importance of the interaction of race and age can be seen by comparing the left and right halves of Figure 2.10.

With the necessary information now imputed into the 2016 comScore North Carolina subset, I could set about building a model of partisanship. Applying the same random forest modeling procedure that I used when modeling race, I wrote a program that built models for partisanship. As arguments, the program takes a comScore sessions file, a comScore demographics file – which

necessarily must already have the four new demographic categories imputed – and two flag arguments, the first telling the program whether to use the two party version of the imputed demographics, and the second telling the program at what threshold to code a user as a Democrat. The latter is extremely important, as it tells the program how to turn the imputed D_pct and D_pct_2p values into a new column, called “Democrat”. The new column was binary: 1 if the user is coded as a Democrat for that version of the model, and 0 otherwise. The program thus first looks at D_pct , or D_pct_2p if the two party flag is set, and codes any rows in the matrix with a D_pct value above the entered threshold as a 1 in the “Democrat” column, and a 0 otherwise. This granted me the flexibility to test against Hersh’s 80% threshold, and also the weaker 50% threshold, in which a user is simply coded as being more likely to be a Democrat than not.

I began by using the program’s defaults: setting the two party flag as false, and using Hersh’s 80% threshold. Thus, a user was coded as a Democrat if and only if that user’s D_pct value was above .8. In other words, this model coded an individual from the comScore dataset as a Democrat if and only if more than 80%

of people
that

who match
individual’s

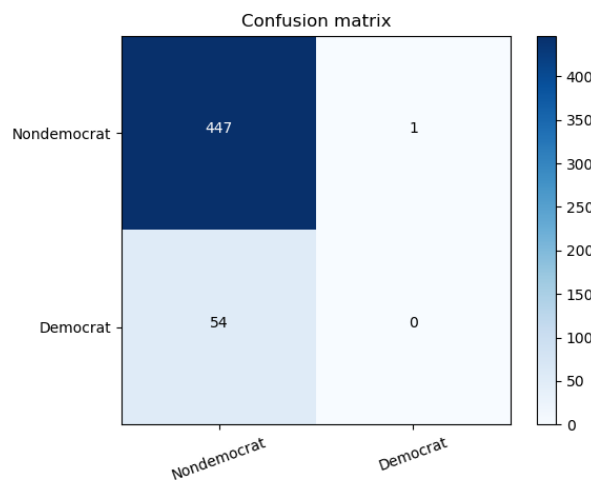


Figure 2.11: $D_pct > 0.8$

demographics in the North Carolina voter file were registered as Democrats. Due to the smaller number of users in the North Carolina subset, I used all the individuals from North Carolina in the 2016 comScore dataset, and did no sampling, unlike with the race models. The confusion matrix, with categories “Democrat” and “Nondemocrat,” can be seen in Figure 2.11, on the previous page. The first thing one might note is that the model achieves a fairly high degree of classification accuracy, as it classifies almost 88% of individuals correctly. However, the model is simply classifying every single user as a nondemocrat, much like how my initial race models coded almost every single user as White. The issue is almost identical: once coding is complete, there are simply not enough users who are labeled as Democrats for the model to be able to effectively train itself to classify both categories. I then decided to lower the threshold to .5 – the 50% standard described above. This helped alleviate the issue, as more individuals would be coded as Democrats, giving the model a more balanced training set. The program now coded any individual as a Democrat if their *D_pct* value was greater than 0.5, according to demographic matches in the North Carolina voter file, or, in plain English, if they were more likely to be a registered Democrat than not. The confusion matrix for this model can be seen in Figure 2.12, and though the model now began to code individuals as Democrats, the original problem still persisted, without a solution. The overall classification accuracy dropped, due to the fact that there were more Democrats after coding,

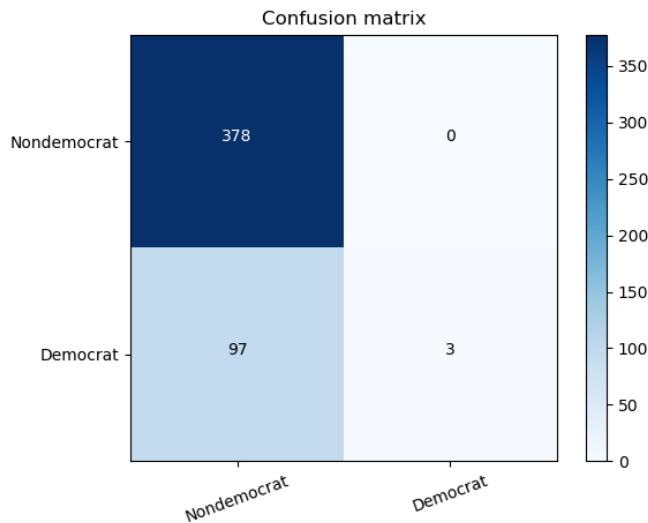


Figure 2.12: $D_pct > 0.5$

and thus more individuals to misclassify. Though there were more coded Democrats upon lowering the threshold, the subset of people coded as Democrats was still only roughly one quarter the size of the subset coded as nondemocrats. Even with the weakest standard that still gives usable information, setting the two party flag to false gave a virtually useless model.

Given my inability to make meaningful predictions using the imputed statistics the included more than just the two major American parties, I began attempting to model the electorate in North Carolina by only looking at voters who were registered to either the Democratic or Republican party. This necessarily excludes individuals in the voter file who were registered Independents, registered members of smaller parties like the Green and Libertarian parties, and individuals who are not registered to any party. In North Carolina specifically, voters without a party registration and independents are all lumped together in a category called “Unaffiliated.” I then repeated the process that I had run with the D_pct based models, this time using models that made Democrat vs. nondemocrat classifications using the imputed D_pct_2p column

instead. Using the two party column, the plain English equivalency is to code as a Democrat any user for whom 80% of the people in the North Carolina voter file with matching demographics *who were registered as a Democrat or a Republican* were registered Democrats. Since by definition, for any individual user, $vf_k \geq vf_{k_2p}$, which, due to the registered number of Democrats not changing, in turn implies $D_pct \leq D_pct_2p$, switching to setting the two party flag to true helps the issue of not enough of the sample being coded as Democrats. We would expect in the vicinity of half of the sample to be coded as Democrats with the threshold

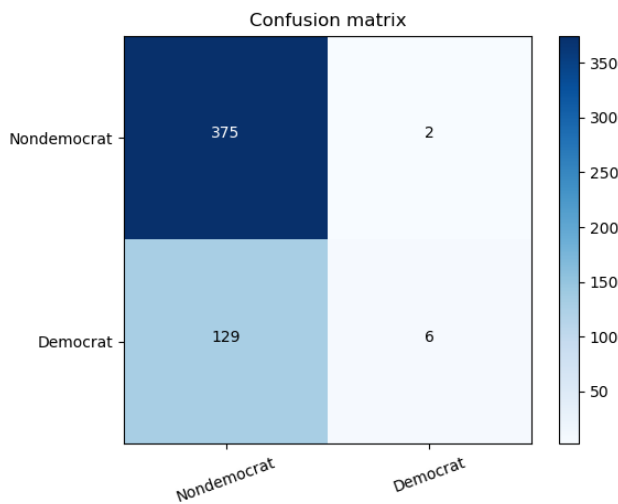


Figure 2.13: $D_pct_2p \geq 0.8$

around 0.5. Again first attempting to reach Hersh's 80% standard, I first set the threshold for Democrat classification at 0.8, of course setting the two party flag to true, coding any user who had

$D_pct_2p \geq 0.8$ as a Democrat, but still, the original problems persisted, though again they were slightly improved. As can be seen in the confusion matrix in Figure 2.13, the random forest model still stubbornly coded almost every user as a nondemocrat. Importantly, this time, about 20% of the sample was coded as a Democrat, compared to about 10% when the threshold was set at 0.8 but the two party flag was set to false. This actually again dropped the overall classification accuracy from the $D_pct > 0.8$ model, to 75%. Still, no important inferences

about an individual can be drawn by running their browsing history through this model.

I then ran a model with the 50% standard; the classification threshold set to 0.5 and the two party flag set to true would indicate simply whether someone is more likely to be a

Democrat or a Republican
– not of small importance
in the political sphere. The
overall classification
accuracy came in at just
over 65%, classifying 47%
of Democrats correctly,

and 82% of

nondemocrats correctly. Though this represents a marked improvement, the old issues still remain. The confusion matrix for this model can be found in Figure 2.14. For the sake of curiosity, I then ran a model with the same parameters, but allowing race, which I showed can be predicted from random forest rather effectively, as a predictor. This made a world of difference. The model now classified 71% of Democrats correctly, and 89% of nondemocrats correctly, giving a slight bias to classify an individual as a nondemocrat, with an over 80% overall classification accuracy. This bias is likely more useful for an industrial setting than the opposite tendency, as the 50% standard is much weaker than Hersh’s 80% standard. Though random forests seem rather unable to predict

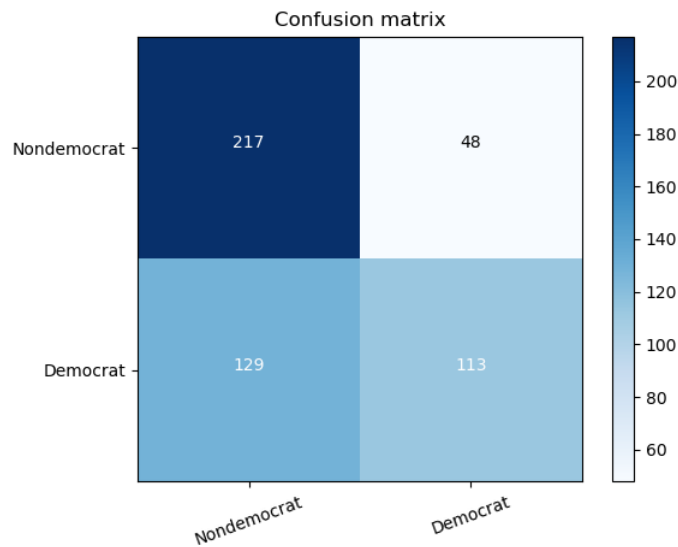


Figure 2.14: $D_pct_2p \geq 0.5$

partisanship directly, using race as a stepping stone appears to have potential as a way to increase the effectiveness of this modeling paradigm.

Using the `sklearn` `RandomForestClassifier` implementation, I was able to pull out the most influential websites to the forest's classifications. The 10 most influential sites for the above model are reproduced in Table 2.1. These can be thought to outline the partisan divides of the Internet, though the `RandomForestClassifier` object does not specify in which way they are influential.

That is, unlike the next class of model I use, the sites in Table 2.1 do not have an explicit side of the divide associated with them.

Additionally, the dimensionality of the dataset is very high compared to the depth and number of trees. Indeed, this is why random forests are a good choice in modeling, but it means that

the influence ratings are liable to change in any given run of the forest, and this table

only applies to one modeling run. In the end random forests were unable to definitively identify Democrats from their browsing histories, though they offer some predictive power, even if at a weaker standard than I might like.

Exclusivity Indices

Random forests in the end turned into a somewhat effective model for predicting both partisanship and race, though with severe limitations for each outcome variable. In trying to hone my predictions even further, and make them

Most Influential Sites	
Two-party, $D_pct > 0.5$	
1	google.com
2	lowes.com
3	msn.com
4	ncdot.gov
5	youtube.com
6	soundcloud.com
7	pinterest.com
8	brassring.com
9	taleo.net
10	etsy.com

Table 2.1: Sites by Influence

potentially more robust, I turned to work done by Latanya Sweeney with Jinyan Zang, using the same comScore data source, and making distinct measurements about how exclusively a website is visited by one category of a demographic axis. Sweeney and Zang define the *exclusivity index* of a website i for a specific demographic group j , in the demographic axis J , to be:

$$EI_{ik} = \frac{\frac{v_{ik}}{m_k}}{\sum_{j \in J} \frac{v_{ij}}{m_j}}$$

where v_{ik} is the number of users in group k who visit website i , and m_k is the total number of machines, or users, in group k .⁹¹ Essentially, the exclusivity index for a website for a group is the proportion of users in a group who visit a site, divided by the sum of all the proportions for groups on the same demographic axis. This allows easy comparisons between indices – a higher exclusivity index for a site for one group, say white individuals compared to Asian individuals, means that that site is more likely to be visited by a given white individual than by any given Asian individual. It should be clear why exclusivity indices may offer predictive power for demographics, including partisanship; the metric is quite literally a measure of the differential likelihood of different groups to visit a website.

The Sweeney-Zang definition of exclusivity indices came very close to suiting my purposes, but does not quite meet the specific needs my analysis has. I was not interested in simply seeing which websites are most exclusively visited by different groups, but in then using that information to make predictions about

⁹¹ Latanya Sweeney. “Online ads roll the dice.” Tech@FTC (blog). September 25, 2014. <https://www.ftc.gov/news-events/blogs/techftc/2014/09/online-ads-roll-dice>

different individuals. As such, I needed to make a slight tweak to the exclusivity index formula. The Sweeney-Zang definition allows a high exclusivity index for a website with a low number of overall visits. Picture a website that is visited by literally one person, and imagine that person is white. Due to the denominator in the fraction scaling the result by the total number of visits, the exclusivity index for that site is 1 for the group of white people, and 0 for all other groups. Though the site is literally exclusively visited by white individuals, its presence in a browsing history offers no predictive power whatsoever, as it is only present for one individual in the group. Thus, I created a slightly modified definition of the exclusivity index:

$$EI_{ik} = |V_i| \frac{\frac{v_{ik}}{m_k}}{\sum_{j \in J} \frac{v_{ij}}{m_j}}$$

In this modified version, each exclusivity index is scaled by the number of users that visit the site in question: $|V_i|$.

We can now define the *exclusivity index classifier criterion*:

For a user x with browsing history B_x on a given demographic axis J , with N_j being the n most exclusive websites for the group $j \in J$, classify x as:

$$\operatorname{argmax}_{\forall j \in J} (|N_j \cap B_x|)$$

Next, I wrote a program that, given a transformed matrix of comScore browsing histories as described above, calculates the exclusivity indices for a given demographic axis, and builds each set N_j . Initially, this process took an incredibly long amount of computer time, but by using Boolean-integer casting, I

was able to get my program to calculate the modified exclusivity indices for a 10,000 user set in about 25 minutes. I was then confronted with another problem: my new metric was not quite right either. Websites with very high numbers of visitors tended to bubble to the top, regardless of the exclusivity indices of the site for each group in the demographic axis. I modified my program to use the following algorithm:

- 1) Calculate the Zang-Sweeney exclusivity index for each website in the sample's training set, for each group in the demographic axis.
- 2) Define a threshold t and drop and for each group j in the demographic axis:
 - a. Drop any site where $EI_j < t$
 - b. Sort the remaining sites by their modified exclusivity index
 - c. Take the top n sites as the n most exclusive in the sample, and for each group j call it N_j
- 3) Classify users in the testing set according to the presence of the top n sites from each group their browsing history, using the exclusivity index classifier criterion.

For the remainder of this thesis, any time I refer to an “exclusivity index,” I refer to the quantity calculated in the above manner. The exclusivity index classifier family of models can be seen as coloring the Internet, either red or blue, by finding the domains visited most exclusively by members of different political groups. If the Internet is a map, these models mark out territories for each party, and classify users by determining which territories their browsing histories fall into.

Race Classification with Exclusivity Indices

Using the algorithm outlined above, I began by modeling race using exclusivity indices. First, I pulled 10 different samples of 10,000 users from the 2016 comScore dataset, and calculated the exclusivity indices for each domain in

the sample, storing the top 100 as detailed in step 2 of the algorithm above.⁹² I then took the 100 most frequently occurring domains for each racial group, and wrote them to a file, in order to not have to go through the time consuming process of calculating exclusivity indices for each individual sample. Though this does increase the danger of test set/training set contamination, it also allowed me to bootstrap exclusivity indices for larger sample sizes than just the about 10,000 that I could sample given my computational resources. I did this setting the threshold value – t in the above algorithm – to 0.7, 0.8, and 0.9.

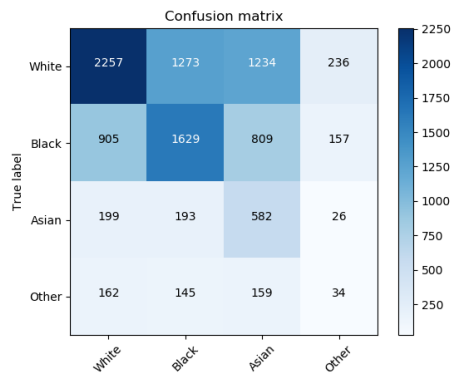


Figure 2.16: Early Exclusivity Index Model

After storing the bootstrapped exclusivity indices in files in order to speed up subsequent runs of the exclusivity classifier models, I began by running a sample of 10,000 users⁹³ through the modeling

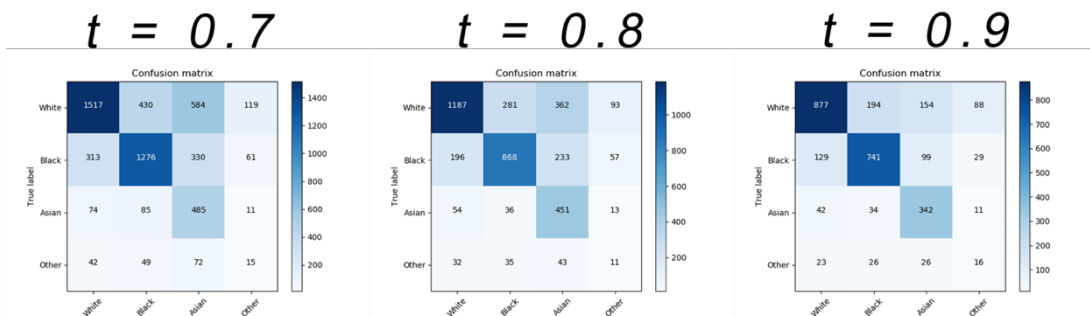


Figure 2.15: Confusion matrices among classified users, varying t

⁹² Code referenced in this section can be found in Appendix 5.

⁹³ Importantly, because the exclusivity indices were already calculated, I did not need to worry about the demographic sampling for these models, and used the 50%-35%-10%-5% weights.

t	Portion not classified	Classification accuracy
0.7	45.37%	65.2%
0.8	60.48%	63.7%
0.9	71.69%	69.8%

Table 2.2: Classification Details

process, using the exclusivity index classifier criterion. I quickly realized I had a problem. The confusion matrix for one of my early model runs is produced in Figure 2.15, and one will quickly notice the high degree of misclassification into the “White” category. I believed this was due to a quirk in the code, that categorized every user who did not visit any of the websites in any of the sets of most visited websites to the lowest comScore code value. As such, I edited the models to only report the results for individuals who had visited any of the sites reported in the sets of the 100 most visited sites. The resultant confusion matrices are in Figure 2.16, on the previous page. Two things immediately jump out: first, the misclassification to “White” drops off after making the small change, and second, the higher the threshold, the more accurate the classifier, but the fewer get classified. Table 2.2 shows the number of people who did not get classified for each value of t , as well as the overall classification accuracy for each of the models. The drop in the number of users who were classified makes sense, as the lower the threshold, the more commonly visited the websites will be, given the way that the exclusivity indices are used to filter out sites with low values on the Sweeney-Zang index. The biggest gain in accuracy is seen between $t = 0.8$ and $t = 0.9$, though many fewer people are classified. Clearly, there is a tradeoff; a

model based on exclusivity indices can accurately categorize some users, but the more accurate the model needs to be, the fewer users it can classify.

After running the above models, I decided to make a small tweak to the exclusivity index classifier criterion. The *modified exclusivity index classifier criterion* takes into account where in the ordered list of exclusivity indices each visited website sits for a demographic group, instead of just the number of sites in a browsing history present in the list. Notated as before, the modified criterion is as follows:

$$\operatorname{argmax}_{\forall j \in J} \left[\sum_{i=1}^n (\mathbf{N}_{ji} \in \mathbf{B}_x)(n + 1 - i) \right]$$

In the formula, the term $(\mathbf{N}_{ji} \in \mathbf{B}_x)$ as a Boolean value: true or false. The term takes “true,” or 1, if the i th site in group j ’s top n sites is in user x ’s browsing history, taking “false,” or 0 otherwise. This criterion differentially weights websites according to their relative exclusivity in each group’s set of exclusive

t	Portion not classified	Classification accuracy
0.7	45.82%	55.6%
0.8	59.82%	58.6%
0.9	71.85%	67.3%

Table 2.3: Classification Details, Modified Criterion

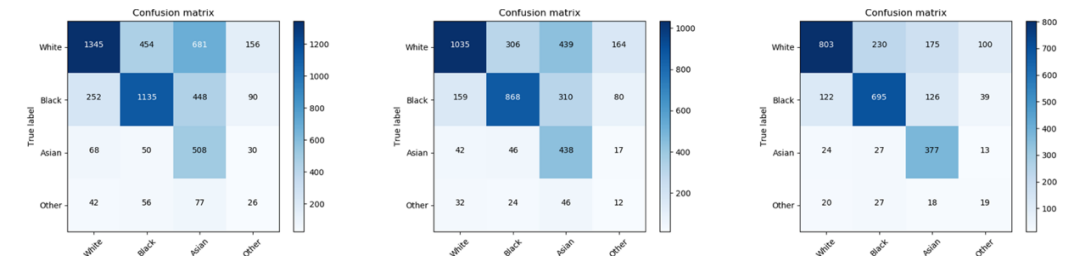


Figure 2.17: Confusion Matrices for Modified Criterion

domains. The sets N themselves do not change for the modified criterion, but merely the way in which their presence in a browsing history is weighted. Imagine a browsing history that contained the single most exclusive website visited by black individuals, and then two websites visited by white people, but in positions 99 and 100 respectively. The original criterion would classify the user as white, while the modified criterion will classify the user as black. I then ran models with the modified classifier, and found that it was actually slightly less effective than the original criterion. The results are summarized in Table 2.3 and Figure 2.17, above. As expected, the overall number of users classified did not change very much, though they do change slightly, reflecting the random nature of sampling. The decrease in overall classification accuracy on each of the three different t values indicates that a model for race should use the original classification criterion.

Modeling Partisanship with Exclusivity Indices

After editing my exclusivity index modeling code so that it could both impute and model partisanship, using the same paradigms as the random forest models did, I began modeling partisanship, once again via partisan identification, using the exclusivity index classifiers. One major difference between the random forest models and the exclusivity index models is that the latter did not require using race as a stepping stone to get to the final outcome of partisanship effectively. Using the 2016 North Carolina user subset from the comScore data, I ran models in which the threshold value t for building exclusivity indices was in the set $\{0.7, 0.8, 0.9\}$, the democrat vs. nondemocrat classification threshold was

set at either 0.5 or 0.8, and with the two party flag set to both true and false. I also ran each model using the modified exclusivity index classifier criterion. Thus, I had 24 different models to run – their results can be found below in Table 2.4.

		Original Criterion			Modified Criterion		
		0.7	0.8	0.9	0.7	0.8	0.9
D_pct	$t > 0.8$	74.6%	79.4%	82.4%	69.8%	71.3%	82.1%
	$t > 0.5$	71.4%	75.5%	76.8%	67.7%	72.7%	78.3%
D_pct_2p	$t > 0.8$	67.2%	72.3%	74.3%	74.0%	75.1%	77.8%
	$t > 0.5$	66.2%	68.0%	66.1%	65.5%	71.4%	63.4%

Table 2.4: Classification Accuracies for Partisan Identification

Though as in the race modeling, increasing the value of t causes fewer people be classified by the algorithm, the above table makes clear that increasing the value of t can cause very large increases in the accuracy of those users who are classified. Interestingly, this depends on the criterion used, and whether or not the two-party system is in effect for a model. The Modified Criterion models show larger gains than the Original Criterion models. One can interpret the better accuracies for higher values of t as the algorithm being more “sure” of its classifications at higher values of t . Extremely importantly, though, is that the exclusivity index classifiers succeed where the random forest classifiers failed: at the Hersh 80% standard, and in the non-two party system. In fact, the original criterion classified over 82% of individuals at the $t = 0.9$ level of exclusivity. Even though there are still far more nondemocrats than Democrats when the coding threshold is set to 0.8, the exclusivity index classifier correctly classifies more of the few Democrats than the random forest classifiers. Another key

difference when comparing the exclusivity index classifiers to the random forest classifiers is that the exclusivity index classifiers actually perform *better* at the Hersh 80% standard, rather than the lower 50% standard. This points to potentially different use cases for the models in the professional and academic worlds: exclusivity index classifiers appear to be better at identifying individuals who are *very* likely to be Democrats, while random forests are better at identifying whether an individual is more likely to align with one party or another.

Looking at overall trends in Table 2.4, the $t = 0.7$ level of exclusivity does not have a very high level of accuracy. Though both $t = 0.8$ and $t = 0.9$ have fairly high degrees of accuracy, $t = 0.9$ is higher, though at the cost of some classifications; in the realm of 30-50 fewer people are classified when t is 0.9 than when it is 0.8. As such, and this is now a normative judgment, I imagine individuals in the political arena would find the $t = 0.8$ set of models most useful, as they make more classifications, and still maintain a relatively high degree of overall accuracy. Equally importantly, however, is the fact that the $t = 0.9$ models are still making predictions for over half of the users in the testing set. For the race models, only about 30% of the users were classified at the highest levels of the exclusivity threshold. This represents a significant improvement when modeling along party lines when compared to modeling racial groups – almost twice as many users are classified in a party model than in a race model, at the same level of the exclusivity threshold. Additionally, for party modeling, the modified criterion actually performs slightly better than the original criterion, a strict reversal from the race modeling results. Importantly, these models were able

to achieve high levels of accuracy *without using race*, which was required in the random forest models to even get close to these numbers.

The Partisan Internet

Taken as a whole, the results portrayed in Table 2.4 present a picture of the Internet that is divided upon party lines, in ways that it is not divided upon racial lines. The high degree of accuracy, and the ability to still make predictions about the majority of users at high levels of exclusivity, point to an Internet that is sharply divided along partisan lines. The decreased ability of lower exclusivity values to predict the partisanship of computer users points to the same phenomenon; if the users can be predicted by highly exclusive domains, but that predictive ability is drowned out when the exclusivity is lowered, and thus more commonly visited sites are used, then the Internet appears divided. Furthermore, the fact that so many users can be classified by domains with high exclusivity indices indicates that most users are indeed visiting those highly exclusive domains. The race models in this chapter demonstrate that about 70% of individuals do not visit the 100 sites that are the most heavily exclusive for different racial groups, but for party models and partisan identification based groups, that number drops to 35%. The 10 most exclusive domains for Democrats and nondemocrats, at the $t = 0.9$, and with the Hersh 80% standard as the coding threshold value in the two party system are printed below in Table 2.5. The websites in Table 2.5 can be looked at in a similar light to the sites in Table 2.1, which were the 10 most important features to a random forest model. That being said, I would consider these explicitly calculated exclusivity indices in Table 2.5

to be a more accurate description of the partisan divide of the Internet in North Carolina, due to the number of trees in the random forest being relatively small in comparison to the extremely high dimensionality of the data, with each website as its own predictor. One will notice that the websites below look very different from the random forest’s most influential features, which contained many websites that are visited by nearly everyone. This phenomenon was likely caused by the way feature importance was calculated in the random forest models, which necessarily takes the magnitude of the predictor into account. Since the websites listed below are calculated based on a binary “visited vs. did not visit” paradigm, they should be a better sketch of the true partisan layout of the Internet. Indeed, Table 2.5 shows results that seem to fit neatly into a preconceived idea of American partisanship: a faith centric website and a “fake news” website are the two most exclusive sites for nondemocrats, and a hip-hop site, likely a proxy for race, is one of the most exclusive for Democrats.

	nondemocrat	Democrat
1	faithtap.com	jimmyjazz.com
2	madworldnews.com	tagged.com
3	newsbake.com	worldstaruncut.com
4	potterybarnkids.com	searchincognito.online
5	itsthevibe.com	fashionnova.com
6	hickoryrecord.com	google.co.in
7	koa.com	einthusan.com
8	twinkledeals.com	vizury.com
9	mycokerewards.com	trkrwiz.com
10	charlottemotorspeedway.com	cartoonhd.website

Table 2.5: Top Exclusivity Indices by Party

The fact that random forests based on browsing history can be used to rather effectively predict race, and exclusivity indices can be used to predict partisan identification indicates that browsing histories can be a powerful predictor to the political affiliations of an individual. That said, these results are just for North Carolina; in the next chapter, I extend my analysis to another state: Florida.

Chapter 3

The obvious issue with the results presented in the preceding chapter is that they only pertain to one state. If the goal of this project was simply to make inferences about the partisanship and race of North Carolinians from their browsing history, then one could consider the job done. The goal, however, was and remains to predict the partisanship and race of Americans writ large from their browsing histories. I selected North Carolina as the first state to analyze due to the construction and availability of its statewide voter file, which contains explicit partisan identification, as well as ZIP code, race, and age. The latter three pieces of information allowed me to partially re-identify the comScore dataset, and then use the party identification information to create a measure of how likely an individual was to be a Democrat. To expand my analysis past the state of North Carolina, I needed a state with a similarly structured voter file. Luckily, one exists: Florida. Though the Florida statewide voter file is not available freely online, it can be requested via a disk from the Florida Secretary of State's office.

Creating predictive models on the browsing histories of Floridians, in addition to those pertaining to North Carolina that I created in Chapter 2, offers a myriad of benefits, as well as the potential to drastically increase the robustness of the entire project. Obviously creating models of equivalent accuracies for the state of Florida would demonstrate that the abilities of random forest and exclusivity index classifiers to predict quantities of interest from browsing histories extend beyond the borders of just one state. Just as importantly, adding Florida models to the proverbial equation opens up the possibility of testing one state's models

against the users from another state. Should models trained on North Carolinian browsing histories be comparably effective at predicting the partisanship of Floridians, or vice versa, it would indicate that one may not need the same level of re-identification for every single state in the nation. Such a scenario would be a boon to researchers and political operatives attempting to use the procedures from the previous chapter to predict quantities of interest about voters, as not every state publicly reports a voter file, and not every voter file contains the same level of detail as the North Carolina and Florida files. Though one could likely piece together similar information from other sources, such as labeling partisans by the primaries they voted in, doing so would introduce additional degrees of uncertainty into an already fairly uncertain set of models.

This final chapter proceeds as follows: first, I describe the Florida dataset, and the process of extracting and imputing the necessary statistics. I then model partisanship for the Florida users in the comScore dataset from 2016 as I did for those from North Carolina, with both random forests and exclusivity index classifiers. Next, I test the North Carolina models on Floridians, the Florida models on North Carolinians, and finally test a combination model, trained on the North Carolina *and* Florida subsets of the comScore data. After discussing the implications of the results, I turn to a discussion of two different ensemble modeling methods using the many different models that have been created for this project, attempting to derive a way to use all the models – random forest and exclusivity index, race and partisan identification – to best predict the likelihood

of a given individual to be a Democrat. Finally, I offer my concluding thoughts on the project as a whole.

The Florida Data

I began, as I did with the North Carolina data, by first extracting the list of users from the 2016 comScore dataset that lived in the state of Florida, identified by their ZIP code. I chose to focus on the 2016 data once again because it is the most immediate to the voter file that I have access to for the state of Florida, and because there was a major election that year. The 2016 comScore dataset is also by far the largest of the three years that I have access to, and so contains the largest training set for any model based on only one year. The extracted Florida dataset contains, once rows containing missing values in key columns were removed,⁹⁴ upwards of 6,000 individuals, more than twice the number of users in the North Carolina subset.⁹⁵

I proceed by beginning to fit partisanship models to the Florida data. The race model that I fit in Chapter 2 was national, and fit without using any partisanship data whatsoever, and so it makes little sense to refit; the Florida voter file would not add any information that could be incorporated into the analysis.

As with the North Carolina models, the first step in this process was to use the

⁹⁴ To do the matching to partisan identification with the voter file, I required a value in the ZIP code, race, and age columns. Thus, I dropped any row from the dataset that did not contain the requisite pieces of demographic information. Another paradigm would be to match on only the available pieces of information, but this would have led to vf_k values for those rows that would have been outliers from the rest of the users’.

⁹⁵ The North Carolina subset of the 2016 comScore data contained just over 2,500 individuals.

voter file to partially re-identify the users in the comScore dataset from the demographic information that the company collected. I used an identical process to impute the same four new categories into the demographics from the voter file: vf_k , D_pct , vf_k_2p , and D_pct_2p . These quantities were calculated in the same manner as they were for the North Carolina subset, and carry the same meanings: vf_k is the number of individuals in the requisite voter file with the same demographic characteristics as the comScore user, D_pct is the portion of those who are registered Democrats, and vf_k_2p and D_pct_2p are the same, respectively, but when the analysis is restricted to only registered Democrats and Republicans, thus ignoring independents and unaffiliated individuals. Indeed, the only thing that changed in the code that did the partial re-identification was the names of the columns pertaining to the important pieces of information, and some of the codes that were matched on.

After imputing the new partisanship columns, I recreated the same graphs from Chapter 2, this time with the Florida demographics instead of those from North Carolina. Immediately obvious was that the vf_k values were comparable to those in North Carolina, once the rows with missing demographic values were dropped. These values were perhaps slightly more right skewed, but not enough to worry about a breakdown of the modeling process. Once again, the data tracked with what one would expect. As can be seen in Figure 3.1, Black members of the dataset skewed overwhelmingly in the direction of the Democratic party, while white members of the dataset display a slight Republican lean, although it appears

to be a smaller shift than that in North Carolina (see Figure 2.9). One worrying thing about the data in the Florida subset is that many more people are labeled as “Other” in the racial background column when compared to the North Carolina

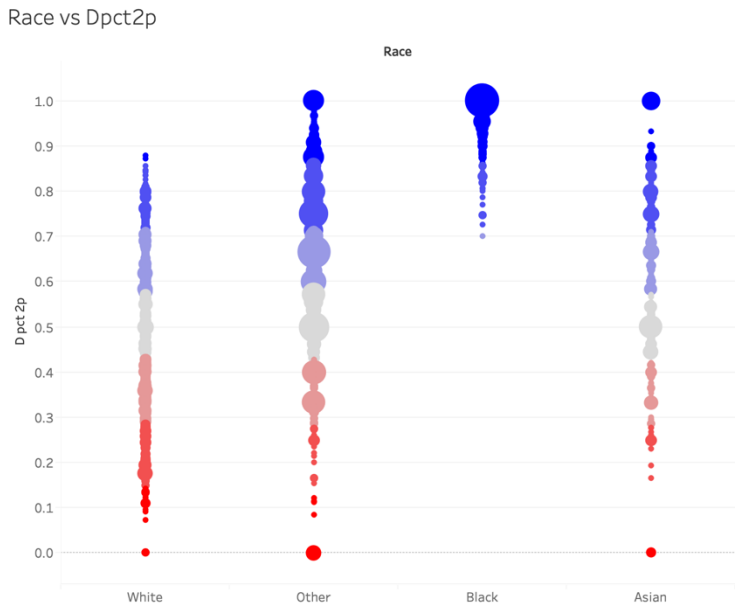


Figure 3.1: Race vs D_pct_2p for Florida

data. When modeling partisanship without using race, as I do in the subsequent sections, this should not present a major issue, but should be a concern to a researcher who is using race as a proxy for partisanship,

as discussed in the first half of Chapter 2. Interestingly, when looking at the age breakdowns in the Florida dataset, which can be seen in Figure 3.2, we see a slightly different story than the one in North Carolina, in Figure 2.10. In Florida, one can still see the leftward tilt at all age levels, which is perhaps a bit more pronounced at younger ages, but the similarities stop when filtering to only white members of the dataset. In North Carolina, the leftward lean completely disappears at all age levels when restricting the graph to white individuals only, but this is not the case in Florida. In Florida, the partisanship of whites is more right leaning than the partisanship of all individuals, but not nearly as far right leaning as in North Carolina. In fact, looking at the right half of Figure 3.2, one

would be hard pressed to say that it demonstrates any sort of significant rightward tilt. Older age groups shift to the right in Florida more significantly when the graph is restricted to only whites, which lends credence to the idea that age could be used in a partisanship model, where Figure 2.10 points to race as the more significant predictor. Taken together, the two again point to a relatively obvious, yet important truth: the more you know about someone’s demographics, the easier it should be to predict their partisan affiliation.

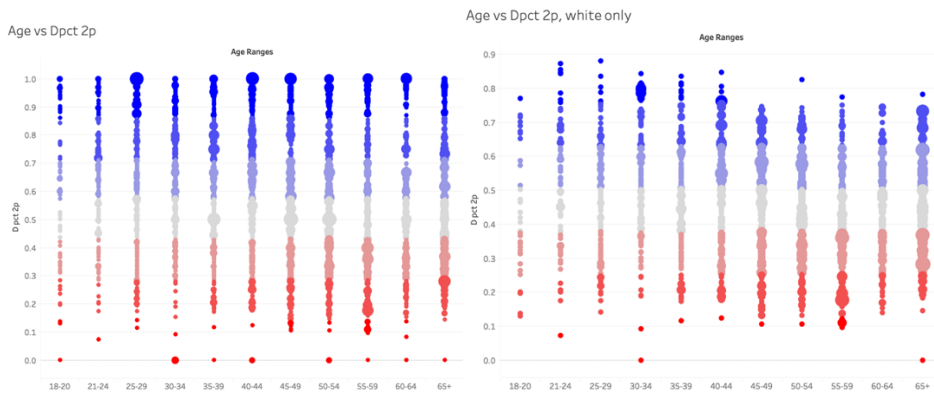


Figure 3.2: Florida D_pct_2p by age, all races (left) and white only (right)

Modeling Partisanship in Florida

Random Forest Modeling

When modeling the partisanship of Floridians using random forests, I ran into the same issues that I had in North Carolina: using either the Hersh 80% standard, or with any sort of model not in the two party system, the overall accuracy was high, but only because the model simply predicted everyone to be a nondemocrat. The only time that the model had any balance between its in-group classification accuracies was when the conditions were the most relaxed: predicting simply whether or not someone was more likely to be a Democrat or a Republican, with the modeling program set to use a coding threshold of D_pct_2p

> 0.5 , when the model came in with an overall classification accuracy of 64%.

The confusion matrix for this model can be found in Figure 3.3, and it contains

one major difference when

compared to the model with the

same parameters in North

Carolina (see Figure 2.14). In

North Carolina, the random forest

model using $D_pct_2p > 0.5$ was

biased towards accurately

predicting someone to be a

nondemocrat, getting 82% of true nondemocrats correct, while only predicting

47% of true Democrats correctly, while in the Florida model, the exact opposite

was true. The Florida version of this model was actually biased in the opposite

direction, classifying only 45% of true nondemocrats correctly, but over 80% of

true Democrats correctly. The opposite bias is likely due to the leftward tilt of the

Florida comScore subset, where the North Carolina subset leaned rightwards, as

discussed above. Each of these models likely has its own place in an overall

ensemble, discussed later in this chapter.⁹⁶

Exclusivity Index Modeling

After running the Florida data through the exclusivity index modeling

programs, I saw very similar results to the North Carolina exclusivity index

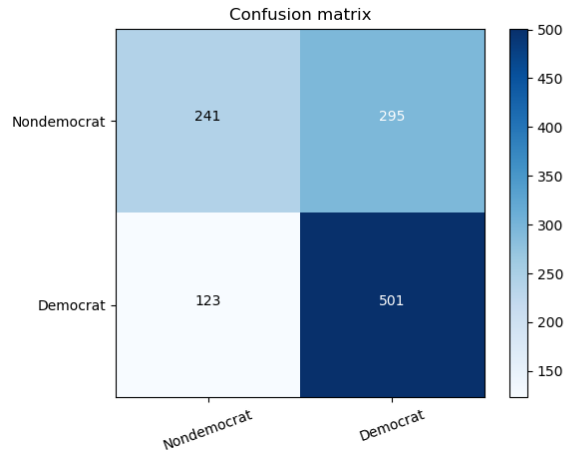


Figure 3.3: $D_pct_2p > 0.5$

⁹⁶ I elect not to report the most important features in the random forest model, for reasons discussed in Chapter 2.

		Original Criterion			Modified Criterion		
t		0.7	0.8	0.9	0.7	0.8	0.9
D_pct	> 0.8	60.0%	71.0%	87.8%	59.4%	69.1%	84.4%
	> 0.5	56.2%	74.9%	84.0%	61.6%	69.5%	76.6%
D_pct_2p	> 0.8	67.0%	76.1%	79.4%	66.3%	71.3%	81.7%
	> 0.5	69.8%	75.4%	75.8%	67.0%	76.7%	74.2%

Table 3.1: Exclusivity Index Model Accuracies, Florida

models. The results are summarized in Table 3.1, and they display similar trends to those in North Carolina. The higher levels of exclusivity resulted in more accurate models, while actually making a classification for fewer of the users in the dataset. This time around, the $t = 0.9$ models classify a smaller percentage of the testing set, but achieve very high accuracies, including in-group accuracies. Once again, at the more stringent coding thresholds for Democrats, there are many more nondemocrats than democrats, but again, the exclusivity index classifiers result in more accurate classifications for both nondemocrats and

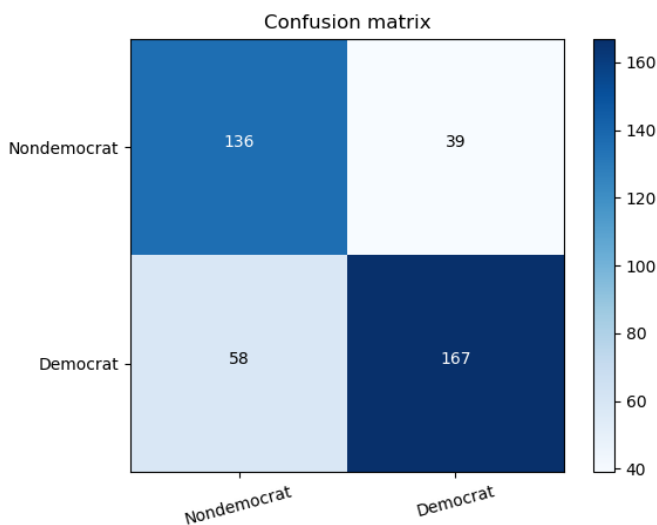


Figure 3.4: $D_pct_2p > 0.5, t = 0.9$, Original Criterion

Democrats when compared to their random forest counterparts. The model with perhaps the best looking confusion matrix was the $D_pct_2p > 0.5, t = 0.9$ model using the original

criterion, shown in Figure 3.4. This model made classifications for only about a third of the users in the dataset, but was highly accurate for both Democrats and nondemocrats alike. The Florida models on the whole classified fewer users than the North Carolina models with the same parameters. Overall, the results of the exclusivity index modeling for Florida mirror those from North Carolina quite well, and generally support the notion that exclusivity index based classifiers can be an effective way to measure partisanship, especially as part of an overall ensemble model, as discussed shortly.

Multi-State Modeling

With 50 states in the United States of America, and voter files with enough demographics to re-identify reasonably well available for only a few of them, the ability to build a national model hinges on the portability of a model built on one state to the testing set of another state. With rather successful models built for two states, Florida and North Carolina, I could now test that portability. Though North Carolina and Florida are both southeastern states, and the implications for said portability would thus be limited, proving that models can be ported to other states would be an important step in demonstrating the robustness of my approach. I thus modified my code to run on two states at once. For this section, I used the random forest model with the $D_pct_2p > 0.5$ threshold, and the exclusivity index classifier model with $D_pct_2p > 0.5, t = 0.9$, with both the original and modified criterion, as these parameters produced the best models from their respective families at the single state level.⁹⁷

⁹⁷ Cross-state modeling code can be found in Appendix 6.

North Carolina models, Florida users

I began by fitting models on the North Carolina users, and then using the Florida users as the testing set. The random forest model fit on North Carolina users had a 59.4% overall classification accuracy when tested on Florida computer users - only 5% less accurate than a North Carolina model fit and tested on North Carolina users. The accuracy for true Democrats in the sample was 41.2%, and the accuracy for true nondemocrats was 81.2% - both numbers that track well with the group level accuracies that were reported for the in-state model in Chapter 2. The exclusivity index models, both run with the parameters $D_pct_2p > 0.5$ and $t = 0.9$, also showed results that indicated they maintained a fair amount of their predictive power. The original criterion model dropped from 66.1% accuracy when tested on North Carolina residents to 63.4% accurate when tested on Florida residents, and the modified criterion model actually increased in its accuracy, from 63.4% to 64.1% accurate. Both models maintained group-level classification accuracies of over 50% for each group. Due to the nature of the exclusivity index classifier, I worried that the models would make relatively few classifications. In Table 2.5, which had the most exclusive sites for the state of North Carolina, one can see websites that indicate local biases in the Internet, like *charlottemotorspeedway.com*. As such, I was worried that local interest sites would cause the classifier to be unable to make predictions for a larger number of users in a different state. Though the number of classified individuals did drop when tested between states, the models still made predictions for over one third of the users in the state of Florida, despite being set at the highest level of

exclusivity. Additionally, it is worth noting that this problem would likely decrease if multiple out-of-state models were used on a browsing history. Say there was an individual from Oregon: one could test the browsing history with both the Florida and North Carolina exclusivity index classifiers, making a prediction even more likely. The confusion matrices for the random forest model, and both exclusivity index classifier models are below in Figure 3.5.

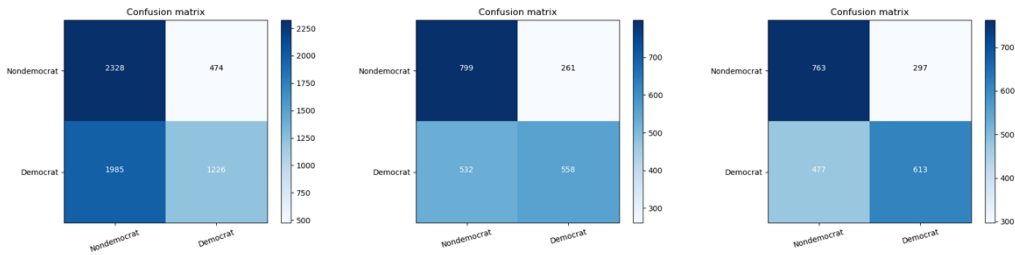


Figure 3.5: North Carolina models tested on Florida users. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion

Florida models, North Carolina users

Next I tested models that had been fit on the Florida subset of the 2016 comScore dataset on the North Carolina users, and observed very similar trends. The random forest model that was fit on Florida individuals saw a slight drop in overall accuracy, from 64% to just under 62%, while maintaining very similar group level classification accuracies. The original criterion exclusivity index model resulted in a 68.1% accuracy overall, a drop from the 75.8% the similarly parameterized model had on individuals within Florida. The modified criterion model dropped in overall accuracy as well, from 74.2% to 69.1%. Both of these numbers remain respectable – just a hair south of 70% - and their confusion matrices, reported in Figure 3.6, remain encouraging. However, the Florida models, when set to the highest exclusivity level of $t = 0.9$, only classified about

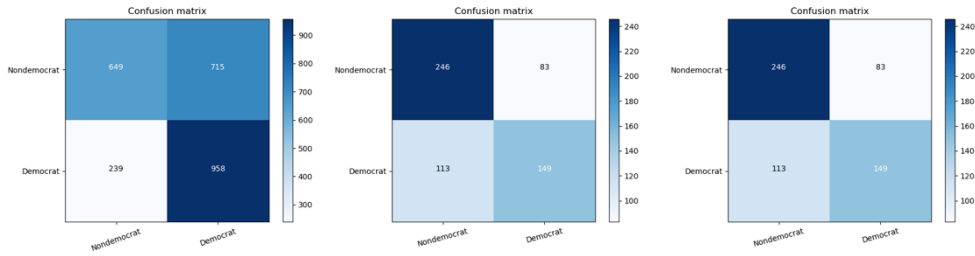


Figure 3.6: Florida models tested on North Carolina users. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion

25% of the individuals in the North Carolina dataset. As previously discussed, this issue is likely to decrease in severity as more models are added, but it does present a significant challenge to the usefulness of exclusivity index based classifiers, especially across states.

A Combined Model

Finally, I fit a combined model, trained on a mix of Florida data and North Carolina data, producing the most encouraging results of the multi-state modeling process. The random forest model achieved an accuracy of 64.5%, which was comparable to both of the single state random forest models. The exclusivity index models also had high overall accuracies: 72.9% for the original criterion model and 73.5% for the modified criterion model. Once again, though the models only actually classified just over one quarter of the individuals in the testing set. Without the locality problem at the state level, however, it is safe to

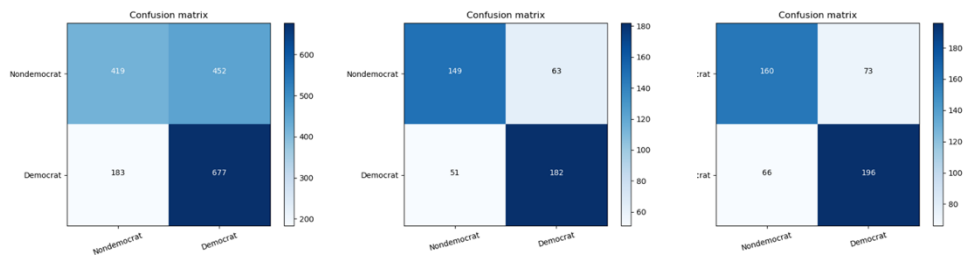


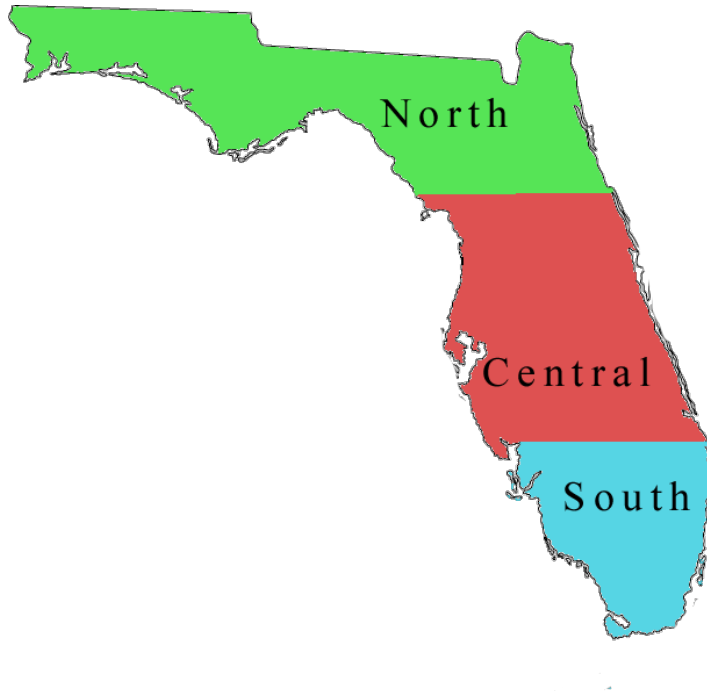
Figure 3.7: Combined Models. Left: Random Forest, Center: Original EI Criterion, Right: Modified EI Criterion

assume in this case that the portion of individuals classified would likely increase substantially as the exclusivity constraints were lowered. Though this classification rate may seem low, we can trade accuracy for a higher classification rate by varying the exclusivity levels. The confusion matrices for each of the three models are above, in Figure 3.7.

Stratified Florida

Florida is significantly larger in size and population than North Carolina, and also has a fundamentally different demographic makeup, all of which could have contributed to the drop in accuracy of the models when testing across states.

In this section, I stratify Florida into three smaller sub-states: North Florida and



the panhandle (henceforth simply North Florida), Central Florida, and South Florida). With only ZIP codes in the comScore dataset to inform me of the location of each computer user, I stratified the state using

Figure 3.8: Stratified Florida

two latitude lines. I set the line between North and Central Florida at 29.56°N , with the majority of Dixie, Alachua, and Flagler counties included in the North Florida portion. I set the line between South Florida at the northern edge of Palm

Beach and Glades counties, at 27.03°N, which puts Lee County in South Florida. The stratified map of Florida is shown in Figure 3.8.⁹⁸ After splitting the 2016 comScore data into the new sub-Floridas, I was left with 2,708 users in Central Florida, 2,334 users in South Florida, and 971 users in North Florida.

Following the old adage “the further North you go in Florida, the further South you get,” I expected demographic differences to potentially make the North Florida subset more accurate with respect to North Carolina than the Central or South Florida subset. North Florida demographically resembles the rest of the American South more than Central Florida, which has a higher portion of Hispanics, or South Florida, which is dominated by Miami-Dade and Broward

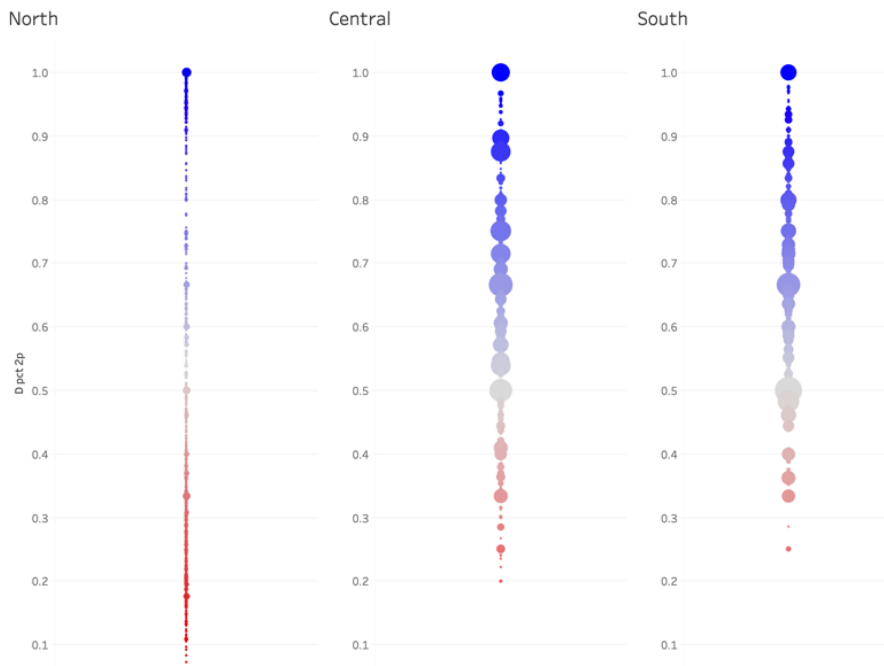


Figure 3.9: Stratified Florida D_pct_2p, size represents count

⁹⁸ This stratification is fairly standard. See https://www.researchgate.net/figure/Regions-of-Florida-with-Cities-Map-showing-Floridas-travel-regions-subregions_fig4_312588628 and https://www.americagofishing.com/shop/index.php?main_page=maps_counties_cities.

counties – two Democratic strongholds with heavily black populations. Splitting the state, and creating graphs of each sub-Florida based on D_pct_2p , in Figure 3.9, I saw that this was indeed the case: South Florida was significantly more Democratic than North or Central Florida, with North Florida being the most Republican of the three. Using the same cross-state modeling code as when testing North Carolina against Florida as a whole, I ran the same three models, this time testing North Carolina against the three different regions of Florida individually. The classification accuracies when models are built on North Carolina data and then tested on North, Central, and South Florida users, respectively, are presented in Table 3.2, and they tell the expected story. In each case, about 63% of the sub-Florida dataset’s users were not able to be classified at the $t = 0.9$ threshold, a number in line with the overall number from testing on the overall Florida dataset. The accuracy levels are highest when testing on North Florida, and lowest when testing on South Florida, results that point to the importance of demographic similarity when testing models across states. The

	Random Forest	Original EI	Modified EI
North Florida	68.7%	72.7%	72.7%
Central Florida	62.5%	64.3%	65.0%
South Florida	52.1%	57.9%	59.3%

Table 3.2: North Carolina models tested on stratified Florida. $D_pct_2p > 0.5$, $t = 0.9$.

results also suggest that a state like Virginia would be best suited to cross state modeling with North Carolina, and that North Carolina and Florida may not be of much use when cross state modeling with states like Utah, or North Dakota.

The Implications of Cross State Modeling

The above section demonstrates some rather important findings towards the overall goal of this project. With voter files from only some states containing the information necessary – namely race, age, and partisan affiliation - to partially re-identify the comScore dataset, for this partisanship prediction scheme to succeed on a national level, predictions will have to be for individuals in the absence of training data from their state of residence. The results of the cross state modeling above suggest that this is indeed possible. Though in general, the models lose accuracy when they are tested against out of state individuals, they retain a significant amount of their original predictive power. The success of the combined models is also an encouraging sign for the creation of a more robust set of models as this analysis is expanded to other states. Though the best performing models I created, the exclusivity index classifier models, maintained their accuracy at the cost of actually classifying many users, this problem is likely to be lessened as exclusivity index classifier models are made on many different states. Due to the laws of probability, we know the following: for an individual, $P(FL = 0, NC = 0) \leq P(FL = 0)$ where $P(FL = 0)$ indicates the probability of a Florida model returning no classification for the individual. Note that this remains true even if the events $FL = 0$ and $NC = 0$ are not independent. This implies: $P(FL = 1 \cup NC = 1) \geq P(FL = 1)$, and so on as more states are added. Thus, as more models are built, the likelihood of a classification increases for every individual in the testing set. In any event, the cross state modeling results presented above

indicate that these methods at the very least hold potential for models to be built that can function on a national level.

The Partisan Internet 2.0

As discussed in Chapter 2, the results of the exclusivity index classifier modeling process, especially at the level of the single state, points to an Internet that mirrors the partisanship of American politics. Table 3.3 contains the 10 most exclusive domains, with Democrats coded using $D_pct_2p > 0.5$ (note that this is different from the Table 2.8) for runs of the Florida, North Carolina, and combined models. These models were parameterized using $t = 0.9$, making these websites the most exclusive I calculated. Since these models simply predict whether an individual is more likely to be a Democrat or a Republican – a nondemocrat with these parameters can simply be viewed as a Republican – I present this table as a first sketch of the explicit partisan bounds of the Internet in the American political system.⁹⁹

Potential Analytic Improvements

There are several improvements that could be done to the analysis that have potentially high impacts for the accuracy and rigor of the analysis presented thus far in this project. As currently constituted, my analysis has at least two bottlenecks of uncertainty, and more depending on which quantity of interest is used. Note that this is in addition to the inherent uncertainty of predicting who is likely to be a Democrat – we cannot actually know *ex ante*, if someone is a

⁹⁹ One will notice the domination of the combined columns by Florida sites, due to the number of Florida users being more than double the number of North Carolina users in the comScore dataset in 2016.

	North Carolina		Florida		Combined	
	<i>Republican</i>	<i>Democrat</i>	<i>Republican</i>	<i>Democrat</i>	<i>Republican</i>	<i>Democrat</i>
1	tacticalrepublic.com	dpsnc.net	newsbake.com	browardschools.com	heatst.com	browardschools.com
2	cheapertandirt.com	moviesub.net	swflaml.com	broward.edu	altdriver.com	broward.edu
3	ncagr.gov	eesysoft.com	pnj.com	federate365.com	leepa.org	federate365.com
4	faithtap.com	ukessays.com	genealogybank.com	browardhealth.com	sheriffleefl.org	browardhealth.org
5	itsthevibe.com	daclips.in	dexwebsolutions.com	pediatricassociates.com	proxyvote.com	recruitmentplatform.com
6	sierratradingpost.com	einthusan.com	wholesalemarine.com	spinrilla.com	getitfree-samples.us	elconfidencial.com
7	odometer.com	tutorialspoint.com	proxyvote.com	browardlibrary.org	husqvarna.com	browardlibrary.org
8	beoyonddiet.com	google.co.in	villages-news.com	v11ndpin.com	villages-news.com	gameplaycloud.com
9	mycookerwards.com	xerox.jobs	brevardtaxcollector.com	sitel.com	godtoday.com	miramarfl.gov
10	activebeat.co	tudou.com	altdriver.com	recruitmentplatform.com	webervations.com	stereoday.com

Table 3.3: The Partisan Internet

Democrat, we can only predict with some probability who is a Democrat. This uncertainty is the type that Hersh discusses, in which a campaign will contact an individual if they believe there is a greater than 80% chance that the individual would vote for the campaigns candidate.¹⁰⁰ This kind of uncertainty is captured in the actual modeling process – the portion of trees in the forest that decide an individual is in a certain class, or the portion of exclusively visited sites by that group, is what makes the actual prediction.

If one is directly modeling partisanship based on the browsing histories of individuals, there are two additional places where uncertainty leaks into the analysis: the coding of who is a Democrat and who is a nondemocrat, and the actual uncertainty in the accuracy of each model. If one is attempting to use race, the other quantity of interest I predict, as a stepping stone to get to browsing history, then there is an additional level of uncertainty – not every white individual votes Republican, and not every nonwhite individual votes Democrat, even if they both might be considered relatively safe bets. Improving the accuracy and rigor of the analysis presented in this paper can be done by decreasing the amount of uncertainty introduced at each of these points, and offers the potential for significant future research. Below, I offer what I believe are some avenues into decreasing the entry of additional uncertainty into my modeling process.

Better Linking with comScore

Put simply, no reliable method currently exists to easily link the comScore dataset to the voter files of states in a manner that offers rigorous re-identification.

¹⁰⁰ Hersh, *Hacking the Electorate*, 72

I used an approximate method, but in many cases was not able to make a user less than 300-anonymous. The existence of some columns in the comScore dataset, like education level and household income, that could be used to identify individuals but do not exist in the voter files indicates the possibility of better re-identification, as do the presence of similar columns in the voter file but not the comScore data, such as gender. The addition of a third dataset to the linking process could be helpful here. Imagine a dataset containing information on the income and education levels of a household that could be linked to the voter file for a state before the attempted linking with the comScore dataset, or a reliable method for imputing the gender of an individual in the comScore dataset. The more demographic columns that two datasets share in column, the lower the value of k -anonymity that can be reached via linking the two.¹⁰¹ Importantly, the linking process does not necessarily need to reach a level of anonymity where $k = l$. Since there are very few values in the sensitive column, which is either party or race, the dataset will have low l -diversity to begin with. If we could reach a value where $l = l$, wherein each of the k individuals in the voter file matching a comScore user has the same value, then we can simply impute that value into the comScore data with relatively high certainty. Reaching a low enough value of k to achieve $l = l$ would greatly reduce the uncertainty entering at the point where values for partisanship are imputed into the training data, one of the three

¹⁰¹ In this manner, re-identification quite obviously produces the opposite goal of de-identification. A lower k for a k -anonymous dataset means that the dataset has been more effectively re-identified, whereas a higher k would mean that the dataset had been more effectively de-identified.

uncertainty bottlenecks described above. If gender were to be reliably predicted from the comScore dataset, then the value of k would be cut in half, at least in the average case, across the board, getting a researcher significantly closer to the necessary goalpost.

More Data Collection with Browsing Histories

The section immediately above describes a way to improve the linking from the comScore dataset of browsing histories to the voter file as a way to measure partisanship. However, there is a way that this problem could be circumvented entirely: direct collection of partisan identification along with the browsing history. If comScore had collected partisan identification as one of the demographics in their dataset, the linking problem would disappear entirely – no re-identification would actually be necessary. This would represent a further violation of the privacy of the users present in the comScore dataset, but it would allow a better identification for the training and testing values. Such information being present in a browsing history dataset at the outset would also eliminate the need to use voter files for relinking, rendering some of the cross-state concerns null. With the fall of net neutrality, the likelihood that such a dataset will arise, perhaps separate from comScore, is increasing.

More States

As discussed above, another way to improve the rigor and robustness of the modeling process would be to simply add more states to the analysis. Any voter file containing partisan identification would be a candidate for use in this modeling process, though the degree to which the data could be re-identified

would depend on the other demographic columns in said voter file – the more columns in common with the comScore data, the better the re-identification possible. The addition of more states to the modeling process, especially states not located in the Southeast, would make overall models more robust, and likely make them more accurate for states that were not in the original training set.

Ensemble Modeling

Another area for future research to tackle is the use of all the individual models in this project in combination to make the most accurate predictions possible. The analysis in this project has resulted in two different classes of models, on two different quantities of interest, with many different combinations of tuning parameters and classification criteria. A key question for the use of these models in a professional setting would be how to use all of these models in concert to make the single best overall classification. Indeed, the use of multiple models is at the heart of the random forest modeling process, in which the mode is taken from the decisions of a large number of decision trees. I believe there are two main methods for creating an ensemble model out of the disparate models for race and partisanship created so far for this project. I will refer to them as the *modular forest* and the *unbalanced decision tree*. As I built my testing set for the individual models in ways that inherently introduce extra levels of uncertainty into the modeling process, I choose not to attempt to discover which ensemble strategy is the more effective here, as the conclusions would have a very high degree of uncertainty and would thus not be substantive. Instead, I will discuss the conditions under which each ensemble strategy would likely be the most

effective, and the merits and failings of each strategy. It is important to note that for both of these strategies, only the models that predict to the same threshold should be used in concert. For example, a random forest model for partisanship that predicts whether or not someone is more likely to be a Democrat than a Republican – this model would have the two party flag set and the threshold at 0.5 – should not be used in the same ensemble as a model that is meant to predict to Hersh’s 80% standard using D_pct and not D_pct_2p . That said, models that predict to the same threshold level, say the Hersh 80% threshold, but use different levels of exclusivity, could indeed be used in the same ensemble. The same goes for models with the same threshold, but different settings for the two party flag, and of course for exclusivity index and random forest models that use the same threshold.

The Modular Forest

Similar to a random forest, which takes the modal class prediction of a large number of decision trees, the modular forest strategy for ensemble modeling would take all the models in question, run them on an individual’s browsing history, and make a final classification based on the mode of the different class predictions. The decision rule can be summarized as follows:

Modular Forest Decision Rule

Given a modular forest F^* made of models $\{F_1, F_2, \dots, F_n\}$, and a browsing history B for user u , pass B through each of F_1, F_2, \dots, F_n , and classify u with the modal prediction.

The modular forest ensemble should be used if there are many models that all have similar levels of accuracy. The modular forest ensemble functions

essentially under foundational principles of democratic theory, namely the Condorcet Jury Theorem. The theorem states that if any individual juror has a greater than 50% chance at reaching the correct verdict, and all jurors have the same probability of the correct answer, then all else equal, a larger jury will reach a correct verdict with a higher probability than a smaller jury. Put in statistical terms, a binomial distribution, $Bin(n,p)$, is more likely to have come out with more than $\frac{n}{2}$ successes the larger the value of n , given that $p > 0.5$. Consider each model as a juror, and the modular forest itself as the jury. The more models that are added to the forest, so long as they are accurate more than 50% of the time, the better the prediction of the forest, based on the mode of the models, is likely to be. This principle, however, breaks down when one model has a significantly higher accuracy than the others. Imagine a contrived jury where there is one juror who gets the correct answer to the trial 99% of the time, and 8 jurors who get the answer correct only 51% of the time. Condorcet's Jury Theorem now breaks down, as the jurors have differential accuracies. By simply choosing the "best" juror's decision every time, the jury would achieve 99% accuracy, but taking the mode of this jury would yield only a 65% chance of achieving the correct decision. In a case where the accuracies models resemble such a contrived jury as this, the modular forest could conceivably be weighted tree by tree, or a version of the unbalanced decision tree ensemble could be used.

Unbalanced Decision Tree

The second method of forming an ensemble, the unbalanced decision tree, is especially helpful for models that may or may not make a classification, such as

many of the exclusivity index classifier models described in detail in chapter 2. The unbalanced decision tree method of forming an ensemble can be thought of as a sort of flow chart.¹⁰² Say we have four models: A, B, C, and D, with classification accuracies of 95%, 85%, 70%, and 65%, respectively. Now imagine that model A makes a classification 50% of the time, model B 60% of the time, model C 70% of the time, and model D 80% of the time. Intuitively, we should take a user and first feed their browsing history into model A. If model A spits out a prediction, we can quite reasonably assume it to be correct. If model A does not spit out a prediction, we can then feed the browsing history into model B, and so on and so forth. Formally, the decision rule for this tree is as follows, for a user u with browsing history B :

- 1) Feed B into A. If A returns a classification, classify u accordingly. Else proceed to 2.
- 2) Feed B into B. If B returns a classification, classify u accordingly. Else proceed to 3.
- 3) Feed B into C. If C returns a classification, classify u accordingly. Else proceed to 4.
- 4) Feed B into D. If D returns a classification, classify u accordingly. Else do not classify.

If we were to take the modular forest approach to building the ensemble, we would be essentially creating the contrived jury, for an ensemble that expects a Condorcet jury. Model A could conceivably not make a prediction, and then model B could conceivably be the opposite of both models C and D. In this case, a

¹⁰² I describe the decision tree as unbalanced, because I envision it essentially being one long branch, rather than an attempt at balancing a tree where one makes $\log(n)$ decisions in each case (for n models).

modular forest would classify according to the mode, which would be the choice of models C and D. There would be a 19.5% chance that both C and D would be incorrect – and this is assuming that their accuracies are independent of one another - whereas the probability that model B made an incorrect choice would be only 15%. One can envision a world in which the modular forest created the joint probabilities for all the models classifying in each manner being correct and incorrect, and then choosing based on those numbers, but such a calculation would have an incredibly large number of assumptions about independence and covariance woven into it. In cases such as the one described above, the unbalanced decision tree can help remove the necessity to make, or blatantly ignore, such assumptions.

When analyzing many of the early models that I created for this project, I noted that while they may have had high overall classification accuracies, the model accuracies differed for each class of the target quantity. This phenomenon was especially prevalent during the early race modeling process with random forests. In an unbalanced decision tree, however, differential classification accuracies can actually be leveraged as a feature, and not a bug. Imagine a model that predicted 95% of individuals to be a nondemocrat, and only 5% to be Democrats. This model, however, is correct *every single time* that it predicts someone to be a Democrat. It's classification accuracy for nondemocrats may hover around 50%, and the overall classification accuracy may be in that vicinity as well, but every time it predicts someone to be a Democrat, the model gets it right. This model would be virtually useless in a modular forest setting – indeed,

if its overall accuracy is under 50% it may actually make the forest worse. In an unbalanced decision tree, however, this model can be very effective. Recall our imaginary unbalanced decision tree with models A-D from the prior paragraph. We can add this new model, call it M, at the root of the tree, with a rule along the lines of, “If M classifies as Democrat, classify as Democrat, else proceed to A.” Our set of classification rules from the prior paragraph then has the following step as its new first step:

- 0) Feed B into M. If M classifies u Democrat, label u accordingly. Else proceed to 1.

Even though M might make a prediction, nondemocrat, that we do not have any confidence in the accuracy of, we can leverage its differential accuracy across classes to weed out the individuals who we think are Democrats, one by one.

Such a structure for an ensemble model would have a few important implications. First, the unbalanced decision tree allows the modeler a certain degree of extra power over the decision rules of the process, allowing them to vary from model to model. The user could even turn the tree into a more complicated sort of flow chart, making the tree more balanced in the process, if they so desire. The unbalanced decision tree paradigm also has a recursive nature to it. Each joint on the tree’s branch can contain its own ensemble model, whether that be another, smaller unbalanced decision tree, or even a modular forest.

Imagine we still have model M from above, and now we have models F_1 , F_2 , F_3 , and F_4 . Each F model has the same classification accuracy, and they all have the same group-level accuracies as well. With these models, it makes the most sense to use the following classification algorithm for a user u with browsing history B :

- 1) Pass B to M . If M classifies as Democrat, label u as a Democrat. Else, proceed to 2.
- 2) Create a modular forest F^* with models $\{F_1, F_2, F_3, F_4\}$. Use the Modular Forest Decision Rule when passing B to F^* , and classify u with the result.

An important note is that there is no Unbalanced Decision Tree Decision Rule, but rather an individual decision rule for each branch of the tree itself. This makes the process more complicated, and potentially more time consuming, but also more robust and adaptable. Fundamentally, the unbalanced decision tree is more efficient in the statistical sense than the modular forest because it can make use of models that the modular forest cannot, namely those with differential group-level accuracies, and thus theoretically requires less data to be effective. However, in the computer science sense of the word, the unbalanced decision tree is less efficient as it is less parallelizable, and likely requires more operations to complete than the modular forest. The final major difference, between two ensemble paradigms stems from the fact that different users will necessarily be classified at different levels in the unbalanced decision tree. If the tree is constructed properly, then users who fall out the tree into classifications near the top of the tree, after fewer branches have been traversed, should have been classified with higher degrees of confidence than users who fall out of the tree lower down the branch. This property could be quite valuable to political professionals, as it conveys extra information along with the simple classification of individuals, which is all the modular forest can do.

All this being said, an unbalanced decision tree should be used only if it offers distinct advantages over a modular forest. If the conditions for a modular

forest are met – many models with similar classification accuracies, and group-level accuracies, all above 50% - then using an unbalanced decision tree could quite conceivably result in worse results, rather than better ones. In the simplest example, imagine three models, that each classify any user correctly 75% of the time. If we created an unbalanced decision tree, we could get the answer wrong at the first branch 25% of the time, whereas we expect a modular forest to be correct over 84% of the time in this case.¹⁰³

Given the results of the models I have presented in this thesis, I believe that an unbalanced decision tree structure would be the most effective at creating an ensemble model. The differing levels of exclusivity used in the exclusivity index models would allow individuals who are very likely to be Democrats to be filtered out near early in the modeling process using the $t = 0.9$ threshold. The exclusivity threshold could then be slowly reduced as an individual gets further down the tree without being classified. The last branch on the tree would be a modular forest created from the random forest partisanship models, for those models guarantee a prediction, though less confidently than the exclusivity index classifiers. This structure would also allow a researcher to use the race model to predict the race, and then use that prediction in one of the partisanship models, as discussed in Chapter 2.

Conclusions

At a bare minimum, the analysis in this project demonstrates that browsing histories can be used to somewhat reliably infer both the race and the partisanship

¹⁰³ Easily calculated by $X \sim \text{Bin}(.75, 3)$, $\Pr(X \geq 2) = .84375$.

of an individual. As discussed at the outset of this paper, such a result has potentially severe ramifications for both the political industry and academics in the political sphere. The results presented in this paper point to an Internet that suffers from the same partisan divides that are rending the American populace in two in our current politics. Liberals and conservatives do not just vote differently, but also visit different websites – the partisan divide in the Internet mirrors that present in American society. With news, shopping, entertainment, and so much more increasingly moving to the online realm, the partisan divides already present point to the possibility of a positive feedback loop that could further divide the American public. The ability to use browsing histories as a predictive dataset could present political professionals with the potential for valuable gains in advertising efficiency during campaigns or legislative actions. Partisanship matters incredibly deeply to American politics – indeed one could arguably even call it the foundation of the modern political system – and thus so too does a deeply divided partisan Internet.

The vast majority of this paper has focused on using browsing history as a predictive measure of partisanship, with an eye towards the future. A campaign benefits if they can determine who someone *will* vote for. Here, I consider the possible ramifications of using browsing history to determine who someone *did* vote for in a past election. Partisanship, and as a corollary the choice someone makes when voting, is not a protected class under American law. The repeal of net neutrality under the Trump administration, as discussed in the opening chapter of this paper, gives Internet service providers the ability to treat customers

differentially. Combining these facts with the results from this paper, a significant stretch of the imagination is not required to picture a world in which an ISP begins treating Democrats and Republicans differently. An individual's vote is meant to be secret and private; a secret vote is one of the core tenets of a robust democracy, and is enshrined in American law.¹⁰⁴ The current American legal system, when combined with the results I have reached, indicate that the privacy of the American voter may no longer be respected by the legal and political system.

This project is inherently concerned with the concept of privacy. The re-identification of the comScore dataset by linking to state voter files is in some ways an inherent violation of privacy, especially if k -anonymity begins to be violated for smaller and smaller values of k . The political and legal realities of the current time, however, require a much deeper question of privacy: is privacy a right or a tradable commodity? Can an individual trade away his, her, or their rights to privacy, in this case the right to the privacy of their vote? If society conceives of privacy as a right, instead of a tradable commodity, then the answer to that question is a firm no. In a world rife with technologists committing privacy violation¹⁰⁵ after privacy violation,¹⁰⁶ society seems to be trickling away from the

¹⁰⁴ Gerber, "The Adoption of the Secret Ballot," 1994.

¹⁰⁵ Jennifer Valentino-DeVries, et al. "Your Apps Know Where You Were Last Night, and They're Not Keeping It Secret." *The New York Times*. December 10, 2018. <https://www.nytimes.com/interactive/2018/12/10/business/location-data-privacy-apps.html>

¹⁰⁶ Gabriel J.X. Dance, et al. "As Facebook Raised a Privacy Wall, It Carved an Opening for Tech Giants." *The New York Times*, December 18, 2018. <https://www.nytimes.com/2018/12/18/technology/facebook-privacy.html>

idea of privacy as a right. If privacy is a tradable commodity, then the lack of secrecy of the vote can just be read as a cost in the bill that one pays their ISP. Economists could read the worth of a private vote off of differential ISP prices, the way they infer the prices of other goods that are not explicitly on the market, and there would be nothing inherently wrong with an ISP differentially treating customers based on their inferred voting patterns. Indeed, the “economically rational” ISP would probably begin to charge people more money in order to not have their voting patterns predicted and sold away, in order to reap the gains of the monetary value of privacy. If, however, we conceive of privacy as a right, then there should be something deeply disturbing about the realities discussed above. The current legal framework prevents a political machine from entering the voting booth and intimidating voters, but it says nothing about modern computational machines doing essentially the exact same thing.¹⁰⁷

On February 26, 2019, almost exactly one month before this project was finished, the city of Chicago, once synonymous with machine politics, held its mayoral primary. In the jungle primary, a son of the Daley family lost to two African-American women, one of whom ran a specifically anti-machine campaign.¹⁰⁸ For many, the election results highlighted the decreasing power of the old political machines in the increasingly modern political world. The fact of

¹⁰⁷ Special thanks to Kate Vredenburg for taking the time to sit down and discuss this topic, and the distinctions therein, with me.

¹⁰⁸ Chicago Tribune Editorial Board. “Lightfoot cracks the machine, setting up this historic finale. Chicago’s next mayor will be an African-American woman.” *The Chicago Tribune*. February 27, 2019. <https://www.chicagotribune.com/news/opinion/editorials/ct-edit-lori-lightfoot-chicago-mayor-runoff-20190226-story.html>

the matter is simply this: the old machines are not needed anymore, at least not to the same degree. A campaign does not need to pledge their undying allegiance to the machine in order to whip votes, or even just identify voters. The technological advancements of this century mean campaigns can do that for themselves. Though the classic political machine may be dead or dying, American politics is on its way to being governed by the computer. The age of the old machine politics is over, and the age of the new machine politics is upon us.

Appendix 1: comScore codebook

Source: comScore, Inc.

comScore Web Behavior Database

The comScore Web Behavior Database captures detailed browsing and buying behavior by 100,000 Internet users across the United States at the domain level. The panel is based on a random sample from a cross-section of more than 2 million global Internet users who have given comScore explicit permission to confidentially capture their Web-wide activity.

The unique panel identifier is Machine ID and there are cases where multiple configured machines exist within a household, but all demographic information is based upon the associated household. All sessions are aggregated by machine in the household, so that individual breakdowns are not available and a particular individual could use more than one machine. The items below are associated with each machine-household.

Demographics

Variables	Type	Label/Description
machine_id	num	machine identifier
hoh_most_education	num	most education – head of household
census_region	num	census region
household_size	num	household size
hoh_oldest_age	num	oldest age – head of household
household_income	num	household income
children	num	presence of children
racial_background	num	racial background
connection_speed	num	connection speed
country_of_origin	num	country of origin
zip_code	num	zip code

A product-transaction table shows online purchases if and only if there was a transaction in the session. The following items are records for each transaction.

Transaction Information

Variable	Type	Label/Description
prod_category_id	num	product category ID
prod_num	char	product name
prod_qty	num	product quantity
prod_totprice	num	product total price
basket_tot	num	basket total

User sessions are recorded with date and time stamps and the most detailed clickstream item is a Session ID, which represents a domain visit that may have more than one 'page view.' The items below are associated with each Session ID.

Session Information

Variable	Type	Label/Information
user_session_id	num	identifies a session of activity
domain_id	num	domain ID
ref_domain_name	char	referring domain name
pages_viewed	num	pages viewed
duration	num	duration at site
event_date	char	date of activity
event_time	char	time of activity

The items in the tables above are linked in a WRDS web query using a four step process:

1. The demographics table is linked to session table using 'machine_id'
2. The domain name lookup table is joined to the traffic and transaction tables on 'domain_id'
3. The session table is linked to the transaction table using 'site_session_id' ('machine_id' is also provided in the transaction table for convenience, but not needed to link the tables)
4. The product category lookup table is linked with the transaction table using 'prod_category_id'

NOTE: All .txt files are tab delimited

Most Educated Head of Household

0	Less than a high school diploma
1	High school diploma or equivalent
2	Some college but no degree
3	Associate degree
4	Bachelor's degree
5	Graduate degree
99	Missing

Household Income

1	Less than 15k
2	15k-24.999k
3	25k-34.999k
4	35k-49.999k
5	50k-74.999k
6	75k-99.999k
7	100k+

Age of Eldest Head of Household and Age of User

1	18-20
2	21-24
3	25-29
4	30-34
5	35-39
6	40-44
7	45-49
8	50-54
9	55-59
10	60-64
11	65 and over

Household Size

1	1
2	2
3	3
4	4
5	5
6	6+

Racial Background

1	White
2	Black
3	Asian
5	Other

Connection Speed

0	Not broadband
1	Broadband

Census Region of Residence

1	Northeast
2	North Central
3	South
4	West

Country of Origin

1	Hispanic
0	Non-Hispanic

Child Present

0	No
1	Yes

Appendix 2: Matrix Transformation Code

cleaning.py

```
#####  
# Jack Deschler  
# Data Cleaning for Senior Thesis  
# cleans comScore data  
#####  
import sys  
import pandas as pd  
import numpy as np  
  
def process_data(csv):  
    token = csv[:-4]  
    df = pd.read_csv(csv)  
    # extract demographics, save separately for re-linking later  
    df_demos = df[['machine_id', 'hoh_most_education', 'census_region',  
                  'household_size', 'hoh_oldest_age', 'household_income',  
                  'children', 'racial_background', 'connection_speed',  
                  'country_of_origin', 'zip_code']]  
    df_demos = df_demos.drop_duplicates('machine_id')  
    df_demos.to_csv(token + '_demographics.csv', index = False)  
    # drop columns we don't need, and demos, bc we already saved those  
    df = df.drop(['site_session_id', 'domain_id', 'ref_domain_name', 'duration',  
                'tran_flg', 'hoh_most_education', 'census_region',  
                'household_size', 'hoh_oldest_age', 'household_income',  
                'children', 'racial_background', 'connection_speed',  
                'country_of_origin', 'zip_code'], axis = 1)  
    # use pivot to get the data in the format we want  
    df = df.pivot_table(index='machine_id', columns='domain_name', values='pages_viewed',  
                        aggfunc = np.sum)  
    df = pd.DataFrame(df.to_records())  
    df = df.fillna(0)  
    # merge demographics back, and write final  
    final = df.merge(df_demos, on = 'machine_id')  
    final.to_csv(token + '_cleaned.csv', index = False)  
  
process_data(sys.argv[1])
```

cleaning_manual.py

```
# Data Cleaning for Senior Thesis, manual  
# Jack Deschler  
  
import pandas as pd  
import sys  
  
def process_data(csv):  
    token = csv[:-4]  
    cols = ['machine_id', 'domain_name', 'pages_viewed']  
    df = pd.read_csv(csv, usecols=cols, encoding = "ISO-8859-1")  
    machines = df['machine_id'].unique()  
    res = pd.DataFrame(index=machines)  
    # process data frame row by row instead of all at once
```

```
# saves memory at the cost of speed
for m in machines:
    df_m = df.loc[df['machine_id'] == m]
    for _index, row in df_m.iterrows():
        site = row['domain_name']
        if site not in list(res):
            # add the column if necessary
            res[site] = 0
            # now add to the cell
            res.at[m, site] += row['pages_viewed']
res.columns = res.columns.astype(str)
res.reindex(sorted(res.columns), axis=1)
res.to_csv(token + '_cleaned_manual.csv', index_label='machine_id')

process_data(sys.argv[1])
```

Appendix 3: Party Imputation Code

```
nc_party_imputation.py
#####
# Jack Deschler, Senior Thesis
# code to impute party characteristics into comScore data
# only for use with North Carolina voter files
#####

import pandas as pd
import sys

# from https://stackoverflow.com/questions/7088009/python-try-except-as-an-expression
# wrapper for Try-Except calls
def try_except(success, arg, failure, *exceptions):
    try:
        return success(arg) if callable(success) else success
    except exceptions or Exception:
        return failure() if callable(failure) else failure

# in NC, we can match on zip code, race, and age
# have to use the comScore coding (complicated wrt age)
def impute_demos(vf, demos):
    # first do some preprocessing on the voter file
    vf['race_code'] = vf['race_code'].apply(lambda x: 'O' if x in ['I','O','U','M'] else x)
    # have the upper level of age ranges for the switch case below
    age_codes = {0: 0, 1: 20, 2: 24, 3: 29, 4: 34, 5: 39, 6: 44, \
                 7: 49, 8: 54, 9: 59, 10: 64, 11: 120}
    race_codes = {1: 'W', 2: 'B', 3: 'A', 5: 'O'}
    count = 0
    for idx, row in demos.iterrows():
        count += 1
        if count % 100 == 0:
            print(str(count) + '/' + str(len(demos)))
        # by zip code
        curr = vf[vf['zip_code'] == row['zip_code']]
        # by race
        curr = curr[curr['race_code'] == race_codes[row['racial_background']]]
        # by age
        age_code = row['hoh_oldest_age']
        curr = curr[curr['birth_age'].between(age_codes[age_code-1]+1, age_codes[age_code])]
        # now calculate the statistics
        demos.loc[idx, 'vf_k'] = len(curr)
        f = lambda s: curr['party_cd'].value_counts()[s]
        d_count = try_except(f, 'DEM', 0, KeyError)
        r_count = try_except(f, 'REP', 0, KeyError)
        k = d_count + r_count
        demos.loc[idx, 'vf_k_2p'] = k
        demos.loc[idx, 'D_pct'] = 0 if k == 0 else d_count / len(curr)
        demos.loc[idx, 'D_pct_2p'] = 0 if k == 0 else d_count / (d_count + r_count)
    return demos

def main():
    if len(sys.argv) != 3:
```

```

    print('Usage: python nc_party_imputation.py NC_voterfile.txt NC_demos.csv')
    exit(1)
# read in voter file
vf = pd.read_csv(sys.argv[1], sep='\t',\
                 usecols=['zip_code','race_code','party_cd','birth_age'],\
                 lineterminator='\n', encoding = "ISO-8859-1")
demos = pd.read_csv(sys.argv[2])
demo_cols = list(demos)
# impute necessary columns
if 'D_pct' not in demo_cols:
    demos['D_pct'] = 0
if 'D_pct_2p' not in demo_cols:
    demos['D_pct_2p'] = 0
if 'vf_k' not in demo_cols:
    demos['vf_k'] = 0
if 'vf_k_2p' not in demo_cols:
    demos['vf_k_2p'] = 0
# do the imputation in a separate function
demos = impute_demos(vf, demos)
demos.to_csv(sys.argv[2])
exit(0)

if __name__ == '__main__':
    main()

```

fl_party_imputation.py

```

#####
# Jack Deschler, Senior Thesis
# code to impute party characteristics into comScore data
# only for use with Florida voter files
#####

import pandas as pd
import sys

# from https://stackoverflow.com/questions/7088009/python-try-except-as-an-expression
# wrapper for Try-Except calls
def try_except(success, arg, failure, *exceptions):
    try:
        return success(arg) if callable(success) else success
    except exceptions or Exception:
        return failure() if callable(failure) else failure

# in FL, we can match on zip code, race, and age
# have to use the comScore coding (complicated wrt age)
def impute_demos(vf, demos):
    # first do some preprocessing on the voter file
    vf['race_code'] = vf['race_code'].apply(lambda x: 6 if x in [1, 6, 7, 9] else x)
    vf['race_code'] = vf['race_code'].apply(lambda x: 5 if x in [4, 5] else x)
    # have the upper level of age ranges for the switch case below
    age_codes = {0: 0, 1: 20, 2: 24, 3: 29, 4: 34, 5: 39, 6: 44, \
                 7: 49, 8: 54, 9: 59, 10: 64, 11: 120}
    race_codes = {1: 5, 2: 3, 3: 2, 5: 6}
    # fix birth date to birth age
    for idx, row in vf.iterrows():
        idx_tmp = idx

```

```

    bd = row['birth_date']
    if type(bd) == float or bd == '*':
        vf.at[idx, 'birth_age'] == -1
    else:
        vf.at[idx, 'birth_age'] = 2018 - int(bd[-4:])
count = 0
for idx, row in demos.iterrows():
    count += 1
    if count % 500 == 0:
        print(str(count) + '/' + str(len(demos)))
    # by zip code
    curr = vf[vf['zip_code'] == str(int(row['zip_code']))]
    # by race
    curr = curr[curr['race_code'] == race_codes[row['racial_background']]]
    # by age
    age_code = row['hoh_oldest_age']
    curr = curr[curr['birth_age'].between(age_codes[age_code-1]+1, age_codes[age_code])]
    # now calculate the statistics
    demos.loc[idx, 'vf_k'] = len(curr)
    f = lambda s: curr['party_cd'].value_counts()[s]
    d_count = try_except(f, 'DEM', 0, KeyError)
    r_count = try_except(f, 'REP', 0, KeyError)
    k = d_count + r_count
    demos.loc[idx, 'vf_k_2p'] = k
    demos.loc[idx, 'D_pct'] = 0 if k == 0 else d_count / len(curr)
    demos.loc[idx, 'D_pct_2p'] = 0 if k == 0 else d_count / (d_count + r_count)
return demos

def main():
    if len(sys.argv) != 3:
        print('usage: python fl_party_imputation.py VF demos')
        exit(1)
    # read in the voter file, name columns appropriately
    vf = pd.read_csv(sys.argv[1], sep = '\t', lineterminator='\n', encoding = "ISO-8859-1", header =
None, names = ['zip_code', 'race_code', 'birth_date', 'party_cd'], usecols = [11, 20, 21, 23])
    demos = pd.read_csv(sys.argv[2])
    demo_cols = list(demos)
    # ensure necessary columns are in dataframe
    if 'D_pct' not in demo_cols:
        demos['D_pct'] = 0
    if 'D_pct_2p' not in demo_cols:
        demos['D_pct_2p'] = 0
    if 'vf_k' not in demo_cols:
        demos['vf_k'] = 0
    if 'vf_k_2p' not in demo_cols:
        demos['vf_k_2p'] = 0
    # do the imputation in a separate function
    demos = impute_demos(vf, demos)
    demos.to_csv(sys.argv[2], index = False)
    exit(0)

if __name__ == '__main__':
    main()

```

Appendix 4: Random Forest Modeling Code

race_model.py

```
#####  
# Jack Deschler, Senior Thesis  
# code to fit random forest models on comScore data  
# in order to predict race from browser history  
#####  
  
import numpy as np  
import pandas as pd  
import scipy as sp  
import argparse  
import gc  
import matplotlib  
matplotlib.use('TkAgg')  
import matplotlib.pyplot as plt  
import itertools  
from sklearn.linear_model import LogisticRegressionCV  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.metrics import confusion_matrix  
parser = argparse.ArgumentParser()  
from nmp_helpers import *  
  
# Fit the random Forest model!  
def fit_rf_model(df_train, df_test, demos = ['machine_id', 'hoh_most_education', 'census_region',  
      'household_size', 'hoh_oldest_age', 'household_income',  
      'children', 'connection_speed',  
      'country_of_origin', 'zip_code'], response = 'racial_background', n_estimators = 256,  
max_depth = 48):  
  
    rf_model = RandomForestClassifier(n_estimators = n_estimators, max_depth = max_depth)  
    print("fitting random forest model")  
    preds = list(set(list(df_train)) - set([response] + demos))  
  
    rf_model.fit(df_train[preds], df_train[response])  
  
    y_hat = rf_model.predict(df_test[preds])  
    print("Overall accuracy: {}".format(classification_accuracy(df_test['racial_background'].values,  
y_hat)))  
    # write feature importances console, top 10  
    impts = rf_model.feature_importances_  
    indices = np.argsort(impts)[::-1]  
    print("Top 10 Features")  
    for i in range(0,10):  
        print('{} . {}'.format(i+1, preds[indices[i]]))  
    return rf_model  
  
def fit_models(df_final):  
    train, test = split_data(df_final)  
    rf_model = fit_rf_model(train, test)  
    return {'rf': rf_model}
```

```

def main():
    parser.add_argument('-n', type=int, help='size of subsample')
    parser.add_argument('Sessions', help='comScore Sessions file')
    parser.add_argument('demos', help='comScore demographics file')
    args = parser.parse_args()
    n = 20000 if not args.n else args.n
    sample = get_subsample(args.Sessions, args.demos, n=n)
    fit_models(sample)
    exit(0)

if __name__ == '__main__':
    main()

```

party_model_rf.py

```

#####
# Jack Deschler, Senior Thesis
# code to fit random forest models on comScore data
# in order to predict partisanship from browsing history
#####

import argparse
import numpy as np
import pandas as pd
import scipy as sp
import gc
import itertools
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
parser = argparse.ArgumentParser()
from nmp_helpers import *

def fit_rf_model(df_train, df_test, demos = ['machine_id', 'hoh_most_education', 'census_region',
      'household_size', 'hoh_oldest_age', 'household_income',
      'children', 'connection_speed',
      'country_of_origin', 'racial_background', 'zip_code', 'vf_k', 'vf_k_2p', 'D_pct',
      'D_pct_2p'], response = 'democrat', n_estimators = 256, max_depth = 48):

    rf_model = RandomForestClassifier(n_estimators = n_estimators, max_depth = max_depth)
    print("fitting random forest model")
    preds = list(set(list(df_train)) - set([response] + demos))
    rf_model.fit(df_train[preds], df_train[response])

    y_hat = rf_model.predict(df_test[preds])
    print("Overall accuracy: {}".format(classification_accuracy(df_test['democrat'].values, y_hat)))
    conf_mat(df_test['democrat'].values, y_hat)
    # write feature importances console, top 10
    impts = rf_model.feature_importances_
    indices = np.argsort(impts)[-10:]
    print("Top 10 Features")
    for i in range(0,10):
        print('{} . {}'.format(i+1, preds[indices[i]]))
    return rf_model

```

```

def fit_models(df_final):
    train, test = split_data(df_final)
    rf_model = fit_rf_model(train, test)
    return {'rf': rf_model}

def main():
    parser.add_argument("-tp", "--twoparty", help='run on 2 party stats', action='store_true')
    parser.add_argument("Sessions", help = "comScore Sessions for NC")
    parser.add_argument("demos", help = "NC users demographic data")
    parser.add_argument("-n", type=int, help = 'number to subsample')
    parser.add_argument("-l", "--threshold", type=float, help = 'threshold if not 0.8 for D
classification')
    args = parser.parse_args()
    n = -1 if not args.n else args.n
    threshold = 0.8 if not args.threshold else args.threshold
    sample = get_subsample(args.Sessions, args.demos, n = n, party = True)
    sample = classify_party(sample, threshold = threshold, two_party = args.twoparty)
    fit_models(sample)
    exit(0)

if __name__ == '__main__':
    main()

```

nmp_helpers.py

```

# Jack Deschler
# The New Machine Politics
# nmp_helpers.py
# contains functions used commonly in modeling code
# 3 sections:
# - Confusion Matrices and Accuracy
# - Party Classification
# - Subsampling
# - Matrix Transformation
# Dependencies:
import numpy as np
import pandas as pd
import scipy as sp
import gc
import itertools
import matplotlib
from sklearn.metrics import confusion_matrix
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

##### Confusion Matrices and Accuracy #####

# classification_accuracy(y_true, y_pred)
# returns the percentage matched between y_pred and y_true
# which is the overall classification accuracy of a model
def classification_accuracy(y_true, y_pred, party = False):
    plot_conf_mat(y_true, y_pred)
    total_missed = 0
    for i in range(len(y_true)):
        if y_true[i] != y_pred[i]:
            total_missed += 1

```

```

    return 1 - (total_missed/len(y_true))

# conf_mat(y_real, y_pred)
# prints the confusion matrix to the console
# only works for 2-class classifiers
def conf_mat(y_real, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_real, y_pred).ravel()

    print("\nCONFUSION MATRIX:")
    print("{0:6} {1:6} {2:6}".format("", "pred +", "pred -"))
    print("{0:6} {1:6} {2:6}".format("true +", tp, fn))
    print("{0:6} {1:6} {2:6}".format("true -", fp, tn))

    if tp == 0 and fp == 0:
        tpr = 0
    else:
        tpr = tp / (tp + fp)

    if tn == 0 and fn == 0:
        tnr = 0
    else:
        tnr = tn / (tn + fn)

    print("\nTRUE POSITIVE:", tpr)
    print("TRUE NEGATIVE:", tnr)

# plot_conf_mat to plot the confusion matrix
# adapted from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py
def plot_conf_mat(y_true, y_pred, classes=["White", "Black", "Asian", "Other"],
                 normalize = False, title = 'Confusion matrix', cmap = plt.cm.Blues, party = False):
    cm = confusion_matrix(y_true, y_pred)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)
    if party:
        classes = ['Nondemocrat', 'Democrat']
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=15)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")

```

```

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.savefig('confmat.png')
plt.clf()

##### Party Classification #####
# classify_party
# Adds column to dataframe for party classification
# 'democrat': 1 if classified as a democrat, 0 if not
# Inputs:
# threshold: value at which D_pct[_2p] must be to classify as D
# default set at 0.8 as per Eitan D. Hersh in 'Hacking the Electorate'
# 2p: boolean, True if to use 2 party values
# Returns:
# dataframe with the 'democrat' column
def classify_party(df, threshold = 0.8, two_party = True):
    comp_col = 'D_pct_2p' if two_party else 'D_pct'
    df['democrat'] = df.apply(lambda row: 1 if row[comp_col] > threshold else 0, axis = 1)
    return df

##### Subsampling #####
# get_subsample(sessions, demos, n)
# returns a subsample of the sessions dataframe of size n, if n is set
# samples by machine number, so we get n users
# if n is set, uses subsample() to weight race a certain way (see below)
# if n is not set, simply uses the entire set of demographics
# calls respective transform mat, collects garbage and returns transformed matrix
def get_subsample(sessions, demos, n = -1, targets = [{1: .766, 2: .134, 3: .058, 5: .042}], party =
False):
    # get the subsample
    demos = pd.read_csv(demos)
    demos_sample = demos if n == -1 else subsample(n, demos, targets = targets)
    subsample_numbers = demos_sample['machine_id']

    # read in the actual file and clean it here
    df_full = pd.read_csv(sessions)
    sample = df_full[df_full['machine_id'].isin(subsample_numbers)]
    df_final = transform_mat_party(sample) if party else transform_mat(sample)

    # force memory garbage collection
    demos = None
    df_full = None
    sample = None
    gc.collect()
    return df_final

# splits DataFrame into testing and training sets
def split_data(df, threshold = 0.8):
    msk = np.random.rand(len(df)) < threshold

    data_train = df[msk]
    data_test = df[~msk]

    return (data_train, data_test)

# subsample: returns a subsample of a dataframe weighted by certain targets

```

```

# IN:
# n:    number of rows to be in subsample
# df:   dataframe to sample from
# demos: list of demographics to weight on (comScore data column names)
# targets: list of dictionaries of weights (comScore code: weight)
# defaults set to bet census weights
# OUT:
# final dataframe subsample
def subsample(n, df, demos = ['racial_background'], targets = [{1: .766, 2: .134, 3: .058, 5:
.042}]):
    # start with code for just one demographic axis
    demo = demos[0]
    target = targets[0]
    keys = list(target.keys())
    dfs = {k: df[df[demo] == k] for k in keys}

    cols = list(df)
    agg = pd.DataFrame(columns = cols)

    for k in keys:
        n_k = round(target[k] * n)
        agg = agg.append(dfs[k].sample(n_k))
    return agg

##### Matrix Transformation #####
# transforms comscore sessions into usable matrix
# transform_mat for race, transform_mat_party for party
def transform_mat(df):
    # extract demographics, save separately for re-linking later
    df_demos = df[['machine_id', 'hoh_most_education', 'census_region',
                    'household_size', 'hoh_oldest_age', 'household_income',
                    'children', 'racial_background', 'connection_speed',
                    'country_of_origin', 'zip_code']]
    df_demos = df_demos.drop_duplicates('machine_id')
    # drop columns we don't need, and demos, bc we already saved those
    df = df.drop(['hoh_most_education', 'census_region',
                  'household_size', 'hoh_oldest_age', 'household_income',
                  'children', 'racial_background', 'connection_speed',
                  'country_of_origin', 'zip_code'], axis = 1)
    try:
        df = df.drop(['site_session_id', 'domain_id', 'ref_domain_name', 'duration', 'tran_flg'], axis =
1)
    except KeyError:
        pass
    # use pivot to get the data in the format we want
    df = df.pivot_table(index='machine_id', columns='domain_name', values='pages_viewed',
aggfunc = lambda x: np.sum(x).astype(bool).astype(int))
    df = pd.DataFrame(df.to_records())
    df = df.fillna(0)
    print("{} columns in the transformed matrix".format(len(list(df))))
    # merge demographics back, and write final
    final = df.merge(df_demos, on = 'machine_id')
    return final

def transform_mat_party(df):

```

```

# extract demographics, save separately for re-linking later
# key difference is we must include the imputed columns (vf_k, etc.)
df_demos = df[['machine_id', 'hoh_most_education', 'census_region',
              'household_size', 'hoh_oldest_age', 'household_income',
              'children', 'racial_background', 'connection_speed',
              'country_of_origin', 'zip_code', 'D_pct', 'D_pct_2p',
              'vf_k', 'vf_k_2p']]
df_demos = df_demos.drop_duplicates('machine_id')
# drop columns we don't need, and demos, bc we already saved those
df = df.drop(['hoh_most_education', 'census_region',
             'household_size', 'hoh_oldest_age', 'household_income',
             'children', 'racial_background', 'connection_speed',
             'country_of_origin', 'zip_code'], axis = 1)
# use pivot to get the data in the format we want
df = df.pivot_table(index='machine_id', columns='domain_name', values='pages_viewed',
                    aggfunc = lambda x: np.sum(x).astype(bool).astype(int))
df = pd.DataFrame(df.to_records())
df = df.fillna(0)
print("{} columns in the transformed matrix".format(len(list(df))))
# merge demographics back, and write final
final = df.merge(df_demos, on = 'machine_id')
return final

```

Appendix 5: Exclusivity Index Model Code

ei_model.py

```
#####
# Jack Deschler, Senior Thesis
# code to fit models on comScore data
# based on Zang/Sweeney's Exclusivity Indices
#####

import numpy as np
import pandas as pd
import scipy as sp
import argparse
import gc
import matplotlib
import operator
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import itertools
from collections import Counter
from sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import confusion_matrix
parser = argparse.ArgumentParser()
from nmp_helpers import *

# global for party, needed to factor some pieces of code out
global party_bool

##### EXCLUSIVITY INDEX MODELING
#####

# calc_exclusivity() and calc_exclusivity_v2()
# inputs:
# df: the sample dataframe, with both demographics and matrix transformation done
# axis: racial axis to calculate indices on
# n: number of top exclusivity indices to return
# returns:
# dictionary of axis codes: top n domains by exclusivity index, over 95%, and by # of visits
# use calc_exclusivity_v2, calc_exclusivity is deprecated

def calc_exclusivity(df, axis, n = 100, outfile = 'exclusivity_indices.csv', threshold = 0.9):
    codes = list(np.unique(df[axis].values))
    domains = set(list(df)).difference(set(['machine_id', 'hoh_most_education', 'census_region',
'household_size', 'hoh_oldest_age', 'household_income', 'children', 'racial_background',
'connection_speed', 'country_of_origin', 'zip_code']))
    domains = list(domains)
    ph = [0] * len(domains)
    final_df = pd.DataFrame({'domain': domains, 'visits': ph})
    for c in codes:
        final_df[c] = ph
    counter = 0
    for idx, row in final_df.iterrows():
```

```

d = row['domain']
visits = df[d].sum()
fracs = {c: len(df[df[axis] == c][d].nonzero()[0])/len(df[df[axis] == c]) for c in codes}
denom = sum(fracs.values())
for c in codes:
    final_df.at[idx, c] = fracs[c]/denom
final_df.at[idx, 'visits'] = visits
counter += 1
if counter % 10000 == 0:
    print("{} / {} domains processed".format(counter, len(domains)))
if outfile:
    final_df.to_csv(outfile, index = False)
final = {c: list(final_df[final_df[c] > threshold].sort_values(by=['visits'], ascending =
False)['domain'])[:n] for c in codes}
return final

def calc_exclusivity_v2(df, axis, n = 100, outfile = 'exclusivity_indices.csv', threshold = 0.7):
    # must get the domains in a usable format
    codes = list(np.unique(df[axis].values))
    domains = set(list(df)).difference(set(['machine_id', 'hoh_most_education', 'census_region',
'household_size', 'hoh_oldest_age', 'household_income', 'children', 'racial_background',
'connection_speed', 'country_of_origin', 'zip_code']))
    if axis == 'democrat':
        domains = domains.difference(set(['vf_k', 'vf_k_2p', 'D_pct', 'D_pct_2p', 'democrat']))
    domains = list(domains)
    ph = [0.] * len(domains)
    visits = {c: df[df[axis] == c].sum() for c in codes}
    lengths = {c: len(df[df[axis] == c]) for c in codes}
    lengths['total'] = len(df)
    visits_df = pd.DataFrame({'domain': domains, 'visits': ph})
    for c in codes:
        visits_df[c] = ph
    counter = 0
    # now that we have the domains in a df, calculate the numerator for each domain for each group
    for idx, row in visits_df.iterrows():
        tot = 0
        d = row['domain']
        for c in codes:
            tmp = float(visits[c][d])
            tmp2 = tmp/float(lengths[c])
            visits_df.at[idx, c] = tmp/float(lengths[c])
            tot += tmp
        visits_df.at[idx, 'visits'] = tot
        counter += 1
    # have visit fractions, must process them now
    visits_df['tot'] = visits_df.drop(['visits', 'domain'], axis = 1).sum(axis = 1)
    for c in codes:
        visits_df[c] = visits_df[c] / visits_df['tot']
    if outfile:
        visits_df.to_csv(outfile, index = False)
    final = {c: list(visits_df[visits_df[c] > threshold].sort_values(by=['visits'], ascending =
False)['domain'])[:n] for c in codes}
    if axis == 'democrat':
        final_df = pd.DataFrame.from_dict(final)
        final_df.to_csv('party_eis_{}.csv'.format(int(threshold*10)), index = False)
    return final

```

```

# ei_classifier
# IN:
# eis: dataframe of exclusivity indices by race or party
# df: the df to test on (must be the testing set)
# outcome: the outcome variable, racial_background or democrat
# mod: whether or not to use the modified criterion
# RETURNS:
# Nothing, but saves confusion matrix and prints accuracies
def ei_classifier(eis, df, outcome, mod = False):
    y_true = df[outcome].values
    df['pred'] = 0
    counter = 0
    y_hat_classified = []
    y_true_classified = []
    # check each browsing history for the presence of the exclusive domains
    for idx, row in df.iterrows():
        counts = {c: 0 for c in list(eis)}
        for c in list(eis):
            domains = eis[c]
            for d in range(len(domains)):
                domain = domains[d]
                try:
                    # order by rank if using the modified criterion
                    if row[domain] > 0:
                        counts[c] += (101 - d) if mod else 1
                except KeyError:
                    pass
            if sum(list(counts.values())) == 0:
                counter += 1
            classify = max(counts.items(), key=operator.itemgetter(1))[0]
            classify = int(classify)
            df.at[idx, 'pred'] = classify
            # actually make the classification
            if sum(list(counts.values())) > 0:
                y_hat_classified += [classify]
                y_true_classified += [df.at[idx, outcome]]
    y_hat = df['pred'].values
    party = False
    if outcome == 'democrat':
        global party_bool
        party_bool = True
    print("{} had none of the domains".format(counter))
    print("Overall accuracy (among classified):
    {}".format(classification_accuracy(y_true_classified, y_hat_classified)))

def main():
    parser.add_argument('-n', type=int, help='size of subsample')
    parser.add_argument('Sessions', help='comScore Sessions file')
    parser.add_argument('demos', help='comScore demographics file')
    parser.add_argument('-o', '--outfile', help='slug for outfile for exclusivity indices')
    parser.add_argument('-f', '--infile', help='file to read in EIs')
    parser.add_argument('axis', help='axis to calculate on')
    parser.add_argument('-m', '--modified', help='use modified criterion', action='store_true')
    parser.add_argument('-t', '--threshold', help='D_pct[_2p] threshold')
    parser.add_argument('-e', '--ei_threshold', help='EI sorting threshold')

```

```

args = parser.parse_args()
mod = False if not args.modified else True
ei_thresh = 0.8 if not args.ei_threshold else float(args.ei_threshold)
n = 20000 if not args.n else args.n
outcome = args.axis
threshold = 0.8 if not args.threshold else float(args.threshold)
out = 'exclusivity_indices.csv' if not args.outfile else str(args.outfile)
excl_indices = None
# deal with race and party modeling differently
if args.axis == 'D_pct' or args.axis == 'D_pct_2p':
    # no reading from a file of EIs for party
    # first get the demos and impute party
    demos = pd.read_csv(args.demos)
    subsample_numbers = demos['machine_id']

    # read in the actual file and clean it here
    df_full = pd.read_csv(args.Sessions)
    sample = df_full[df_full['machine_id'].isin(subsample_numbers)]
    df_final = transform_mat_party(sample)
    demos = None
    df_full = None
    sample = None
    gc.collect()
    tp = False
    if args.axis == 'D_pct_2p':
        tp = True
    sample = classify_party(df_final, threshold = threshold, two_party = tp)
    # split data and calculate EIs from training set
    train, sample = split_data(sample)
    excl_indices = calc_exclusivity_v2(train, 'democrat', n = 100, outfile = None, threshold =
ei_thresh)
    outcome = 'democrat'
# race modeling
else:
    # we must re-calculate indicex
    if not args.infile:
        eis = []
        ei_df = pd.DataFrame()
        # pseudo 10-fold validation
        for i in range(10):
            sample = None
            gc.collect()
            print(i+1)
            sample = get_subsample(args.Sessions, args.demos, n=n)
            eis.append(calc_exclusivity_v2(sample, outcome, n = 100, outfile = None, threshold =
ei_thresh))
        for c in eis[0].keys():
            l = [a[c] for a in eis]
            l = [a for b in l for a in b]
            counts = dict(Counter(l))
            counts = sorted(counts.items(), key=lambda kv: kv[1])
            counts = [c for (c, _) in counts[:100]]
            tmp_df = pd.DataFrame({c:counts})
            ei_df = pd.concat([ei_df, tmp_df], axis=1)
        ei_df = ei_df.fillna(value = "")
        ei_df.to_csv(out, index = False)

```

```
    excl_indices = ei_df
else:
    excl_indices = pd.read_csv(args.infile)
print("Getting testing subsample")
sample = None
gc.collect()
sample = get_subsample(args.Sessions, args.demos, n = n)
ei_classifier(excl_indices, sample, outcome, mod = mod)
exit(0)

if __name__ == '__main__':
    main()
```

Appendix 6: Cross-state Modeling Code

```
cross_state_models.py
#####
# Jack Deschler, Senior Thesis
# cross state modeling code
# fits random forest and exclusivity index models
# between two states, and combined between states
# works with any two slugs as long as:
# demos csv path is in form ../demos/X_demos.csv
# sessions csv path is in form ../X_Sessions.csv
#####
import argparse
import os
import numpy as np
import pandas as pd
import scipy as sp
import gc
import itertools
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from party_model_rf import fit_rf_model
from ei_model import *
from nmp_helpers import *
parser = argparse.ArgumentParser()

# split_by_state(demos1, demos2, df)
# IN:
# demos1: demographic list for the training set
# demos2: demographic list for the testing set
# df: combined transformed comscore sessions data frame
# RETURN:
# train: training dataset, pertaining to demos1
# test: testing dataset, pertaining to demos2
def split_by_state(demos1, demos2, df):
    train_machines = demos1['machine_id']
    test_machines = demos2['machine_id']
    train = df[df['machine_id'].isin(train_machines)]
    test = df[df['machine_id'].isin(test_machines)]
    return (train, test)

def main():
    parser.add_argument("state1", help='first state')
    parser.add_argument("state2", help='second state')
    args = parser.parse_args()
    state1 = args.state1
    state2 = args.state2
    demos1 = pd.read_csv('../demos/' + state1 + '_demos.csv')
    demos2 = pd.read_csv('../demos/' + state2 + '_demos.csv')
    sessions1 = pd.read_csv('../' + state1 + '_Sessions.csv')
```

```

sessions2 = pd.read_csv('../ + state2 + '_Sessions.csv')
sessions = pd.concat([sessions1, sessions2], sort = False)
sessions1 = None
sessions2 = None
gc.collect()
# add cols so transform_mat doesn't break
sessions['site_session_id'] = 0
sessions['domain_id'] = 0
sessions['ref_domain_name'] = 0
sessions['duration'] = 0
sessions['tran_flg'] = 0
sessions = transform_mat_party(sessions)
sessions = classify_party(sessions, threshold = 0.5, two_party = True)

# State 1 on State 2
print(state1 + ' on ' + state2)
train, test = split_by_state(demos1, demos2, sessions)
eis = calc_exclusivity_v2(train, 'democrat', n = 100, outfile = None, threshold = 0.9)
print("EI original criterion")
ei_classifier(eis, test, 'democrat', mod = False)
os.rename('confmat.png', 'combo_confmat/' + state1 + 'on' + state2 + '_ei_orig.png')
print("EI modified criterion")
ei_classifier(eis, test, 'democrat', mod = True)
os.rename('confmat.png', 'combo_confmat/' + state1 + 'on' + state2 + '_ei_mod.png')
print("RF Model")
fit_rf_model(train, test)
os.rename('confmat.png', 'combo_confmat/' + state1 + 'on' + state2 + '_rf.png')
print()

# State 2 on State 1
print(state2 + " on " + state1)
train, test = split_by_state(demos2, demos1, sessions)
eis = calc_exclusivity_v2(train, 'democrat', n = 100, outfile = None, threshold = 0.9)
print("EI original criterion")
ei_classifier(eis, test, 'democrat', mod = False)
os.rename('confmat.png', 'combo_confmat/' + state2 + 'on' + state1 + '_ei_orig.png')
print("EI modified criterion")
ei_classifier(eis, test, 'democrat', mod = True)
os.rename('confmat.png', 'combo_confmat/' + state2 + 'on' + state1 + '_ei_mod.png')
print("RF Model")
fit_rf_model(train, test)
os.rename('confmat.png', 'combo_confmat/' + state2 + 'on' + state1 + '_rf.png')
print()

# Combined Model
print("Combined Model")
train, test = split_data(sessions)
eis = calc_exclusivity_v2(train, 'democrat', n = 100, outfile = None, threshold = 0.9)
print("EI original criterion")
ei_classifier(eis, test, 'democrat', mod = False)
os.rename('confmat.png', 'combo_confmat/combo_ei_orig.png')
print("EI modified criterion")
ei_classifier(eis, test, 'democrat', mod = True)
os.rename('confmat.png', 'combo_confmat/combo_ei_mod.png')
print("RF Model")
fit_rf_model(train, test)

```

```
os.rename('confmat.png', 'combo_confmat/combo_rf.png')
```

```
if __name__ == '__main__':  
    main()
```

Bibliography

- Allison, Paul. "Convergence Failures in Logistic Regression." SAS Global Forum 2008. 360.
- Ansolabehere, Stephen, Shanto Iyengar, Adam Simon, and Nicholas Valentino. "Does Attack Advertising Demobilize the Electorate?" *The American Political Science Review* (1927) 88, no. 4 (1994): 829-838.
- Barbu, Oana. "Advertising, Microtargeting and Social Media." *Procedia - Social and Behavioral Sciences* 163, no. C (2014): 44-49.
- Beauchamp, Nicholas. "Predicting and Interpolating State-Level Polls Using Twitter Textual Data." *American Journal of Political Science* 61, no. 2 (2017): 490-503.
- Buolamwini, Joy, and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification." *Proceedings of Machine Learning Research* 81, (2018). 1-15.
- Carmichael, Jason, T. Brulle, and Robert Huxster. "The Great Divide: Understanding the Role of Media and Other Drivers of the Partisan Divide in Public Concern over Climate Change in the USA, 2001–2014." *Climatic Change* 141, no. 4 (2017): 599-612.
- Cohen, Marty, et al. *The Party Decides: Presidential Nominations Before and After Reform*. Chicago: University of Chicago Press, 2008.
- Condorcet, Jean-Antoine-Nicolas De Caritat, Iain. McLean, and Fiona Hewitt. *Condorcet: Foundations of Social Choice and Political Theory*. Aldershot, Hants, England ; Brookfield, Vt.: E. Elgar, 1994.

- Dancey, Logan, and Geoffrey Sheagley. "Partisanship and Perceptions of Party-Line Voting in Congress." *Political Research Quarterly* 71, no. 1 (2018): 32-45.
- Daries, Jon, Justin Reich, Jim Waldo, Elise Young, Jonathan Whittinghill, Andrew Ho, Daniel Seaton, and Isaac Chuang. "Privacy, Anonymity, and Big Data in the Social Sciences." *Communications of the ACM* 57, no. 9 (2014): 56-63.
- Enns, Peter K., Paul M. Kellstedt, and Gregory E. McAvoy. "The Consequences of Partisanship in Economic Perceptions." *Public Opinion Quarterly* 76, no. 2 (2012): 287-310.
- Enos, Ryan, and Anthony Fowler. "Aggregate Effects of Large-Scale Campaigns on Voter Turnout." *Political Science Research and Methods* 6, no. 4 (2018): 733-51.
- Eren, Ozkan, and Naci Mocan. "Emotional Judges and Unlucky Juveniles." *American Economic Journal: Applied Economics* 10, no. 3 (2018): 171-205.
- Gerber, Alan S. "The Adoption of the Secret Ballot." Ph.D. dissertation, Massachusetts Institute of Technology. 1994.
- Gerber, Alan S, Gregory A Huber, and Ebonya Washington. "Party Affiliation, Partisanship, and Political Beliefs: A Field Experiment." *American Political Science Review* 104, no. 4 (2010): 720-44.
- Gerber, Alan S., Gregory A. Huber, David Doherty, Conor M. Dowling, and Seth J. Hill. "Do Perceptions of Ballot Secrecy Influence Turnout? Results from a Field Experiment." *American Journal of Political Science* 57, no. 3 (2013): 537-51.
- Gosnell, Harold Foote. *Machine Politics: Chicago Model*. New York: Greenwood Press, 1968.

- Hamilton, Alexander, James Madison, John Jay, Elizabeth Cobbs, and Louis Fisher. *The Federalist Papers and the Constitution of the United States: The Principles of the American Government*. New York: Skyhorse, 2016.
- Hersh, Eitan D. *Hacking the Electorate: How Campaigns Perceive Voters*. New York, NY: Cambridge University Press, 2015.
- Highton, Benjamin, and Cindy D. Kam. "The Long-Term Dynamics of Partisanship and Issue Orientations." *The Journal of Politics* 73, no. 1, 202-15.
- Kalla, Joshua L., and David E. Broockman. "The Minimal Persuasive Effects of Campaign Contact in General Elections: Evidence from 49 Field Experiments." 112, no. 1 (2018): 148-66.
- Keyssar, Alexander. *The Right to Vote: The Contested History of Democracy in the United States*. Rev. ed. New York: Basic Books, 2009.
- Kleinberg, Jon, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. "Human Decisions and Machine Predictions." *The Quarterly Journal of Economics* 133, no. 1 (2017): 237-93.
- Machanavajjhala, Ashwin, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. "L-diversity: Privacy beyond K-anonymity." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, no. 1 (2007): 3-Es.
- Mackuen, Michael B., Robert S. Erikson, and James A. Stimson. "Peasants or Bankers? The American Electorate and the U.S. Economy." *The American Political Science Review* 86, no. 3 (1992): 597-611.
- McCain, John. *Political Ad: Chicago Machine (McCain)*. New York: N/a, 2008.

- McCarty, Nolan, Keith T. Poole, and Howard Rosenthal. "Does Gerrymandering Cause Polarization?" *American Journal of Political Science* 53, no. 3 (2009): 666-80.
- McGhee, Eric, Seth Masket, Boris Shor, Steven Rogers, and Nolan McCarty. "A Primary Cause of Partisanship? Nomination Systems and Legislator Ideology." *American Journal of Political Science* 58, no. 2 (2014): 337-51.
- Mladenka, Kenneth R. "The Urban Bureaucracy and the Chicago Political Machine: Who Gets What and the Limits to Political Control." *The American Political Science Review* 74, no. 4 (1980): 991-98.
- Narayanan, Arvind, and Vitaly Shmatikov. "How To Break Anonymity of the Netflix Prize Dataset." 2006.
- Nichter, Simeon. "Vote Buying or Turnout Buying? Machine Politics and the Secret Ballot." *American Political Science Review* 102, no. 1 (2008): 19-31.
- Riker, William. "The Two-party System and Duverger's Law - an Essay on the History of Political Science." In *Discipline and History - Political Science in the United States*, 345-62. 1993.
- Seabrook, Nicholas R. "The Obama Effect: Patterns of Geographic Clustering in the 2004 and 2008 Presidential Elections." *The Forum* 7, no. 2 (2009): The Forum, 2009, Vol.7(2).
- Schattschneider, E.E. *The Semisovereign People: A Realist's View of Democracy in America*. Hinsdale, Ill.: Dryden Press, 1975.
- Sweeney, Latanya. "Discrimination in Online Ad Delivery." *Queue* 11, no. 3 (2013): 10-29.

Sweeney, Latanya. "K-anonymity: A Model for Protecting Privacy." *International Journal Of Uncertainty Fuzziness And Knowledge-Based Systems* 10, no. 5 (2002): 557-70.

Szwarcberg, Mariela Laura. *Mobilizing Poor Voters: Machine Politics, Clientelism, and Social Networks in Argentina*. Structural Analysis in the Social Sciences; 38. New York, 2015.

Washington, George. *George Washington Papers, Series 2, Letterbooks -1799: Letterbook 24, April 3, 1793 - March 3, 1797*. 1793. Manuscript/Mixed Material. <https://www.loc.gov/item/mgw2.024/>.

Webster, James G., and Ksiazek, Thomas B. "The Dynamics of Audience Fragmentation: Public Attention in an Age of Digital Media." *Journal of Communication* 62, no. 1 (2012): 39-56.

Datasets

2016 comScore WRDS dataset. Used at permission of Prof. Latanya Sweeney, Harvard Department of Government. comScore, Inc.

Florida Voter Registration and Voting History Extract File, February 2019. Used at request from Florida Secretary of State office. Redacted March 5, 2019.

Free Zipcode Database. Last updated January 22, 2012. Last accessed March 26, 2019. Online at: <http://federalgovernmentzipcodes.us>.

North Carolina Statewide Voter File. Last updated: May 1, 2017. North Carolina State Board of Elections. Last accessed March 26, 2019. Online at: https://s3.amazonaws.com/dl.ncsbe.gov/data/ncvoer_Statewide.zip