PH125.9x Data Science Capstone

Predicting Federal District Court Outcomes

Adam C. Riley

11/30/2020

Contents

1 Executive Summary **2**

1.1 Artificial Intelligence in the Legal Field . . . 2

1.2 Data . . . 3

1.2.1 Dependent Variable . . . 3

1.2.2 Independent Variables . . . 3

1.3 Process . . . 3

1.4 Goal . . . 4

2 Data Analysis **4**

2.1 Data Import & Cleaning . . . 4

2.1.1 Tables found in the codebook: . . . 4

2.1.2 Non-Carp-Manning Data . . . 6

2.2 Data Exploration & Visualization . . . 6

2.2.1 Jurisdiction . . . 7

2.2.2 Judge Demographics . . . 9

2.2.3 Appointment Demographics . . . 13

2.2.4 Tenure . . . 15

2.2.5 Subject Matter . . . 16

2.2.6 Limitations . . . 17

2.3 Data Analysis Conslusions . . . 18

3 Supervised Learning Methods **18**

3.1 Modeling Approach . . . 18

3.1.1 Cross Validation of Training Data . . . 18

3.2 Models & Algorithms . . . 18

3.2.1 Linear (Logisitic) Model . . . 18

3.2.2 Naive Bayes . . . 19
3.2.3 Randon Forest . . . 20
3.2.4 Ensemble . . . 21

4 Results **21**
4.1 Metrics . . . 21
4.1.1 Accuracy . . . 22
4.1.2 F1 score . . . 22
4.1.3 Matthews correlation coefficient . . . 23
4.2 Performance . . . 23
4.2.1 Linear (Logisitic) . . . 23
4.2.2 Naive Bayes . . . 24
4.2.3 Random Forest . . . 25
4.2.4 Ensemble . . . 26

5 Conclusion **27**
5.1 Selected model . . . 27
5.2 Future Work . . . 27
5.3 Impact . . . 27

The R Markdown code used to generate this pdf document and associated files can be found my Github.

1 Executive Summary

1.1 Artificial Intelligence in the Legal Field

Artificial Intelligence is booming in the legal field as firms scramble to research more quickly and leverage technology to provide their clients with a more complete experience. Any edge that a firm can bring to the litigation landscape become a crucial factor in determining how they negotiate, allocate resources, and communicate with their clients.

This project will focus on assisting legal practitioners to predict the outcome of cases with social impact components. This litigation is often costly and can take years to complete. With predictive information, applied to the facts of their particular issue, they can either temper the expectations of a plaintiff with a case that will not likely be ruled in their favor or to be more aggressive with a defendant that is facing a potential loss and is looking for a way to mitigate the impact of the associated monetary award.

We are not aiming to create a model that can replace legal judgment, but will focus on a tool that can be utilized alongside traditional legal research to provide additional insights.

1.2 Data

Robert A. Carp (Professor of Political Science at the University of Houston) and Kenneth L. Manning (Professor of Political Science at the University of Massachusetts-Dartmouth) examined and collected a subset of the judicial opinions published in the *Federal Supplement* that contained a question of law with a traditional liberal-conservative dimension. The database contains more than 110,000 entries decided between 1927 and 2012. It is estimated that (as of 2016) approximately half of the decisions published in the *Federal Supplement* have been included in the collection.

The authors would read each decision and, if appropriate, coded various demographic information and cataloged the decision as either liberal or conservative using criteria that can be found in the database's companion codebook. If a case could not clearly be classified as either liberal or conservative of if there were multiple issues or multiple parties that made the determination uncertain, the case was not included in the database.

1.2.1 Dependent Variable

The dependent variable in the data is the ordinal-level liberal-conservative classification, "libcon." The variable has a value of either 0 (Conservative) or 1 (Liberal) and takes the form of a Bernoulli distribution.

1.2.2 Independent Variables

Independent variables include details about the judge authoring the opinion, information about the court, topical classifications about the subject matter of the case, and the date of ruling. Each of the independent variables has been encoded but can be decoded using the above referenced companion codebook. We will look further into each of the independent variables as we inspect and process the data below.

1.3 Process

I will follow the following steps to create the model for this project:

1. Data Analysis
    1. Data Import & Cleaning
    2. Data Exploration & Visualization
    3. Analysis Conclusion
2. Supervised Learning Methods
    1. Modeling Approach
    2. Models & Algorithms
3. Results
    1. Metrics
    2. Performance
4. Conclusion

1.4 Goal

This Capstone Project will use supervised machine learning to create a classifier based on data contained in the Carp Manning U.S. District Court Database that predicts the outcome of a certain United States Federal District Court cases as either liberal or conservative.

2 Data Analysis

2.1 Data Import & Cleaning

The Carp-Manning data is split between two different documents, so we need to import them both and combine them to be able to effectively explore the data in a meaningful way. The first document is the coded data set (we will utilize the excel version, although the data is available in several formats) and the second is the PDF companion codebook which we will use to decode and map the data to make it readable.

We will do a quick check of our files to see if there are locally available copies of these files and if there are not, we will automatically download them from the the University of Massachusetts-Dartmouth project site. The excel file can be imported directly as a tribble and the PDF document is initially converted into a character vector.

Now that we have both documents, we need to process the codebook and scrape the the tables within to be able to manipulate them. The first step is to inspect the codebook and identify the tables we want to target and note what pages they appear on. While doing that, I noticed that each row of each table is on its own line and each entry appears to be in similar format: a numerical identifier, then a dash, then decoded information. After cross referencing to check that the values within the codebook match exactly to the raw excel data and verifying the formats, everything looks promising. We can make a notes based on our visual inspect and can then create a function that will scan the pages we identified and collect the table information from them.

2.1.1 Tables found in the codebook:

crtpoint
Encoded Format: three numbers
Pages: 5, 6, 7, 8, 9, 10, 11, 12, 13

circuit
Encoded Format: two numbers
Page: 15

state
Encoded Format: two numbers
Pages: 16, 17

statdist
Encoded Format: three numbers
Pages: 17, 18, 19, 20

casetype
Encoded Format: two numbers

Pages: 24, 25

category
Encoded Format: one number
Page: 39

appres
Encoded Format: two number
Page: 43

demographics
Encoded Format: one numbers
Page: 44
Note: There are several tables on page 44 and we'll have to split the data in further processing

We are really lucky that there is minimal cross over between the pages. I noted above in the demographics block about page 44, but there are no other instances that will require further processing because where there are two tables on one page, the formats are different and can be parsed automatically.

I initially envisioned this as a three stage process:

Stage one: Split the character vector text by line to create the rows of the table
Stage two: split the rows using the dash to create the columns
Stage three: verify the data and return it

After some trial and error, I realized that the 'dash' character was not being scanned in a uniform way and needed to be cleaned so that any horizontal line between the formatted number and the definition would be converted to a 'dash' and stage two would work as intended. I also needed to strip white space from the beginning and end of the strings to make sure the verification at stage three matched as I expected. Finally, I also removed any rows with empty values.

We will name our function 'get__CM__tables':

```r
get_CM_tables <- function(CM_page, CM_format) {
    stage0 <- gsub("\\s[[:punct:]]+\\s)", " - ", CM_text[[CM_page]])
    stage0 <- gsub(" - ", " - ", CM_text[[CM_page]])
    stage1 <- str_split(stage0, "\n", simplify=TRUE)
    stage2 <- str_split(stage1, " - ", simplify=TRUE)
    stage2[,] <- trimws(stage2[,])
    stage3 <- stage2[which(grepl(CM_format, stage2[,1])),]
    stage3 <- as.data.frame(stage3[which(stage3[,2] != ''),])
    return(stage3)
}
```

Then we will loop over the function for each table using a vector of page numbers and a regular expression for the format of the number we are trying to decode. Once processed, we will combine the processed results from each page into a single table and then rename the columns using more descriptive titles.

These tables can be joined to the raw excel data to create a master data set with all Carp-Manning data.

2.1.2 Non-Carp-Manning Data

I added two additional variables to the table during processing. The first was a calculation of the difference between the year the judge was appointed and the the year the case opinion was written. My gut is tells me that the length of time a jurist has served on the bench may change how they rule either from the impact of age or additional experience. Second, I looked up and added the political party of the president that appointed the judge. This was done completely out of curiosity.

There are a lot of other additional classifiers that could be added to this data to improve the fit and accuracy of the model. States could be assigned different regions (which could be distinct from circuits), case types could be further sub-divided based on topic or assigned into sub-categories, or more demographic information could be added about the judges. These steps could add value, especially subdividing the topics, and should be explored if the model would have a commercial application which would require a very high degree of accuracy based on the business case (as described above).

We have the all the data compiled in a single table, now we need to partition the data to hold out a small portion for unbiased final testing to allow us to determine how accurate our predictions are against data that was never involved in the training or exploration and will remain unknown to both myself and the algorithm. Although there is no hard rule about the amount of data that should be selected for a testing hold out, typical values are between 10-20%. I will select 15% of the data and set it aside because while the data set isn't large, there are several parameters that will need to be tuned. We will train the models on the remaining 85%.

2.2 Data Exploration & Visualization

We should begin our exploration by taking a look at how our independent variables influence our dependent one.

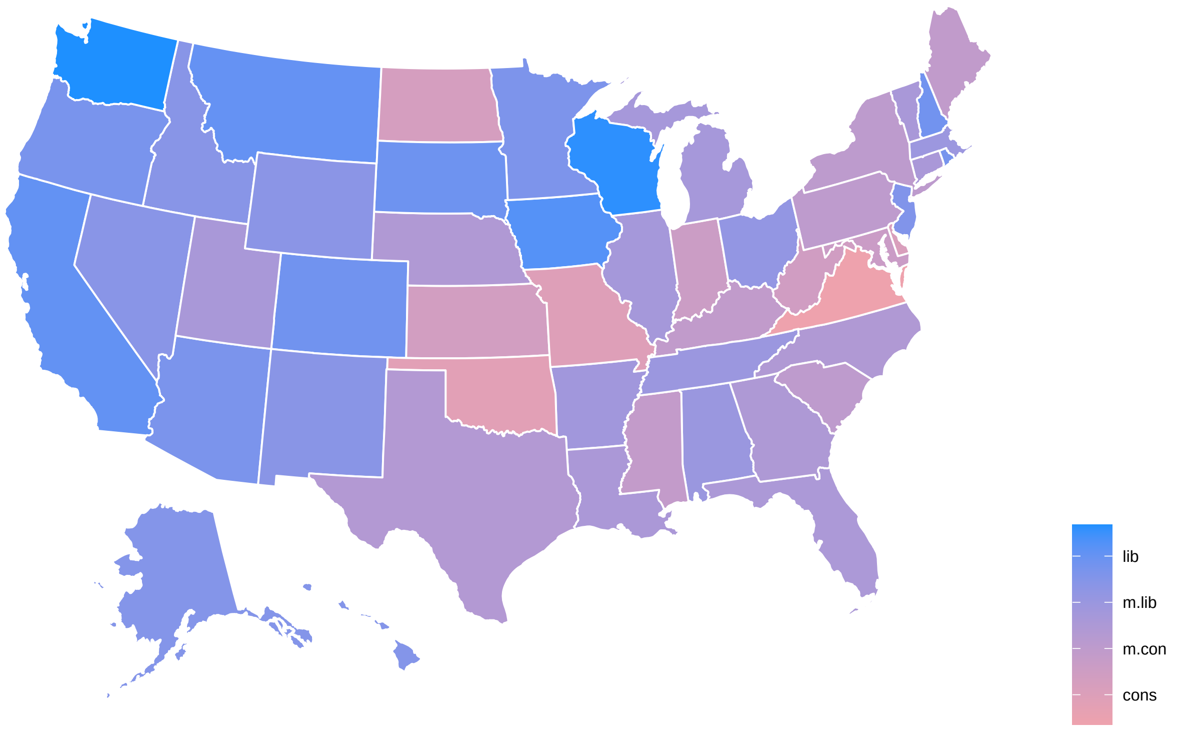
```r
glimpse(CMtraining)
```

```
## Rows: 94,340
## Columns: 30
## $ judge           <fct> 3002, 5126, 2077, 5076, 5046, 2105, 3017, 2061, 2...
## $ crtpoint        <fct> 044, 115, 037, 111, 146, 159, 173, 040, 159, 111,...
## $ numjudge        <dbl> 1, 1, 5, 1, 1, 8, 3, 1, 8, 1, 1, 1, 3, 5, 5, 8, 1...
## $ circuit         <fct> 03, 05, 02, 05, 05, 02, 03, 02, 02, 05, 05, 09, 0...
## $ state           <fct> 30, 09, 32, 43, 01, 33, 38, 32, 32, 43, 10, 03, 0...
## $ statdist        <fct> 301, 093, 323, 432, 013, 332, 381, 324, 322, 432,...
## $ month           <int> 7, 9, 7, 8, 6, 6, 9, 8, 8, 9, 7, 9, 6, 10, 10, 12...
## $ year            <int> 1932, 1932, 1932, 1932, 1931, 1932, 1931, 1932, 1...
## $ libcon          <int> 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1...
## $ casetype        <fct> 06, 05, 03, 20, 03, 21, 18, 03, 18, 05, 18, 03, 2...
## $ category        <fct> 2, 1, 1, 3, 1, 3, 3, 1, 3, 1, 3, 1, 3, 3, 1, 1, 1...
## $ casnum          <dbl> 10010028, 10010033, 10010104, 10010156, 10010213,...
## $ apyear          <int> 1929, 1931, 1925, 1931, 1917, 1929, 1914, 1931, 1...
## $ appres          <fct> 31, 31, 30, 31, 28, 31, 28, 31, 28, 31, 29, 29, 3...
## $ party           <fct> 2, 1, 2, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 2, 2, 2...
## $ gender          <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ race            <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ tenure          <int> 3, 1, 7, 1, 14, 3, 17, 1, 14, 1, 10, 9, 3, 3, 3, ...
## $ location        <fct> "Camden, NJ", "Jacksonville, FL", "Brooklyn, NY",...
## $ courtofappeals     <fct> Third Circuit, Fifth Circuit, Second Circuit, Fif...
## $ statename          <fct> New Jersey, Florida, New York, Texas, Alabama, No...
## $ districtcourt      <fct> New Jersey District of NJ, Florida Southern Distr...
## $ typeofcase         <fct> "Alien Petitions", "(Non)Conv-Criminal Case", "Cr...
## $ generaltypeofcase <fct> Civil Liberties/Rights case, Criminal Justice cas...
## $ presname           <fct> Herbert Hoover, Herbert Hoover, Calvin Coolidge, ...
## $ prespartyname      <chr> "Republican", "Republican", "Republican", "Republ...
## $ judgeparty         <fct> Republican, Democrat, Republican, Republican, Dem...
## $ judgegender        <fct> male, male, male, male, male, male, male, male, m...
## $ judgerace          <fct> white/Caucasian, white/Caucasian, white/Caucasian...
## $ presparty          <dbl> 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0...
```

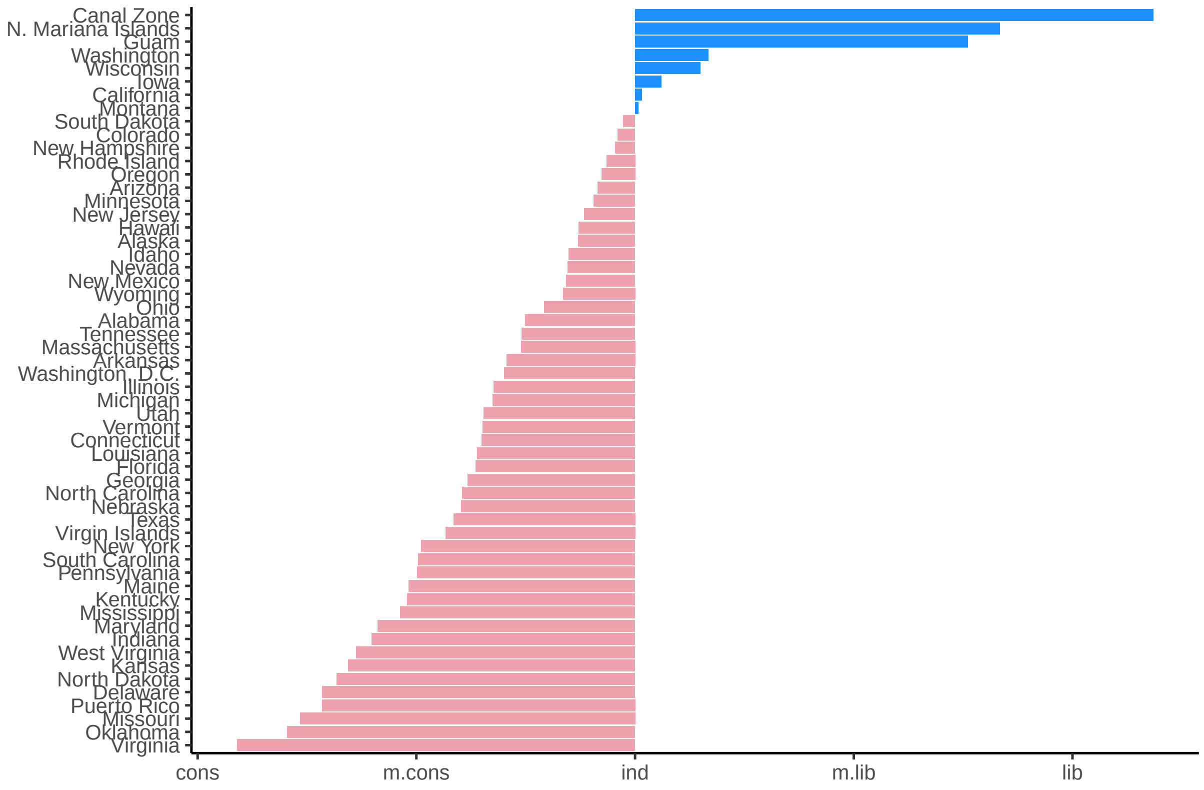
The training set consists of 94,340 entries of 30 columns (11 of which are duplicate and a result of the decoding process). The average case is outcome is a conservative one (~58% of the time). With 19 independent variables to consider, we need to visually inspect how our outcome is being influenced. Many of our independent variables might be related, so we will briefly look at each group.

2.2.1 Jurisdiction

Our first chart will take a look at jurisdiction to see if the state the case in which the case was decided is significant, the hypothesis is that there is a large impact. Different states are bound to follow different precedent and each state can have different laws, both of which could change the outcomes one way or the other:



The are a lot of different shades of color in that representation, but there are a couple of states that stand out. Additionally, as we look into the data more closely, we can notice that there are several US territories that were not represented on the original map. We should take a different look and switch to a view that will allow us see a more direct comparison between all jurisdictions.

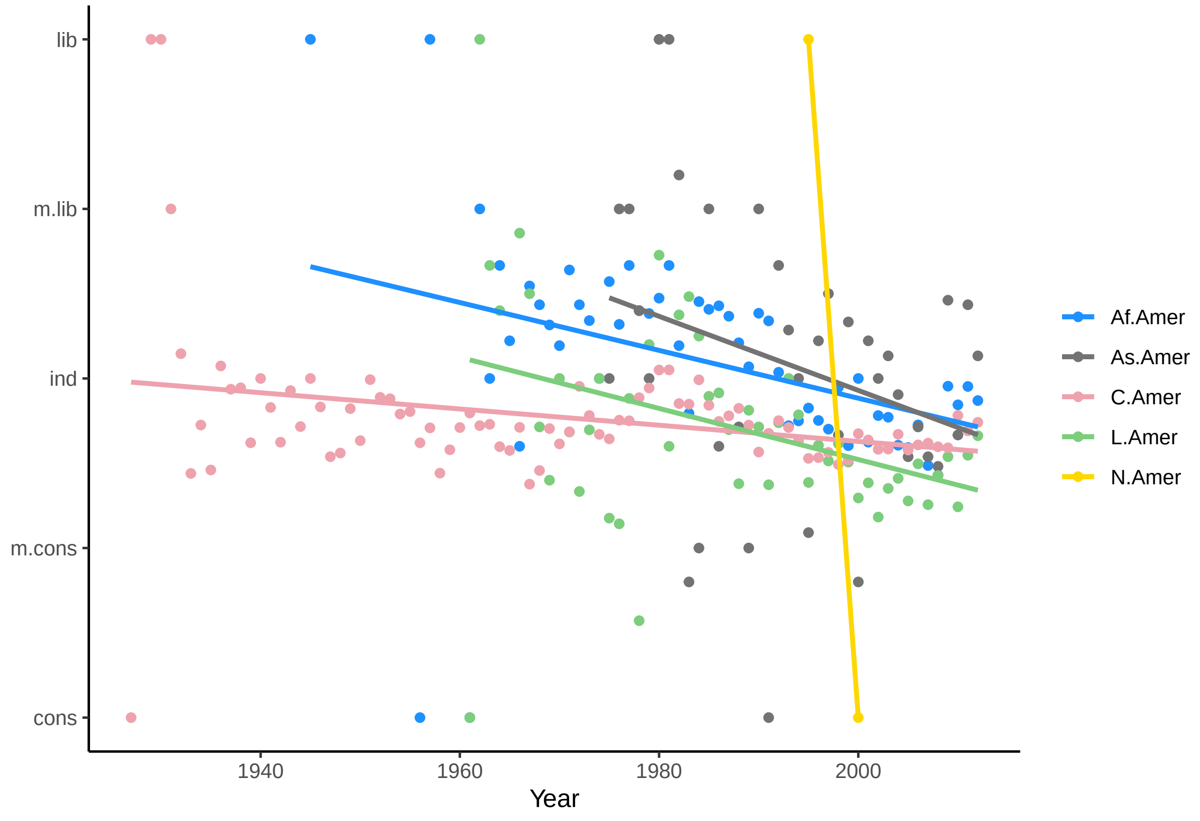


As expected, jurisdiction seems to be an important independent variable, which matches exactly with the expectation of anyone with legal training.

2.2.2 Judge Demographics

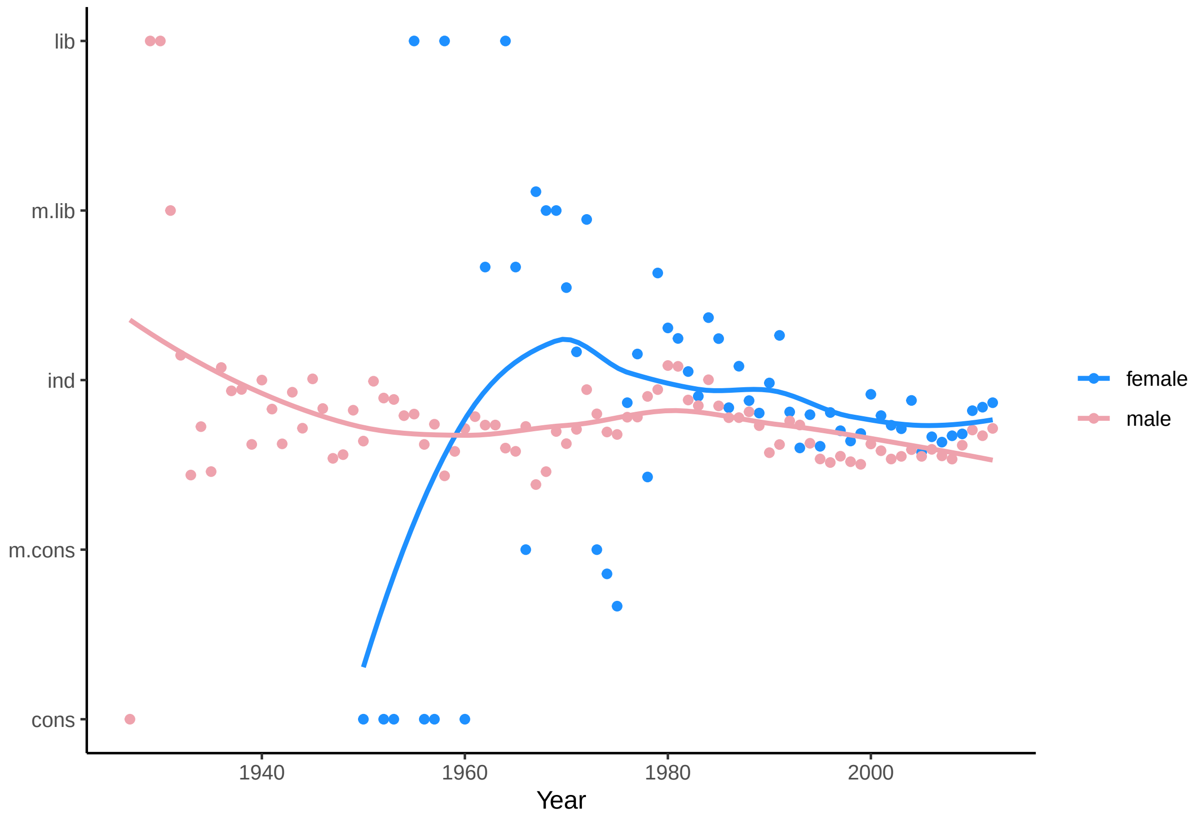
Another obvious set of variables that could influence the outcome of a given case are the demographic specifics of the judge presiding over the matter. This group of visualizations will be summarized so that the average for each characteristic will be shown for a given year and each characteristics will be plotted over time. This significantly simplifies what is shown.

The first demographic we will examine is race:

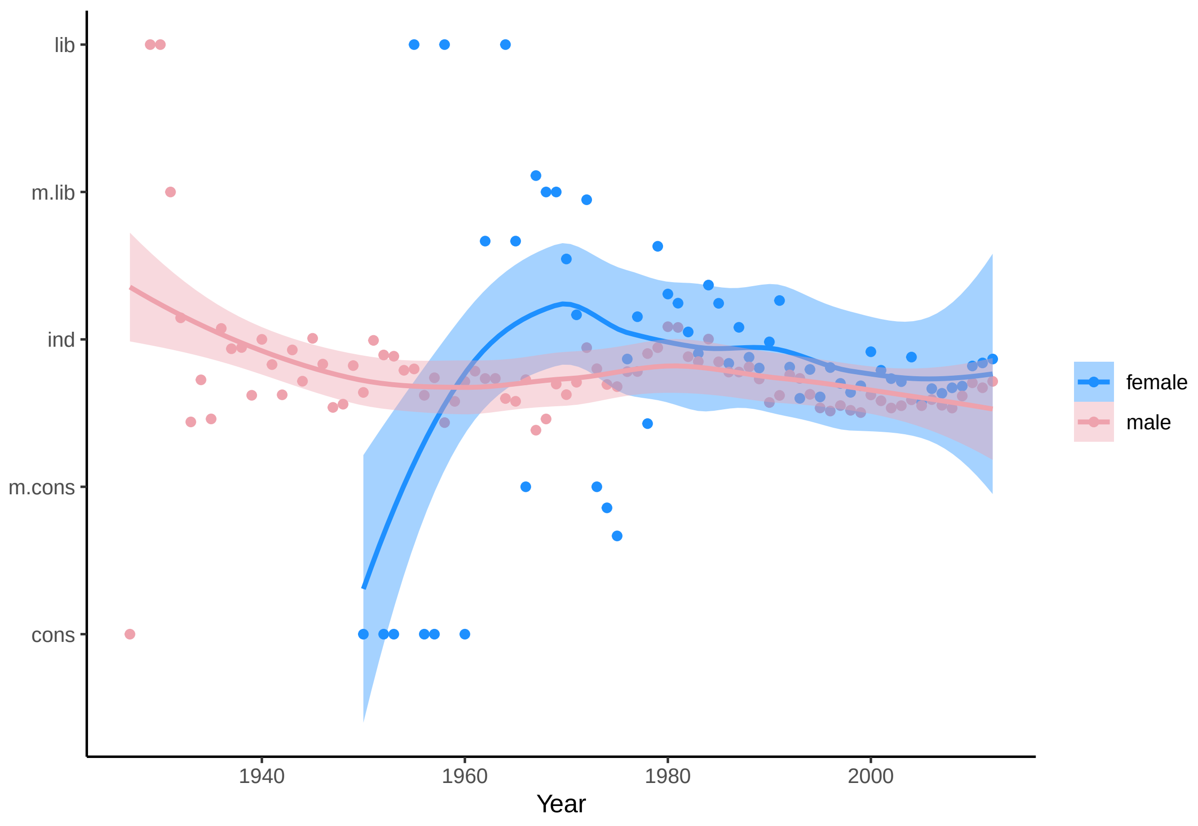


There are five possible values for race, which I abbreviated in the above figure: “African-American/black” (“Af.Amer”), “white/Caucasian” (“C.Amer”), “Latino/Hispanic” (“L.Amer”), “Native American” (“N.Amer”), and “Asian American” (“As.Amer”). While I had planned to add smoothed trend lines to all the graphs, some of the race data was too skewed for smoothing and linear approximations were required. There are clear trends contained in the racial breakdown so it is a trait that will likely add value to our model.

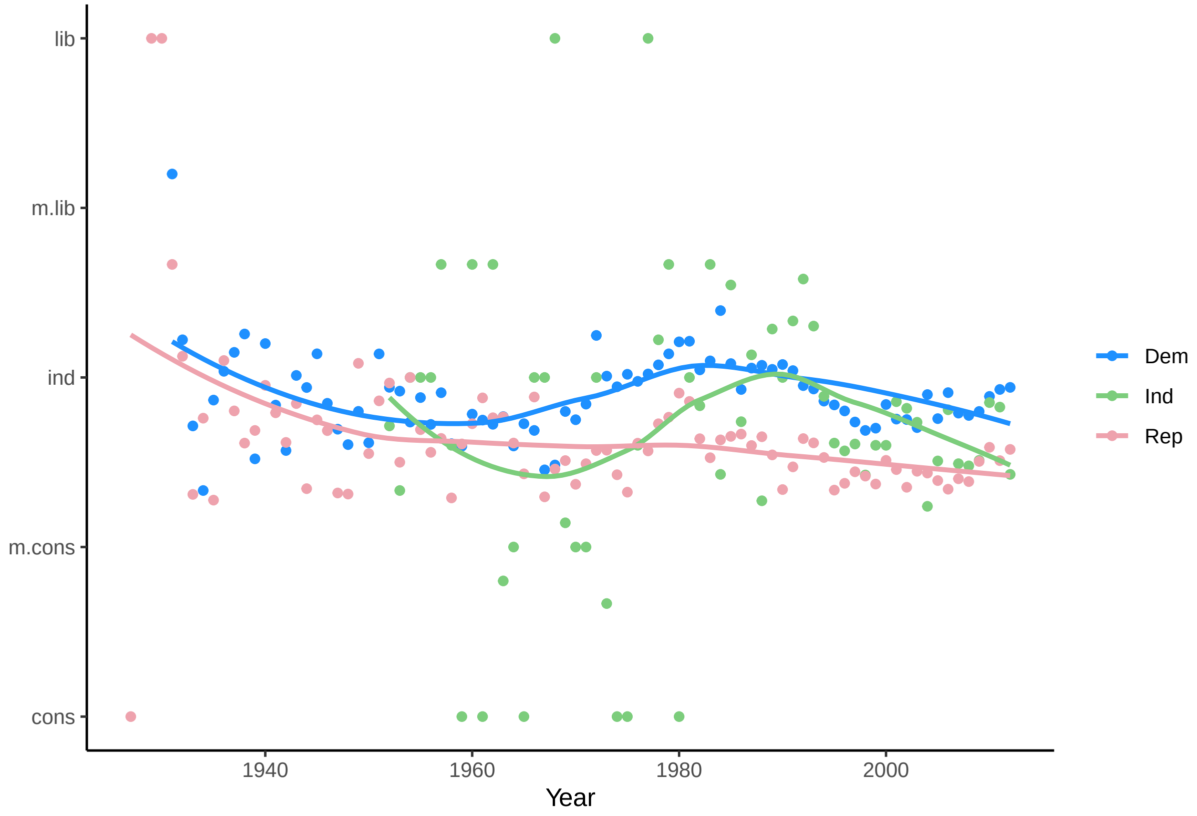
Next, we will examine gender:



The Carp-Manning authors have used the binary “male” and “female” designations for this study. Again, there is a strong trend visible with this data, even though the data has some variability in the first half of the 20th century. Actually if we look at the graph with confidence intervals displayed (next page), we can see that while the data for the male judges seems pretty steady, data for the female judges has a lot more uncertainly. This is probably due in large part to the smaller number of female judges, but I think the trend is worth adding to the model regardless.


lib
m.lib
ind
m.cons
cons
1940
1960
1980
2000
Year
female
male

The final piece of judge demographic information we should explore is the correlation of the ruling tendency and the judge's political party:

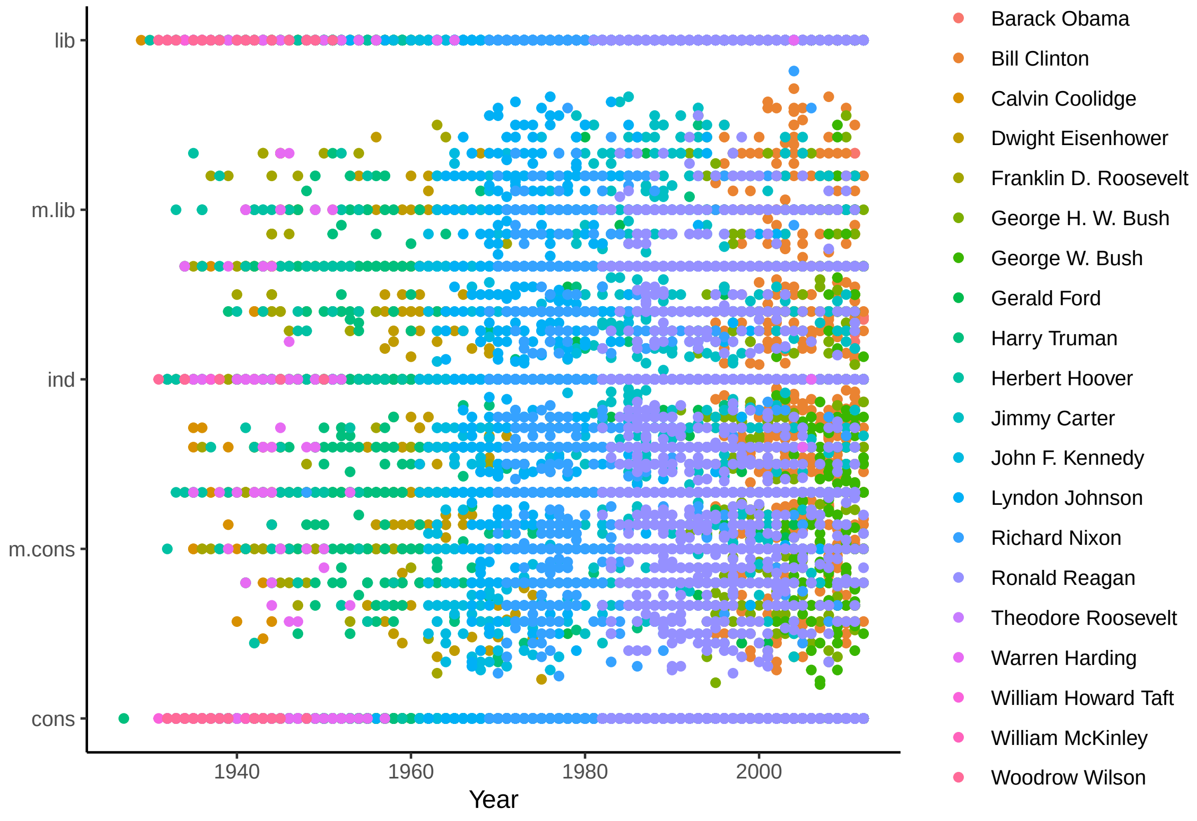


There were 4 designations provided by the authors for this characteristic: "Democrat" ("Dem"), "Republican" ("Rep"), "Independent/Other/Unknown" ("Ind"), and "99". I am not sure, nor do the authors indicate what the the "99" designation is associated with, so I removed that factor from the chart.

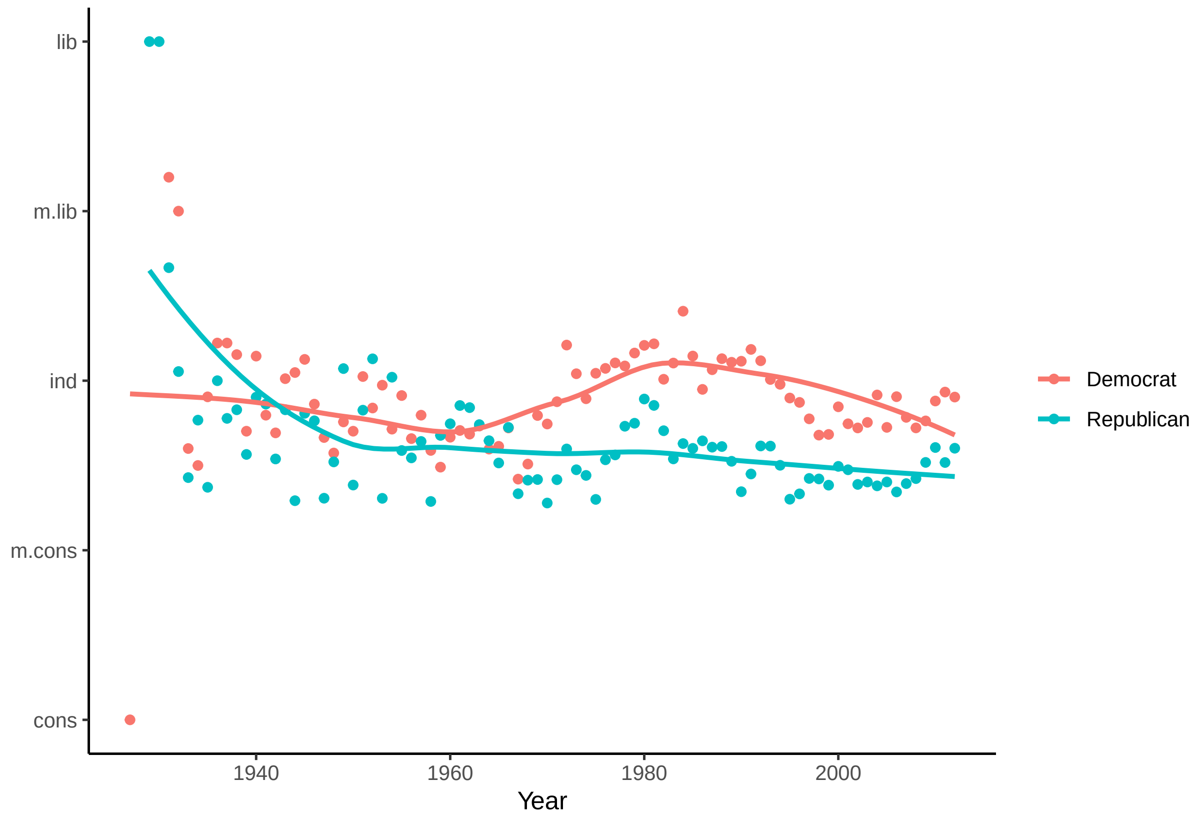
This display matches with expectations almost perfectly, with Democrats being traditionally being perceived as more liberal and Republican as being more conservative over this time frame. Additionally, Independents seem not to follow a specific trend which is also an expected pattern. These trends make common sense as political party is a feature personally chosen by the judge and it logically follows that the expression of the political party would also provide good information about the ruling tendencies of each judge.

2.2.3 Appointment Demographics

Federal judges of all levels in the United States are nominated by the President and confirmed by the Senate. While this action gets a lot of press coverage for high level appointments, I suspect there is just as much of an impact for lower court judges as well:



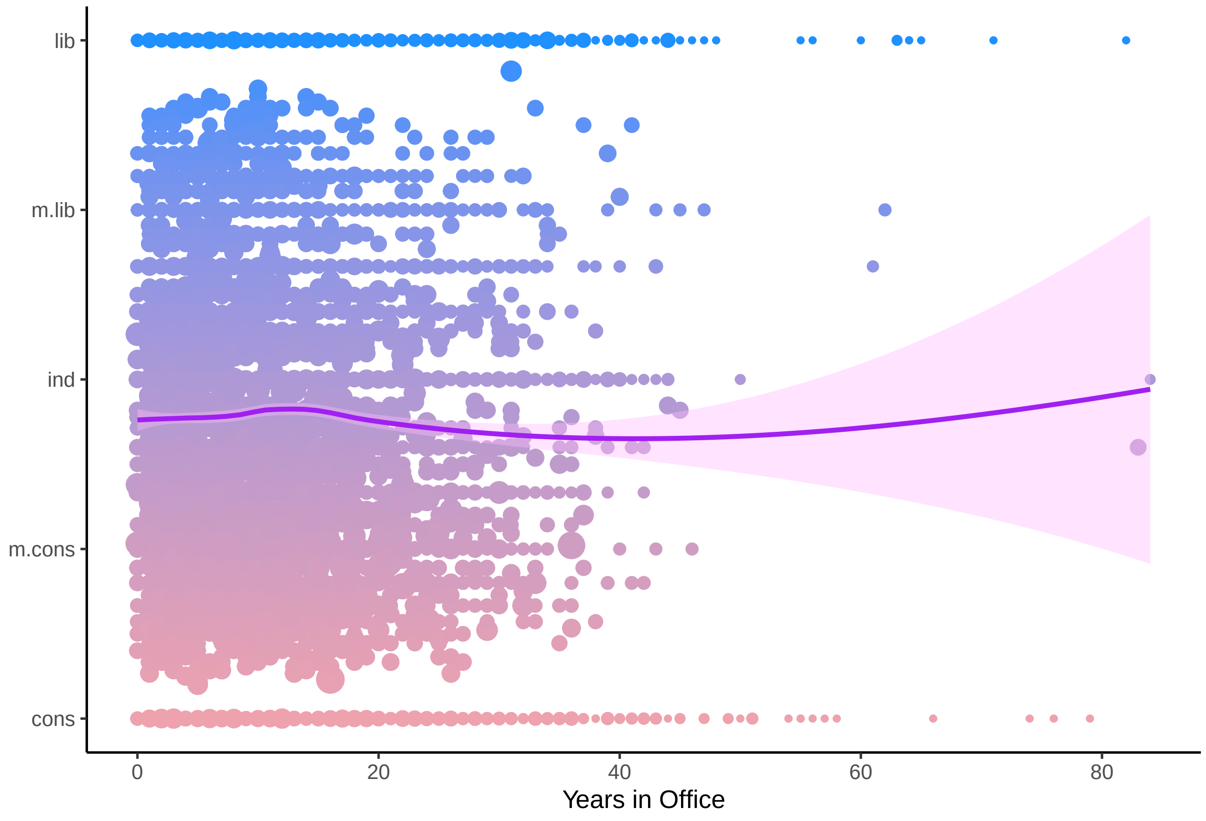
This visualization is too crowded to be able to make sense of and no trends are apparent from a quick examination of the graph. The are so many presidents listed that adding trend lines makes the graph more confusing and even removing the data points all together so as to just leave the trend lines was still too much. We should peel the information back a layer and group the presidents by their political party and see if a trend emerges. I feel like there should be some sort of correlation, as presidents would be more likely to appoint judges that have views similar to their own:



We can now see a clear trend in the data and the shape of our trend lines mirrors the trend lines of the judge's political party. This also makes common sense and tells us that the appointing president designation might be conflated because presidents are likely to pick judges with political parties matching their own. We need to be caution of that when selecting variables for our model.

2.2.4 Tenure

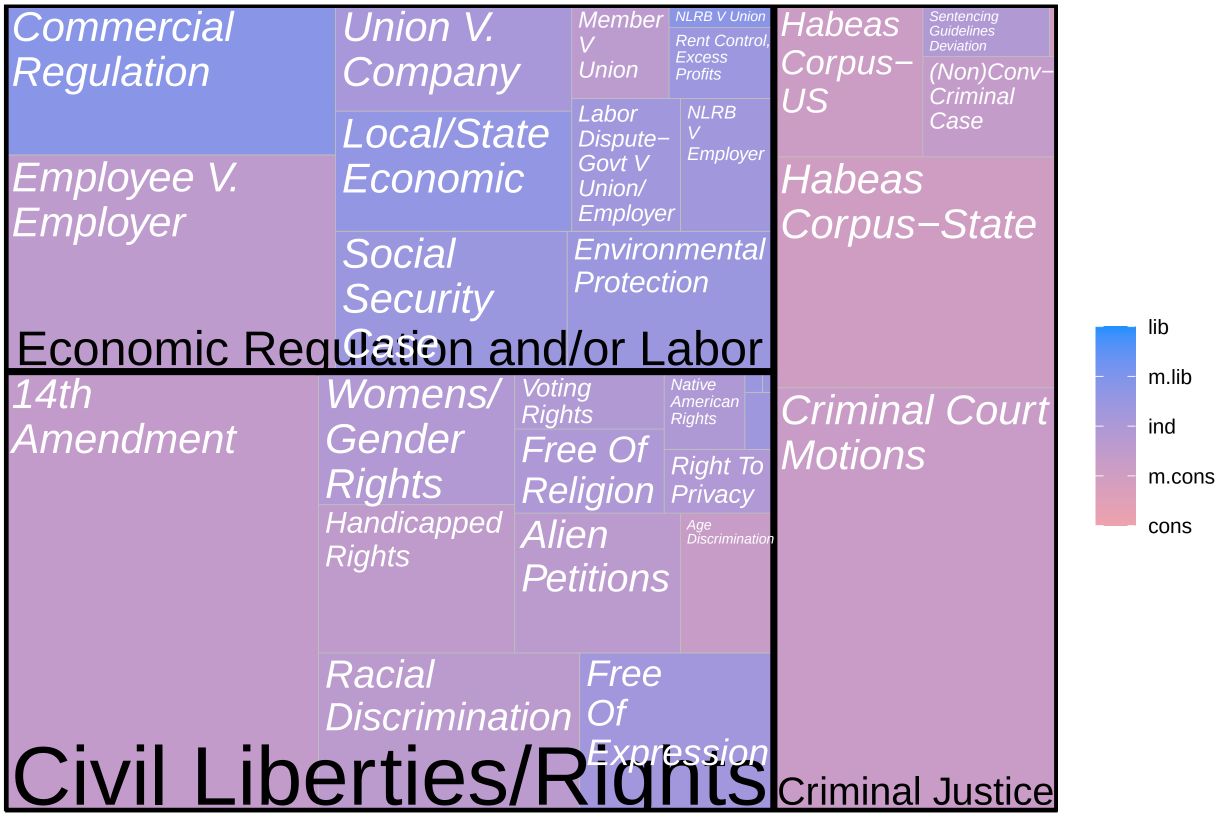
We can also test the hypothesis that length of time in office changes the tendency of a judge to rule in a certain way:



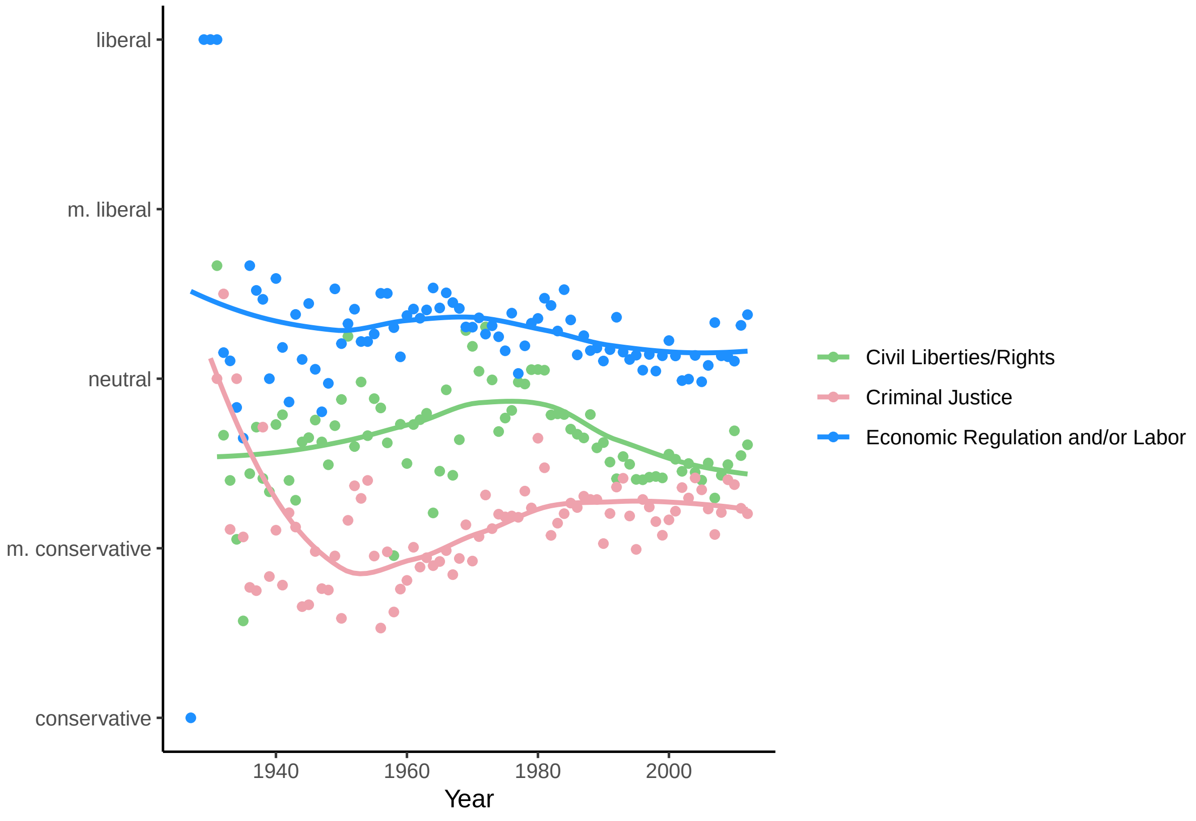
On this chart, each dot represents a group of judges at that particular ideological lean for that number of years in office. The size of the dot represents the number of opinions written by the group. We do see a slight conservative shift after about 15 years on the bench, but before that we see a slight upward trend, which I did not expect. This measure does look impactful.

2.2.5 Subject Matter

The final independent variables we want to take a look at involves the subject matter of the law suit:



The tree diagram is useful here because it allows us to take a look at the variance from each of the 31 case type designations while keeping them grouped within the 3 case type categories. There are noticeable trends here, which is expected because of the style of visualization, but if we view the overall categories as a whole over time, we can see that they seem to have a stronger effect:



The case type categories almost form bands, which meets our anecdotal expectations, with judges treating economic and labor matters much more liberally than criminal justice issues.

2.2.6 Limitations

One final point before we move on. Legal cases involve one or more plaintiffs against one or more defendants contesting a specific factual pattern. While there are obviously trends in the data based on factors outside of the case itself, at the end of the day we will not be able to always determine the outcome of a case without knowing the specific facts. This unknown variability needs to be taken into account when fitting our model.

We also mentioned confounding earlier when looking at political parties, but in fact many of the independent variables may be related or may seem to be associated because underlying factors may be the real cause of both the presentation in the dependent and independent variables.

While I intended to use Linear discriminant analysis (LDA) as one of the models for this project, the data does not fit several of the assumptions required, which is worth mentioning here. Firstly, the data's independent variables do not seem to be normally distributed, with much of the data skewed towards one of the classifiers. Additionally, some of the data sets that contain specific classifiers are too small for the model to function, because there are less data points than degrees of freedom in the model.

I had also originally also wanted to use a k-nearest neighbor approach, but the categorical nature of the data and small number of selected factors compared to the amount of data meant that the distance function used to calculate the neighborhood was too saturated and was causing model failures, even with a neighborhood at $k = 1$.

2.3 Data Analysis Conslusions

All of the features we examined look promising as predictors so we will select them all except the judge's political party, which we discussed may be too confounded with other predictors. We will use 8 predictors:

- Appointing President - appres
- President's Political Party - presparty
- State - state
- Case Topic - casetype
- Case Topic Category - category
- Judge's Gender - gender
- Judge's Ethnicity - race
- Number of Years on the Bench - tenure

3 Supervised Learning Methods

3.1 Modeling Approach

We will build several machine learning algorithms and additionally ensemble them into a final predictor. We will include three models, a linear (logistic) model, a Naive Bayes classifier, and a random forest model. After testing, we will select the most accurate result.

3.1.1 Cross Validation of Training Data

Using the same ideas we discussed when we set aside a portion of our data to do unbiased testing, we can use a similar approach when working to tune our model parameters. We can randomly select a portion of our training data to train from, say 85%, model it, and then test it against the 15% of data that was left over. We can do this multiple time to build confidence in our parameters, When the first pass is completed, we replace the sample we used and sample again from the entire training set. This will allow us to find parameters that are more stable and robust than just modeling against the entire training set at once. Randomly selecting the sample from the whole population is a method called "bootstrapping." For these models we will bootstrap the samples with replacement 10 times. This control method is defined and then we can use it for tuning in all of our models.

```r
control <- trainControl(method = "boot", number = 10, p = .85)
```

3.2 Models & Algorithms

3.2.1 Linear (Logisitic) Model

Linear models are simple but are widely used and can be effective. These model presume some form of a linear relationship between the factors and make predictions based on that assumption.

Linear models fit the general form of:

$$\hat{Y}_n = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon_n$$

Where $\hat{Y}$ is the predicted tendency, is β_0 is μ is the true population mean of the tendency, β_n is the true bias effects of x_n, the features selected, and ϵ_n is the error distribution.

The individual bias effect(s) β_n take for the form of:

$$\hat{\beta}_n = \frac{1}{N}\sum_{i=1}^{N}(y_n - \hat{\mu})$$

where $\hat{\beta}_n$ is the predicted value of β_n, N is the number of samples, y_n is the predicted ruling, and $\hat{\mu}$ is the predicted mean. Since we aim to measure a particular feature, we treat that each item in that category as a small set. We "take away" our population mean and then calculate the mean across the deviation from that prediction. The mean of the deviations is used to correct our prediction for all items in the category.

If we can combine two or more features together to estimate the bias of further terms:

$$\hat{\beta}_n = \frac{1}{N}\sum_{i=1}^{N}(y_n - \hat{\beta}_{n-1} - \cdots - \hat{\beta}_1 - \hat{\mu})$$

where $\hat{\beta}_n$ is the prediction for the second factor, N is the number of samples, y_n is the predicted ruling, $\hat{\beta}_{n-1}$, $\hat{\beta}_1$, etc are the predicted bias from the other selected features, and $\hat{\mu}$ is the predicted mean. We do the exact same math here, picking a second category but this time removing the predicted mean and the bias effect of the first characteristic. After those factors are removed, we are again left with a set of deviations based on a feature and then we can assign the mean of those deviations for that item. We can continue to add terms in this way and we should increase our accuracy each time.

One thing to note here, while we can continue to increase our accuracy for the whole set, at some point the features we select will have such small groups that they might contain only 1 member. This could lead to a model that is so precise it will not be able to account for new variations. This is called 'over-fitting.'

Because we have a Bernoulli distribution, we can use a special case of the linear model called logistical Regression. Logistical regression is a variation of the general linear model specifically used to classify binary events, like the outcome of a case being liberal or conservative. as we have here.

The logistical regression model is defined as follows:

$$p\,(x) = P[Y = 1 \mid X = x]$$
$$1 - p\,(x) = P[Y = 0 \mid X = x]$$

$$ln\left(\frac{p\,(x_n)}{1 - p\,(x_n)}\right) = \beta_0 + \beta_1 x_1 + .. + \beta_n x_n$$

Where $p\,(x_n)$ is the probability that the output is 1 given our set of x independent variables and β_n is still a bias effect for a given characteristic variable x_n. This will be our first model.

3.2.2 Naive Bayes

The Naive Bayes is a simple probabilistic classifier and is a popular baseline model for classification. Its most important assumption is that all of the features are *equally important* and *independent.* While this rarely happens in real world data, we will use it here because the assumptions seem to hold somewhat based on our visual inspection.

Given a vector of predictor variable values x, the algorithm assigns conditional probabilities for each of the response variable's Y_i classes:

$$p(Y_i|x)\,.$$

Then we can apply Bayes' rule from probability theory:

$$p(Y_i|\mathbf{x}) = \frac{p(Y_i)p(\mathbf{x}|Y_i)}{p(\mathbf{x})} \propto p((Y_i)p(\mathbf{x}|Y_i)$$

In practice we ignore the denominator, which is a constant in any given analysis.

The next step is to expand $p(\mathbf{x}|Y_i)$:

$$p(\mathbf{x}|Y_i) = p(x_1, \ldots, x_n|Y_i)$$
$$p(\mathbf{x}|Y_i) = p(x_1|x_2, \ldots, x_n, Y_i)p(x_2|x_3, \ldots, x_n, Y_i) \cdots p(x_n|Y_i)$$

Then we can apply our assumption that the factors are independent of each other:

$$p(x_1|x_2, \ldots, x_n, Y_i)p(x_2|x_3, \ldots, x_n, Y_i) \cdots p(x_n|Y_i) \rightarrow p(x_1|Y_i) \cdots p(x_n|Y_i)$$

and can simplify:

$$p(Y_i|\mathbf{x}) \propto p(Y_i) \prod_{n=1}^{n} p(x_n|Y_i)$$

We will use this as our second model.

3.2.3 Randon Forest

Random Forest is a model built from an ensemble of random decisions trees. Decision trees are predictive models which split the data into various subsets based on characteristics and assigning a probable predicted outcome based on which subset the data point falls into; this process is called recursive partitioning. Visually, this can be represented as nodes with "branches" coming off of them. We can view them as follows:

$$\hat{Y}_i = \begin{cases} x_{i,1} > m \implies \begin{cases} x_{i,2} > n \implies \begin{cases} x_{i,3} > c \implies 1 \\ x_{i,3} < c \implies \begin{cases} x_{i,4} > s \implies 1 \\ x_{i,4} < s \implies 0 \end{cases} \end{cases} \\ x_{i,2} < n \implies \begin{cases} x_{i,4} > s \implies 1 \\ x_{i,4} < s \implies 0 \end{cases} \end{cases} \\ x_{i,1} < m \implies \begin{cases} x_{i,2} > n \implies \begin{cases} x_{i,4} > s \implies 1 \\ x_{i,4} < s \implies 0 \end{cases} \\ x_{i,n} < n \implies \begin{cases} x_{i,3} > c \implies 1 \\ x_{i,3} < c \implies \begin{cases} x_{i,4} > s \implies 1 \\ x_{i,4} < s \implies 0 \end{cases} \end{cases} \end{cases} \end{cases}$$

Where $\hat{Y}_i$ is the predicted value and $x_{i,1}$, $x_{i,2}$, $x_{i,3}$, and $x_{i,4}$ are characteristics with values of in the range of m, n, c, and s respectively. Decision trees work by following the “tree” down the “branches”; each time you get to a node you examine a (set of) characteristic(s) that causes you to go down the path one way or the other. You continue following your choices until you get to the end of the chart, also called a “leaf”. The leaf will then assign a prediction to the data point.

Random forest generate many random decision trees from the data, allowing the model to be more flexible and avoiding the common problem of over-fitting that can happen if just one tree is created for the model.

3.2.4 Ensemble

Finally, an ensemble model uses predictions from other models to make its prediction. The model looks at all the other predictions and then uses a majority voting approach to determine how it should predict the outcome. If a majority of the models predicts the outcome of the case as liberal, the ensemble will classify it that way as well. We are using our three underlying models and so we will always have a majority and do not need to consider a tie breaker mechanism. We can view the model this way:

$$\hat{Y}_i = \begin{cases} y_{i,1} + y_{i,2} + y_{i,3} > 1 \implies 1 \\ y_{i,1} + y_{i,2} + y_{i,3} < 2 \implies 0 \end{cases}$$

Where $\hat{Y}_i$ is the prediction of the ensemble and $y_{i,1}$, $y_{i,2}$, and $y_{i,3}$ are the predictions from our previous models.

4 Results

4.1 Metrics

Before we can examine how our models performed, we need to define the metric we will use to compare them. A reason that the legal industry has been slow to adopt artificial intelligence and machine learning is that users expect very accurate predictions and the business will not compromise or allow for a general weakness within models.

Because we are predicting a single outcome as either liberal or conservative, there are 4 possible results from our prediction:

- True positive (TP): We predict the case is ruled as liberal and the case is actually ruled as liberal
- False positive (FP): We predict the case is ruled as liberal and the case is actually ruled as conservative
- False negative (FN): We predict the case is ruled as conservative and the case is actually ruled as liberal
- True negative (TN): We predict the case is rules as conservative and the case is actually rules as conservative

There are many ways to evaluate this set of outcomes to measure success and we would normally need to be overly cautious in selecting a quality measure based on the demands of the industry. Best practice would be to work with industry experts and find a balance between several metrics, but for simplicity's sake, we will focus on measures that can be expressed as a single value and then we will further narrow down the possibilities to three options for our metric, accuracy, F_1 score, and Matthews correlation coefficient.

4.1.1 Accuracy

Accuracy measures the total of the correct outcomes over the total number of outcomes:

$$Accuracy = \frac{\Sigma TP + \Sigma TN}{\Sigma predictions}$$

Accuracy can be a good metric if the data is balanced overall, meaning that possible outcome are approximately equally likely to occur. If one outcome is much more likely, accuracy can be increased by focusing on the result with the highest prevalence at the expense of the other result. Our earlier examination of the Carp-Manning data shows that conservative outcomes are much more likely overall, so this could be a problem if we selected accuracy as our metric. We need to find a metric that balances the approach to can account for a skew in the result distribution.

4.1.2 F1 score

F_1 measures the harmonic mean between precision and recall and can be a more balanced approach to performance measurement. We can begin by defining the components and then we will define the metric in full:

$$recall = \frac{\Sigma TP}{\Sigma TP + \Sigma FP}$$
$$precision = \frac{\Sigma TP}{\Sigma TP + \Sigma FN}$$
$$F_1 = 2\left(\frac{(precision)\,(recall)}{precision + recall}\right)$$

Which can be simplified to:

$$F_1 = \frac{\Sigma TP}{\Sigma TP + \frac{1}{2}\left(\Sigma FP + \Sigma FN\right)}$$

We notice that while f_1 is more balanced, it does not take the true negative value into consideration, meaning we can still do better in our hunt for a metric that would work for this use case.

4.1.3 Matthews correlation coefficient

The Matthews correlation coefficient (MCC) is regarded as one of the most balanced and complete single value metrics available for data considering a binary outcomes. The metric shows the correlation between predicted and observed values:

$$MCC = \frac{(\Sigma TP)(\Sigma TN) - (\Sigma FP)(\Sigma FN)}{\sqrt{(\Sigma TP + \Sigma FP)(\Sigma TP + \Sigma FN)(\Sigma TN + \Sigma FP)(\Sigma TN + \Sigma FN)}}$$

This is the most robust option available to us and because of the skew in our sample outcomes, this is the option we will select.

Because MCC is a coefficient, the possible values range from -1 to 1 with 1 being perfect predictions and 0 being a random guess (a negative score would mean that we were predicting the incorrect answer). The closer the MCC is to 1, the better our model is performing.

we can build a function in r to calculate this from the confusion matrix output from the caret library:

```r
MCC <- function (conf_matrix) {
TP <- conf_matrix$table[1,1]
TN <- conf_matrix$table[2,2]
FP <- conf_matrix$table[1,2]
FN <- conf_matrix$table[2,1]

mcc_num <- (TP*TN - FP*FN)
mcc_den <- as.double((TP+FP))*as.double((TP+FN))*as.double((TN+FP))*as.double((TN+FN))

mcc_final <- ifelse(mcc_den == 0, 0, mcc_num/sqrt(mcc_den))
return(mcc_final)
}
```

4.2 Performance

With our metric selected, we can now evaluate how our models perform.

4.2.1 Linear (Logisitic)

There are no parameters that can be adjusted or tuned in this model, so we will simply run it against our validation cluster and check our results:

```
## Generalized Linear Model
##
## 94340 samples
##     8 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 94340, 94340, 94340, 94340, 94340, 94340, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6467917  0.2468381
```

We can use the results outlined above to calculate our final score:

```
## # A tibble: 1 x 2
##   method              MCC
##   <chr>             <dbl>
## 1 Linear (Logistic) 0.249
```

This result is promising, but not exceptional. It is a good first effort.

4.2.2 Naive Bayes

The Naive Bayes model has three parameters we can tune in the model, "useKernel", "laplace", and "adjust". "useKernel" allows us to use a kernel density estimate for continuous variables versus a guassian density estimate. Because we have a discrete dependent variable, this will probably perform better as false, but we will test both versions. "Laplace" allows us to incorporate a laplace smoother, which will help us deal with the potential of a zero probably when using factored predictors like we have in our data set. And finally, "adjust" allows us to adjust the bandwidth of the kernel density with larger numbers meaning a more flexible estimate.

- usekernel = True or False
- laplace = 0, 0.5, or 1
- adjust = 0.75, 1, 1.25, or 1.5

We can look to see how this model performed:

```
## Naive Bayes
##
## 94340 samples
##     8 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 94340, 94340, 94340, 94340, 94340, 94340, ...
## Resampling results across tuning parameters:
##
##   usekernel  laplace  adjust  Accuracy   Kappa
##   FALSE      0.0      0.75    0.6234591   2.040593e-01
##   FALSE      0.0      1.00    0.6234591   2.040593e-01
##   FALSE      0.0      1.25    0.6234591   2.040593e-01
##   FALSE      0.0      1.50    0.6234591   2.040593e-01
##   FALSE      0.5      0.75    0.6234591   2.040593e-01
##   FALSE      0.5      1.00    0.6234591   2.040593e-01
##   FALSE      0.5      1.25    0.6234591   2.040593e-01
##   FALSE      0.5      1.50    0.6234591   2.040593e-01
##   FALSE      1.0      0.75    0.6234591   2.040593e-01
##   FALSE      1.0      1.00    0.6234591   2.040593e-01
##   FALSE      1.0      1.25    0.6234591   2.040593e-01
##   FALSE      1.0      1.50    0.6234591   2.040593e-01
##    TRUE      0.0      0.75    0.5760505  -1.259794e-05
##    TRUE      0.0      1.00    0.5760505  -2.097593e-05
##    TRUE      0.0      1.25    0.5760620   0.000000e+00
##    TRUE      0.0      1.50    0.5760620   0.000000e+00
##    TRUE      0.5      0.75    0.5760505  -1.259794e-05
##    TRUE      0.5      1.00    0.5760505  -2.097593e-05
##    TRUE      0.5      1.25    0.5760620   0.000000e+00
##    TRUE      0.5      1.50    0.5760620   0.000000e+00
##    TRUE      1.0      0.75    0.5760505  -1.259794e-05
##    TRUE      1.0      1.00    0.5760505  -2.097593e-05
##    TRUE      1.0      1.25    0.5760620   0.000000e+00
##    TRUE      1.0      1.50    0.5760620   0.000000e+00
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were laplace = 0, usekernel = FALSE
##  and adjust = 0.75.
```

We will again use the results to calculate the final score:

```
## # A tibble: 1 x 2
##   method        MCC
##   <chr>       <dbl>
## 1 Naive Bayes 0.214
```

This model performed less well than the linear model, but it still a decent improvement over just guessing. The tuning parameters were less impactful than expected. Regardless, it should add value to our ensemble but it will not be selected as the best model.

4.2.3 Random Forest

The Random forest has 2 parameters that can be tuned when testing the model, the first is "predFixed", which determines the minimum number of predictors at any giving split in the tree and "minNode", which determine the minimum number of distinct row references required to allow for a node to split. We can will test several values for each:

- predFixed = 1, 3, 5 or 7
- minNode = 2, 50, 100

We can see the optimal values and model information here:

```
## Random Forest
##
```

```
## 94340 samples
##     8 predictor
##     2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (10 reps)
## Summary of sample sizes: 94340, 94340, 94340, 94340, 94340, 94340, ...
## Resampling results across tuning parameters:
##
##   predFixed  minNode  Accuracy   Kappa
##   3           50      0.6445427  0.2221285
##   3          100      0.6442136  0.2220819
##   5           50      0.6533326  0.2626525
##   5          100      0.6536949  0.2627346
##   7           50      0.6515601  0.2661782
##   7          100      0.6531718  0.2671397
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were predFixed = 5 and minNode = 100.
```

The MCC associated with these results is:

```
## # A tibble: 1 x 2
##   method          MCC
##   <chr>         <dbl>
## 1 Random Forest 0.274
```

Great! We've improved over the linear fit and have a new top candidate for best fit.

4.2.4 Ensemble

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7745 4073
##          1 1805 3014
##
##                Accuracy : 0.6467
##                  95% CI : (0.6394, 0.654)
##     No Information Rate : 0.574
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2465
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8110
##             Specificity : 0.4253
##          Pos Pred Value : 0.6554
##          Neg Pred Value : 0.6254
##               Prevalence : 0.5740
##           Detection Rate : 0.4655
##     Detection Prevalence : 0.7103
##        Balanced Accuracy : 0.6181
##
##         'Positive' Class : 0
##
```

Our final MCC score is:

```
## # A tibble: 1 x 2
##   method     MCC
##   <chr>    <dbl>
## 1 Ensemble 0.258
```

This turns out to be our second best model, coming in slightly better than than the linear model and well above our last place model, naive Bayes.

5 Conclusion

Our goal was to create a classifier that predicted if a case would have liberal or conservative ruling and we were able to build several models that performed consistently better than guessing when utilizing a handful of diverse variables in the Carp-Manning data set, despite the limitations that we outlined above.

5.1 Selected model

Our best performing model was the random forest.

5.2 Future Work

We did not consider several of the variables or even explore all available variables during our visualization step. Future work could utilize additional variables to increase the accuracy of the predictions. Additionally, improvements could be made to the case type categorization system to increase the detail and complexity of the classification system which would increase the model accuracy. Another area of improvement would be to increase our computational power and focus on building prediction profiles for individual judges which could be even more useful than the general approach we have taken.

5.3 Impact

Even though we were not able to create a model with a high degree of accuracy, our lower quality models could still be impactful in the legal space. This information could be leveraged to help law firms build settlement strategies or help determine which cases the firms should focus its resources on. If the model were even more accurate, this could have a larger impact, but the model will not replace the human act of analyzing facts and applying laws to determine how justice should best be served.