# U.S. House Floor Votes Project

Casey Crouch

9/20/2020

## Author Introduction

This report was drafted by Casey Crouch in accordance with the 2nd capstone requirement of the HarvardX Data Science Professional Certificate. HarvardX is an online learning initiative by Harvard University through edX.

Mr. Crouch is a third-year college student studying Political Science in Los Angeles. He aspires to a career in public policy.

## Project Overview

### Desired Outcome

The goal of this project is to develop a model for political party affiliation in the 98th Session of the United States Congress utilizing data compiled by the Congressional Quarterly Almanac.

To achieve this end, the author undertakes a 3-step process of 1) Data Preparation, 2) Data Exploration, and 3) Data Modeling. This project's primary models are developed using the machine learning algorithms provided with the "caret" package, including the Logistic Regression, K-Nearest Neighbors, and Random Forests.

Success is measured with Accuracy, a common metric used to evaluate machine learning algorithms assigned to classification problems. The author considers an Accuracy of 90% to be a sufficient threshold for modeling success.

A second goal of this project is to improve the author's proficiency with R--a programming language typically used in statistical analysis.

### The Congressional Voting Records Dataset

The Congressional Quarterly Almanac is a privately-operated journal that has assembled information on federal legislation since the mid-20th Century. Volume XL (1985) identifies 16 key floor votes that characterized the state of each party in the 2nd Session of the 98th Congress. Dr. Jeffrey Schlimmer of Washington State University compiled this information into a categorical dataset for his 1987 dissertation: "Concept Acquisition Through Representational Adjustment." This dataset is comprised of 6,960 floor votes given by 435 members of the U.S. House of Representatives on 16 unique bills.

Today, the information is publicly available through the University of California, Irvine, Machine Learning Repository. You can access the dataset for yourself by clicking on the following link:

https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

## Section 1: Data Preparation

The author commences the project by briefly assessing the state of the data and pinpointing areas requiring adjustment. The dataset provided by Dr. Schlimmer is already quite clean, and the only real issue is a significant number of missing values. The author temporary recodes these "?"s into NAs, which are transformed once again in the following section. Additionally, the author capitalizes political party names and compiles the data into the "voting" dataframe for use in the "data exploration."

## Section 2: Data Exploration

Having prepared the data for analysis, the purpose of this section is to identify insights upon which machine learning algorithms can be applied and models developed. The author carries out this task by completing a series of "Exploration Activities," such as:

- Plotting the 1982 midterm election results
- Calculating bill passage & failure
- Developing a "Party-Line Index" to quantify partisanship
- Applying an "Inverse Party-Line Fix" to recode NAs
- Differentiating partisanship and ideology in floor voting
- Identifying legislation with a clear partisan consensus

This section concludes with the recognition of 2 bills, "Medical" & "Budget," as strongly partisan, and 1 bill, "ElSalvador," as strongly ideological. The author states his intention to utilize the two former items as a starting point in the modeling process that will likely yield significant initial Accuracy.

## Section 3: Data Modeling

The purpose of this final section is to employ the findings of the "data exploration" to construct the most accurate model possible.

First, the author partitions the "voting" dataframe into training and testing sets, with 90% of the data being fed into "training" in order to maximize Accuracy given the relatively small number of observations. With more data to train upon, this 9:1 train-test ratio will give the machine learning algorithms sufficient information from which to make their predictions.

Furthermore, the author chose Accuracy as the primary metric for model evaluation because the nature of this project is relatively simple--it is a binary classification. Accuracy can be defined with the following formula:

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Observations}$$

Additionally, since we are not concerned with predicting Democrats more efficiently than Republicans, or vice versa, Sensitivity, Specificity, and F-1 Scores are not needed to gauge model performance.

Once the data is ready, the author proceeds to develop 7 models, in the following order:

1. **Random Guessing** (sample())
2. **Party-Line Index** (ifelse())
3. **Logistic Regression** (train(), method = "glm")
4. **K-Nearest Neighbors** (train(), method = "knn")
5. **Random Forest** (train(), method = "rf")
6. **Neural Network** (train(), method = "nnet")
7. **Majority-Vote Ensemble** (ifelse())

While the traditional machine-learning algorithms utilized in this section are only trained upon the 16 floor votes, Model #2 takes a different approach with the use of the Party-Line Index. This value is recalculated at the start of this section with the "training" data alone so as to avoid the possibility of overtraining on the "testing" data.

The report ends with a brief summary of the results and a series of concluding remarks, including a discussion of limitations and suggestions for future research. The most successful model obtains an Accuracy of 97.7%, well surpassing the 90% Accuracy threshold and deemed satisfactory by the author as a final product.

# Methods & Analysis

## Section 1: Data Preparation

The purpose of this first section is to prepare the data for analysis. This dataset consists of 435 rows and 17 columns. Each row represents a member of the House of Representatives, and the columns are of two types. The first column is the name of the member's political party, and the remaining columns are the member's "Yea" or "Nay" vote regarding each item.

If we look at the state of the data in its original downloaded form, we can see that it already looks pretty good. Here are the first 3 entries:

| party | Infants | H2O | Budget | Medical | ElSalvador | |
|---|---|---|---|---|---|---|
| republican | n | | y | n | y | y |
| republican | n | | y | n | y | y |
| democrat | ? | | y | y | ? | y |
| Religion | Testban | | Contras | Missile | Immigration | Synfuels |
| y | n | | n | n | y | ? |
| y | n | | n | n | n | n |
| y | n | | n | n | n | y |
| Education | Superfund | | Crime | Dutyfree | SAfrica | |
| y | y | | y | n | y | |
| y | y | | y | n | ? | |
| n | y | | y | n | n | |

Votes are classified as either "y," "n," or "?"--that is, "Yea," "Nay," or Position Unknown. In that dataset comments, Dr. Schlimmer notes that these categories are condensed versions of the full range of voting options identified by the Congressional Quarterly Almanac. essentially, positive votes and endorsements count as "Yeas," while the negative counterparts to these actions count as "Nays." A "?" is provided 1) when a member voted "Present" (read: both a positive or negative would be politically damaging), 2) when a member voted "Present" when there would otherwise be a conflict of interest, and 3) when a member for whatever reason did not make their position known.

These unknown cases would serve to hinder the modeling process if they were left unaddressed, but we do not yet have any justification to recode these values one way or another. Therefore, we will temporarily recode the "?"s as "NAs" until we have developed a valid line of reasoning to transform them further.

While the vote data requires some tweaking, the predictors are already in good shape. The names of the two political parties contained in the first column--Democrats and Republicans--are not capitalized, and we will make that change for the sake of more aesthetically-pleasing graphs plots in the "data exploration." When we read-in the data from the UCI Machine Learning Repository, we rename each column for the sake of simplicity. Here are the item names and what we know about them from the data comments and from www.Congress.gov, a website that tracks the activities of the U.S. Congress:

1. **Infants (H.R.808)**: Handicapped Infants Protection Act of 1982; This item directs an agency to study child abuse and neglect in facilities linked to the federal government. It also establishes protections for victims. (https://www.congress.gov/bill/98th-congress/house-bill/808)

2. **H2O (H.R.3678)**: Water Resources Conservation…Act of 1983; This item protects public water resources in the United States. (https://www.congress.gov/bill/98th-congress/house-bill/3678)

3. **Budget (Ambiguous)**: This item is labeled as the "adoption of the budget resolution," but the author cannot identify exactly which item the dataset is referring to.

4. **Medical (H.R.4170)**: Deficit Reduction Act of 1984; Among many other things, this item places a freeze on physician fees for Medicare recipients. The overall bill is designed to cut benefits in order to shrink the federal deficit. (https://www.congress.gov/bill/98th-congress/house-bill/4170)

5. **ElSalvador (H.R.4042)**: A bill to continue…the current certification requirements with respect to El Salvador…; This item temporarily extends financial aid to El Salvador at its current level. (https://www.congress.gov/bill/98th-congress/house-bill/4042)

6. **Religion: (Ambiguous)**: This item is labeled as "religious groups in schools," but the author cannot identify exactly which item the dataset is referring to. This item is likely some iteration of the Equal Access Act.

7. **Testban: (H.R.5167)**: Department of Defense Authorization Act, 1985; The researcher believes that this item refers to a proposed amendment to this bill that would prevent the use of anti-satellite missiles (ASAT) by the U.S. military. (https://www.congress.gov/bill/98th-congress/house-bill/5167)

8. **Contras: (H.R.5167)**: Department of Defense Authorization Act, 1985; The researcher believes that this item is another proposed amendment to the DoD Authorization Act. This amendment seeks to restrict U.S. assistance to the Nicaraguan Contras. (https://www.congress.gov/bill/98th-congress/house-bill/5167)

9. **Missile: (H.R.4952)**: A bill to authorize…assistance to certain Indian tribes…relating to the planned deployment of the MX missile system; This item attempts to compensate Native American tribes that were negatively impacted by domestic U.S. weapons testing. (https://www.congress.gov/bill/98th-congress/house-bill/4952)

10. **Immigration (H.R.1510)**: Immigration Reform and Control Act of 1983; This item outlaws the employment of undocumented immigrants in the United States. (https://www.congress.gov/bill/98th-congress/house-bill/1510)

11. **Synfuels (Ambiguous)**: This item is labeled as "synfuels corporation cutback," but the author cannot identify exactly which item the dataset is referring to. The item definitely relates to efforts to reduce federal investment in Synfuels, an energy corporation created by Congress in 1980.

12. **Education (Ambiguous)**: This item is labeled as "education spending," but the author cannot identify exactly which item the dataset is referring to. It is unclear whether this is a measure to increase or decrease federal appropriations in education.

13. **Superfund (H.R.5640)**: Superfund Expansion and Protection Act of 1984; This item establishes a new tax on the disposal of hazardous materials into the ground and

increases funding for the Hazardous Substance Superfund. (https://www.congress.gov/bill/98th-congress/house-bill/5640)

14. **Crime (Ambiguous)**: This item is labeled as "crime," and the author cannot identify exactly which item the dataset is referring to. This predictor may be connected to the Victims of Crime Act of 1984.

15. **Dutyfree (H.R.3398)**: Omnibus Tariff and Trade Act of 1984; This item amends U.S. tariff schedules. Note: the author is not totally certain that this bill is the same as the item referenced in the dataset. The predictor might be a proposed amendment to this bill. (https://www.congress.gov/bill/98th-congress/house-bill/3398)

16. **SAfrica (H.R.4230)**: Export Administration Act Amendments of 1984; This bill amends U.S. trade policies, and it specifically declares American opposition to apartheid. It also restricts certain financial loans to South Africa. (https://www.congress.gov/bill/98th-congress/house-bill/4230?s=1&r=92)

### Data Cleaning

As previously described, the data does not require much pre-processing in order for the analysis to proceed. Here is the full list of actions taken by the author to prepare the data at this stage:

- Binarize the "Yeas" and "Nays"
- Replace "?" with "NA"
- Capitalize the first letter of political party names
- Compile the transformed data into the primary "voting" dataframe to facilitate the "data exploration"

In case you're curious, here is the code we use to binarize the votes and prepare the NAs:

```
data <- data.frame(sapply(data, function(x){
  ifelse(x == '?', NA, ifelse(
    x == 'y', 1, ifelse(
      x == 'n', 0, ifelse(
        x == 'republican', 'republican', 'democrat'
)))))}))
```

Now we can view the pre-processed data:

| party | Infants | H2O | Budget | Medical | ElSalvador |
|---|---|---|---|---|---|
| Republican | 0 | 1 | 0 | 1 | 1 |
| Republican | 0 | 1 | 0 | 1 | 1 |
| Democrat | NA | 1 | 1 | NA | 1 |
| Religion | Testban | Contras | Missile | Immigration | Synfuels |
| 1 | 0 | 0 | 0 | 1 | NA |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| Education | Superfund | Crime | Dutyfree | SAfrica | |
| 1 | 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | NA | |
| 0 | 1 | 1 | 0 | 0 | |

Besides the missing values, the data is looking great, and we're ready to proceed into the exploratory stage.

## Section 2: Data Exploration

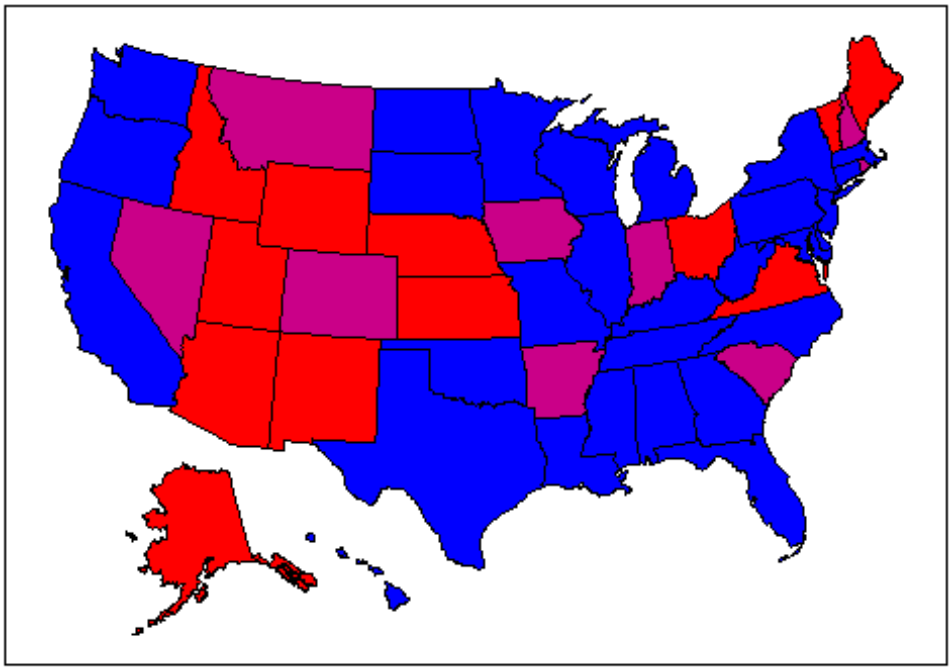
### Activity 1): Plot 1982 Midterm Election Results

Before we begin analyzing Dr. Schlimmer's voting data, it would be beneficial to put the 98th Congress in context. We can do this by reviewing the 1982 midterm election results, which we can load into R using the "politicaldata" package, like this:

```
data('house_results')
```

Oftentimes the best way to obtain a general impression of the state of a political system is to look at maps depicting control of various geographical constituencies. In this case, we can look at a map depicting party control of House delegations.

United States Congress, 98th Session (1983-1985)
Party Control of U.S. House Delegations

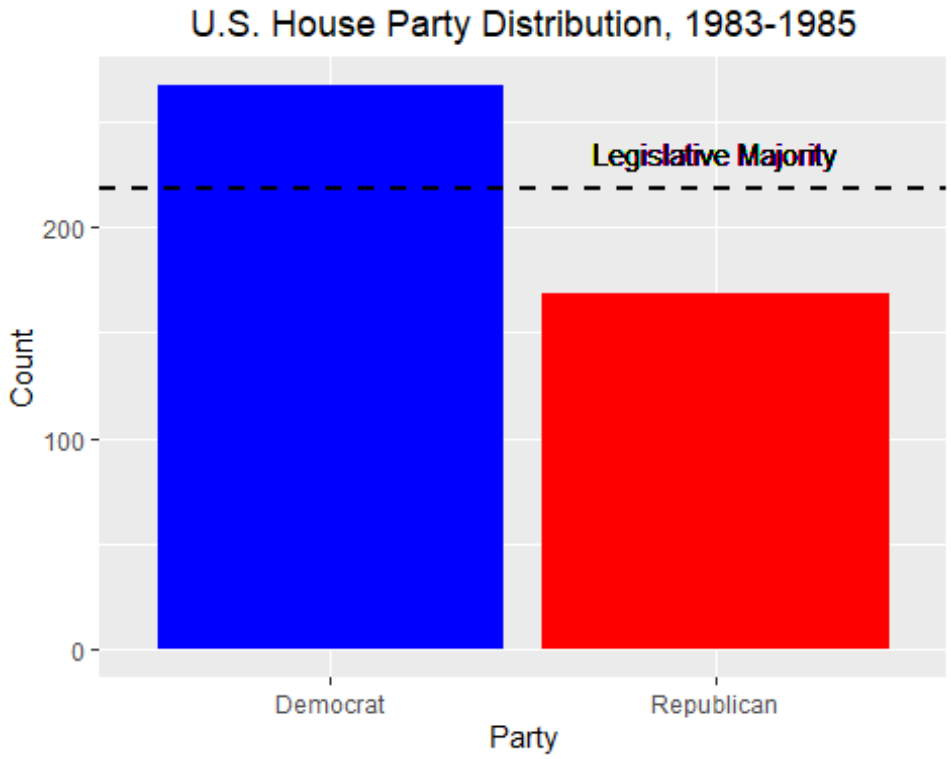


If you are familiar with modern U.S. politics, you'll be seeing a lot more blue across this map then you're probably used to. In fact, this party distribution is relatively typical for the mid-20th century. Democrats controlled the House of Representatives for 40 consecutive years from 1954-1994, and at the time of the 98th Congress from 1983-1985, House Speaker Tip O'Neill was nearing the end of his 10 year speakership. Based on this map alone, we would assume that Democratic legislation would nearly always pass and Republican legislation would nearly always fail. But as we are to soon discover, that was often not the case.

Speaker O'Neill presided over a much different Congress than what we have today. Political parties were much less polarized, and the conservative Democrats that helped to maintain the long blue majority would easily be considered Republicans by modern standards. While the parties were deeply divided on domestic economic concerns, they were more unified on social issues and matters of foreign policy than members are today. This dynamic of party-ideological mixing makes it more difficult to classify members by party for the 98th Congress than it would for the 116th Congress of 2020. We will address this problem in more detail later in this section.

### Activity 2): Analyze party makeup of the 98th Congress

Having briefly summarized the state of the Congress, let's transition into an analysis of the dataset in question. Below, you will find a bar graph displaying party distribution in the House of Representatives.



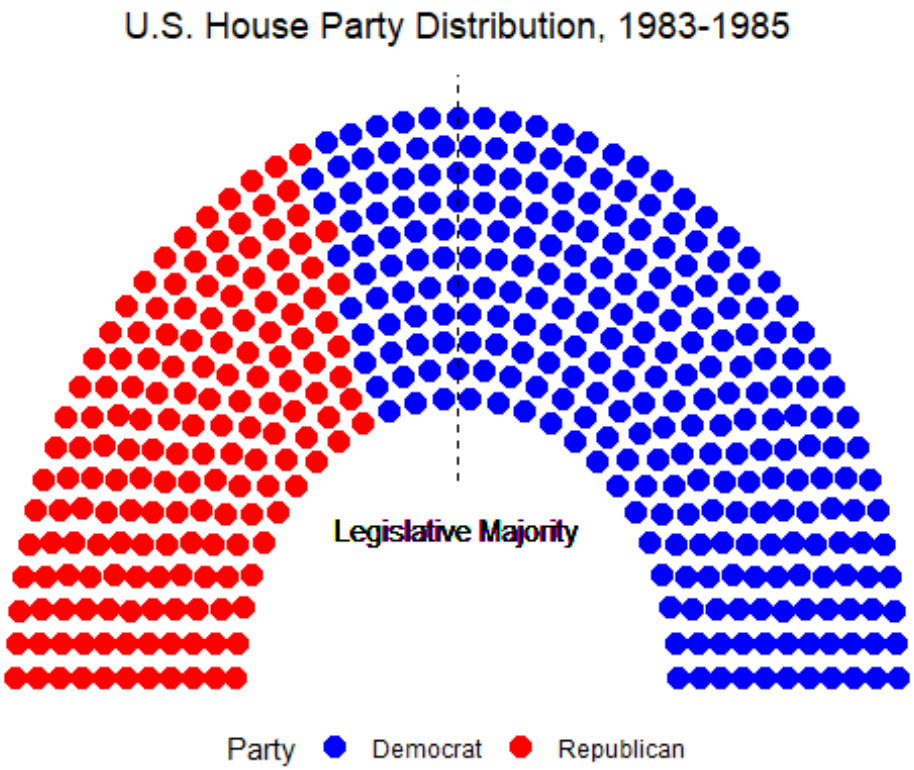

In accordance with the 1982 midterm results, we see that Democrats enjoy a strong majority in the House. The number of representatives is set by law at 435, so a majority of these at 218 members constituents a governing majority. Here is the exact number of seats controlled by each party:

| Democrat | Republican |
|---|---|
| 267 | 168 |

Additionally, here is the proportion of seats controlled by each party:

| Democrat | Republican |
|---|---|
| 0.6137931 | 0.3862069 |

Another way to visualize the distribution of seats by party is with the use of a parliament plot. We can generate one with the “ggparliament” package.




The advantage of this method is that you can see exactly how many seats each party is above & below a majority. Again, if you follow modern U.S. politics, you might be surprised by how large the Democratic majority is. In the 116th Congress, Democrats have only maintained control of ~235 seats, around 15 seats above the number required for control of the chamber.

But remember: the Democratic majority of the 1980s was maintained with conservative Democrats who often voted with Republicans. Therefore, Democratic control in the 98th Congress did not always translate to Democratic victories.

### Activity 3): Calculate whether or not each bill passed

Next, we determine whether or not the House passed each bill according to the tallies of the "Yeas" and the "Nays." Note that we assume a quorum to always be present. According to House rules, in the presence of a quorum a measure passes when the "Yeas" supersede the "Nays" by 1 vote. Therefore, an item does not always require 218 votes to pass. Please see *United States v. Ballin* (1892) for more information on quorums and thresholds for item approval.

Here is the code we use to determine whether each item passed or failed:

```
measure_outcomes <- sapply(voting[,2:17], function(x){
  yea <- x[x == 1]
  nea <- x[x == 0]
  ifelse(length(yea) > length(nea), 'Pass', 'Fail')
})
```

Using this information, we can determine how many items passed and failed:

| Passed Legislation | Failed Legislation |
|---|---|
| 11 | 5 |

Additionally, here is the breakdown for how each bill fared.

Again, since the Democrats are in the majority, we would expect of their legislation to pass. And, conversely, we would expect most Republican legislation to fail. We test this assumption in the next activity.

### Activity 4): Assess bill partisanship

Here we provide a complete tabulation on party votes for each item in the dataset.

| Item | Result | Yeas | Nays | NVPR | (D)Y | (R)Y | (D)N | (R)N | (D)? | (R)? |
|---|---|---|---|---|---|---|---|---|---|---|
| Religion | Pass | 272 | 152 | 11 | 123 | 149 | 135 | 17 | 9 | 2 |
| SAfrica | Pass | 269 | 62 | 104 | 173 | 96 | 12 | 50 | 82 | 22 |
| Budget | Pass | 253 | 171 | 11 | 231 | 22 | 29 | 142 | 7 | 4 |
| Crime | Pass | 248 | 170 | 17 | 90 | 158 | 167 | 3 | 10 | 7 |
| Contras | Pass | 242 | 178 | 15 | 218 | 24 | 45 | 133 | 4 | 11 |
| Testban | Pass | 239 | 182 | 14 | 200 | 39 | 59 | 123 | 8 | 6 |
| Immigration | Pass | 216 | 212 | 7 | 124 | 92 | 139 | 73 | 4 | 3 |
| ElSalvador | Pass | 212 | 208 | 15 | 55 | 157 | 200 | 8 | 12 | 3 |
| Superfund | Pass | 209 | 201 | 25 | 73 | 136 | 179 | 22 | 15 | 10 |
| Missile | Pass | 207 | 206 | 22 | 188 | 19 | 60 | 146 | 19 | 3 |
| H2O | Pass | 195 | 192 | 48 | 120 | 75 | 119 | 73 | 28 | 20 |
| Infants | Fail | 187 | 236 | 12 | 156 | 31 | 102 | 134 | 9 | 3 |
| Medical | Fail | 177 | 247 | 11 | 14 | 163 | 245 | 2 | 8 | 3 |
| Dutyfree | Fail | 174 | 233 | 28 | 160 | 14 | 91 | 142 | 16 | 12 |
| Education | Fail | 171 | 233 | 31 | 36 | 135 | 213 | 20 | 18 | 13 |
| Synfuels | Fail | 150 | 264 | 21 | 129 | 21 | 126 | 138 | 12 | 9 |

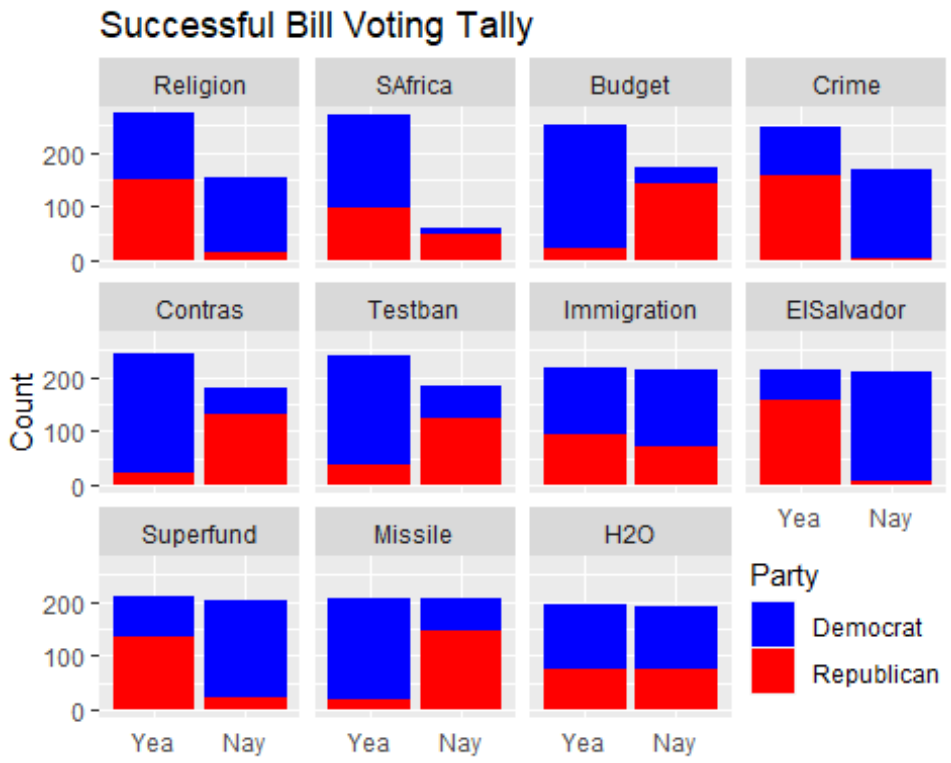
NVPR refers to counts of the missing data. (D)? and (R)? fulfill this same purpose for their respective parties.

Surprisingly, we see that not all bills in which Democrats comprised a majority of the "Yeas" passed. In these instances, such as with the "Synfuels" and "Dutyfree" items, many rogue Democrats sided with united Republicans in opposition to the bill and caused it to sink. This form of internal dysfunction will characterize the parties in this study.

### Activity 5): Plot voting results for each bill, stacking by party using real votes

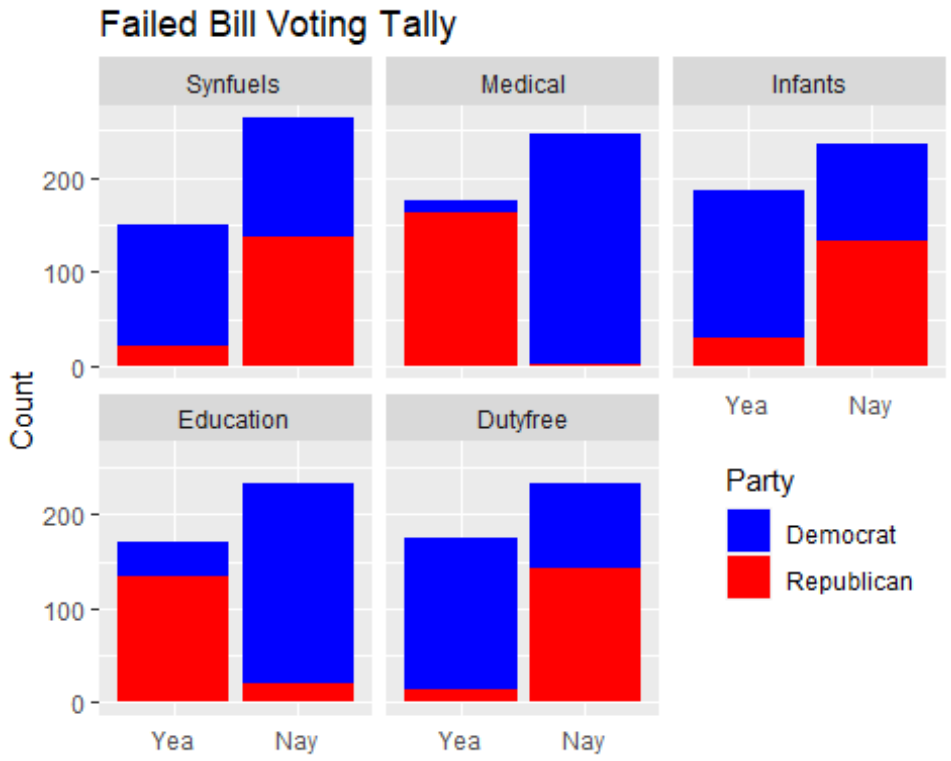
Up next, we plot the "Yea" and "Nay" counts for each item, and we use the "ggplot2" group_by() function to color the bars y party. The result is quite striking.

Here is the distribution of votes for each passing item:




We can immediately see that certain votes were taken straight down party lines, such as the vote on the "Budget." This item received only 22 Republican votes, according to the table generated in Activity 4). We can also see that other items enjoyed a clear consensus from one party and strong divisions from another. For example, the "Crime" bill was supported by all but 3 Republicans, while about 1/3 of Democrats broke from the party majority and voted "Yea." Finally, we see some bills where both parties are equally divided. The "H2O" bill is a perfect example of this dynamic, with about 51% of each party voting "Yea."

Here is the same plot for the items that failed the House:




We see some interesting trends within these items as well. There are 2 patterns. First, "Synfuels," "Infants," and "Dutyfree" all appear to have failed due to strong Republican resistance and divided Democratic support. Second, "Medical" and "Education" both seem to have failed due to near perfect Democratic opposition.

The "Medical" item is especially interesting, as only 2 Republicans voted "Nay" alongside the vast majority of Democrats. Given the extreme partisanship of this vote, "Medical" will likely be a very useful item for party classification once we begin to model for party affiliation.

### Activity 6): Develop Categories for Bill Partisanship

Our next step is to determine how the majority of each party voted for each item. We refer to this dynamic as "party-majority voting." We define the orthodox party position on each item by how each party majority votes. Having defined partisan orthodoxies, we will be able to quantify deviations present among individual members from their party. We will use this metric as an approximation of ideology.

Here is the code we use to calculate party-majority voting. Note that we use the term "Party-Line" interchangeably with this concept.

```
for(i in 1:16){
  dem_majority <- ifelse(vote_details[i,6] > vote_details[i,8], 1, 0)
  gop_majority <- ifelse(vote_details[i,7] > vote_details[i,9], 1, 0)
  bill_party_affiliation <- ifelse(dem_majority == gop_majority,
'Bipartisan', ifelse(
```

```
    dem_majority > gop_majority, 'Democratic', ifelse(
      dem_majority < gop_majority, 'Republican', 'Minority Support')))
  bill_partisanship[nrow(bill_partisanship) + 1,] <- c(vote_details[i,1],
bill_party_affiliation, vote_details[i,2])
}
```

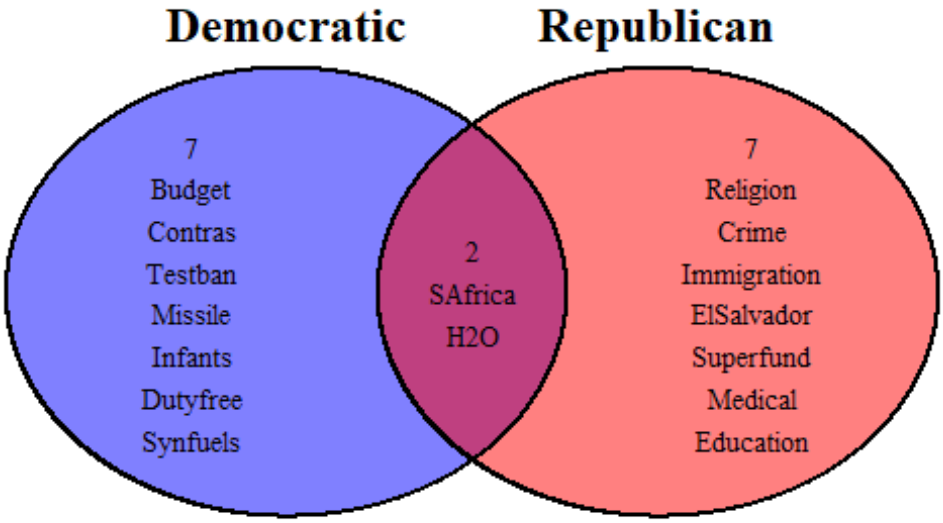
Here are the results of the party-majority voting calculation:

| Item | Partisanship | Result |
|---|---|---|
| SAfrica | Bipartisan | Pass |
| H2O | Bipartisan | Pass |
| Budget | Democratic | Pass |
| Contras | Democratic | Pass |
| Testban | Democratic | Pass |
| Missile | Democratic | Pass |
| Infants | Democratic | Fail |
| Dutyfree | Democratic | Fail |
| Synfuels | Democratic | Fail |
| Religion | Republican | Pass |
| Crime | Republican | Pass |
| Immigration | Republican | Pass |
| ElSalvador | Republican | Pass |
| Superfund | Republican | Pass |
| Medical | Republican | Fail |
| Education | Republican | Fail |

We see that most bills were partisan, and the only 2 bipartisan bills found in the dataset both passed. Bipartisanship is defined for this project as being an instance in which a majority of both parties voted “Yea” on an item.

We also see that Democrats and Republicans were about equally likely to have an item fail. Interestingly, despite being in the minority Republicans passed 5 partisan bills, while Democrats only passed 4. We are beginning to see the influence of complex ideologies, as empowered Democrats have been found to be less successful at passing legislation (at least within this sample.)

To create a clear visual depicting item relationship to party, we generate a Venn Diagram using the “RAM” package.




### Activity 7): Generate Party-Line Index, depicting the ideal Democratic member

For the Activity, we want to develop an index that describes how well each member's voting record aligns with the perfect Democrat, according to the party-majority voting classifications previously identified. We refer to this index as the Party-Line Index (PL-Index).

First, however, we need to recode the NAs in the voting data. With our new understanding of party-majority voting, we have a justification for transforming these values. We know the orthodox party vote for each item, and we also know that partisan legislators support their party when they vote according to this orthodoxy.

When legislators do not vote according to the orthodoxy, they can therefore be understood to be voting against the interest of their party. For example, an individual Democratic "Yea" vote on an orthodox Democratic "Nay" vote would be a clear violation of the Party-Line. This logic extends to "Present" votes as well.

Since an item requires a majority of "Yea" votes to pass and a "Present" vote serves only to provide a quorum, then "Present" votes always contradict the party orthodoxy. This conclusion is based on the reality that, except in the instance of an absolute tie, there will always be a "Yea" or "Nay" party majority for each item. Voting "Present" is not a "Yea" or "Nay," so it cannot fall in line with the party orthodoxy.

Therefore, it is appropriate to recode the NAs as the opposite of the party-majority vote for each item. We refer to this process as the "Inverse Party-Line Fix" (IPL-Fix). Here is the code we use to apply the fix to the "voting" dataframe:

First, we calculate the Inverse Party-Line Fix for each party. This code generates a vector that contains the vote opposite of the party orthodoxy.

```
ipl_fix_dem <- sapply(bill_partisanship_edit[,2], function(x){
  ifelse(x == 'Bipartisan', 1, ifelse(
    x == 'Republican', 1, 0))
})

ipl_fix_gop <- sapply(bill_partisanship_edit[,2], function(x){
  ifelse(x == 'Bipartisan', 1, ifelse(
    x == 'Democratic', 1, 0))
})
```

Second, we apply the IPL-Fix to each NA within the "voting" dataframe.

```
for(i in 1:435){
  if(voting2[i,1] == 'Republican'){
    for(j in 2:17){
      voting2[i,j] <- ifelse(is.na(voting2[i,j]), ipl_fix_gop[j - 1],
voting2[i,j])}}
  if(voting2[i,1] == 'Democrat'){
    for(j in 2:17){
      voting2[i,j] <- ifelse(is.na(voting2[i,j]), ipl_fix_dem[j - 1],
voting2[i,j])}}
}
```

If we look at the first few rows of "voting2"--a copy of "voting" used to preserve the original data--we will see that the NAs have been appropriately filled-in:

| party | Infants | H2O | Budget | Medical | ElSalvador |
|---|---|---|---|---|---|
| Republican | 0 | 1 | 0 | 1 | 1 |
| Republican | 0 | 1 | 0 | 1 | 1 |
| Democrat | 0 | 1 | 1 | 1 | 1 |

| Religion | Testban | Contras | Missile | Immigration | Synfuels |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |

| Education | Superfund | Crime | Dutyfree | SAfrica |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 |

Having recoded the NAs, we are ready to calculate the Party-Line Index. Here is a formula which describes the process:

$$Party\ Line\ Index = \frac{Democratic\ Orthodox\ Votes}{Total\ Votes}$$

A score of "1" is a "Perfect Democrat," and a score of "0" is a "Perfect Republican." Each member incurs a point for every vote they cast alongside the majority of Democrats. The index is a result of summing these points, divided by the total possible number of points. Also, the index ignores votes for or against the two bipartisan bills--"SAfrica" and "H2O."

Here is the code we use to generate the index:

```
for(i in 1:435){
  index <- vector()
  for(j in c(2, 4:16)){
    index <- c(index, ifelse(voting2[i,j] == ipl_fix_gop[j - 1], 1, 0))}
  total <- sum(index)
  dem_score <- total / length(index)
  dem_index <- c(dem_index, dem_score)
}
```

We can see the first few entries of the Party-Line Index with head():

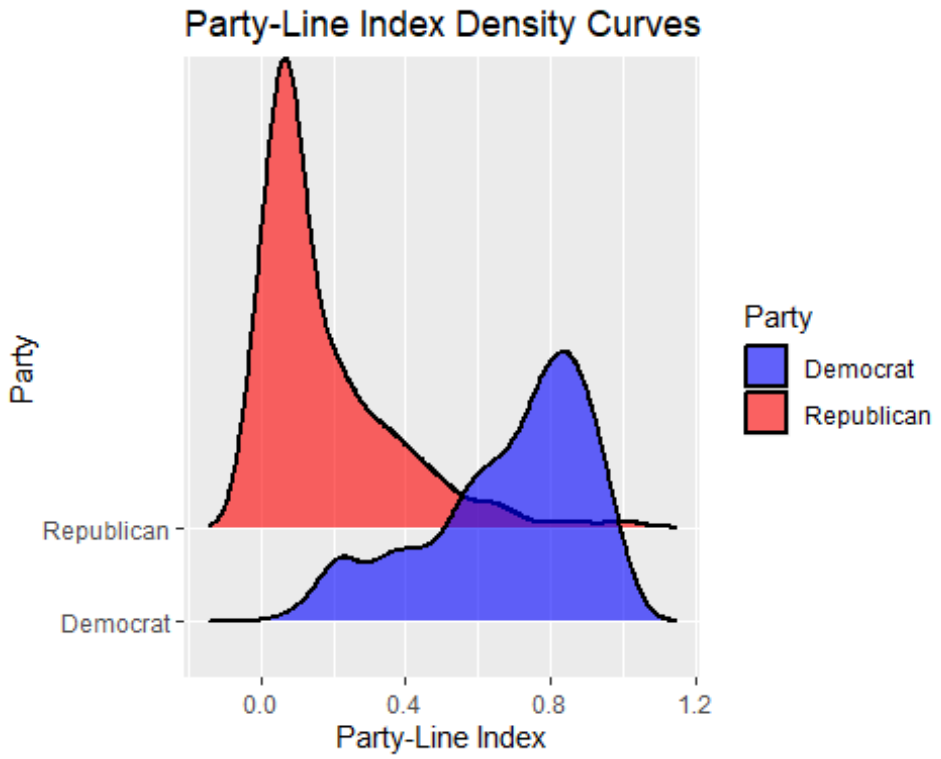
```
## [1] 0.07142857 0.07142857 0.28571429 0.42857143 0.42857143 0.35714286
```

And we can view the men, median, etc. with summary():

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1429  0.5000  0.4867  0.7857  1.0000
```

Interestingly, the median is right at 0.5000--a perfect centrist voter. However, the mean is slightly below the median at 0.4867. This dynamic could be due to the influence of "Reagan Democrats," otherwise known as the "Blue Dogs." These are the Democrats that we already discussed who maintain their Democratic identity but regularly vote with Republicans.

We can see the distribution of Party-Line Index scores by party using the "ggridges" package:




We see that there is a significant difference between how Republicans and Democrats vote on average. However, Republicans, who have one major peak in their distribution, seem much more unified than Democrats, who have several minor peaks at the conservative end of their density curve.

Despite this, the overall gap between Democratic and Republican voting implies that we will be able to successfully predict most legislators' parties based upon their voting record.

### Activity 8): Classify for party-ideological "tilt" using the Party-Line Index

In order to explain the overlap among Party-Line Index scores for some Democrats and Republicans, we take a deeper look at ideology by identifying six potential ideological categories for House members. These classifications are determined by how often each member votes with their party orthodoxy.

Here are the Democratic categories:

1. **Far-Left (D)**: Democrats who vote with the party 90% of the time or more; This term comes from the liberal 1960s-70s policy positions of strong Democrats in the Reagan era.

2. **New Deal (D)**: Democrats who vote with the party 60%-89.99% of the time; This term refers to the New Deal of President F.D.R.--standard economic policies for most Democrats in the 1980s.

3. **Blue Dog (D)**: Democrats who vote with the party less than 60% of the time; Otherwise known as "Reagan Democrats," these members sided with President

Reagan on many issues in the 1980s when the country was growing more conservative. In the coming years, many "Blue Dogs" would officially realign to the Republican Party.

Here are the Republican categories:

1. **Far-Right (R)**: Republicans who vote with the party 90% of the time or more; This term comes from the extremely conservative social and economic policies of Republican hard-liners in the 1980s.

2. **Goldwater (R)**: Republicans who vote with the party 60%-89.99% of the time; The namesake of the category, Barry Goldwater, ran for president in 1964 on a fiscally conservative platform. These Republicans support neoliberal economic policies, but they are not as conservative as the far-right.

3. **Rockefeller (R)**: Republicans who vote with the party less than 60% of the time; This category is named after V.P. Nelson Rockefeller, who led the progressive wing of the Republican Party in the mid 20th century.

Here is the For-Loop we use to classify each member:

```
for(i in 1:435){
  if(voting[i,1] == 'Democrat'){
    tilt <- ifelse(voting[i,18] >= 0.9, 'Far-Left', ifelse(
      voting[i,18] < 0.9 & voting[i,18] >= 0.6, 'New Deal', ifelse(
        voting[i,18] < 0.6, 'Blue Dog', 'No Affiliation'
      ))))}
  if(voting[i,1] == 'Republican'){
    tilt <- ifelse(voting[i,18] > 0.4, 'Rockefeller', ifelse(
      voting[i,18] <= 0.4 & voting[i,18] > 0.1, 'Goldwater', ifelse(
        voting[i,18] <= 0.1, 'Far-Right', 'No Affiliation'
      ))))}
  tilt_results[nrow(tilt_results) + 1,] <- tilt
}
```

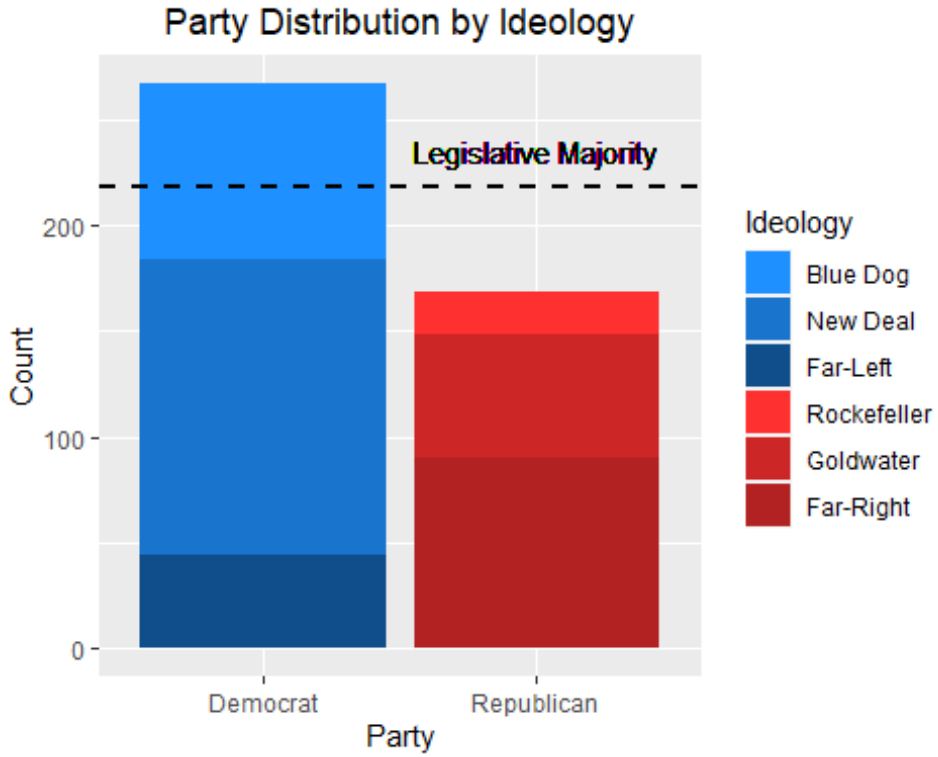
And here is the tally of ideological class membership:

```
## tilt_analysis
##    Far-Left    New Deal    Blue Dog Rockefeller   Goldwater   Far-Right
##          44         140          83          20          58          90
```

We can see that moderates make up the largest share of Democrats, while the far-right makes up the largest share of Republican members. Additionally, the 2nd largest group for Democrats is conservatives, while the 2nd largest group for Republicans is moderates. The left wing is the smallest class for each party. Therefore, while the 98th Congress was managed by a Democratic "Partisan Majority," individual members trended towards a conservative "Ideological Majority."

This dynamic explains why Republicans passed more partisan bills in the sample than Democrats--Republicans were less willing to shift to liberal policies than Democrats were to shift to conservative policies. The ideological House of Representatives skewed right.

We can visualize this dichotomy by plotting party membership among these ideological categories.



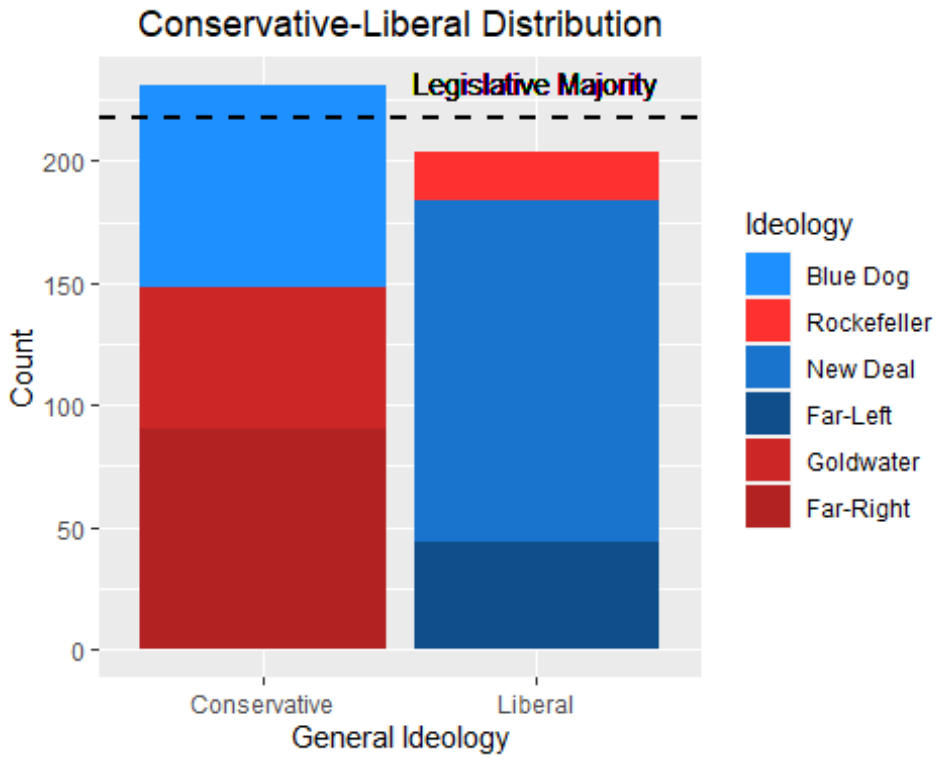

We can generalize this partisan-ideological inequality by summarizing how many generally liberal and generally conservative members were in the House at this time.

We calculate the number of conservatives in the House by adding together the moderate-conservative Republicans + the Blue Dog Democrats. Similarly, we calculate the number of liberals by adding together the moderate-liberal Democrats + the Rockefeller Republicans.

Here is a tally of the results:

```
## ideology_results
## Conservative      Liberal
##          231          204
```

And here is a bar plot of the conservative-liberal ideological scale grouped by the 6 party-based ideologies:




We can see that, despite having a Democratic Speaker, the 98th Congress was controlled by a conservative ideological majority. Again, this finding helps to explain why Republicans passed 5 partisan bills while Democrats only passed 4.

Additionally, the partisan-ideological mixing between conservative Democrats and liberal Republicans could hinder the process of party classification. On the other hand, it is possible that the machine learning algorithms to be employed will detect more subtle trends in the voting data that better indicate party affiliation than proximity to pure party-majority voting.

### Activity 9): Identify Particularly Divisive Items with Partisanship Scores (PART-Scores)

In order to work around the party-ideological mixing that we uncovered in the last section, we will try to identify bills that exhibit a clear partisan consensus. These pieces of legislation should be able to overcome ideological ambiguity to correctly identify party affiliation.

To achieve this, we will calculate "Partisanship Scores" (PART-Scores) for each bill to measure its level of partisanship in member voting. Here is the formula:

$$PART\ Score = 100 * |(\frac{1}{N_D}\sum_{i=1}^{N_D} D\ emocratic\ Vote_i) - (\frac{1}{N_R}\sum_{i=1}^{N_R} R\ epublican\ Vote_i)|$$

In the above formula, $i$ is the index for each member, $N_D$ is the number of Democratic House members, $N_R$ is the number of Republican House members, and $Democratic\ Vote$ & $Republican\ Vote$ are the "Yea" (1) and "Nay" (0) votes cast by each member for each item.

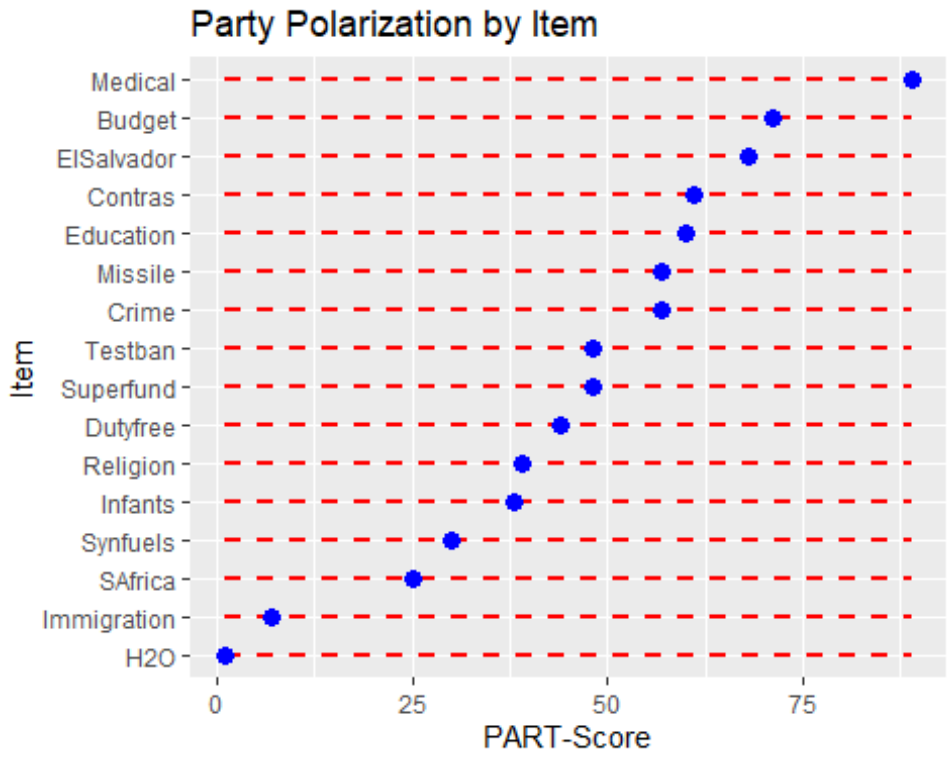
These scores are the absolute value of the difference between Democratic and Republican mean voting for each bill, multiplied by 100. The 0-100 scale of the PART-Scores measures how the absolute difference in voting between the parties, so larger values imply larger differences. Additionally, these scores mimic the Variable Importance values which will be generated by some of the machine learning algorithms we use later on.

Here is the code we use to calculate the PART-Scores:

```
for(i in 2:17){
  abs_diff <- 100 * round(abs(mean(voting2[,i][voting2$party == 'Democrat'])
- mean(voting2[,i][voting2$party == 'Republican'])), digits = 2)
  part_scores[nrow(part_scores) + 1,] <- abs_diff
}
```

Below you will find the PART-Score calculated for each item in the dataset, as well as a dot plot which shows the results graphically:

| Item | PART-Score |
|---|---|
| Medical | 89 |
| Budget | 71 |
| ElSalvador | 68 |
| Contras | 61 |
| Education | 60 |
| Missile | 57 |
| Crime | 57 |
| Testban | 48 |
| Superfund | 48 |
| Dutyfree | 44 |
| Religion | 39 |
| Infants | 38 |
| Synfuels | 30 |
| SAfrica | 25 |
| Immigration | 7 |
| H2O | 1 |




We see that there is a clear partisan consensus on the "Medical" bill, with a PART-Score of 89 / 100. The second highest PART-Score is for the "Budget" item at 71, and the third is for "ElSalvador" at 68. Due to their extreme partisanship, we should be able to expect these three bills to overcome ideological ambiguity and effectively classify members by party, especially with regards to the "Medical" item. This aligns with our earlier discussion regarding the "Medical" bill as a clear instance of straight party-line voting.

Just to be safe, we will recalculate PART-Scores, now grouping by conservative-liberal ideology. If the top few scored items are different for these "IDEA-Scores" than for the PART-Scores, then we can be confident that the items identified with the latter can correctly identify party affiliation. Here is the formula:

$$IDEA\ Score = 100 * |(\frac{1}{N_L}\sum_{i=1}^{N_L} Liberal\ Vote_i) - (\frac{1}{N_C}\sum_{i=1}^{N_C} Conservative\ Vote_i)|$$

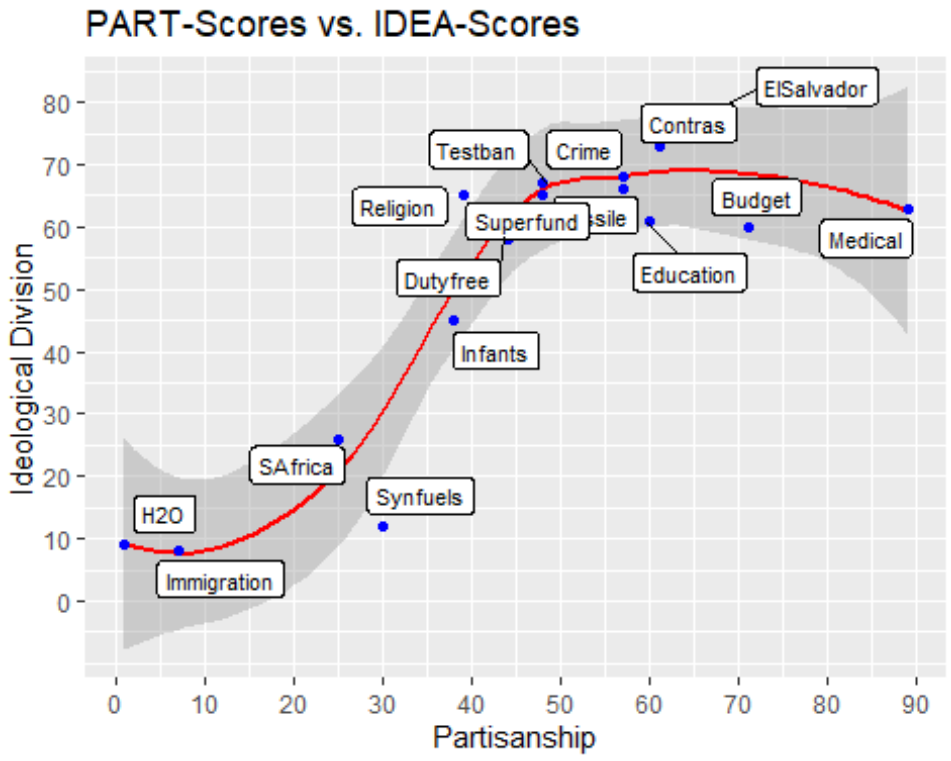
In the above formula, $i$ is the index for each member, $N_L$ is the number of liberal House members, $N_C$ is the number of conservative House members, and $Liberal\ Vote$ & $Conservative\ Vote$ are the "Yea" (1) and "Nay" (0) votes cast by each member for each item.

And here are the most ideologically-divisive items:

| Item | PART-Score | IDEA-Score |
|---|---|---|
| Medical | 89 | 63 |
| Budget | 71 | 60 |
| ElSalvador | 68 | 79 |
| Contras | 61 | 73 |
| Education | 60 | 61 |
| Missile | 57 | 66 |
| Crime | 57 | 68 |
| Testban | 48 | 67 |
| Superfund | 48 | 65 |
| Dutyfree | 44 | 58 |
| Religion | 39 | 65 |
| Infants | 38 | 45 |
| Synfuels | 30 | 12 |
| SAfrica | 25 | 26 |
| Immigration | 7 | 8 |
| H2O | 1 | 9 |

As opposed to earlier, “ElSalvador” is now the most divisive item, followed by “Contras” and “Crime.” However, the IDEA-Scores are less extreme than the PART-Scores, so these top items might not have as much predictive power for ideology as the top PART-Scored items are for party. “Medical” and “Budget” are now in #8th and #10th place respectively, far from being the most important items. Therefore, it seems like “ElSalvador” will likely not be very effective at identifying party, as many ideologically-aligned members of both parties voted for it. This leaves “Medical” and “Budget” as the two most likely bills to have the strongest predictive power.

To conclude the data exploration, we will plot PART-Scores versus IDEA-Scores, visualizing the 98th Congress’s party-ideological split. Remember: if parties and ideologies were perfectly aligned, the resulting plot would form a straight line.




We see that there is not a clear linear relationship between party and ideology. Again, the observe the "Medical" and "Budget" bills as strong indicators of party, while "ElSalvador" is a strong indicator of ideology. We will keep all of this in mind as we transition into the analysis.

## Section 3: Data Modeling

The purpose of this section is to develop a model for party affiliation in the 98th Congress. This model should have an accuracy of 90% correct predictions or higher.

### Summary of the Modeling Approach:

This project's modeling process involves 5 steps:

1. The "voting" dataframe is transformed into a 90%-sized training set and a 10%-sized testing set using createDataPartition(). These partition values are selected in order to maximize the number of observations allocated for training, thereby increasing the likelihood of accurate model construction while maintaining a reasonably large testing set.

2. The Party-Line Index and Inverse Party-Line Fix are recalculated so as to avoid overtraining on the "testing" set. This step has the incidental consequence of removing the NAs from the post-partition data, which will facilitate modeling.

3. A baseline model is developed upon which new models can be said to improve. This baseline is constructed with sample() in order to generate a random guess for each

member. The expected Accuracy for this model is 50%, as any random guess in a binary situation would have a 50%-50% chance of success.

4. Several intermediate models are developed in order to maximize Accuracy. These models are of two types: 1) an ifelse()-based model that uses the Party-Line Index to predict party affiliation, and 2) models developed using various traditional machine learning algorithms from the "caret" package. We hope that at least of one these intermediate models achieves an Accuracy of 90% or better.

5. An ensemble model is developed using the results of each successful intermediate model. This model uses the majority-vote system, in which a prediction is made for each member based on the majority opinion of the constituent models. We expect this model to have an Accuracy approximating or improving upon the most successful individual models.

If any of the models developed in this section achieve an Accuracy of 90% or higher, this project shall be considered a success, and the most accurate model(s) will be presented in the "Results" section as the final product.

### Preparation for Modeling

To get started, we will partition the data into the training and testing sets. Here is the code. (Note: we set the code before this process because it involves random number generation).

```
set.seed(1, sample.kind="Rounding")
partition <- createDataPartition(y = voting$party, times = 1, p = 0.1, list = FALSE)
training <- voting[-partition,]
testing <- voting[partition,]
```

Next, we calculate the distribution of Democratic and Republican members in each partitioned set relative to the original "voting" data. If the proportions of party membership are relatively consistent across all divisions, then we will be safe to proceed.

| Set | Party | True % | New % | Difference |
|---|---|---|---|---|
| Training | Democrat | 0.61379 | 0.61381 | -2e-05 |
| Training | Republican | 0.38621 | 0.38619 | 2e-05 |
| Testing | Democrat | 0.61379 | 0.61364 | 0.00016 |
| Testing | Republican | 0.38621 | 0.38636 | -0.00016 |

As we can see, the distribution of party membership has remained almost identical across the pre-partition and post-partition data (as well as among the training and testing sets). We are justified to proceed.

### Calculate New Party-Line Index & Inverse Party-Line Fix Values

In order to avoid potential overtraining, we will now re-calculate the Party-Line Index and Inverse Party-Line Fix with the "training" data alone. This process will ensure that these values are not biased by the entries set aside for the "testing" data in any way.

Additionally, the calculation of the PL-Index requires the replacement of NAs with the Inverse PL-Fix, so this step will also serve to replace all of the NAs is the "training" and "testing" sets.

Here is the "training" data with the new Party-Line Index.

| party | Infants | H2O | Budget | Medical | ElSalvador |
|---|---|---|---|---|---|
| 0 | nay | yea | nay | yea | yea |
| 0 | nay | yea | nay | yea | yea |
| 1 | nay | yea | yea | yea | yea |
| Religion | Testban | Contras | Missile | Immigration | Synfuels |
| yea | nay | nay | nay | yea | yea |
| yea | nay | nay | nay | nay | nay |
| yea | nay | nay | nay | nay | yea |
| Education | Superfund | Crime | Dutyfree | SAfrica | dem_index_train |
| yea | yea | yea | nay | yea | 0.0714 |
| yea | yea | yea | nay | yea | 0.0714 |
| nay | yea | yea | nay | nay | 0.2857 |

Note that the PL-Index applied to the "testing" data is based on the distribution of party-majority votes from the "training" data alone. This ensures that no bias will enter into the models that utilize the PL-Index.

Here is the distribution of the new PL-Index for both the post-partition datasets:

```
summary(training$dem_index_train)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1429  0.5714  0.4914  0.7857  1.0000

summary(testing$dem_index_test)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1250  0.4286  0.4448  0.7321  0.9286
```

We can see that the values appear relatively different for each set. While the 1st and 3rd Quartiles appear relatively similar, the mean and median of the "testing" data are much lower than that of the "training" data. This implies that more Republicans (or conservatives) ended up in "testing" relative to "training", which could lead to weaker predictive power for the PL-Index.

We're now ready to begin modeling!

### Baseline Model

## Model 1: Guessing

Our first model is a random guess of each member's partisanship, carried out with sample() random number generation. For the length of the "testing" set, each member is randomly assigned "1" for Democrat and "0" for Republican, with replacement. We set the seed in order to maintain consistency with random sampling. Here is the code:

```
set.seed(1, sample.kind = 'Rounding')
guess_model <- as.factor(sample(c(0,1), nrow(testing), replace = TRUE))
```

We calculate the Accuracy for each model using the Confusion Matrix function. Additionally, we will use the "pixiedust" package to display the results after each model has been fit and applied to the "testing" data. Here are the results:

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |

Since this first attempt was a completely random guess, it makes sense that we received an Accuracy of 50%. As we transition into the intermediate models, we will strive to maximize this value.

### Intermediate Models

The goal of this subsection is to develop a model with an Accuracy of at least 90%. After having fit each model and compiled the Accuracy results, we will combine all of our predictions into a majority-vote ensemble.

## Model 2: Party-Line Index

Our second model is based on the Party-Line Index as it was recalculated on the training data at the start of Section 3. As the Party-Line Index quantifies how often a member votes alongside the Democratic orthodoxy, we can assume that a PL-Index > 50% is a Democratic-leaning member, while a PL-Index <= 50% is a Republican-leaning member. We test this assumption with an ifelse() statement below:

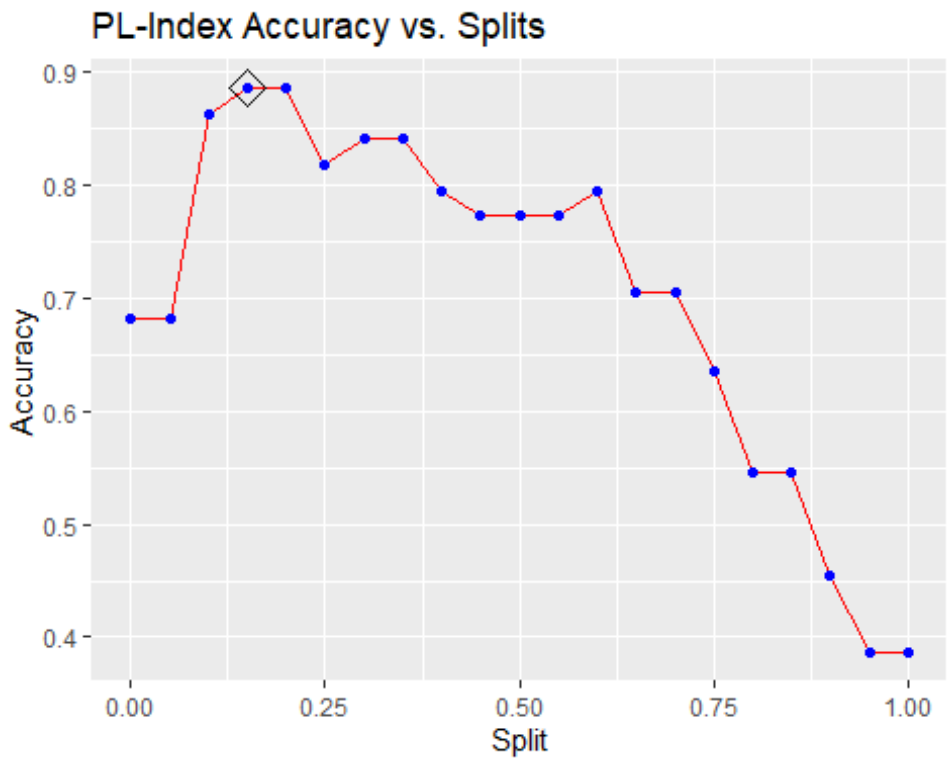
```
partyline_model <- as.factor(ifelse(testing$dem_index_test > 0.50, 1, 0))
```

Here are the results:

```
## [1] 0.7727273
```

We can see that the explanatory power of the Party-Line Index has risen our Accuracy from 50% to 77%. That's really good! However, we might be able to improve this model by tuning the split parameter. In order to do this, we run a For-Loop that calculates the Accuracy returned by all splits from 0 to 1, by 0.05.

Here is a plot of Accuracy vs Model 2 Splits:




Additionally, we can use which.max() to see that we obtain a maximum Accuracy of nearly 89% when we split at 0.15.

```
best_split[which.max(best_split$Accuracy),]

##   Split  Accuracy
## 4  0.15 0.8863636
```

Here is the current standing of model Accuracies.

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |

## Model 3: Logistic Regression

At this point, we transition into using traditional machine learning algorithms accessible through the "caret" package. Our first traditional algorithm is the Logistic Regression, which is a form of linear regression analysis intended for classification problems. Instead of dealing with continuous data, the logistic regression assumes that each observation is a member of a definite class, of which there are several. It is generally advised to use the Logistic Regression instead of simple Linear Regression when you have categorical or binary data, as a Linear Regression model could return predictions which are below 0 or greater than 1. These continuous outputs might not be very useful for classification.

We will start by running a Logistic Regression for the 3 items identified in the "data exploration" as being most notable for their ideological or partisan divisiveness. First, we consider "ElSalvador"--here is the code:

```
set.seed(1, sample.kind = 'Rounding')
glm_fit <- train(party ~ ElSalvador, method = 'glm', data = training[,1:17])
glm_model <- predict(glm_fit, newdata = testing)
```

As seen below, "ElSalvador" returns an accuracy of 75%. This isn't too bad, especially considering our earlier discussion of the partisan-ideological ambiguity of "ElSalvador" as an item with support from conservative Democrats and nearly all Republicans. Due to (limited) bipartisan support, this bill is not the most effective predictor of party affiliation.

```
## [1] 0.75
```

We will likely see greater success with "Budget," which was previously found to be a highly partisan Democratic bill. Here is the result of a Logistic Regression for the "Budget" item alone:

```
## [1] 0.7727273
```

Surprisingly, we see that Accuracy has risen by only ~0.02 from 75% to 77% Accuracy. However, we still have a chance to do better with the "Medical" bill, which we earlier found to be the most partisan item in the dataset. Here is the result of a Logistic Regression with the "Medical" votes alone:

```
## [1] 0.9545455
```

We see that Accuracy has risen to 95.5%. This is great! We have successfully developed a model that surpasses our 90% Accuracy threshold.

Next, we'll see if we can improve this value further by running a combined Logistic Regression of "Medical" + "Budget" + "ElSalvador." Here are the results:

```
## [1] 0.9318182
```

Unfortunately, this combined model drops overall Accuracy down to 93%. While this value remains quite strong, it's not as good as when we used "Medical" alone.

For our last Logistic Regression, we will fit the model upon all 16 bills included in the dataset. This will be our default method from now on. Here is the code:

```
set.seed(1, sample.kind = 'Rounding')
glm_fit <- train(party ~ ., method = 'glm', data = training[,1:17])
glm_model <- predict(glm_fit, newdata = testing)
```

Here are the results:

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |

Logistic 0.9545

Happily, we can see that the Accuracy has risen once again to 95.5%.

In case you're curious, here are the first few predictions generated by our final Logistic Regression model:

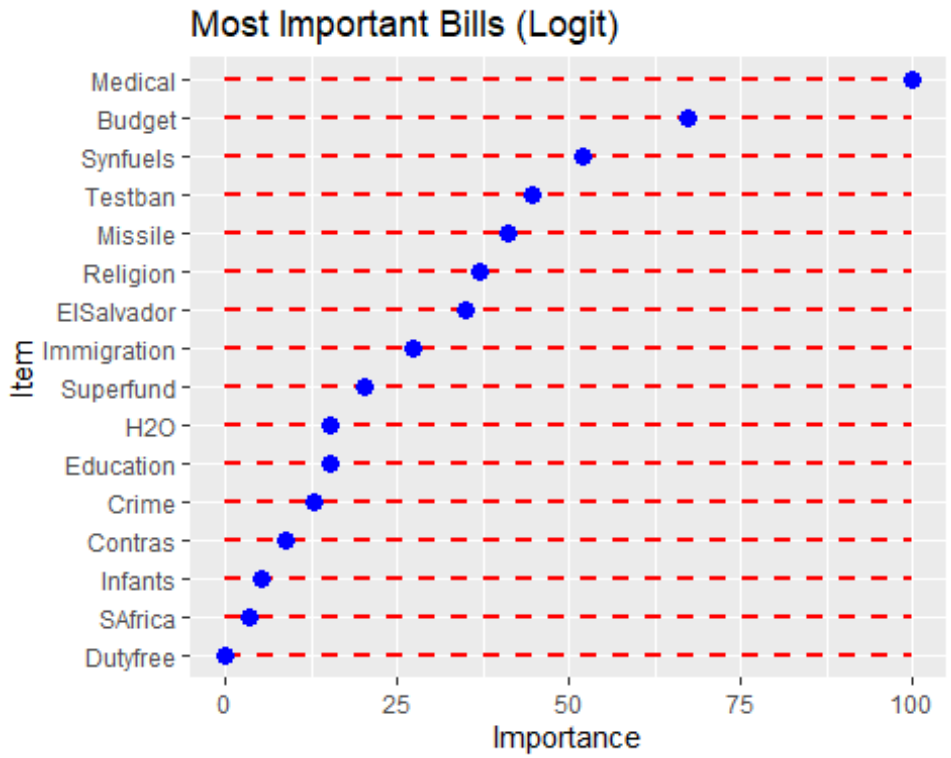
```
head(glm_model)

## [1] 1 0 1 1 0 1
## Levels: 0 1
```

Additionally, we can see which variables were identified by the Logistic Regression as having the most predictive power with the varImp() function.

```
##           Item Importance
## 1      Medical 100.000000
## 2       Budget  67.309364
## 3     Synfuels  51.924418
## 4      Testban  44.764657
## 5      Missile  41.164668
## 6     Religion  37.052885
## 7   ElSalvador  34.847989
## 8  Immigration  27.305556
## 9    Superfund  20.265999
## 10         H2O  15.352262
## 11   Education  15.238955
## 12       Crime  12.886359
## 13     Contras   8.782406
## 14     Infants   5.173696
## 15     SAfrica   3.527715
## 16    Dutyfree   0.000000
```

We can also make a dot plot to visualize the rankings:




As we earlier ascertained, the "Medical" item was by far the most important bill, with an importance value of 100. The second and third most significant items identified by the algorithm were "Budget" and "Synfuels," with respective scores of 67.3 and 51.9.

Note that these importance values take the same form as the PART-Scores that we derived earlier.

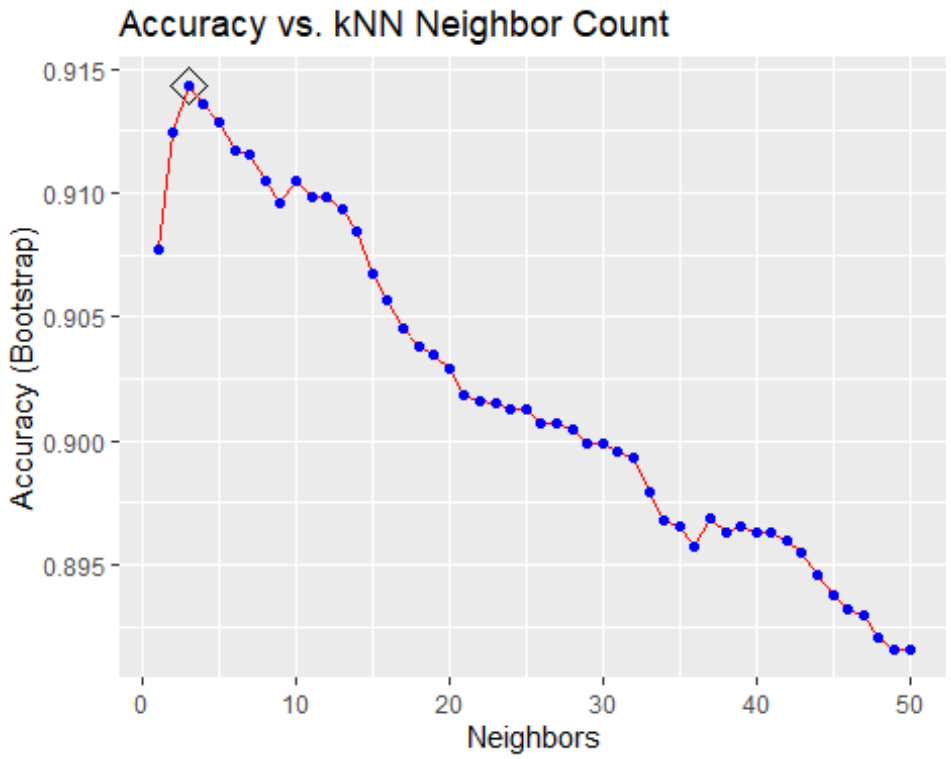
## Model 4: K-Nearest Neighbors

Our next model uses the K-Nearest Neighbors (KNN) algorithm. KNN computes the distance between a point in the testing data and several other points, *k*, originating from the training set. For classification problems, a prediction is made by majority-vote of *k* local predictors.

KNN is considered a reliable algorithm for classification, and we thus expect it to perform well with our relatively simple dataset. Below, you will find the code we use to train the KNN fit over 50 values of *k* from 1-50. The author selected this relatively large number of trials because the dataset is not very large, and *k* was capped at 50 so as to not use too much of the "training" set for every prediction such that the concept of "locality" would retain its purpose.

```
set.seed(1, sample.kind = 'Rounding')
knn_fit <- train(party ~ ., method = 'knn', data = training[,1:17], tuneGrid
= data.frame(k = seq(1,50,1)))
```

Here is a plot of Accuracy vs. the number of neighbors for each KNN trial:




Using the "bestTune" variable, we can see that the algorithm identified $k = 3$ as the optimal parameter. This implies that the data is relatively volatile, as the most accurate predictions are coming out of very small neighborhoods.

```
##   k
## 3 3
```

Next, we feed the "testing" data into the model to generate our predictions, and we save and present results:

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |
| Logistic | 0.9545 |
| KNearest | 0.8636 |

Unfortunately, this model was not as successful as the Logistic Regression, and it was not able to surpass the 90% Accuracy threshold. This weaker outcome could be due to the fact that KNN does not evaluate the predictive power of the various items involved, so it not able to identify key votes to parse the partisan-ideological ambiguity which characterizes the 98th Congress.

## Model 5: Random Forest

We should expect to see better results with the Random Forest algorithm, which, like the Logistic Regression, takes the efficacy of the individual predictors into account.

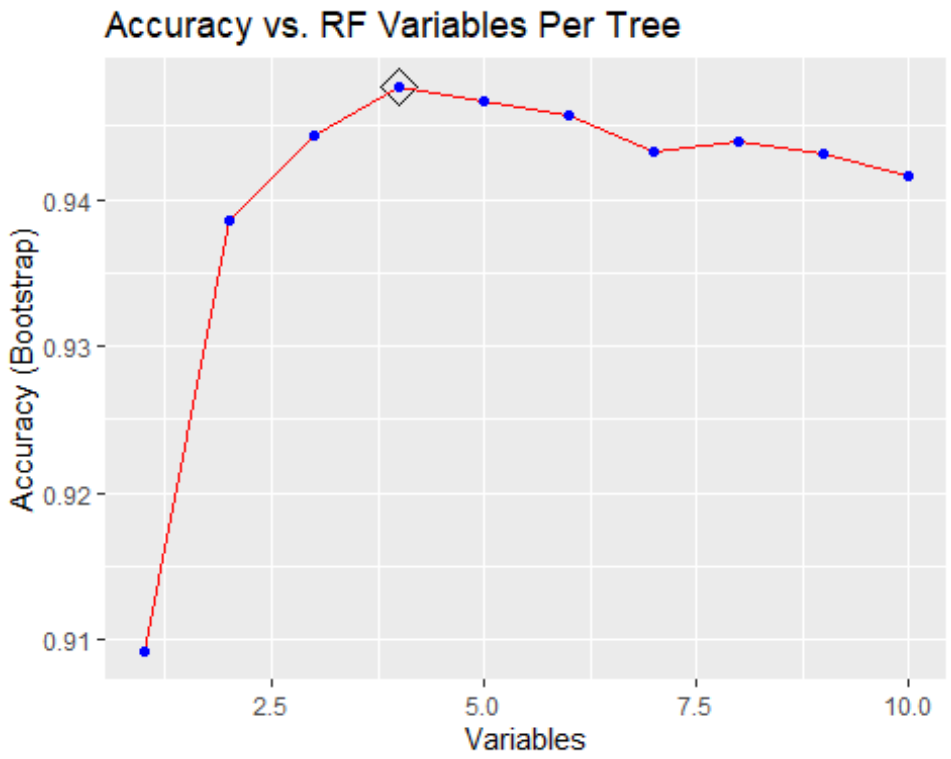
Random Forests compile the results of many classification trees, which assemble hierarchical models of the data based upon key predictors. Random Forests are generally more accurate than any individual tree because they can contain many trees with different numbers of branches and stopping thresholds, which determine whether any given "node" will function as a final result or as a relay to another branch.

Here is the code to fit the Random Forest model using train():

```
set.seed(1, sample.kind = 'Rounding')
rf_fit <- train(party ~ ., method = 'rf', data = training[,1:17], ntree =
100, tuneGrid = data.frame(mtry = seq(1:10)), importance = TRUE)
```

We instruct the algorithm to construct 100 trees in order to balance sampling variability with processing time, and we specify that trials of 1-10 variables per tree should be carried out to determine the optimal number. Given that this dataset only has 16 predictor variables, we would not want a number higher that 10, as the resulting trees would be unnecessarily complex.

Here is a plot of Accuracy vs. the number of variables per tree:




We can see that Accuracy peaks with a relatively few number of variables, which we identify with "bestTune" as 4 items per tree.

```
##   mtry
## 4    4
```

Here is the final Accuracy output of the Random Forest model:

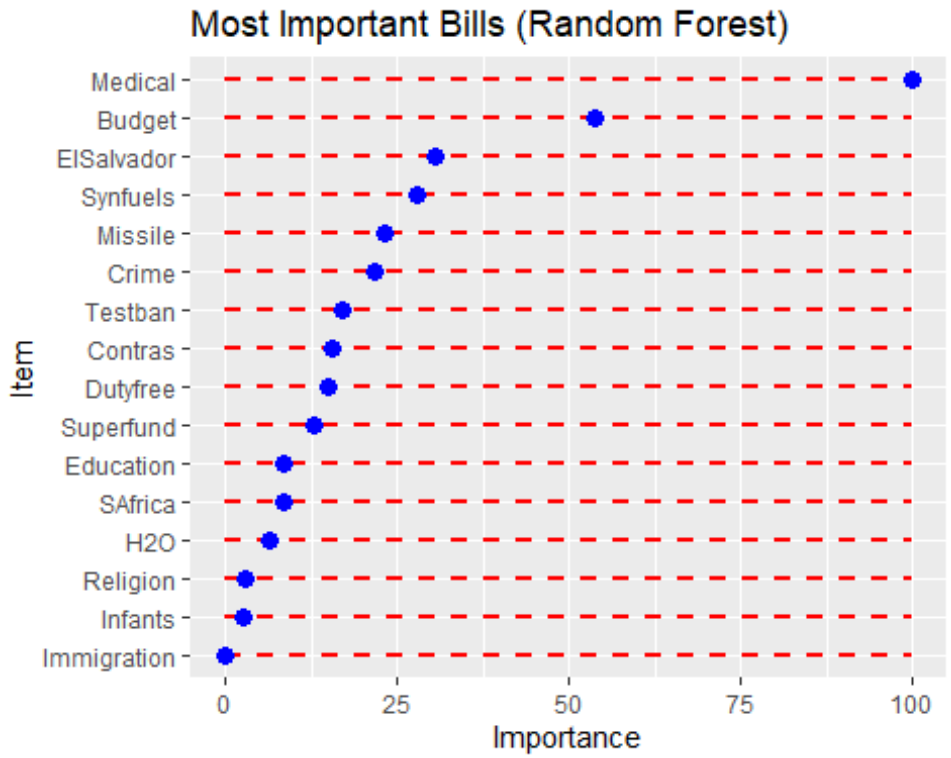
| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |
| Logistic | 0.9545 |
| KNearest | 0.8636 |
| R-Forest | 0.9545 |

Our tuned Random Forest provides an Accuracy of 95.5%, just like the Logistic Regression. This is a very strong figure, and it could be difficult for any other machine learning algorithm to beat.

Additionally, the small number of variables per tree requested by the algorithm is likely due to the extreme partisanship of "Medical" and "Budget"--the two items which we have previously discussed. We can confirm this speculation with the varImp() function.

```
##           Item Importance
## 1      Medical 100.000000
## 2       Budget  53.773055
## 3   ElSalvador  30.660443
## 4     Synfuels  28.031809
## 5      Missile  23.346941
## 6        Crime  21.734484
## 7      Testban  17.005676
## 8      Contras  15.446328
## 9     Dutyfree  14.951750
## 10   Superfund  12.790753
## 11   Education   8.615598
## 12     SAfrica   8.535613
## 13         H2O   6.539970
## 14    Religion   2.944670
## 15     Infants   2.762897
## 16 Immigration   0.000000
```

Additionally, here is a plot of the varImp() results:




Indeed, we can see that "Medical" and "Budget" were the two most important items identified by the algorithm. It's interesting how strongly each algorithm gravitates towards the "Medical" bill as the #1 predictor of party affiliation. The Random Forest considers "Medical" almost twice as important as "Budget," which is itself almost twice as important as "ElSalvador," the third most relevant item according to this algorithm.

Then again, as we discussed earlier all but 2 Republicans voted "Yea" on this item, and the vast majority of Democrats voted "Nay." This item is about as partisan as any vote in the House of Representatives is likely to get.

## Model 6: Neural Network

For the final intermediate model, the author decided to try out a new machine learning algorithm that we have not covered in the HarvardX series. After reading up on several algorithms that are commonly used for classification, the author decided to use the "nnet" Neural-Network algorithm from the "caret" package.

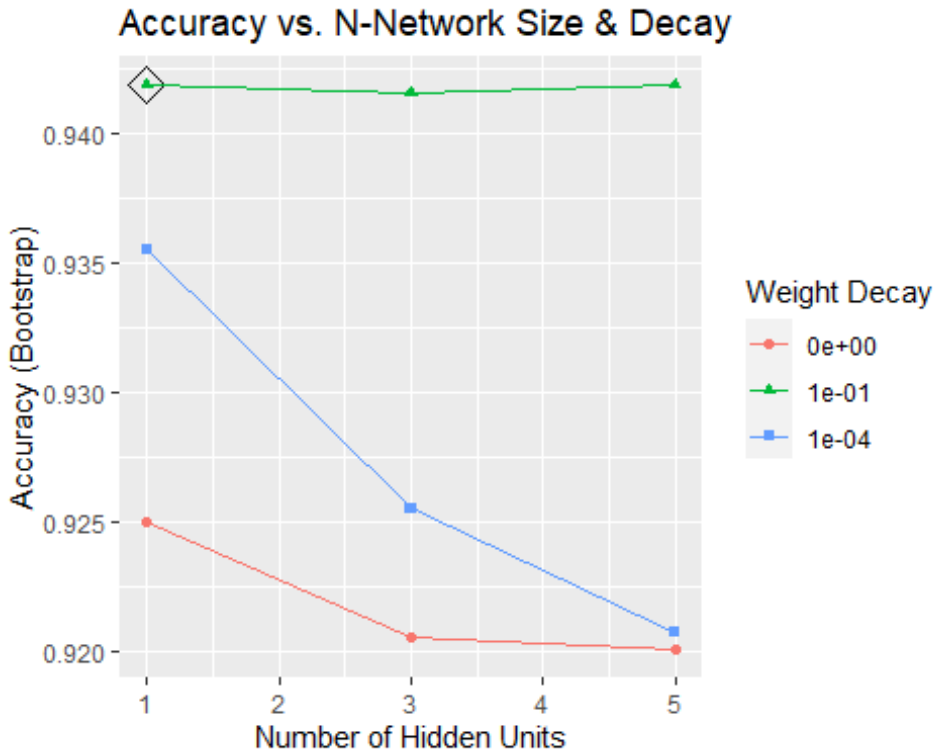
Neural Networks function by studying the nature of the training data through repeated exposure to many examples of classes for which predictions are desired. The "network" created by the algorithm has 3 features: an input layer, an output layer, and a hidden layer, which sorts observations according to the algorithm's understanding of the data. When correctly applied, Neural Networks can be extremely powerful.

Below, you will find the code used to train our Neural Network. Please note that the author decided to allow train() to calculate the optimal parameters & hyperparameters without the use of manual adjustments, as the author is only beginning his study of this concept. By

default, “nnet” utilizes a Bootstrapped resampling method to tune a “size” parameter, which refers to the number of hidden units, and a “decay” parameter, which refers to the weight decay that prevents weights in the model from getting too large.

```
set.seed(1, sample.kind = 'Rounding')
nnet_fit <- train(party ~ ., method = 'nnet', data = training[,1:17], metric
= 'Accuracy')
```

Here is a plot depicting model Accuracy vs. the Neural Network size and rate of weight decay:




We identify the optimal rate of decay and number of hidden units using “bestTune.”

```
##   size decay
## 3    1   0.1
```

We can see that the algorithm has determined that a small network with a weight decay of 0.1 will provide the best result.

Here is our final Accuracy using the tuned model:

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |
| Logistic | 0.9545 |
| KNearest | 0.8636 |

R-Forest 0.9545

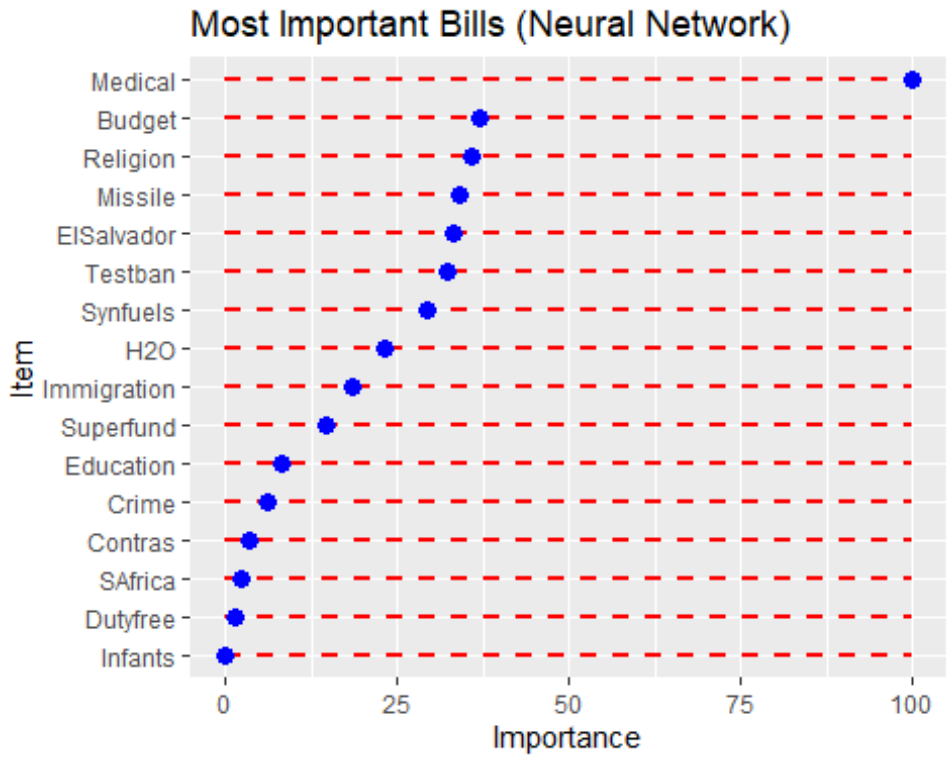
NNetwork 0.9773

We can see that the Neural Network returns an Accuracy of 97.7%. This is an outstanding result, well above the 90% threshold for success.

Again, we can which variables were deemed most important by this algorithm using varImp():

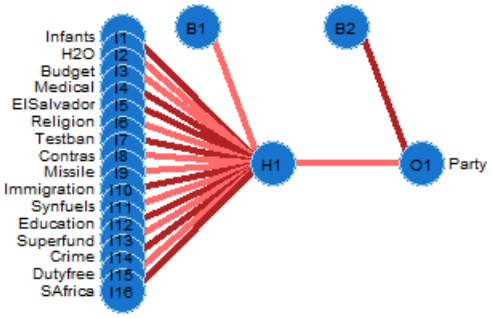
```
##           Item Importance
## 1      Medical 100.000000
## 2       Budget  37.085121
## 3     Religion  35.963691
## 4      Missile  34.047828
## 5   ElSalvador  33.342439
## 6      Testban  32.399560
## 7     Synfuels  29.315909
## 8          H2O  23.114015
## 9  Immigration  18.398867
## 10   Superfund  14.546233
## 11   Education   8.334586
## 12       Crime   6.285977
## 13     Contras   3.549278
## 14     SAfrica   2.269666
## 15    Dutyfree   1.602697
## 16     Infants   0.000000
```

Here is the corresponding dot plot:




Unsurprisingly, we see that the Neural Network identified the "Medical" bill as the most significant. The disparity between "Medical" and "Budget" (which again came in second place) is even more extreme in this model. That is, "Medical" was given an importance score of 100 while "Budget" was given an importance score of only 37. Clearly, the Neural Network took full advantage of the "Medical" item's intensely partisan voting.

For those interested, here is a visual of the shape of the optimized network, with 1 hidden layer and two biases:



To finish the analysis, we will combine the results of our successful models in a basic majority-vote ensemble. This Neural Network Accuracy is going to be very difficult to beat, but we will give it a shot.

### Cumulative Model

## Model 8: Majority-Vote Ensemble

Our final model attempts to merge the various perspectives taken by each model to assemble a single, powerful predictive system.

This Ensemble is developed around the collective insights of the 5 intermediate models: 1) Party-Line Index, 2) Logistic Regression, 3) K-Nearest Neighbors, 4) Random Forest, and 5) Neural Network. For each member assigned to the "testing" set, the Ensemble polls each intermediate model for their prediction, and the majority vote is adopted as the Ensemble prediction.

Hypothetically, the Ensemble should be the best-performing model, as intermediate model mistakes are overruled by the wise majority--thereby increasing overall Accuracy. However, a majority-vote ensemble will not improve upon constituent model accuracy if certain outlier observations are so difficult to classify that a majority of models all make the same mistake.

Before we are able to construct the Ensemble, we must put together a dataframe containing each individual model's list of predictions.

```
combined_predictions <- data.frame(PLI = as.numeric(partyline_model) - 1, GLM
= as.numeric(glm_model) - 1, KNN = as.numeric(knn_model) - 1, RF =
as.numeric(rf_model) - 1, NET = as.numeric(nnet_model) - 1, TEST =
as.numeric(testing$party) - 1)
```

Here are the first few entries of the cumulative dataset:

```
head(combined_predictions, n = 6)
```

```
##   PLI GLM KNN RF NET TEST
## 1   1   1   1  1   1    1
## 2   1   0   0  0   0    0
## 3   1   1   1  1   1    1
## 4   1   1   1  1   1    1
## 5   0   0   0  0   0    0
## 6   1   1   1  1   1    1
```

Next, we run a For-Loop to compute the final Ensemble predictions.

```
for(i in 1:nrow(combined_predictions)){
  prediction <- ifelse(sum(combined_predictions[i,1:5]) > 2, 1, 0)
  ensemble_model[nrow(ensemble_model) + 1,] <- prediction
}
```

Here is the final result:

| Model | Accuracy |
|---|---|
| Guessing | 0.5000 |
| PL-Index | 0.8864 |
| Logistic | 0.9545 |
| KNearest | 0.8636 |
| R-Forest | 0.9545 |
| NNetwork | 0.9773 |
| Ensemble | 0.9773 |

We can see that the Ensemble tied the Neural Network with an accuracy of almost 98%! Again, that is a great outcome.

The author will examine why the Ensemble was unable to achieve 100% Accuracy--and much more--in the upcoming discussion on the results.

## Results

Once again, here are the results of each model for party affiliation. The models are now ranked from most effective to least effective:

| Model | Accuracy |
|---|---|
| Ensemble | 0.9773 |
| NNetwork | 0.9773 |
| R-Forest | 0.9545 |
| Logistic | 0.9545 |
| PL-Index | 0.8864 |
| KNearest | 0.8636 |
| Guessing | 0.5000 |

In total, we see that 4 of our models passed the 90% Accuracy threshold to be considered sufficient final products, while 3 models failed to meet this value.

Appropriately, the Guessing baseline model was least effective, with an Accuracy of only 50%. This outcome makes sense, as binary guessing has a 50%-50% chance of success. The K-Nearest Neighbor algorithm significantly improved on the Guessing model, but it too was unable to pass the success threshold with an Accuracy of 86%. As discussed previously, we suspect that KNN did not do better than it did due to the nature of the algorithm, which emphasizes proximity instead of predictor efficacy.

Next, the Party-Line Index produced a respectful Accuracy of 88%, only 2 points short of the success threshold. Like KNN, the PL-Index was unable to take advantage of the items with a clear partisan consensus (like “Medical” and “Budget”), thus depriving the model of a simple path to high Accuracy.

The Logistic Regression and Random Forest both resulted in an Accuracy of 95.5%. Any Accuracy above 95% is generally considered to be very good for most classification problems, and this instance is no exception. However, these models were beat by the Neural Network, which achieved an exceptional Accuracy of 97.7%. This model only missed 1 prediction, while the Logistic Regression and Random Forest each missed 2 predictions. The Neural Network most likely surpassed them due to a more complex understanding of the data.

Finally, the majority-vote Ensemble tied the Neural Network, achieving the same Accuracy and missing the same 1 prediction. In several rounds of polling, the Neural Network proved to be the tie-breaking vote that ensured that the correct decision was made. Additionally, upon reviewing the Neural Network predictions, we can identify which House member was incorrectly sorted by these final models.

```
tfs <- combined_predictions$NET == combined_predictions$TEST
which(tfs == FALSE)

## [1] 39
```

Looking at the "testing" entry for this member, we can see that they do not easily fit the Congressional party stereotypes. Despite being a Republican, they have a Party-Line Index over 50%; this means that they voted with Democrats on a majority of items.

| party | Infants | H2O | Budget | Medical | ElSalvador |
|---|---|---|---|---|---|
| 0 | yea | yea | yea | nay | nay |
| Religion | Testban | Contras | Missile | Immigration | Synfuels |
| yea | nay | yea | yea | nay | nay |
| Education | Superfund | Crime | Dutyfree | SAfrica | dem_index_test |
| yea | yea | nay | nay | yea | 0.5714 |

In fact, none of the models were able to classify this member correctly.

```
##    PLI GLM KNN RF NET TEST
## 39   1   1   1  1   1    0
```

Therefore, this is an instance of ideological ambiguity overpowering partisan trends for classification. However, given the widespread partisan-ideological mixing that we have observed in the 98th Congress, it is quite impressive that the Neural Network was able to correctly classify every other member in the testing data.

Overall, with 4 successful models, the author is satisfied with the results of the project. If the author were to hypothetically require a single, final model to use for classification upon a new set of voting data, either the Neural Network or the majority-vote Ensemble would be more than satisfactory.

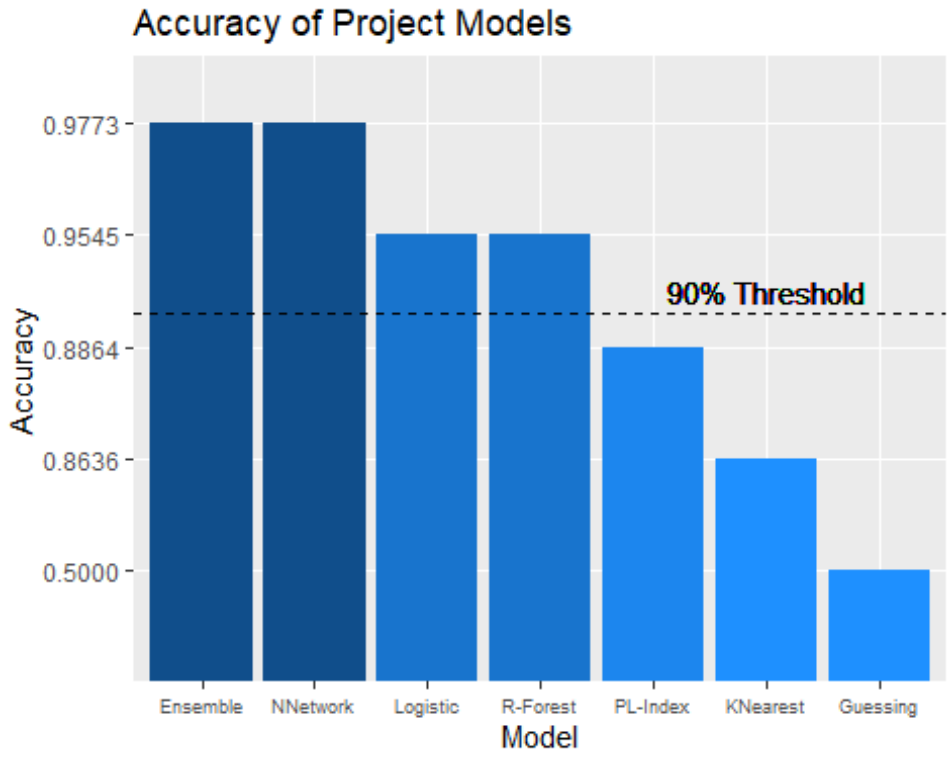
# Conclusion

## Project Overview

This project began with the goal of accurately predicting party affiliations for members of the House of Representatives serving in the 98th Congress. In order to make these predictions, the project involved an extensive exploration of the data. This exploration uncovered significant party-ideological mixing; that is, there were many Democrats in the 98th Congress that tended to vote with the Republicans, and vice versa. Next, the author sought to identify pieces of legislation with a clear partisan consensus for the purpose of party classification in this environment of ambiguous partisan and ideological mixing. The two most significant items along this trajectory were postulated to be the "Medical" bill and the "Budget" bill.

The author proceeded to make predictions for party affiliation by focusing on these bills as well as a partisan index developed by the author. Most of the models were eventually fit with all of these predictors provided by the dataset, and the "Medical" bill consistently remained the most important item for prediction. Using the metric of "accuracy" to measure success, this project exceeded the stated threshold of 90% accuracy by utilizing several machine learning algorithms, including the Logistic Regression, kNN, and Random Forests. The most effective single algorithm came in the form of a Neural Network, which had an accuracy output of 97.7%.

Here is a plot showing the Accuracy metric for each model developed throughout the project:



Overall, 4 models met the success threshold by predicting with an accuracy over 90%, and 3 models failed to meet the threshold. The manually-developed PL-Index model failed to meet the 90% threshold by less than 2 points with an accuracy of 88.6%, and, quite surprisingly, the K-Nearest Neighbors model fared significantly less well than the other traditional machine learning algorithms, achieving an accuracy of only 86.4%. The Logistic Regression & Random Forest each obtained a respectable accuracy of 95.5%.

The author was surprised by how well the Neural Network performed, missing only 1/44 predictions. Additionally, the majority-vote Ensemble tied the Neural Network with an accuracy of 97.7% and only one error. After a brief investigation, the author discovered that this member had a particularly ambiguous identity, affiliating with the Republicans but voting with the Democrats over half of the time.

### Limitations & Future Work:

This project does not reach its full potential due to several limitations, including:

1. The code relies heavily on For-Loops and copy + paste methods, as opposed to custom functions and vectorized / matrix-based calculations. These methods limit the scope of data exploration & analysis, and they are inefficient.
2. The Congressional Voting Records dataset is relatively basic, as floor votes do not always reflect the true disposition of members of Congress. Many political scientists discourage studies of floor votes in isolation of other factors that influence the legislative process, such as committee work and outside deals.
3. There were many NAs present in the original data that were recoded under the assumption that no-votes do not reflect party loyalty and can thus be treated as votes against the party majority. While this assumption is grounded in general political realities, it is likely not true for every single member considered.
4. The variable importance function (varImp()) did not work as intended on the local machine, and the author was forced to program an unwieldy partial-replacement. This new code is likely to be less efficient than the original function.
5. The author is relatively new to programming in R, and he likely did not take advantage of the best possible methods for every operation.

In the future, similar projects could make significant improvements, such as:

1. The Congressional Voting Records dataset should be supplemented with other forms of information to create a more accurate picture of political loyalties in Congress.
2. More machine learning algorithms should be used to maximize accuracy, such as SVM, Naive Bayes, individual Classification Trees, etc.
3. Metrics besides accuracy should be utilized to provide a more comprehensive review of model performance.
4. Future researchers should attempt to focus on vector and matrix-based operations more than For-Loops and copy + pasting, which are generally considered to be less efficient.

The author hopes that this report contributes to the literature on Political Science by showcasing the potential for advanced quantitative analysis of political phenomena using machine learning algorithms, among other techniques of data science.

In sum, this project has been an exciting opportunity to learn more both about Congressional voting habits and programming in R. The author particularly enjoyed finding new packages to utilize in the "data exploration," such as the "ggparliament" package which was used to make the parliament plot. The author has greatly enjoyed the HarvardX Data Science program, and he is excited to apply and expand the programming skills that he has developed in the professional certificate series to graduate studies in the field of public policy. Thank you for reading!

## Code Appendix

Note: This is the bare, functional code with minimal commentary; please view the .R document for a more organized, fully-commented script.

Part 1: Preparing R

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(usmap)) install.packages("usmap", repos = "http://cran.us.r-project.org")
if(!require(politicaldata)) install.packages("politicaldata", repos = "http://cran.us.r-project.org") if(!require(ggparliament)) install.packages("ggparliament", repos = "http://cran.us.r-project.org") if(!require(pixiedust)) install.packages("pixiedust", repos = "http://cran.us.r-project.org") if(!require(sjlabelled)) install.packages("sjlabelled", repos = "http://cran.us.r-project.org") if(!require(RAM)) install.packages("RAM", repos = "http://cran.us.r-project.org") if(!require(ggridges)) install.packages("ggridges", repos = "http://cran.us.r-project.org") if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org") if(!require(ggrepel)) install.packages("ggrepel", repos = "http://cran.us.r-project.org") if(!require(NeuralNetTools)) install.packages("NeuralNetTools", repos = "http://cran.us.r-project.org")

library(tidyverse) library(usmap) library(politicaldata) library(ggparliament) library(pixiedust) library(sjlabelled) library(RAM) library(ggridges) library(caret) library(ggrepel) library(NeuralNetTools)
```

Part 2: Downloading the Data

```
temporary_data <- tempfile() download.file('https://archive.ics.uci.edu/ml/machine-learning-databases/voting-records/house-votes-84.data', temporary_data) data <- read.table(temporary_data, sep = ',', col.names = c('party', 'Infants', 'H2O', 'Budget', 'Medical', 'ElSalvador', 'Religion', 'Testban', 'Contras', 'Missile', 'Immigration', 'Synfuels', 'Education', 'Superfund', 'Crime', 'Dutyfree', 'SAfrica'))
```

Part 3: Data Pre-Processing

```
data <- data.frame(sapply(data, function(x){ ifelse(x == '?', NA, ifelse( x == 'y', 1, ifelse( x == 'n', 0, ifelse( x == 'republican', 'republican', 'democrat' ))))}))
```

data_temp_1 <- data[,1] data_temp_2 <- sapply(data[,2:17], as.numeric) voting <- data.frame(party = data_temp_1, data_temp_2)

voting$party <- str_replace(voting$party, 'republican', 'Republican')voting$party <- −str_replace(voting party, 'democrat', 'Democrat')

Part 4: Data Exploration

data('house_results')

US_House_98 <- house_results %>% filter(year == 1982)

possible_winners <- US_House_98[,5:7] winners_1_3 <- max.col(replace(possible_winners, is.na(possible_winners), 0), ties.method = 'first') dem_win <- ifelse(winners_1_3 == 1, 1, 0)

My_Congress <- data.frame(US_House_98, dem_win)

My_Congress <- My_Congress[-241,] My_Congress <- My_Congress[-241,]

maj_dem_98 <- My_Congress %>% group_by(state_abb) %>% summarize(maj_dem = ifelse(sum(dem_win) > n() / 2, 1, ifelse( sum(dem_win) == n() / 2, 0.5, 0))) %>% .$maj_dem

state_maj_dem <- data.frame(state = unique(My_Congress$state_abb), X = maj_dem_98)

plot_usmap(regions = 'states', data = state_maj_dem, value = 'X') + labs(title = 'United States Congress, 98th Session (1983-1985)', subtitle = 'Party Control of U.S. House Delegations') + theme(legend.position = 'right') + scale_fill_continuous(low = 'red', high = 'blue') + theme(panel.background = element_rect(color = "black", fill = "white"), legend.position = 'none')

voting %>% ggplot(aes(party, fill = party)) + geom_bar() + scale_fill_manual(values = c('blue', 'red')) + geom_hline(yintercept = 218, linetype = 'dashed', size = 1) + geom_text(aes(2, 218, label = 'Legislative Majority', vjust = -1)) + theme(legend.position = 'none') + labs(title = 'U.S. House Party Distribution, 1983-1985', x = 'Party', y = 'Count') + theme(plot.title = element_text(hjust = 0.5))

seat_totals <- data.frame(Democrat = nrow(voting[voting$party == 'Democrat',]), Republican = nrow(voting[voting$party == 'Republican',])) dust(seat_totals)

seat_percents <- data.frame(Democrat = nrow(voting[voting$party == 'Democrat',]) / nrow(voting), Republican = nrow(voting[voting$party == 'Republican',]) / nrow(voting)) dust(seat_percents)

election_summary <- data.frame(year = numeric(), country = character(), house = character(), party_long = character(), party_short = character(), seats = numeric(), government = numeric())

election_summary[nrow(election_summary) + 1,] <- c(1982, 'USA', 'Representatives', 'Republican', 'GOP', nrow(voting[voting$party == 'Republican',]), 0)

```
election_summary[nrow(election_summary) + 1,] <- c(1982, 'USA', 'Representatives', 'Democratic', 'Dem', nrow(voting[voting$party == 'Democrat',]), 1)

election_summary$year <- as.numeric(election_summary$year)election_summary$seats <- -as.numeric(election_summary$seats) election_summary$government <- as.numeric(election_summary$government)

house_parliament_prep <- parliament_data(election_summary, type = 'semicircle', parl_rows = 11, party_seats = election_summary$seats)

ggplot(house_parliament_prep, aes(x = x, y = y, color = party_short)) + geom_parliament_seats() + theme_ggparliament() + scale_color_manual(values = c('blue', 'red'), name = 'Party', labels = c('Democrat', 'Republican')) + geom_segment(aes(x = 0, xend = 0, y = 0.70, yend = 2.15), color = 'black', linetype = 'dashed') + geom_text(aes(0, 0.4, label = 'Legislative Majority', vjust = -1), color = 'black') + labs(color = 'Party', title = 'U.S. House Party Distribution, 1983-1985') + theme(legend.position = 'bottom', legend.direction = "horizontal") + theme(plot.title = element_text(hjust = 0.5))

measure_outcomes <- sapply(voting[,2:17], function(x){ yea <- x[x == 1] nea <- x[x == 0] ifelse(length(yea) > length(nea), 'Pass', 'Fail') })

house_votes <- remove_all_labels(data.frame(Item = colnames(voting[,2:17]), Result = measure_outcomes))

passed_legislation <- count(house_votes, house_votes[,2])[2,2] failed_legislation <- count(house_votes, house_votes[,2])[1,2] legislation_outcomes <- data.frame(pass = passed_legislation, fail = failed_legislation)

dust(legislation_outcomes) %>% sprinkle_colnames(pass = 'Passed Legislation', fail = 'Failed Legislation')

house_votes_arranged <- arrange(house_votes, desc(Result))

dust(house_votes_arranged)

vote_details <- data.frame(item = character(), result = character(), sum_yeas = numeric(), sum_nays = numeric(), no_vote = numeric(), dem_yeas = numeric(), gop_yeas = numeric(), dem_nays = numeric(), gop_nays = numeric(), dem_nv = numeric(), gop_nv = numeric())

for(i in 2:17){ only_dems <- voting %>% filter(party == 'Democrat') dem_yeas <- count(only_dems, only_dems[,i])[2,2] dem_nays <- count(only_dems, only_dems[,i])[1,2] dem_NA <- count(only_dems, only_dems[,i])[3,2] only_gop <- voting %>% filter(party == 'Republican') gop_yeas <- count(only_gop, only_gop[,i])[2,2] gop_nays <- count(only_gop, only_gop[,i])[1,2] gop_NA <- count(only_gop, only_gop[,i])[3,2] vote_details[nrow(vote_details) + 1,] <- c(colnames(voting[i]), house_votes$Result[i - 1], dem_yeas + gop_yeas, dem_nays + gop_nays, dem_NA + gop_NA, dem_yeas, gop_yeas, dem_nays, gop_nays, dem_NA, gop_NA) }

vote_details <- arrange(vote_details, desc(result), desc(sum_yeas))
```

```
dust(vote_details) %>% sprinkle_colnames(item = 'Item', result = 'Result', sum_yeas = 'Yeas', dem_yeas = '(D)Y', gop_yeas = '(R)Y', sum_nays = 'Nays', dem_nays ='(D)N', gop_nays = '(R)N', no_vote = 'NVPR', dem_nv = '(D)?', gop_nv = '(R)?')

vote_long <- gather(voting, item, vote, Infants:SAfrica)

vote_long$vote <- ifelse(vote_long$vote == 1, "Yea", "Nay")

vote_long$party <- as.factor(vote_long$party)

passed_bills <- house_votes$Item[house_votes$Result == 'Pass'] failed_bills <- house_votes$Item[house_votes$Result == 'Fail']

vote_long %>% filter(item %in% passed_bills) %>% drop_na(vote) %>% mutate(item = factor(item, levels = vote_details[1:11,1])) %>% group_by(item) %>% ggplot(aes(reorder(vote, desc(vote)), fill = party)) + geom_bar() + facet_wrap(~item) + scale_fill_manual(values = c('blue', 'red')) + labs(title = 'Successful Bill Voting Tally', y = 'Count', fill = 'Party') + theme(axis.ticks.x = element_blank(), axis.title.x = element_blank(), legend.position = c(0.88, 0.15))

vote_long %>% filter(item %in% failed_bills) %>% drop_na(vote) %>% mutate(item = factor(item, levels = c('Synfuels', 'Medical', 'Infants', 'Education', 'Dutyfree'))) %>% group_by(item) %>% ggplot(aes(reorder(vote, desc(vote)), fill = party)) + geom_bar() + facet_wrap(~item) + scale_fill_manual(values = c('blue', 'red')) + labs(title = 'Failed Bill Voting Tally', y = 'Count', fill = 'Party') + theme(axis.ticks.x = element_blank(), axis.title.x = element_blank(), legend.position = c(0.85, 0.3))

bill_partisanship <- data.frame(item = character(), party_affiliation = character(), result = character())

vote_details[,3:11] <- sapply(vote_details[,3:11], as.numeric)

for(i in 1:16){ dem_majority <- ifelse(vote_details[i,6] > vote_details[i,8], 1, 0) gop_majority <- ifelse(vote_details[i,7] > vote_details[i,9], 1, 0) bill_party_affiliation <- ifelse(dem_majority == gop_majority, 'Bipartisan', ifelse( dem_majority > gop_majority, 'Democratic', ifelse( dem_majority < gop_majority, 'Republican', 'Minority Support'))) bill_partisanship[nrow(bill_partisanship) + 1,] <- c(vote_details[i,1], bill_party_affiliation, vote_details[i,2]) }

bill_partisanship <- arrange(bill_partisanship, party_affiliation)

dust(arrange(bill_partisanship, party_affiliation)) %>% sprinkle_colnames(item = 'Item', party_affiliation = 'Partisanship', result = 'Result')

items <- list(Democratic = bill_partisanship$item[1:9], Republican = c(bill_partisanship$item[10:16], bill_partisanship$item[1:2])) group.venn(items, label = TRUE, fill = c('blue', 'red'), cat.pos = c(0,0), lab.cex = .9, cat.dist = .04)

voting2 <- voting

bill_partisanship_edit <- bill_partisanship
```

```
bill_partisanship_edit <- bill_partisanship_edit[order(match(bill_partisanship_edit$item, colnames(voting2))),]

ipl_fix_dem <- sapply(bill_partisanship_edit[,2], function(x){ ifelse(x == 'Bipartisan', 1, ifelse( x == 'Republican', 1, 0)) })

ipl_fix_gop <- sapply(bill_partisanship_edit[,2], function(x){ ifelse(x == 'Bipartisan', 1, ifelse( x == 'Democratic', 1, 0)) })

for(i in 1:435){ if(voting2[i,1] == 'Republican'){ for(j in 2:17){ voting2[i,j] <- ifelse(is.na(voting2[i,j]), ipl_fix_gop[j - 1], voting2[i,j])}} if(voting2[i,1] == 'Democrat'){ for(j in 2:17){ voting2[i,j] <- ifelse(is.na(voting2[i,j]), ipl_fix_dem[j - 1], voting2[i,j])}} }

dem_index <- vector()

for(i in 1:435){ index <- vector() for(j in c(2, 4:16)){ index <- c(index, ifelse(voting2[i,j] == ipl_fix_gop[j - 1], 1, 0))} total <- sum(index) dem_score <- total / length(index) dem_index <- c(dem_index, dem_score) }

head(dem_index)

summary(dem_index)

voting <- voting %>% mutate(dem_index = round(dem_index, digits = 4))

voting %>% ggplot(aes(dem_index, party, fill = party)) + geom_density_ridges(scale = 5, alpha = .6, size = .8) + scale_fill_manual(values = c('blue', 'red')) + labs(y = 'Party', x = 'Party-Line Index', title = 'Party-Line Index Density Curves', fill = 'Party')

tilt_results <- data.frame(result = character())

for(i in 1:435){ if(voting[i,1] == 'Democrat'){ tilt <- ifelse(voting[i,18] >= 0.9, 'Far-Left', ifelse( voting[i,18] < 0.9 & voting[i,18] >= 0.6, 'New Deal', ifelse( voting[i,18] < 0.6, 'Blue Dog', 'No Affiliation' ))) } if(voting[i,1] == 'Republican'){ tilt <- ifelse(voting[i,18] > 0.4, 'Rockefeller', ifelse( voting[i,18] <= 0.4 & voting[i,18] > 0.1, 'Goldwater', ifelse( voting[i,18] <= 0.1, 'Far-Right', 'No Affiliation' )))} tilt_results[nrow(tilt_results) + 1,] <- tilt }

tilt_analysis <- tilt_results$result tilt_analysis <- factor(tilt_analysis, levels = c('Far-Left', 'New Deal', 'Blue Dog', 'Rockefeller', 'Goldwater', 'Far-Right'))

table(tilt_analysis)

voting <- voting %>% mutate(tilt = tilt_analysis)

voting$tilt <- factor(voting$tilt, levels = c('Blue Dog', 'New Deal', 'Far-Left', 'Rockefeller', 'Goldwater', 'Far-Right'))

voting %>% ggplot(aes(party, fill = tilt)) + geom_bar() + scale_fill_manual(values = c('dodgerblue', 'dodgerblue3', 'dodgerblue4', 'firebrick1', 'firebrick3', 'firebrick')) + geom_hline(yintercept = 218, linetype = 'dashed', size = 1) + geom_text(aes(2, 218, label =
```

‘Legislative Majority’, vjust = -1)) + labs(title = ‘Party Distribution by Ideology’, x = ‘Party’, y = ‘Count’, fill = ‘Ideology’) + theme(plot.title = element_text(hjust = 0.5))

ideology_results <- data.frame(result = character())

for(i in 1:435){ member_ideology <- ifelse(voting[i,19] == ‘Blue Dog’ | voting[i,19] == ‘Goldwater’ | voting[i,19] == ‘Far-Right’, ‘Conservative’, ‘Liberal’) ideology_results[nrow(ideology_results) + 1,] <- member_ideology }

table(ideology_results)

voting <- voting %>% mutate(lcID = ideology_results$result)

voting$tilt <- factor(voting$tilt, levels = c(‘Blue Dog’, ‘Rockefeller’, ‘New Deal’, ‘Far-Left’, ‘Goldwater’, ‘Far-Right’))

voting %>% ggplot(aes(lcID, fill = tilt)) + geom_bar() + scale_fill_manual(values = c(‘dodgerblue’, ‘firebrick1’, ‘dodgerblue3’, ‘dodgerblue4’, ‘firebrick3’, ‘firebrick’)) + geom_hline(yintercept = 218, linetype = ‘dashed’, size = 1) + geom_text(aes(2, 218, label = ‘Legislative Majority’, vjust = -1)) + labs(title = ‘Conservative-Liberal Distribution’, x = ‘General Ideology’, y = ‘Count’, fill = ‘Ideology’) + theme(plot.title = element_text(hjust = 0.5))

part_scores <- data.frame(PART_Score = numeric())

for(i in 2:17){ abs_diff <- 100 * round(abs(mean(voting2,#edit,i][voting2$party == ‘Democrat’]) - mean(voting2,#edit,i][voting2$party == ‘Republican’])), digits = 2) part_scores[nrow(part_scores) + 1,] <- abs_diff }

part_scores <- part_scores %>% mutate(Item = colnames(voting2[2:17])) %>% relocate(Item, PART_Score) %>% arrange(desc(PART_Score))

dust(part_scores) %>% sprinkle_colnames(Item = ‘Item’, PART_Score = ‘PART-Score’)

part_scores %>% mutate(Item = fct_reorder(Item, PART_Score)) %>% ggplot(aes(Item, PART_Score)) + geom_segment(aes(x = Item, xend = Item, y = min(PART_Score), yend = max(PART_Score)), linetype = ‘dashed’, color = ‘red’, size = 1) + geom_point(color = ‘blue’, size = 3) + coord_flip() + labs(title = ‘Party Polarization by Item’, y = ‘PART-Score’)

voting2 <- voting2 %>% mutate(lcID = voting$lcID)

idea_scores <- data.frame(IDEA_Score = numeric())

for(i in 2:17){ abs_diff <- 100 * round(abs(mean(voting2#edit,i][voting2$lcID == ‘Liberal’]) - mean(voting2#edit,i][voting2$lcID == ‘Conservative’])), digits = 2) idea_scores[nrow(idea_scores) + 1,] <- abs_diff }

idea_scores <- idea_scores %>% mutate(Item = colnames(voting2[2:17])) %>% relocate(Item, IDEA_Score) %>% arrange(desc(IDEA_Score))

combined_PART_IDEA <- left_join(part_scores, idea_scores)

```
dust(combined_PART_IDEA) %>% sprinkle_colnames(Item = 'Item', PART_Score = 'PART-Score', IDEA_Score = 'IDEA-Score')

combined_PART_IDEA %>% ggplot(aes(PART_Score, IDEA_Score)) + geom_smooth(color = 'red') + geom_point(color = 'blue') + geom_label_repel(aes(label = Item), position = position_jitter(), size = 3) + scale_y_continuous(breaks = seq(0, 100, by = 10)) + scale_x_continuous(breaks = seq(0, 100, by = 10)) + theme(legend.position = 'None') + labs(title = 'PART-Scores vs. IDEA-Scores', x = 'Partisanship', y = 'Ideological Division')
```

Part 5: Data Analysis

```
voting$party <- ifelse(voting$party == 'Democrat', 1, 0)

voting[,2:17] <- sapply(voting[,2:17], function(x){ ifelse(x == 1, 'yea', 'nay')})

voting$party <- as.factor(voting$party)

set.seed(1, sample.kind="Rounding") partition <- createDataPartition(y = voting$party, times = 1, p = 0.1, list = FALSE) training <- voting[-partition,] testing <- voting[partition,]

partition_validation <- data.frame(set = character(), party = character(), true_prop = numeric(), part_prop = numeric(), difference = numeric())
partition_validation[nrow(partition_validation) + 1,] <- c('Training', 'Democrat', round(seat_percents[1,1], digits = 5), round(nrow(training[training$party == 1,]) / nrow(training), digits = 5), round(seat_percents[1,1] - nrow(training[training$party == 1,]) / nrow(training), digits = 5)) partition_validation[nrow(partition_validation) + 1,] <- c('Training', 'Republican', round(seat_percents[1,2], digits = 5), round(nrow(training[training$party == 0,]) / nrow(training), digits = 5), round(seat_percents[1,2] - nrow(training[training$party == 0,]) / nrow(training), digits = 5)) partition_validation[nrow(partition_validation) + 1,] <- c('Testing', 'Democrat', round(seat_percents[1,1], digits = 5), round(nrow(testing[testing$party == 1,]) / nrow(testing), digits = 5), round(seat_percents[1,1] - nrow(testing[testing$party == 1,]) / nrow(testing), digits = 5)) partition_validation[nrow(partition_validation) + 1,] <- c('Testing', 'Republican', round(seat_percents[1,2], digits = 5), round(nrow(testing[testing$party == 0,]) / nrow(testing), digits = 5), round(seat_percents[1,2] - nrow(testing[testing$party == 0,]) / nrow(testing), digits = 5))

dust(partition_validation) %>% sprinkle_colnames(set = 'Set', party = 'Party', true_prop = 'True %', part_prop = 'New %', difference = 'Difference')

training <- training[,1:17] testing <- testing[,1:17]

training_example <- training

training_example$party <- ifelse(training_example$party == 1, 'Democrat', 'Republican')
training_example[,2:17] <- sapply(training_example[,2:17], function(x){ ifelse(x == 'yea', 1, 0)})

train_details_example <- data.frame(item = character(), result = character(), sum_yeas = numeric(), sum_nays = numeric(), no_vote = numeric(), dem_yeas = numeric(), gop_yeas =
```

```
numeric(), dem_nays = numeric(), gop_nays = numeric(), dem_nv = numeric(), gop_nv =
numeric())

for(i in 2:17){ only_dems <- training_example %>% filter(party == 'Democrat') dem_yeas <-
count(only_dems, only_dems[,i])[2,2] dem_nays <- count(only_dems, only_dems[,i])[1,2]
dem_NA <- count(only_dems, only_dems[,i])[3,2] only_gop <- training_example %>%
filter(party == 'Republican') gop_yeas <- count(only_gop, only_gop[,i])[2,2] gop_nays <-
count(only_gop, only_gop[,i])[1,2] gop_NA <- count(only_gop, only_gop[,i])[3,2]
train_details_example[nrow(train_details_example) + 1,] <-
c(colnames(training_example[i]), house_votes$Result[i - 1], dem_yeas + gop_yeas,
dem_nays + gop_nays, dem_NA + gop_NA, dem_yeas, gop_yeas, dem_nays, gop_nays,
dem_NA, gop_NA) }

train_details_example <- arrange(train_details_example, desc(result), desc(sum_yeas))

bill_partisanship_example <- data.frame(item = character(), party_affiliation = character(),
result = character())

train_details_example[,3:11] <- sapply(train_details_example[,3:11], as.numeric)

for(i in 1:16){ dem_majority <- ifelse(train_details_example[i,6] >
train_details_example[i,8], 1, 0) gop_majority <- ifelse(train_details_example[i,7] >
train_details_example[i,9], 1, 0) bill_party_affiliation <- ifelse(dem_majority ==
gop_majority, 'Bipartisan', ifelse( dem_majority > gop_majority, 'Democratic', ifelse(
dem_majority < gop_majority, 'Republican', 'Minority Support'))) 
bill_partisanship_example[nrow(bill_partisanship_example) + 1,] <-
c(train_details_example[i,1], bill_party_affiliation, train_details_example[i,2]) }

bill_partisanship_example <-
bill_partisanship_example[order(match(bill_partisanship_example$item,
colnames(training))),]

ipl_fix_dem <- sapply(bill_partisanship_example[,2], function(x){ ifelse(x == 'Bipartisan', 1,
ifelse( x == 'Republican', 1, 0)) })

ipl_fix_gop <- sapply(bill_partisanship_example[,2], function(x){ ifelse(x == 'Bipartisan', 1,
ifelse( x == 'Democratic', 1, 0)) })

ipl_fix_gop <- sapply(ipl_fix_gop, function(x){ ifelse(x == 1, 'yea', 'nay') })

ipl_fix_dem <- sapply(ipl_fix_dem, function(x){ ifelse(x == 1, 'yea', 'nay') })

for(i in 1:nrow(training)){ if(training[i,1] == 0){ for(j in 2:17){ training[i,j] <-
ifelse(is.na(training[i,j]), ipl_fix_gop[j - 1], training[i,j])}} if(training[i,1] == 1){ for(j in
2:17){ training[i,j] <- ifelse(is.na(training[i,j]), ipl_fix_dem[j - 1], training[i,j])}} }

for(i in 1:nrow(testing)){ if(testing[i,1] == 0){ for(j in 2:17){ testing[i,j] <-
ifelse(is.na(testing[i,j]), ipl_fix_gop[j - 1], testing[i,j])}} if(testing[i,1] == 1){ for(j in 2:17){
testing[i,j] <- ifelse(is.na(testing[i,j]), ipl_fix_dem[j - 1], testing[i,j])}} }
```

```
dem_index_train <- vector() dem_index_test <- vector()

for(i in 1:nrow(training)){ index <- vector() for(j in c(2, 4:16)){ index <- c(index,
ifelse(training[i,j] == ipl_fix_gop[j - 1], 1, 0))} total <- sum(index) dem_score <- total /
length(index) dem_index_train <- c(dem_index_train, dem_score) }

for(i in 1:nrow(testing)){ index <- vector() for(j in c(2, 4:16)){ index <- c(index,
ifelse(testing[i,j] == ipl_fix_gop[j - 1], 1, 0))} total <- sum(index) dem_score <- total /
length(index) dem_index_test <- c(dem_index_test, dem_score) }

training <- training %>% mutate(dem_index_train = round(dem_index_train, digits = 4))

testing <- testing %>% mutate(dem_index_test = round(dem_index_test, digits = 4))

summary(training$dem_index_train)summary(testing$dem_index_test)

model_accuracy <- data.frame(Model = character(), Accuracy = numeric())

set.seed(1, sample.kind = 'Rounding') guess_model <- as.factor(sample(c(0,1),
nrow(testing), replace = TRUE))

model_1_acc <- str_pad(confusionMatrix(guess_model, testing$party)overall[['Accuracy']],
width = 6, side = 'right', pad = '0') model_accuracy[nrow(model_accuracy) + 1,] <- c(Model
= 'Guessing', Accuracy = model_1_acc) dust(model_accuracy)

partyline_model <- as.factor(ifelse(testing$dem_index_test > 0.50, 1, 0))

confusionMatrix(partyline_model, testing$party)$overall[['Accuracy']]

best_split <- data.frame(Split = numeric(), Accuracy = numeric())

for(i in seq(0, 1, 0.05)){ partyline_model <- ifelse(testing$dem_index_test >
i, 1,0)accuracy <
-confusionMatrix(as.factor(partyline_model), testing$party)$overall[['Accuracy']]
best_split[nrow(best_split) + 1,] <- c(Split = i, Accuracy = accuracy) }

best_split %>% ggplot(aes(Split, Accuracy)) + geom_line(color = 'red') + geom_point(aes(y
= max(Accuracy), x = Split[which.max(Accuracy)]), color = 'black', shape = 5, size = 4) +
geom_point(color = 'blue') + labs(title = 'PL-Index Accuracy vs. Splits')

best_split[which.max(best_split$Accuracy),]

partyline_model <- as.factor(ifelse(testing$dem_index_test > 0.15,1,0))model_2_acc <
-confusionMatrix(partyline_model, testing$party)$overall[['Accuracy']]
model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'PL-Index', Accuracy =
round(model_2_acc, digits = 4)) dust(model_accuracy)

set.seed(1, sample.kind = 'Rounding') glm_fit <- train(party ~ ElSalvador, method = 'glm',
data = training[,1:17]) glm_model <- predict(glm_fit, newdata = testing)

confusionMatrix(glm_model, testing$party)$overall[['Accuracy']]
```

```
set.seed(1, sample.kind = 'Rounding') glm_fit <- train(party ~ Budget, method = 'glm', data = training[,1:17]) glm_model <- predict(glm_fit, newdata = testing)

confusionMatrix(glm_model, testing$party)$overall[['Accuracy']]

set.seed(1, sample.kind = 'Rounding') glm_fit <- train(party ~ Medical, method = 'glm', data = training[,1:17]) glm_model <- predict(glm_fit, newdata = testing)

confusionMatrix(glm_model, testing$party)$overall[['Accuracy']]

set.seed(1, sample.kind = 'Rounding') glm_fit <- train(party ~ Medical + Budget + ElSalvador, method = 'glm', data = training[,1:17]) glm_model <- predict(glm_fit, newdata = testing)

confusionMatrix(glm_model, testing$party)$overall[['Accuracy']]

set.seed(1, sample.kind = 'Rounding') glm_fit <- train(party ~ ., method = 'glm', data = training[,1:17]) glm_model <- predict(glm_fit, newdata = testing)

model_3_acc <- confusionMatrix(glm_model, testing$party)$overall[['Accuracy']] model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'Logistic', Accuracy = round(model_3_acc, digits = 4)) dust(model_accuracy)

impVar_glm <- function(x){ vars <- varImp(glm_fit) vars_df <- data.frame(vars$importance[1])  names_yea <- rownames(vars_df)  names_yea <- str_remove(names_yea, "yea$") vars_df <- vars_df %>% mutate(Item = names_yea, Importance = Overall) %>% select(Item, Importance) arrange(vars_df, desc(Importance)) }

impVar_glm(glm_fit)

impVar_glm_plot <- impVar_glm(glm_fit)

impVar_glm_plot %>% mutate(Item = fct_reorder(Item, Importance)) %>% ggplot(aes(Item, Importance)) + geom_segment(aes(x = Item, xend = Item, y = min(Importance), yend = max(Importance)), linetype = 'dashed', color = 'red', size = 1) + geom_point(color = 'blue', size = 3) + coord_flip() + labs(title = 'Most Important Bills (Logit)')

set.seed(1, sample.kind = 'Rounding') knn_fit <- train(party ~ ., method = 'knn', data = training[,1:17], tuneGrid = data.frame(k = seq(1,50,1)))

ggplot(knn_fit, highlight = TRUE) + labs(title = 'Accuracy vs. kNN Neighbor Count', x = 'Neighbors') + geom_line(color = 'red') + geom_point(color = 'blue')

knn_fit$bestTune

knn_model <- predict(knn_fit, newdata = testing)

model_4_acc <- confusionMatrix(knn_model, testing$party)overall[['Accuracy']] model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'KNearest', Accuracy = round(model_4_acc, digits = 4)) dust(model_accuracy)
```

```
set.seed(1, sample.kind = 'Rounding') rf_fit <- train(party ~ ., method = 'rf', data = training[,1:17], ntree = 100, tuneGrid = data.frame(mtry = seq(1:10)), importance = TRUE)

ggplot(rf_fit, highlight = TRUE) + labs(title = 'Accuracy vs. RF Variables Per Tree', x = 'Variables') + geom_line(color = 'red') + geom_point(color = 'blue')

rf_fit$bestTune

rf_model <- predict(rf_fit, newdata = testing)

model_5_acc <- confusionMatrix(rf_model, testing$party)$overall[['Accuracy']] model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'R-Forest', Accuracy = round(model_5_acc, digits = 4)) dust(model_accuracy)

impVar_rf <- function(x){ vars <- varImp(x) vars_df <- data.frame(vars$importance[1]) names_yea <- rownames(vars_df)  names_yea <- str_remove(names_yea, "yea$") vars_df <- vars_df %>% mutate(Item = names_yea, Importance = X0) %>% select(Item, Importance) arrange(vars_df, desc(Importance)) }

impVar_rf(rf_fit)

impVar_rf_plot <- impVar_rf(rf_fit)

impVar_rf_plot %>% mutate(Item = fct_reorder(Item, Importance)) %>% ggplot(aes(Item, Importance)) + geom_segment(aes(x = Item, xend = Item, y = min(Importance), yend = max(Importance)), linetype = 'dashed', color = 'red', size = 1) + geom_point(color = 'blue', size = 3) + coord_flip() + labs(title = 'Most Important Bills (Random Forest)')

set.seed(1, sample.kind = 'Rounding') nnet_fit <- train(party ~ ., method = 'nnet', data = training[,1:17], metric = 'Accuracy')

ggplot(nnet_fit, highlight = TRUE) + labs(title = 'Accuracy vs. N-Network Size & Decay', x = 'Number of Hidden Units')

nnet_fit$bestTune

nnet_model <- predict(nnet_fit, newdata = testing)

model_6_acc <- confusionMatrix(nnet_model, testing$party)$overall[['Accuracy']] model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'NNetwork', Accuracy = round(model_6_acc, digits = 4)) dust(model_accuracy)

impVar_nnet <- function(x){ vars <- varImp(x) vars_df <- data.frame(vars$importance[1]) names_yea <- rownames(vars_df)  names_yea <- str_remove(names_yea, "yea$") vars_df <- vars_df %>% mutate(Item = names_yea, Importance = Overall) %>% select(Item, Importance) arrange(vars_df, desc(Importance)) }

impVar_nnet(nnet_fit)

impVar_nnet_plot <- impVar_nnet(nnet_fit)
```

impVar_nnet_plot %>% mutate(Item = fct_reorder(Item, Importance)) %>% ggplot(aes(Item, Importance)) + geom_segment(aes(x = Item, xend = Item, y = min(Importance), yend = max(Importance)), linetype = 'dashed', color = 'red', size = 1) + geom_point(color = 'blue', size = 3) + coord_flip() + labs(title = 'Most Important Bills (Neural Network)')

plotnet(nnet_fit, nid = TRUE, x_names = c(colnames(training[2:17])), y_names = 'Party', pad_x = 0.7, circle_col = 'dodgerblue3', pos_col = 'firebrick', neg_col = 'indianred1', circle_cex = 5, rel_rsc = 4, cex_val = 0.8)

combined_predictions <- data.frame(PLI = as.numeric(partyline_model) - 1, GLM = as.numeric(glm_model) - 1, KNN = as.numeric(knn_model) - 1, RF = as.numeric(rf_model) - 1, NET = as.numeric(nnet_model) - 1, TEST = as.numeric(testing$party) - 1)

ensemble_model <- data.frame(maj_vote = numeric())

for(i in 1:nrow(combined_predictions)){ prediction <- ifelse(sum(combined_predictions[i,1:5]) > 2, 1, 0) ensemble_model[nrow(ensemble_model) + 1,] <- prediction }

model_7_acc <- confusionMatrix(as.factor(ensemble_model$maj_vote$), testing$party)$overall[['Accuracy']] model_accuracy[nrow(model_accuracy) + 1,] <- c(Model = 'Ensemble', Accuracy = round(model_7_acc, digits = 4)) dust(model_accuracy)

tfs <- combined_predictions$NET == combined_predictions$TEST which(tfs == FALSE)

testing[39,]

combined_predictions[39,]

Part 6: Conclusion

final_table <- model_accuracy[order(model_accuracy$Accuracy),] final_models <- final_table$Model final_accuracy < -final_table$Accuracy dust(data.frame(Model = rev(final_models), Accuracy = rev(final_accuracy)))

model_accuracy %>% mutate(Model = factor(Model, levels = c('Ensemble', 'NNetwork', 'Logistic', 'R-Forest', 'PL-Index', 'KNearest', 'Guessing'))) %>% ggplot(aes(Model, Accuracy, fill = Accuracy)) + geom_col() + scale_fill_manual(values = c('dodgerblue', 'dodgerblue1', 'dodgerblue2', 'dodgerblue3', 'dodgerblue4')) + labs(title = 'Accuracy of Project Models') + geom_hline(yintercept = 3.3, linetype = 'dashed') + geom_text(aes(6, 3.5, label = '90% Threshold')) + theme(legend.position = 'none', axis.text.x = element_text(size=7))