Predicting Passage of US Constitutional Amendments
Machine Learning Engineer Nanodegree Capstone

Yang Dai
May 17, 2017

I: DEFINITION

Project Overview:

Ratified in 1789, the United States Constitution is the foundational document for the structure of government and law in the United States of America. As the authors foresaw the necessity of updating the Constitution to meet the nation's changing needs, they vested Congress and the states with joint power to append Constitutional amendments. In the intervening 228 years, over 11,000 proposed amendments[1] have been introduced by members of Congress, but only thirty-three have passed Congressional muster, and of those, only twenty-seven have been successfully ratified by the states and made into law. Amendments which have passed range from the enormously impactful, such as abolishing slavery (13th) and granting women the right to vote (19th), to the somewhat mundane, such as delaying Congressional salary raises from taking effect until the following election (27th). The last successful Constitutional amendment ratified was in 1992; with nearly 1000 failed proposed amendments from 1993-2014 alone, a study into which variables are predictors of amendment success is warranted.

Problem Statement:

After converting the proposal titles to numerical features using natural language processing and topic extraction, can a supervised learning classifier (Random Forest or Naïve Bayes) be trained to predict whether a proposed amendment passed Congress? From the resulting model, which features will show the highest significance for predicting amendment passage, and are any of the most heavily weighted features useful for predicting future amendment passage?

Evaluation Metrics:

Accuracy and f-beta scores will be used to evaluate the performance of the supervised classifiers.

The accuracy score is calculated by the following equation[5]:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_\text{samples}} \sum_{i=0}^{n_\text{samples}-1} 1(\hat{y}_i = y_i)$$

This calculates the proportion of predicted outcomes which match the corresponding true outcomes and serves as an intuitive raw score. However, with the large percentage of true negatives in the data, even a naïve predictor which defaults to predicting 100% proposal failure will still score an accuracy of 77.6%, so a more robust scorer is also needed.

The F-beta score is a suitable scorer for uneven class sizes because it is weighted average of the precision and recall scores[5]:

$$F_\beta = (1+\beta^2)\frac{\text{precision} \times \text{recall}}{\beta^2\text{precision} + \text{recall}},$$

where precision = True Positives / (True Positives + False Positives) and recall = True Positives / (True Positives + False Negatives). Because the aim of this study is to investigate factors associated with passage (positives), false positives (predicting a failed amendment had passed) are more detrimental to the analysis than false negatives (predicting a passed amendment had failed), and precision should be weighed more heavily in the F-beta score than recall. Thus, the scorer will be defined with β=0.5, which weighs the precision score more heavily.

II. ANALYSIS

Data Exploration:

The 'us-nara-amending-america-dataset-raw-2016-02-25.csv' dataset contains the 11,797 proposed Constitutional amendments introduced from 1787 to 2014. For each entry, there are 18 features – those features associated with source identification were removed as they are not relevant to proposal performance. The remaining fields relevant to this project are the title/description of the amendment, date of introduction, Congress, sponsor name, sponsor state, and committee of referral.

Manual searches were performed against the proposal descriptions to populate labels for the added 'amendment' column as follows: NaN for non-passing proposals, amendment number for passed and ratified proposals, or "unrat" for passed proposals which did not achieve state ratification. For example, 79 proposals were introduced to lower the voting age to 18, which eventually passed as the 26th amendment, so all 79 proposals were labeled '26'. The total number of proposals matched to passing amendments was 2642, or 22.4% of the entire dataset.

Example entries:

	amendment	title	year	month	day	congress	joint_resolution_chamber	sponsor_name	sponsor_state	committee_of_referral
8000	Unrat	Representation of the District of Columbia in ...	1971.0	1.0	22.0	92.0	House Joint Resolution	Buchanan	Alabama	NaN
8001	NaN	Item veto	1971.0	1.0	22.0	92.0	House Joint Resolution	Byrnes	Wisconsin	NaN
8002	NaN	Congress to function in time of emergency	1971.0	1.0	22.0	92.0	House Joint Resolution	Byrnes	Wisconsin	NaN

For entries with sponsors identified, legislator data was matched from congressional records[3] to populate district, start/total term years, state, type (Senator or Representative), and party of the legislator. 95% of the sponsor names were successfully paired to the corresponding legislator data.

Example congressional records:

	district	end	party	start	state	type	term
Thomas Garrett	5.0	2019	Republican	2017	VA	rep	2
Pramila Jayapal	7.0	2019	Democrat	2017	WA	rep	2
Mike Gallagher	8.0	2019	Republican	2017	WI	rep	2
Liz Cheney	0.0	2019	Republican	2017	WY	rep	2
Luther Strange	NaN	2021	Republican	2017	AL	sen	4

Next, party division data was pulled for the Senate and House[4]. Five features were added to the dataset using the division data – party with majority control in the Senate, percentage of majority party in the Senate, party with majority control in the House, percentage of majority party in the House, and whether the proposal legislator was in the majority for their respective chamber.

Example party division data:

	sen_maj_pct	sen_control	rep_maj_pct	rep_control
congress				
1	69.2	Pro-Administration	56.9	Pro-Administration
2	53.3	Pro-Administration	56.5	Pro-Administration
3	53.3	Pro-Administration	51.4	Anti-Administration
4	65.6	Federalist	55.7	Democratic Republican
5	68.8	Federalist	53.8	Federalist

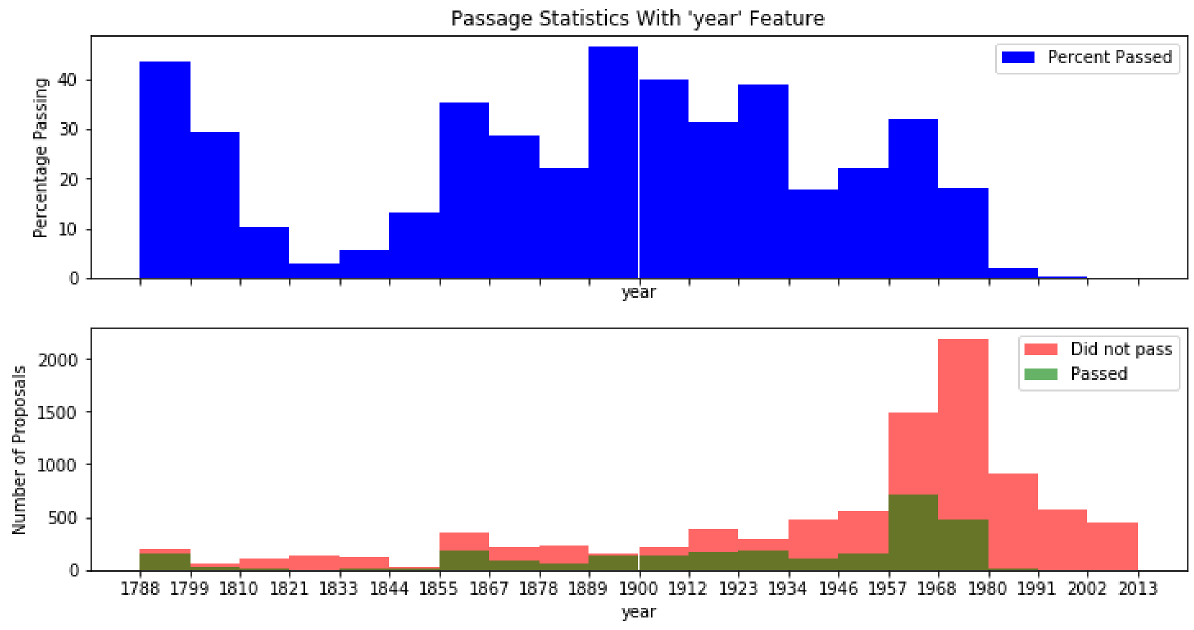
Reviewing the data dataframe summary, some obvious nonsensical values were noted – a 'year' max value of 19931 (in the far future!), 'legis_term' max value of 82 (longest serving legislator had a 59 year term), and negative 'legis_district' min value. These values which were obviously erroneous were set to NaN.

Finally, binomial output data was generated by assigning '0's (non-passing) to all entries with 'amendment'=NaN and '1's (passing) to all other entries.

Exploratory Visualization:

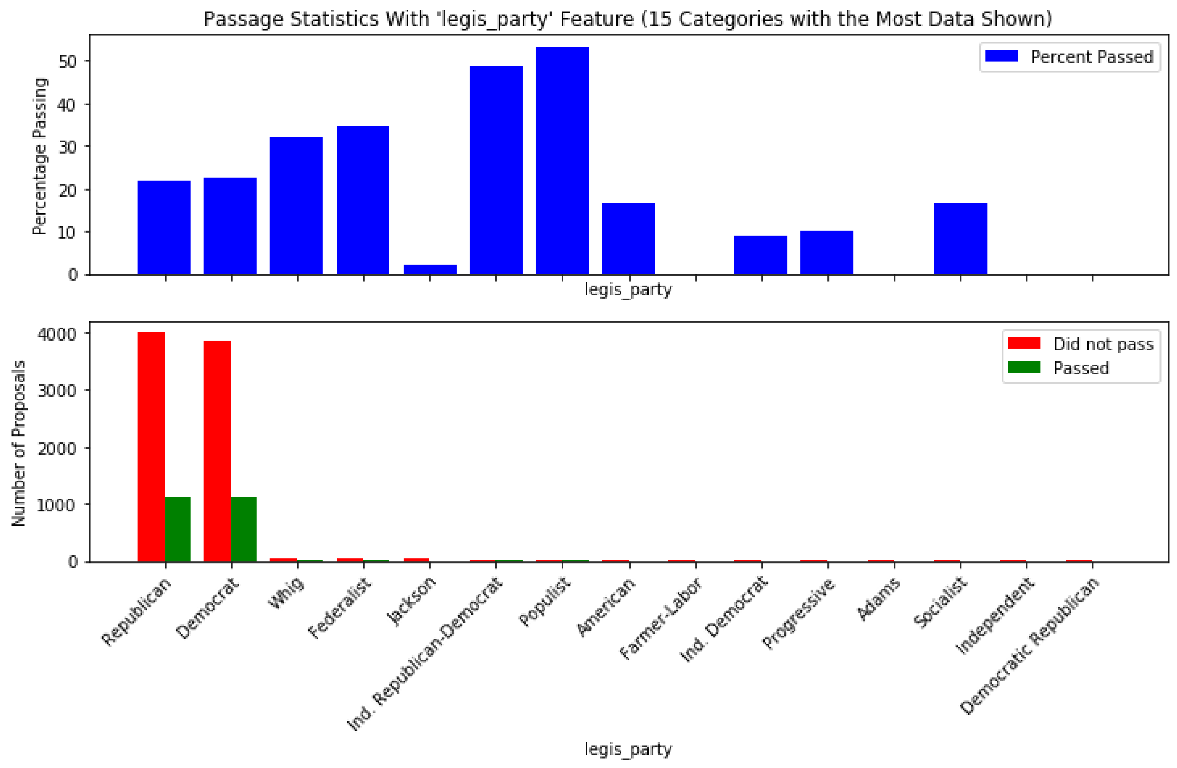
Passage statistics were plotted for each feature by percentage passed and total proposals introduced – the most interesting are shown below.

Figure 1: Passage Statistics by Year



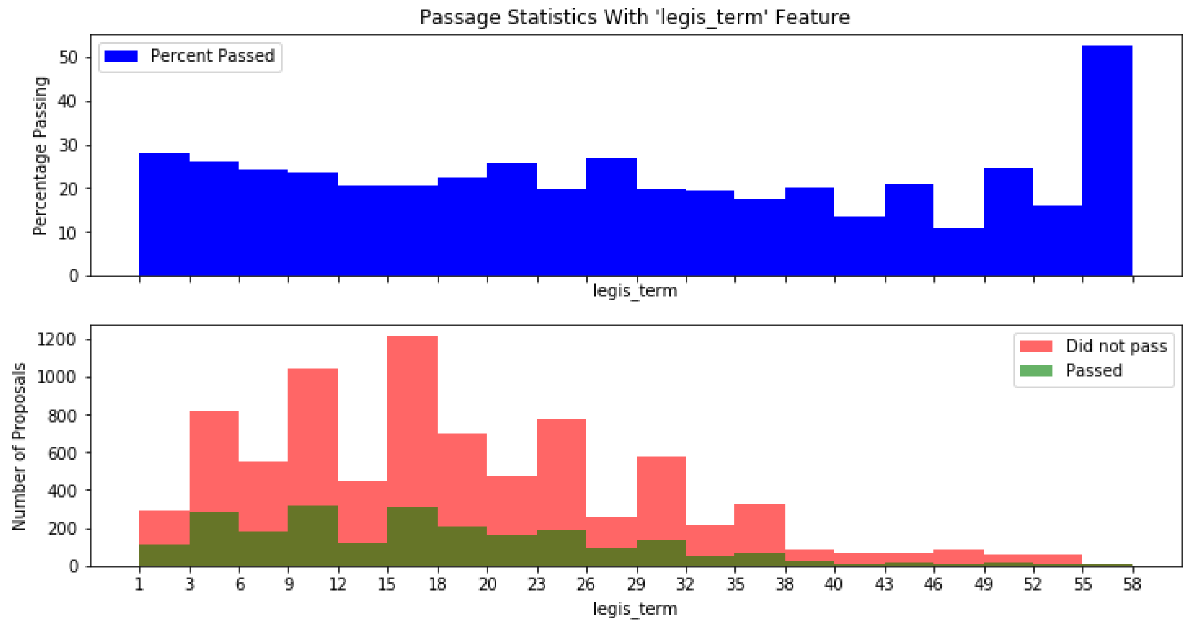
The left-tailed number of proposals plot shows that there were a restrained number of amendment proposal introductions until a massive spike in the mid-20th century, after which proposals gradually declined almost back to the pre-spike levels. While the passage rates don't show any consistent trend, it does decline gradually after 1900 with a total drop off at 1980. This could be reflective of the increasing polarization of parties in the 20th and 21st centuries.

Figure 2: Passage Statistics of the 10 Political Parties with Most Proposals Introduced



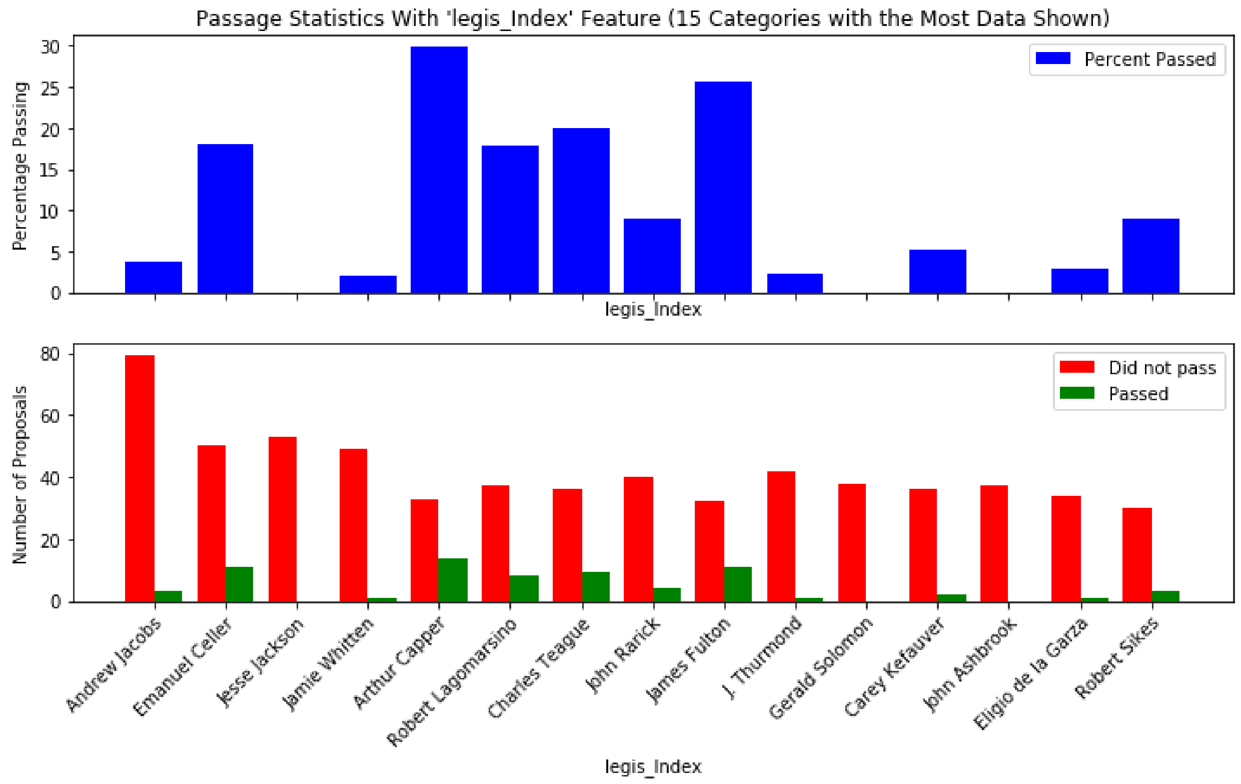
It's immediately evident that the two parties responsible for the vast majority of amendment proposals are the Republican and Democratic parties. This is as expected since these two parties have dominated American politics since the Civil War era. The Whig, Federalist, Ind. Republican-Democrat, and Populist legislators all had higher average passage rates than either Republican or Democrat legislators; however, all those aforementioned parties were very shortly lived and thus generated a significantly smaller sample pool of data not relevant to future predictions.

Figure 3: Passage Statistics of Legislator Term Duration



This plot shows passage rates plotted against the total length of term of the introducing legislator. Presumably those legislators with the longest terms would be most effective at political maneuvering since they were successfully re-elected for so many terms. The plots above however, don't show any obvious advantage to increasing term durations, except at the very end of the chart where the 55-58 year term bin shows a passage rate double that of the remaining bins. Because there are so few proposals in that bin though, it should probably not be used to predict future success.

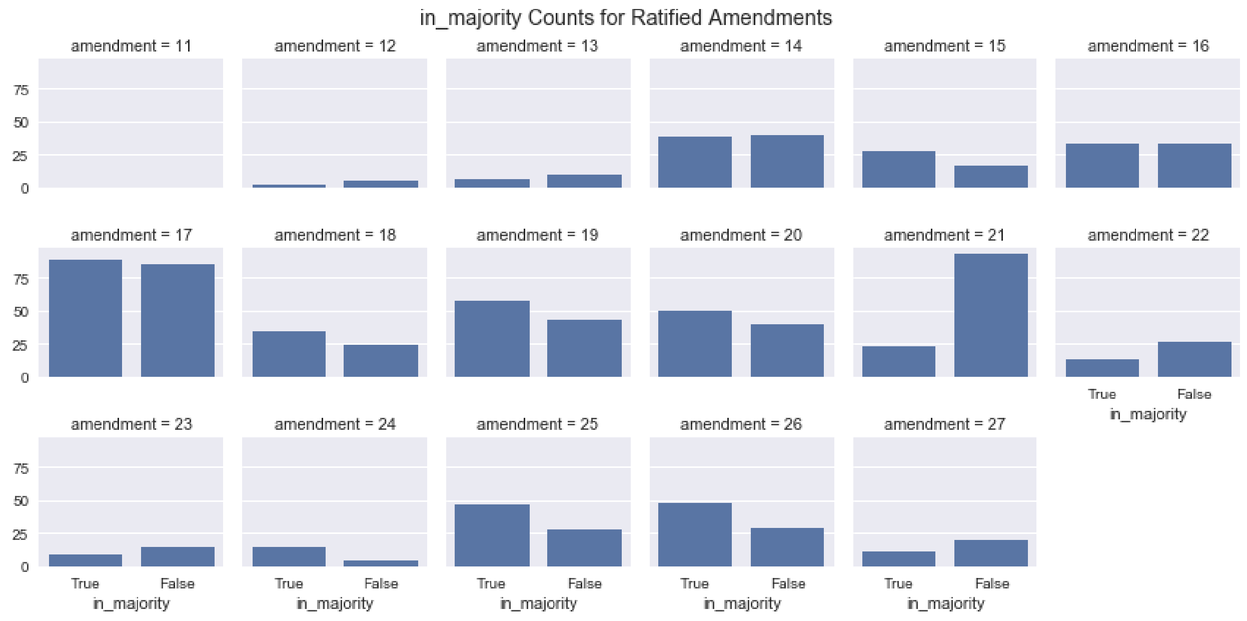
Figure 4: Passage Statistics for the Top 10 Most Prolific Legislators



These plots show the ten legislators who introduced the most number of amendment proposals. Most did not have great success despite their repeated attempts, and only two, Arthur Capper and James Fulton, had passage rates better than the 22.4% average across the entire dataset. This is possibly an indication than the majority of proposals are introduced for grandstanding without actual intent to push them through, or that the legislative gridlock is overwhelmingly biased against change.

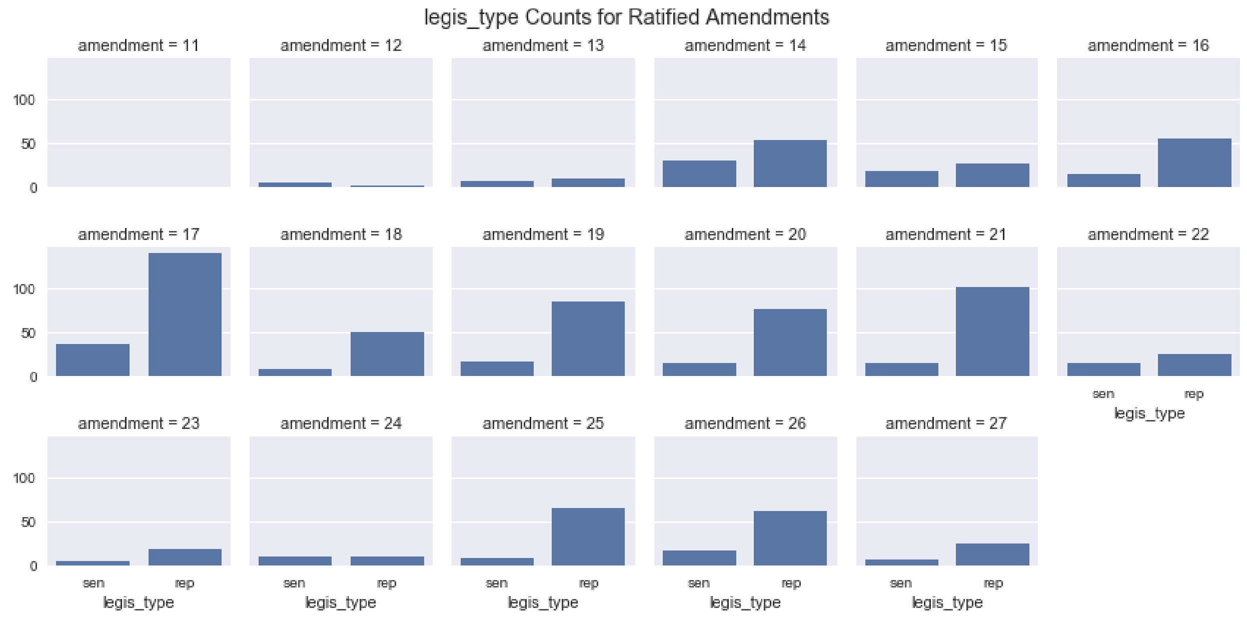
To further investigate success characteristics, the following plots are based on proposals associated with each ratified amendment excluding Amendments 1-10 of the Bill of Rights, which were negotiated during the writing of the original Constitution and thus are circumstantial outliers.

Figure 5: Counts of the 'in_majority' Feature for Each Ratified Amendment



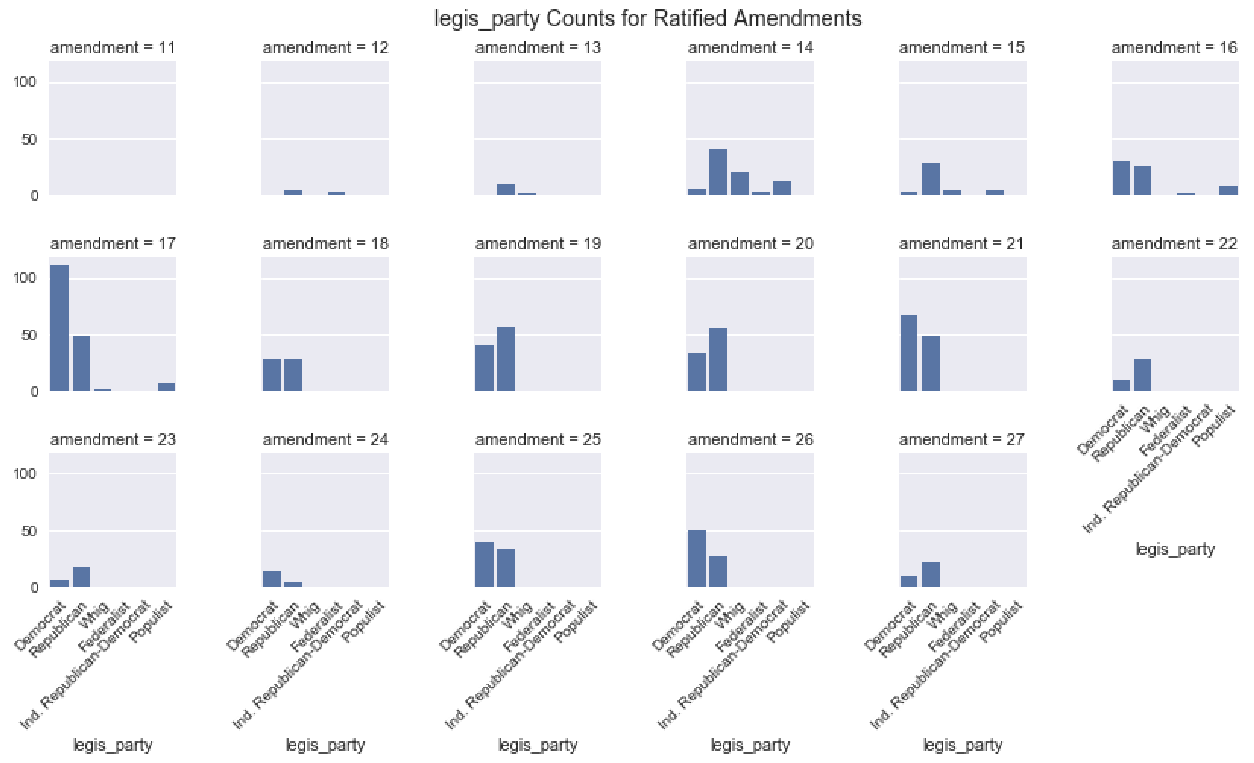
The above histograms show that for ratified amendments, successful proposals are introduced both by the in-majority party and the non-majority party, reflecting a need for broad support across the aisle, rather than a domination by any single party.

Figure 6: Counts of the 'legis_type' Feature for Each Ratified Amendment



With 435 Representatives in the House, and 100 Senators in the Senate, a 4.35:1 ratio of proposal introductions in each chamber would be expected. In the plots above, successful amendments tend to have a higher number of Senate proposals vs House proposals than the naïve ratio would predict.

Figure 7: Counts of the 'legis_party' Feature for Each Ratified Amendment



As with Figure 5, Figure 7 demonstrates that successful amendments require bi-partisan support, as none of the party count histograms for ratified amendments show only a single party putting forth proposals. Although some amendments show higher proportions of proposal introductions from one particular party, all of the histograms with data reflect support from multiple political parties.

<u>Algorithms</u>:

KMeans will be used for unsupervised clustering to observe any trends among the data not obvious from the data exploration plots. KMeans assigns samples to clusters by choosing the smallest average sum of squares to the cluster centroids.

Random Forest and Naive Bayes classifiers will be trained on the dataset to predict passage, and the better performing model was further tuned to improve the final accuracy and F-beta scores. Random Forest was chosen because the ensemble model is very good at preventing overfitting, so the model should be robust enough to avoid becoming overly sensitive to outliers which would be poor predictors of future performance. Naïve Bayes was chosen because it is routinely used in text classification and handles large sets of features well; however, it does not consider correlations between interacting features, which may be a performance degrader.

The Ensemble Random Forest model learns by pulling random sample subsets from the training data and constructing a unique decision tree for each sample subset, aiming to predict the binary outcome (passing or non-passing) correctly for all entries in that subset. The trees are drawn by determining the best feature values (from randomly selected subsets of features) which will split the data meaningfully. For example, if 70% of proposals prior to 1800 passed and 10% after 1800 passed, this would be a good feature node for branching, but if the passage rates before and after were the same, the tree would not branch at this node since it adds no decision value to the outcome. For the final output, the decisions from all the individual subset trees are averaged to leverage the information stored in each tree.

The Naïve Bayes model learns by calculating conditional probabilities of each feature given the output condition. It assumes independence between features, combining all conditional probabilities from the features, then multiplying the sum by the overall probability of the outcome condition to determine the probability of the outcome given the values of the features. The maximum probability outcome is the output of the model. Because Naïve Bayes considers each feature independently of the others, it scales well with large numbers of features, is less computationally expensive, and can achieve high accuracy with a smaller dataset than other learners.

Benchmark:

A congressional bill success study[2] serves as the benchmark model for this project. The benchmark paper attempted to predict 1) whether a bill would get out of committee, and 2) whether an out of committee bill would become enacted. The most analogous to predicting amendment passage (rather than ratification) is the first step of whether a congressional bill makes it out of committee. The benchmark paper investigated logistic regression, SVM, and Naïve Bayes models, ultimately achieving an improvement of 4% in accuracy over baseline and an F-beta score (beta = 0.5) of 88% using Naïve Bayes.

<u>III: METHODOLOGY</u>

<u>Text Processing Techniques</u>:

To convert proposal titles to classifier usable numerical data features, a series of text processing methods from the Natural Language Toolkit were applied to the data. Steps taken are listed below with corresponding examples of a title in process.

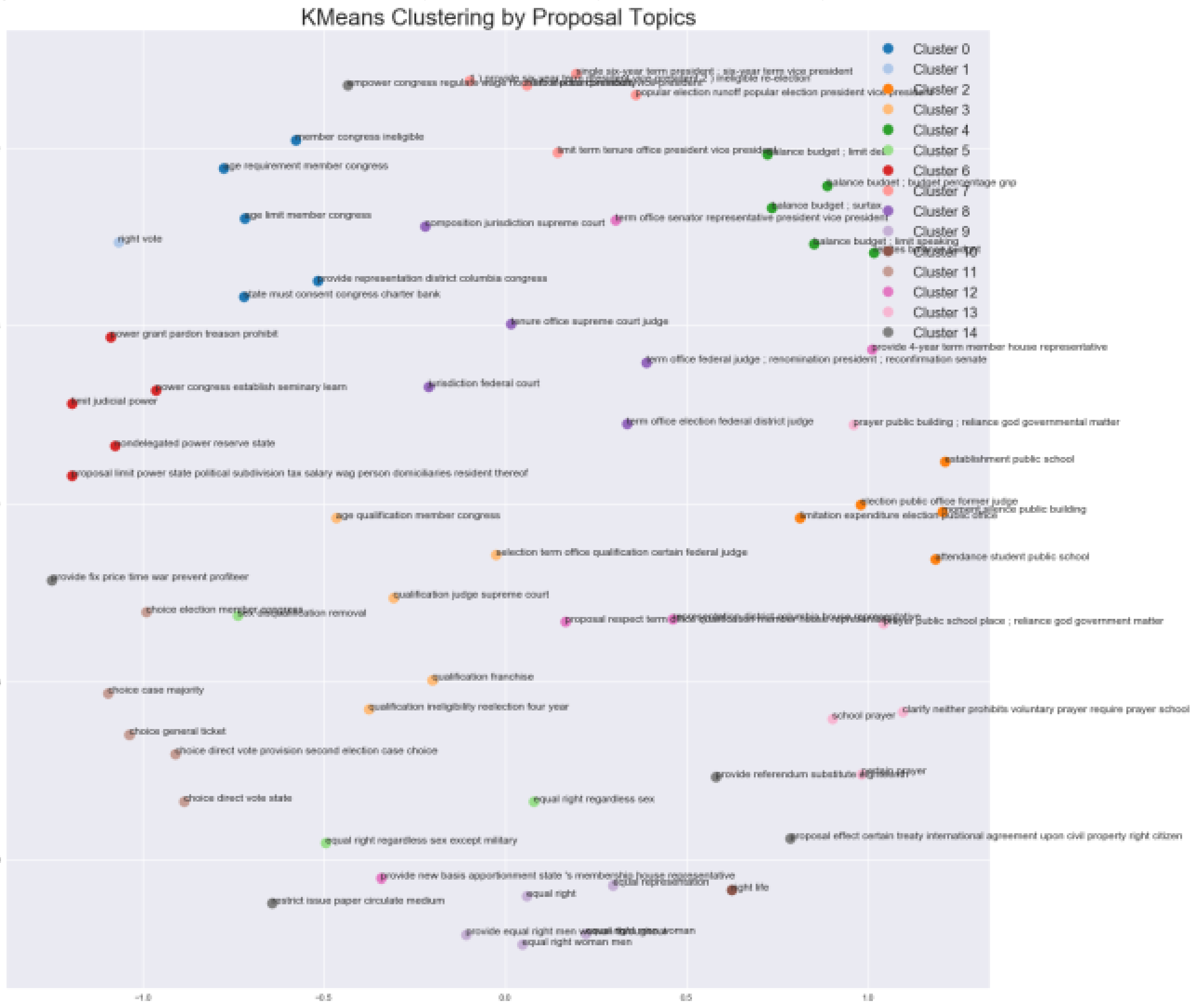
1. All titles were converted to lowercase
    - “a joint resolution proposing an amendment to the constitution of the united states relating to parental rights.”
2. ‘united states’ was stripped from all titles
    - “a joint resolution proposing an amendment to the constitution of the relating to parental rights.”
3. NLTK word_tokenize was used to split each title into lists of individual words
    - ['a', 'joint', 'resolution', 'proposing', 'an', 'amendment', 'to', 'the', 'constitution', 'of', 'the', 'relating', 'to', 'parental', 'rights', '.']
4. NLTK stopwords corpus was used to strip generic English stopwords from the titles (eg. ‘a’, ‘an’, ‘the’, etc.)
    - ['joint', 'resolution', 'proposing', 'amendment', 'constitution', 'relating', 'parental', 'rights', '.']
5. NLTK pos_tag was used to tag each title word as a part of speech
    - [('joint', 'JJ'), ('resolution', 'NN'), ('proposing', 'VBG'), ('amendment', 'NN'), ('constitution', 'NN'), ('relating', 'VBG'), ('parental', 'JJ'), ('rights', 'NNS'), ('.', '.')]
6. POS tags were converted to lemmatizer usable labels
    - [('joint', 'a'), ('resolution', 'n'), ('proposing', 'v'), ('amendment', 'n'), ('constitution', 'n'), ('relating', 'v'), ('parental', 'a'), ('rights', 'n'), ('.', '')]
7. NLTK WordNetLemmatizer was used to stem each title word according to its associated POS rules (default lemmatizer assumes words are all nouns)
    - ['joint', 'resolution', 'propose', 'amendment', 'constitution', 'relate', 'parental', 'right', '.']
8. Generic phrase terms specific to the amendment proposal dataset were removed along with punctuation (eg. 'joint', 'resolution', 'propose', '.')
    - ['parental', 'right']
9. Each title string was rejoined for topic generation
    - “parental right”
10. sklearn.feature_extraction.text.CountVectorizer was used to generate a matrix of word counts from the processed titles, using a min_df=5, meaning a word needed to occur at least 5 times throughout the entire dataset to be added to the dictionary. 867 words were identified for the dictionary.
11. sklearn.decomposition.nmf classifier was used to extract 500 topics from the vectorized titles

Clustering Implementation:

Sklearn.cluster.KMeans was used to group proposals into 15 clusters based on the 500 topic matrix generated from the text processing section above. 5 random proposals were selected from each cluster to be plotted, then cosine distances were calculated between the selected proposals' topic values, fed through an exp function to further separate the distances, and finally converted to a 2D array for plotting using sklearn.manifold.MDS. Redundant proposal titles were excluded to give a more comprehensive range of titles within each cluster.

Note: because of dimensionality reduction, the axes in the following graphs have no meaningful label definition and are an amalgam of topics.

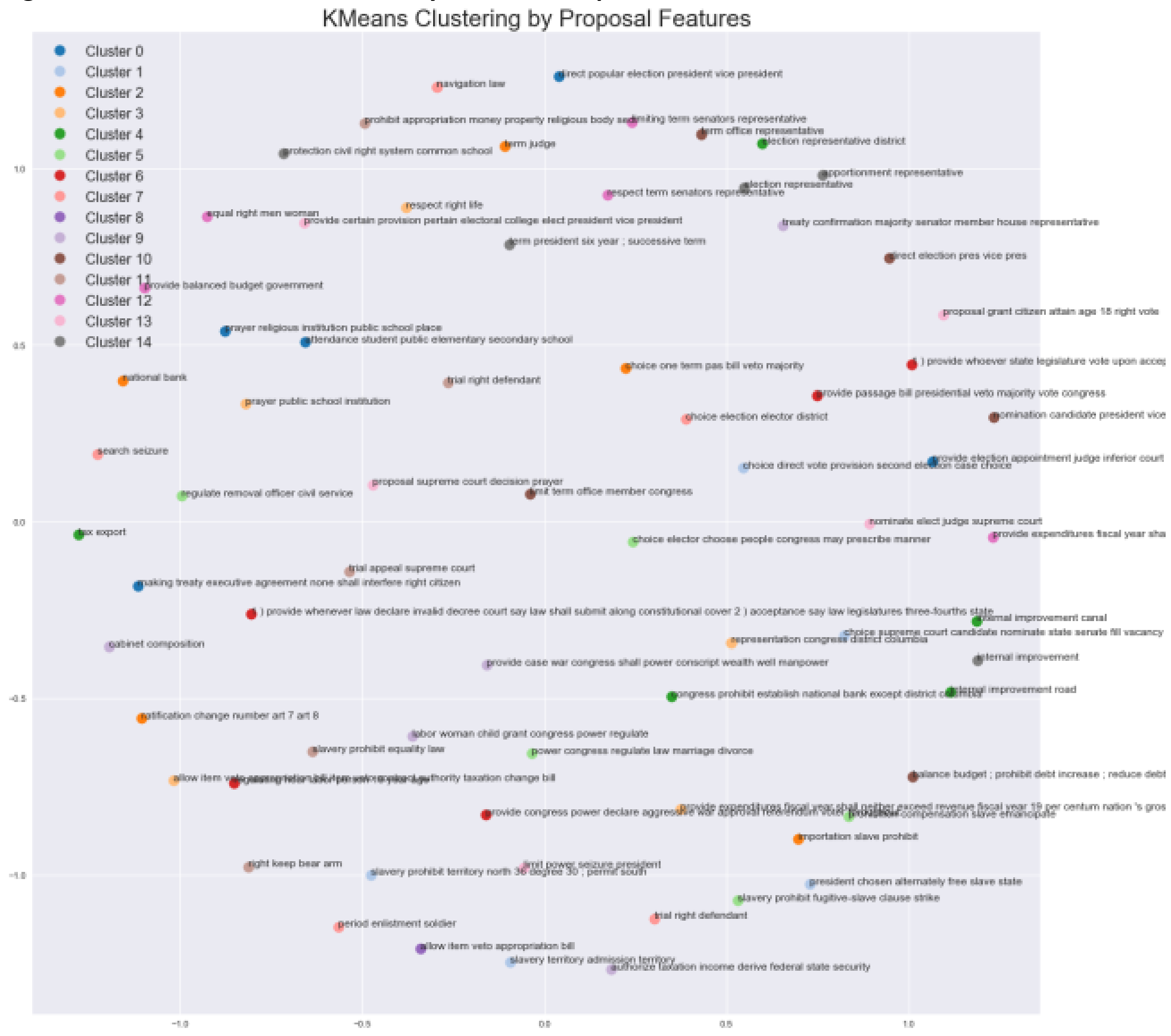
Figure 8: Visualization of 5 Randomly Selected Proposal Titles Per Topic Cluster



It's evident from the clusters that the topic generation and clustering were successful – the clusters have separated proposal topics into congressional requirements (Cluster 0), school prayer (cluster 13), equal rights (cluster 5), and so on. Some clusters, such as 5 and 9, could be combined further, but this visualization serves as a good check that the generated topic data is valid for the supervised learner.

Using the method above for clustering with the non-title related features included (eg. year, sen_maj_pct) generated the following:

Figure 9: Visualization of 5 Randomly Selected Proposal Titles Per Feature Cluster



The features most heavily weighted in this clustering tended to be 'year', 'legis_start', 'sen_maj_pct', 'rep_mat_pct', and 'congress'. Because these proposals were still plotted using distances calculated by topic separation, the increased scattering of cluster colors compared to the previous plot shows that similar topics span many years and congressional shakeups.

Supervised Learning Model Implementation:

The data feature set was reduced for algorithm training as follows:

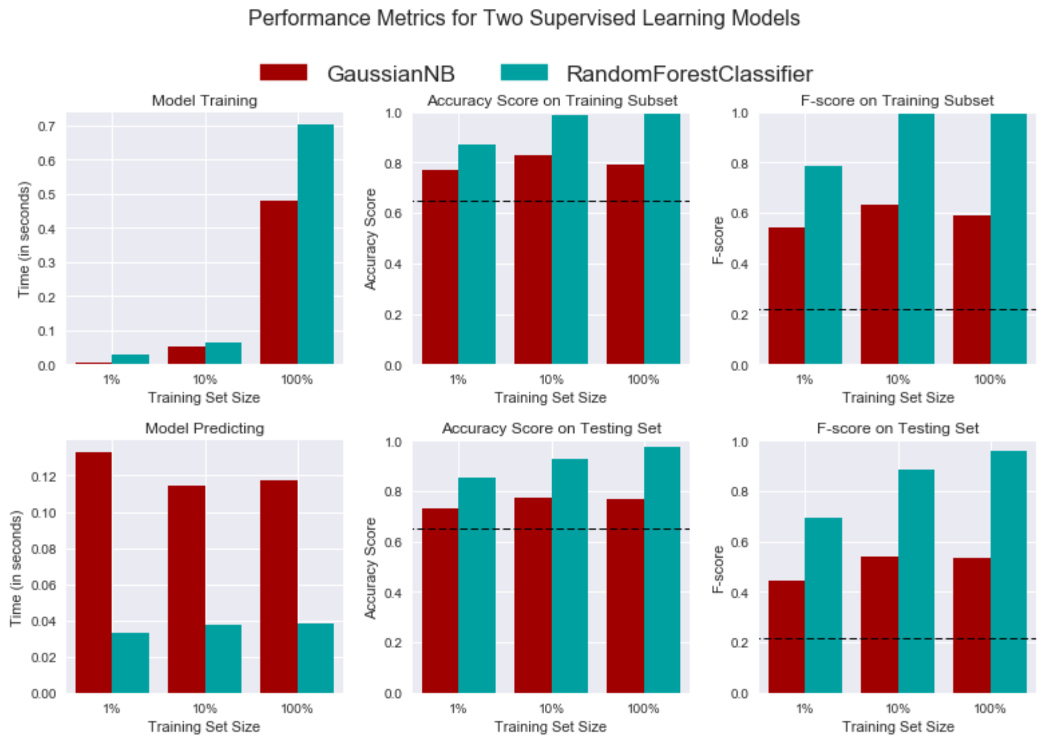
- 'title' and 'clean_title' were removed since they would generate too many unique one-hot encodings; the topic features provide the same information in numerical form
- 'sponsor_name' and 'legis_Index' were removed as individual legislators' names are not relevant to the classifier; each legislator's political term information is captured by the other 'legis_[]' features
- 'legis_state' was removed as it was redundant to 'sponsor_state' data

The remaining categorical features were encoded into numerical values using pandas.get_dummies, and any NaNs remaining in the dataset were filled with 0's. Then, the data was divided into 20% test set and 80% training set using sklearn.model_selection .train_test_split.

For the baseline naïve predictions, an output set with 22.4% positives was randomly generated to reflect a model that predicts entirely by chance, knowing only the number of true positives in the data. This naïve predictor had a 64.9% accuracy score and 21.7% F-beta (β=0.5) score. The same accuracy and F-beta scorers were used to judge performance of the classifiers.

Code was adapted from the MLND Finding Donors project to return results of training a Random Forest classifier and a Gaussian Naïve Bayes classifier (both with default hyper-parameters) on 1%, 10%, and 100% of the training set to observe whether the learners displayed differing advantages given different quantities of training data. For instance, perhaps the Naïve Bayes model would outperform Random Forest when only 1% of the training set was available, but Random Forest would perform better given 10% or 100% of the set.

Figure 10: Performance Comparison of Untuned GaussianNB and RandomForest Classifiers



The Random Forest classifier significantly outperformed both the baseline naïve predictor (dash lines) and the Gaussian Naïve Bayes classifier on all three training set sizes, achieving an accuracy score of 97.5% and an F-beta score of 95.9% on the test set when trained on the full training dataset.

Refinement:

sklearn.model_selection.GridSearchCV with an F-beta scorer was used to tune the Random Forest classifier. The following parameters were evaluated:

- 'n_estimators': [5, 10, 50] – the number of data subset trees generated for the forest
- 'criterion': ['gini', 'entropy'] – scoring criterion for a branch split; 'gini' minimizes misclassifications and 'entropy' maximizes information gain
- 'min_samples_split': [2, 5, 10] – the smallest number of samples allowed to split an internal node
- 'min_samples_leaf': [1, 2, 5] – the smallest number of samples allowed at a terminal leaf node

The default parameters in the initial model implementation were: n_estimators=10, criterion='gini', min_samples_split=2, min_samples_leaf=1.

The final optimal parameters were: n_estimators=50, criterion='entropy', min_samples_split=10, and min_samples_leaf=1, which resulted in a 0.4% improvement in accuracy score and 0.7% improvement in F-beta score over the unoptimized model.

IV. RESULTS

Model Evaluation and Validation:

sklearn.model_selection.StratifiedShuffleSplit was used to make three cross-validation splits from the training data (80% of original dataset) for the GridSearchCV hyper-parameter tuner. The model with the best performing set of hyper-parameters scored on the shuffle splits (training data only) was used to predict outcomes on the withheld test data (20% of the original dataset), achieving an accuracy of 97.9% and F-beta score (β=0.5) of 96.6% on data that was previously unseen to the Random Forest model. This shows that the model is robust even when presented with new data.

In Figure 10, the robustness can also be observed in the untuned Random Forest model, as the drops in accuracy and F-beta scores from training subset (seen data) to test set (unseen data) are less than 5% when the model was trained on 100% of the training data.

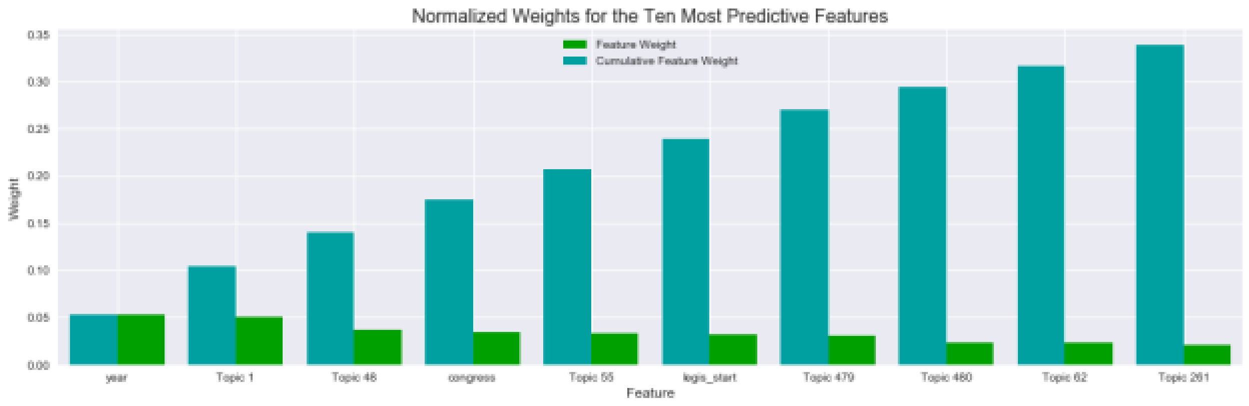
Justification:

The benchmark paper achieved an improvement of 4% in accuracy over their baseline naïve predictor and an F-beta score (β=0.5) of 88% using Naïve Bayes. This project achieved a 33% improvement in accuracy score over the naïve predictor and a final F-beta score of 96.6%, surpassing the benchmark model metrics. The model achieving a 97.9% accuracy score on unseen data shows that the classifier can confidently predict whether a previously introduced amendment proposal passed Congress.

Conclusion:

Feature importances were extracted from the optimized classifier, and the ten most heavily weighted are plotted below.

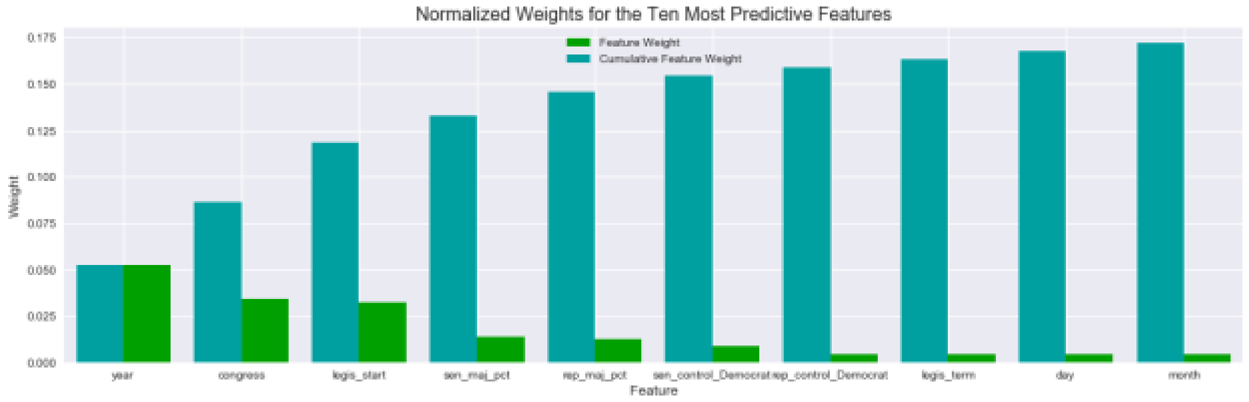
Figure 11: Ten Most Predictive Features for Optimized RandomForest Classifier



The top ten most predictive features were all either time-scale related (year, congress, legis_start) or topic related (Topic 1, Topic 48, Topic 55).

To gain a clearer picture of the influence of non-topic related features, since topic features correspond to already passed amendments which are unlikely to have relevance to future proposals, the following plot shows the ten most important features excluding topics.

Figure 12: Ten Most Predictive Non-Topic Features for Optimized RandomForest Classifier



This plot shows that aside from time-scale features, the percentage of majority control in both chambers and control by Democrats are next most important non-topic features, as well as legis_term, which was investigated in the data exploration section.

Reflection:

This project involved parsing and combining data from a variety of sources to glean a comprehensive set of features for analyzing amendment passage. Working with both csv and yaml files provided insight into how different data structures can be helpful in organizing data (nested levels in the yaml legislator data).

Part of the motivation for choosing this particular dataset was the opportunity to work with text data, and the Natural Language Toolkit and various text visualization techniques were incredibly powerful but also somewhat opaque to use. Working with so many features (500 in topic extraction alone) clearly demonstrated the usefulness of dimensionality reduction for visualizations. This allowed a sanity check at the intermediate clustering step which gave confidence that the data features generated from text processing sufficiently captured the information value of the proposal title data.

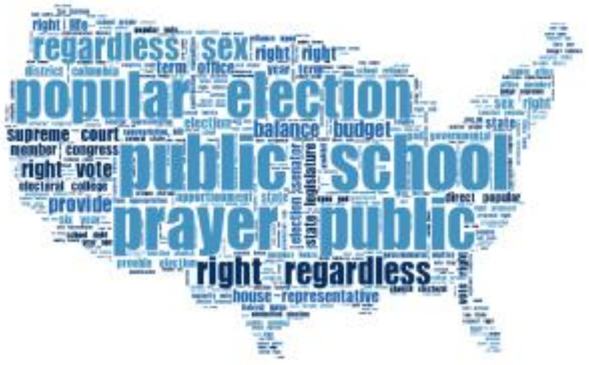
The various plots and visualizations in this project probably required the most amount of troubleshooting time, but once a plotting function was sufficiently vetted, it was simple to generate comparison plots for any number of features, which sped up the data exploration process and dismissed some preliminary prejudices for which features would be most significant in the optimized classifier.

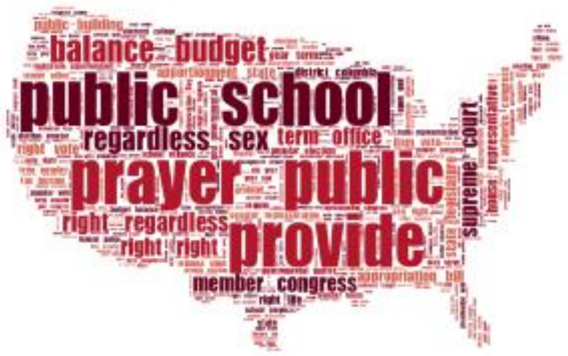
Training the supervised classifiers was relatively straightforward, although some consideration had to be given to how to handle missing values (ultimately the NaNs were all set to 0). There is newfound appreciation for how well documented sklearn is compared to many of the other python libraries used for this project.

The final scores of the optimized classifier were satisfyingly high, although analyzing the weighted features afterwards showed that there was not a great deal of useful information for projecting amendment passage odds into the future, since time-scale features and topic features were ultimately the most important in predicting outcome.

Final Thoughts:

Wordclouds of the proposal titles introduced by Democrats (blue) and Republicans (red):





Despite the intense partisanship in our politics now, most Constitutional amendment proposals had sponsors from both sides of the aisle, shown by the term overlaps in these wordclouds. From the data exploration section, it's evident that the successful amendments tended to have support both from the in-majority and non-majority legislators, and while ultimately the Senate and House majority percentage control were important non-topical features for predicting success, they only accounted for 0.025 of combined weight in the final supervised learner model and were thus much weaker predictors of success than the topic extractions. Thus, further amendment passage should not be predicated on trying to attain a super-majority in either chamber of Congress, but rather by building a bi-partisan coalition to agree on topics for consideration.

Future Work:

Additional features could be added to the dataset for further dependency investigation, such as frequency of occurrence of the highest ranked topic for each proposal, year gap to the most recent war (the ending of the Civil War triggered the 14th, 15th, and 16th amendments), and presidential party/term data.

Performing data exploration on those amendments which were passed by Congress but not ratified by the states would also be interesting, especially comparing state ratification votes against amendment sponsor states to see whether certain states tended to vote together as a block. However, due to the low number of passed but unratified samples, this may not yield much insight.

References:

1. https://www.archives.gov/open/dataset-amendments.html
   'us-nara-amending-america-dataset-raw-2016-02-25.csv'
2. Identifying Factors in Congressional Bill Success (Gingerich, Scher, Dodhia):
   http://snap.stanford.edu/class/cs224w-2014/projects2014/cs224w-62-final.pdf
3. https://github.com/unitedstates/congress-legislators
   'legislators-current.yaml', 'legislators-historical.yaml'
4. https://en.wikipedia.org/wiki/Party_divisions_of_United_States_Congresses
5. http://scikit-learn.org/stable/modules/model_evaluation.html