Predicting Voting Behavior of Supreme Court Nominees

Michael Fine, Jenny Huang, and Chris Rohlicek

October 21, 2019

Contents

1 Introduction

When a Supreme court nominee is confirmed, they are justices for life, and their future votes will affect countless major decisions on cases. They will decide, often by a one vote margin, the fate of abortion rights across the country, the meaning of equal protection, and occasionally, the next president of these United States. These votes hold great of power in shaping the United States, and ramifications echo long after the justice has left the bench.

Because of this, Supreme Court confirmation battles have become a unique touchstone in American politics. Before a Supreme Court justice even formally announces their retirement, the president will begin researching replacements. Dozens of White House staffers comb through millions of pages of writings, cases, and documents on potential nominees, hoping to determine how they will vote in important cases. The nomination hearings themselves have come to reflect the paramount importance of a seat on the Supreme Court. This was kicked into high gear in Ted Kennedy's famous "Robert Bork's America" speech, where he argued that if Bork were confirmed:

> ...women would be forced into back-alley abortions, blacks would sit at segregated lunch counters, rogue police could break down citizens' doors in midnight raids, school children could not be taught about evolution, writers and artists could be censored at the whim of the government, and the doors of the federal courts would be shut on the fingers of millions of citizens for whom the judiciary is-and is often the only-protector of the individual rights that are the heart of our democracy [3].

In modern confirmation hearings, each side will attempt to present dueling visions of a nominees jurisprudence. Modern nominees are chosen with this in mind, and often have little written record that Senators can attack. Indeed, much of the goal of confirmation hearings is to say as little as possible – the act of declining to answer a question is known as the "Ginsburg Rule", after notorious Supreme Court Justice Ruth Bader Ginsburg [2]. It is an open question how much, if any, information do we gain from Supreme Court confirmation hearings.

As such, an important question is: *How well can we predict the voting patterns of Supreme Court nominees at the time of nomination?*

We hope to answer this question by drawing from different data sets and training Logistic Regression and Random Forest classifiers to predict how future justices will vote on particular cases.

2 Background and Related Work

There has been past on work on predicting Supreme Court outcomes, however, most of these models focus on predicting the outcomes of specific cases. The data used in these models are most commonly the text data from the oral arguments of the case or the voting records of the justices ruling on the case. One of the more common approaches across these models is to use random forests, as discussed in the model described by Kaufman et al. [5].

Using these approaches as inspiration, we decided to broaden the scope of our model from predicting individual case outcomes, to predicting how nominees to the Supreme Court would vote generally if confirmed. This is a question of great relevance because Supreme Court nominations are important decisions on which data is kept intentionally sparse. With this as our motivation, we set out to build a model that can give an accurate prediction of how a nominee would vote if confirmed, given only the data available at the time of their confirmation.

We hope to integrate the nominees' confirmation hearing texts in some way. Kraft et al. explored a related problem of predicting legislative roll-call votes from bill texts [6]. They used an embedding-based approach and represented a bill text as the average of the pre-trained word embeddings within that text, something we will do as well.

3 Approach

3.1 Dataset Generation

We looked at three sources of data: demographic data on all past nominees, case data, and text transcripts from all nominees' confirmation hearings (if they had one). Because of the quality of transcriptions available we were limited to using the data from the most recent 16 justices' confirmation hearings (each roughly 40,000 words long).

3.1.1 Demographic Data

Data. The nominee information came from a dataset from WashU, which contains demographic and career information, as well as political science measures of their political/legal valence [8].

Preprocessing. We first eliminated unnecessary or overly specific features such as *Name of Nominees Mother*. Additionally, we used one hot encoding to turn categorical features like *State in which Nominee Served as an Assistant District or County Attorney* into a representative set of dummy variables. Lastly we transformed features based on years to be irrespective of the specific year of a given judge's nomination. For example, we replaced the features *First Year Nominee Served in the Continental Congress* and *Last Year Nominee Served in the Continental Congress* with *Years Nominee Served in the Continental Congress*. After the preprocessing stage, our de-

mographic dataset consisted of 704 constructed features (142 before one hot encoding).

3.1.2 Confirmation Text Hearings

Data. Before the Senate votes whether to confirm a nominee to the Supreme Court, they hold public confirmation hearings. Nominees read their prepared statement before the Senate, and then the Senators ask questions to the nominee pertaining to their judicial philosophy and qualifications. PDF transcripts of the hearings were obtained from the Senate Judiciary committee [4]. These contain all transcripts from Kavanaugh (2018) through Powell (1971). Additional transcripts are available, but these are in an older format that made Optimal Character Recognition impractical.

Preprocessing. To generate utterances from these PDFs, we transcribed them via the program pdftotext, and then parsed them into individual statements with regular expressions. We confirmed by hand that the transcriptions and parsing was roughly accurate, though there were a few transcription errors.

3.1.3 Case Data

Data. Data about court cases and voting patterns came from the Supreme Court Database (SCDB), a third party dataset that contains over 200 years of expertly-coded features for individual cases.

3.2 Feature Set Extraction

We then created multiple feature sets, to determine what kind of features were most informative for our models. Afterwards, we performed feature selection to narrow down the number of features due to computational costs.

3.2.1 Demographic

This feature set includes just the demographic + case data. This data is useful because each justice's past experiences and attributes may play a role in how they vote.

3.2.2 Demographic + N-grams

We decided to look at N-grams to see if the frequency of words in the confirmation hearings provided any information on how they might vote. To produce N-grams (sequences of N words), we looked at each text and cleaned it by removing stop words, converting to lowercase, and more. Then we vectorized the N-gram counts, limiting our feature set to the 5000 most frequent N-grams.

Demographic + 1-grams. This feature set merged the demographic + case data along with the vectorized counts of words within each confirmation hearing text.

Demographic + (1,5)-grams. This feature set merged the demographic data along with the vectorized counts of N-grams, where N can be any value between 1 and 5, within each confirmation hearing text. Some examples of these (1,5)-grams were: 'ago think', 'agree', 'agree disagree', and 'agree senator'.

Demographic + (1,5)-grams + TF-IDF. We then applied a TF-IDF transform to the vectorized counts above. This essentially weights the term frequencies by the inverse document frequency, meaning that words that are more common across documents are weighted less because they may be less specific or informative. This often improves on the standard N-gram approach. The transform is defined as

$$tfidf_{i,j} = tf_{i,j} * \log\left[\left(\frac{1+N}{1+df_i}\right)+1\right]$$

where tf_{ij} = number of occurrences of N-gram i in document j, df_i = number of documents containing i, and N total number of documents [7].

We then merged these N-gram vector counts (vector length 5000) with the demographic + case data features.

3.2.3 Demographic + Mean Word Embeddings

Unlike N-grams, embeddings take into account the semantic relationship between words, such that similar words are closer in the vector space. Following Kraft et al.'s approach, we represented each confirmation hearing text as the average embedding over all the words in that text [6]. We used the embedding model pre-trained on the Google News dataset of 3 million words and phrases [1]. We then merged these mean embeddings (vector length 300) with the demographic + case data features.

3.2.4 L1 Regularization Feature Selection

To find the most predictive features, we trained an SVM with L1-regularization on our feature sets. The penalty term in L1 regularization is $\lambda \sum_i |w_i|$ for model weights $\{w_i\}$, while the penalty term in L2 regularization is $\frac{\lambda}{2} \sum_i w_i^2$. Through gradient descent, the weights will be changed by a step-sized multiple of the gradient. Note that the L1 loss provides a constant gradient while the L2 provides one that diminishes with w_i. As a a result, L1 regularization allows the weights of features to diminish to zero, while those in L2 regularization are minimized but don't reach zero. As a result, linear models with L1-regularization have sparse solutions, meaning that many of the model's estimated coefficients are zero. We can then reduce the dimensionality of the data by selecting the features with non-zero coefficients [7].

After regularization with $C = \frac{1}{\lambda} = 1$ (the inverse regularization constant), we were left with around 500-600 features in each feature set.

3.3 Algorithms and Implementation

To implement the models we chose, we used the scikit-learn framework [7].

3.3.1 Logistic Regression

Motivation. Logistic Regression has a number of advantages like low variance, high interpretability (as outputs are probabilities for classes), and compatibility with linear decision boundaries at the expense of high bias.

However, it is unlikely that the relationship is truly linear – consider a trivial decision model for deciding the outcome of a justice's vote v_a on abortion cases, based on party p of the nominating president and whether the plaintiff is anti-abortion a. A reasonable model is

$$v_a = p \oplus a$$

That is, the justice votes in favor of the plaintiff if the plaintiff is anti-abortion and the justice was appointed by a republican, or if the plaintiff is pro-abortion and the justice was appointed by a democrat. But as we know, exclusive-or cannot be approximated by a linear function, and thus logistic regression could not simulate even this trivial model correctly.

That being said, the interpretability and simplicity benefits of logistic regression classifiers make it a good starting point and baseline for comparison.

Algorithm. To apply logistic regression to our data, we model the probability of a resulting vote as as sigmoided noisy linear function of case and demographic data, where $\sigma(x) = \frac{e^x}{e^x+1}$ is the sigmoid function:

$$\mathbb{P}(\text{Vote}_{i,j} = 1) \sim \sigma(f(\text{case}_i, \text{justice}_j) + \mathcal{N})$$

Because our problem is a two-class classification, we follow the regression model which is a function of the posterior distributions on the two classes, $P(\omega_i|\boldsymbol{x})$ [9]:

$$\ln \frac{P(\omega_1|\boldsymbol{x})}{P(\omega_2|\boldsymbol{x})} = \theta^T \boldsymbol{x} \implies P(\omega_i|\boldsymbol{x})(\theta) = \frac{e^{\theta^T \boldsymbol{x}}}{1 + e^{\theta^T \boldsymbol{x}}} = \sigma(\theta^T \mathbf{x})$$

Under this model, the goal is to minimize the negative log-likelihood with an L2 penalty:

$$-\log L(\mathbf{y}|\theta) = -\sum_{i=1}^{n} (y_i \theta^T x_i - \log(1 + e^{\theta^T x_i})) + \lambda ||\theta||_2^2$$

Hyperparameter Tuning. For tuning, we did a grid search on $C = \frac{1}{\lambda}$, the inverse of regularization strength for an L2 penalty. Adding regularization is important to prevent overfitting. In particular, logistic ridge regression tries to reduce the norm of θ and while also keeping the NLL small [9]. It adds the $\lambda ||\theta||_2^2$ term to our NLL.

Hyperparameter	GridSearch	Best
C	[0.1, 1, 10, 100]	1

Table 1: The process of hyperparameter tuning on Logistic Regression on the feature set with demographic data and 1-grams.

3.3.2 Random Forest

Motivation. We then chose to explore random forest classification for three primary reasons. First, it is an ensemble method so it reduces variance by averaging. Additionally, it is capable of capturing non-linear interactions between features, unlike logistic regression. Lastly, this was a common choice as a good classifier in related work [5].

Algorithm. A random forest is an ensemble of decision trees made independently with bootstrapped subsamples of training data. Although the bias is slightly larger than that of a decision tree (due to bootstrapping sub-samples), ensemble methods reduce variance by averaging, and this outweighs the bias disadvantage in practice [7].

Random forest decision trees are non-parametric models created by recursively partitioning the training data present along the feature and value that leads to minimizing impurity. While we learned about entropy as an impurity measure in class, we chose to use a popular alternative, called the Gini index, which results in a slightly sharper maximum compared to the entropy one [9]. The decrease in node impurity is:

$$\boldsymbol{I}(t) = \sum_{m=1}^{M} P(\omega_m|t)(1 - P(\omega_m|t))$$

where $\boldsymbol{I}(t_Y)$ and $\boldsymbol{I}(t_N)$ are the impurities associated with the two new sets, respectively [9]. The feature and threshold split maximizing the decrease in impurity, $\Delta \boldsymbol{I}(t)$, gives two new descendent nodes t_Y and t_N. After trees stop splitting, classes are assigned to each leaf node by the majority vote of the data present there.

Hyperparameter Tuning. We tuned the tree-based parameters shown in Table 2. Throughout tuning, we used 100 trees for each classifier, which was near our computational limit.

Hyperparameter	GridSearch	Best
MAX DEPTH	[None, 10, 35, 60, 85, 110]	85
MAX FEATURES	[$\log_2$, sqrt]	$\log_2$
MIN SAMPLES SPLIT	[2,5,10]	5
MIN SAMPLES LEAF	[1,5,10]	1

Table 2: The process of hyperparameter tuning on RandomForestClassifier on the demographic only feature set.

4 Experiments

4.1 Testing

The labels were 1 if the justice voted for the plaintiff on a case and 0 if they voted for the defendant. We used 5-fold cross validation on our training set to find training and validation scores. This prevents overfitting and

checks how generalizable these models are to new data for when we are evaluating models in hyperparameter tuning.

We learned about leave-one-out cross-validation in class and showed that it can approximate the expected loss using just the training data (it is unbiased):

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\theta}_{MAP}(D\backslash i)^T x_i)^L$$

$$\rightarrow_{n\rightarrow\infty} \mathbb{E}_{y^N}[||y^N - \hat{\theta}_{MAP}(\xi)||_2^2]$$

However, we decided to use $k-$fold cross validation instead. We made this decision because our data set is large so this reduces computation time and it also reduces the variance of performance on the test set because training sets have less overlap. Given that we are limited by computational resources, it makes sense to choose the method with less variance and less computation time. If we had more time, we would choose an even higher k to reduce variance.

To calculate our final test scores, we trained the model on the whole training set and scored its performance on the test set (data we had not trained on before).

4.2 Results

4.2.1 Feature Set Selection

We created 5 feature sets: Demographic data only, Demographic + 1-grams, Demographic + (1,5)-grams, Demographic + (1,5)-grams + TF-IDF transform, and Demographic + Mean Embeddings. Then, we trained logistic regression and random forest classifiers on each of our feature sets to determine which to use.

Accuracy Scores		
Model	Train	Validate
LR (Demographic)	0.644	0.627
LR (Demographic + 1grams)	0.642	0.629
LR (Demographic + (1,5)-grams)	0.641	0.628
LR (Demographic + (1,5)-grams + TF-IDF)	0.643	0.624
LR (Demographic + Mean Embeddings)	0.644	0.623
RF(Demographic)	0.992	0.703
RF(Demographic + 1grams)	0.991	0.643
RF(Demographic + (1,5)-grams)	0.991	0.643
RF(Demographic + (1,5)-grams + TF-IDF)	0.991	0.651
RF (Demographic + Mean Embeddings)	0.991	0.658

Table 3: Performance of Logistic Regression (LR) and Random Forest (RF) on the training, validation, and test sets for different feature sets.

From the performance seen in Table 3, we can see that for the logistic regression classifier, the feature set that gave the best validation score was Demographic + 1-grams, so we decided to go with that. For the random forest classifier, the feature set that gave the best score was just the demographic data.

4.2.2 Model Selection

We then needed to decide which classifier to use. The performance results before and after tuning are displayed below. (Note that for our final random forest classifier test, we changed the number of trees to 1000, because generally more trees give better performance [7].)

Accuracy Scores			
Model	Training	Validation	Test
LR (Not-tuned)	0.6420	0.6286	N/A
LR (tuned	0.6421	0.6292	0.6249
RF (Not-tuned)	0.9919	0.7028	N/A
RF (Tuned	0.9482	0.7080	0.7228

Table 4: Performance of models on the training, validation, and test sets based on hyperparameter tuning.

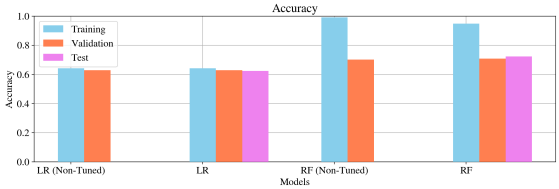

Figure 1: Accuracy Plot of Classifiers for Model Selection.

The Tuned Random Forest Classifier performed the best, with the highest validation accuracy of 70.80% and final test accuracy of 72.28%. Note that tuning using cross validation did improve the accuracy performance of the models as expected, and that since the validation and test scores are very similar, our models are unlikely to be overfit.

4.2.3 Confusion Matrices

We generated confusion matrices for the tuned Logistic Regression and Random Forest estimators performance on the test set.

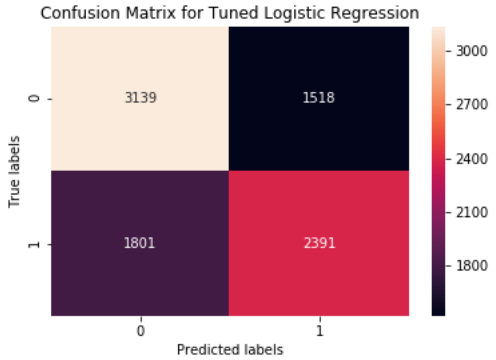

Figure 2: Confusion Matrix for Tuned LR.

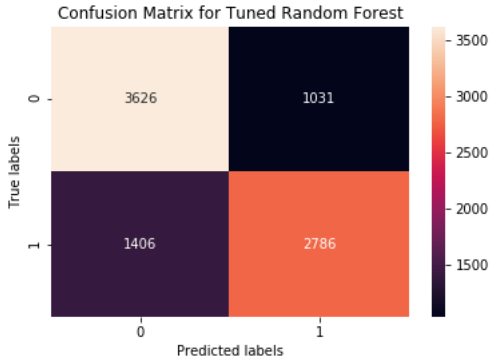

Figure 3: Confusion Matrix for Tuned RF.

4.2.4 Performance Metrics

The generated confusion matrices allow us to calculate key performance metrics of the algorithms. We chose these metrics to give us a better perspective of the success of the algorithm and to ensure that the algorithm was unbiased. The F1 score, which takes into account the sensitivity and precision, also measures accuracy.

Performance Metrics				
Model	Precision	Sensitivity	Specificity	F1-Score
Tuned LR	0.61	0.57	0.67	0.59
Tuned RF	0.73	0.66	0.78	0.70

Table 5: Performance Metrics of Classifiers.

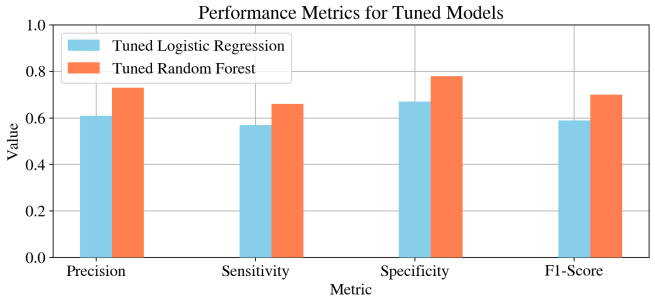

Figure 4: Plot of Performance Metrics of Classifiers.

A visual representation of these trends is shown in Figure 4. For both classifiers, the metrics are not too far apart, which is a good sign that no one classifier is too biased. However, each algorithm interestingly brings a higher specificity than sensitivity. Since there is often a trade-off between these two measurements, this makes sense. Since the specificity is higher, this means that the classifiers are more successful at correctly classifying a justice's vote on a case when they are voting for the defendant. However, since the sensitivity is lower, the classifiers are not as successful at classifying a justice's vote on a case when they are voting for the plaintiff. This effect may be more pronounced because in the dataset, about 53% of the justice, case pairs had a vote for the defendant, so there were slightly more examples like that. Because of the difference in sensitivity and specificity, our algorithms may be a little biased towards classifying votes as for the defendant than they should be.

4.2.5 ROC Curve

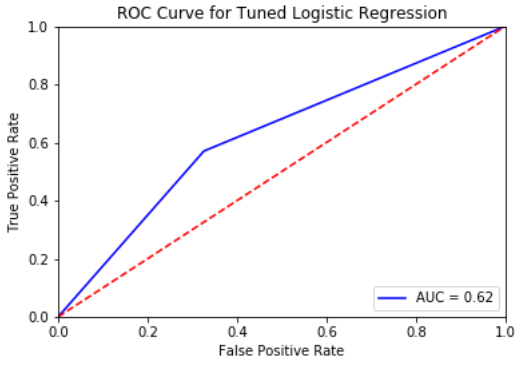

Figure 5: ROC Curve for Tuned LR.

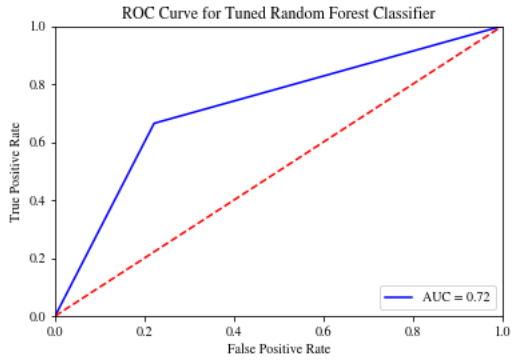

Figure 6: ROC Curve for Tuned RF.

The ROC curves give us a better understanding of the tradeoff between sensitivity and specificity. The area under the curve (AUC) is another good performance metric that lets us see how well the model can distinguish between classes. Here, we see that the Tuned Random Forest classifier had an AUC of 0.72 while the Logistic Regression had an AUC of 0.62.

5 Discussion

5.1 Summary of Approach and Results

After obtaining and pre-processing our data, we then created multiple feature sets to determine which would be the most useful for our models. These feature sets looked at the demographic data along with N-gram counts and mean embedding representations of the confirmation text hearings. For each feature set, we used an SVM with L1 regularization to find the most predictive features to use for training. We found that for logistic regression, the best feature set was the demographic data along with 1-gram vector counts. The random forest classifier performed best on just the demographic data. Note that adding a TF-IDF transform did improve the results of the N-gram approach (as expected), but that the mean embeddings performed better. This makes sense because it takes into account the semantic relationships between words while N-grams do not.

The success of our model on the demographic data alone suggests a nominee's confirmation hearing text may not be very informative about their future voting behavior when in combination with the nominee's demographics. This would confirm the popular view that confirmation hearings contain little substantive information from the nominee.

We then tuned the hyperparameters for these classifiers on the feature sets they performed the best on. The tuned random forest classifier on the demographic feature set performed the best with a final test accuracy of 72.28%, while the tuned logistic regression classifier had an accuracy of 62.49%. As the validation and test scores are very similar, we can be more sure that we have not overfit the data and perhaps there is still room for complexity in our models (like with a neural network). Throughout model evaluation, we found that random forest consistently outperformed logistic regression. This result is intuitively unsurprising as logistic regression has linear decision boundaries, which is simplistic. As such, we select the random forest classifier as our model for predicting voting behavior.

As seen in Table 6, the performance of our best classifier beat the baseline predictor which always guessed in favor of the defendant. Moreover, it performed close to a random forest classifier trained on past votes of Supreme Court Justices, so this suggests that the demographic data seems to be very predictive.

Accuracy Scores		
Model	Train	Test
Baseline (Guess defendant)	0.531	0.531
Baseline (RF on Past Votes)	0.996	0.724
Tuned RF (Demographic)	0.9482	0.7228

Table 6: Comparison of best model with baseline models.

5.2 Hypotheticals

Because our model does not require a voting record to model a justice, we are able predict the behavior of out-of-sample nominees without a voting record on the Supreme Court. This lets us answer the interesting hypothetical: what what cases would have changed if failed nominees had been successfully confirmed?

In the time period we looked at, we had data on three failed nominees: Robert Bork, Harriet Miers, and Merrick Garland. For each nominee we looked at all cases decided by one vote and selected the cases where our model predicted they would have swung the decision by voting differently than the justice who eventually took their seat. Out of the 514 cases with a one vote margin, we predicted 128 where a different outcome would have occurred if the failed nominee were on the bench.

Of those 128 flipped cases, three impactful examples include *Trump v. Hawaii*, where the court failed to enjoin the Trump administration travel ban; *Obergefell v. Hodges*, which declared bans on gay marriage unconstitutional; and *Citizens United v. FEC*, which struck down attempts to limit independent expenditures in political campaigns.

It must be noted that these predictions are counterfactual, therefore not empirically testable. Nor is the predictive accuracy of our model sufficiently high to treat these predictions as dispositive. Still, the forward looking nature of our model provides an empirical grounding for discussing these hypotheticals – a grounding often lacking in the charged popular discourse.

References

[1] Google code archive: Word2vec.

[2] Opinion — The Ginsburg Rule is not an excuse to avoid answering the Senates questions.

[3] Robert Bork's America — C-SPAN.org.

[4] U.S. Senate: Supreme Court Nomination Hearings.

[5] Aaron Russell Kaufman, Peter Kraft, and Maya Sen. Improving supreme court forecasting using boosted decision trees. *Political Analysis*, page 17.

[6] Peter Kraft, Hirsh Jain, and Alexander M. Rush. An embedding model for predicting roll-call votes. In *EMNLP*, 2016.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Jason Roberts, Scott Hendrickson, Nancy Staudt, Thomas G. Walker, and Lee Epstein. The u.s. supreme court justices database., Feb 2019.

[9] Sergios Theodoridis. *Machine Learning: A Bayesian and Optimization Perspective*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 2015.