2016

A Machine Learning Classifier for Corporate Opportunity Waivers

Gabriel V. Rauterberg
cangov@gmail.com

Eric L. Talley
Columbia Law School, etalley@law.columbia.edu

A Machine Learning Classifier for Corporate Opportunity Waivers

Gabriel Rauterberg & Eric Talley
October 2016

Introduction

Rauterberg & Talley (2017) develop a data set of "corporate opportunity waivers" (COWs)—significant contractual modifications of fiduciary duties—sampled from SEC filings. Part of their analysis utilizes a machine learning (ML) classifier to extend their data set beyond the hand-coded sample. Because the ML approach is likely unfamiliar to some readers, and in the light of its great potential across other areas of law and finance research, this note explains the basic components using a simple example, and it demonstrates strategies for calibrating and evaluating the classifier. (Our analysis largely tracks the procedure developed by Talley & O'Kane 2012.) There are six principal steps:

- Raw Text Extraction
- Clean Up
- N-Gram Parsing
- TF-IDF Transformation
- Dimensionality Reduction
- Supervised Calibration, Out-of-Sample Extrapolation, and Evaluation

Raw Text Extraction

The starting point of the analysis is a set of textual documents, assumed here to be in ASCII format (but could be html, pdf, UTF-8 etc.). This set of documents constitutes the set of raw unstructured "inputs" the researcher is interested in (e.g., merger agreements, constitutions, corporate charters, contracts, legal opinions, etc.) In Rauterberg & Talley (2017) the documentary inputs were textual "snippets" identified and extracted from public securities filings as *candidate* COWs. Candidates were flagged through a deliberately general and over-inclusive Boolean key-word search of SEC/Edgar filings. Specifically, document snippets consisted of the flagged key words, plus a 150-word margin of text proceeding and succeeding them in the document. Where the key-word search flagged multiple sections of a single document, the resulting snipped included the union of the collected individual snippets and 150-word margins. While the Boolean key-word query deliberately flagged many "false positive" documents, it also helped both to pare down both the number of documents requiring classification, and to narrow the text to be analyzed within each document. (See Rauterberg & Talley 2017 for details.)

Clean Up

From the set of candidate snippets, Rauterberg & Talley (2017) undertook measures to clean and parse the raw data. These measures typically consist of several steps:

- **Typesetting Code:** If the document was in a non-ASCII format, any typesetting codes (e.g., html tags) are stripped out.[1]
- **Punctuation:** All punctuation is typically stripped out from the raw words (such as periods, commas, exclamation points, question marks, elipses, colons, semi-colons, etc.).
- **Stems:** Word inflections are also typically stripped to their word stem (or base form). For example, the terms "walking", "walked", "walks", and "walkable" would all be reduced to "walk", and treated identically thereafter. A stemming library is necessary for this step (available in many Python modules).
- **Stop Words:** Finally, although not implemented in Rauterberg & Talley (2017), it is often thought desirable to drop "common" words that contribute little to the semantic content of the document. For example, words like "the" or "an" or "are" or "is" are frequently dropped in some applications. Stripping out stop words similarly requires using a library utility.

N-Gram Parsing

From the cleaned and parsed documents, the next step is to extract a numerical matrix of *N-grams*—or a "bag of words"—tabulating raw numerical counts of each unique permutation of "N" consecutive words across the entire set of relevant documents. For instance, a matrix of "1-grams" or "unigrams" would constitute raw frequency counts of single stemmed terms. Denote this matrix by **N**, where representative element n_{ij} represents "count frequency" -- the number of times term *j* appears in document *i*. Matrix **N** is a foundational intermediate result.

To elucidate this step, consider a simple example inspired by a familiar literary canon.[2] Suppose one were interested in a set of five "documents" (labelled D1 through D5, respectively), whose contents are as follows:

- D1: *Hickory, dickory, dock;*
- D2: *The mouse ran up the clock;*
- D3: *The clock struck one;*
- D4: *The mouse ran down;*
- D5: *Hickory, dickory, dock.*

Parsing these documents into 1-grams yields the following matrix **N**:

	hickory	dickory	dock	the	mouse	ran	up	clock	struck	one	down
D1	1	1	1	0	0	0	0	0	0	0	0
D2	0	0	0	2	1	1	1	1	0	0	0
D3	0	0	0	1	0	0	0	1	1	1	0
D4	0	0	0	1	1	1	0	0	0	0	1
D5	1	1	1	0	0	0	0	0	0	0	0

Table 1: Raw 1-gram matrix N

[1] In the current project, strings are converted into bytes and back to strings to clean out non-ASCII characters.
[2] *See* Goose, M. (c. 1697).

Note that in this example, the common word *"the"* appears several times. As previously noted, a stop-word dictionary typically strips this type of term out, and the use of such a dictionary is subject to the preferences of the researcher. (Our preference in this project was to leave stop words in and discount them with a TF-IDF transformation – as detailed below).

Rauterberg & Talley (2017) similarly extract 1-grams from their raw data set. As will become apparent below, their ML classifier was extremely accurate based on 1-grams alone, such that expanding the parsing protocol was unnecessary.

TF-IDF Transformation

Having extracted raw matrix **N,** the next step is to transform the raw frequency counts into "term frequency – inverse document frequency" (or TF–IDF) measures. The purpose and effect of this transformation is to accord greater proportional weight to the counts of terms that appear frequently in a *particular* document and yet are relatively *uncommon overall*. The resulting transformed matrix, **T**, contains representative element t_{ij} for document i and term j, defined by the expression

$$t_{ij} = \left(\frac{n_{ij}}{\Sigma_m n_{mj}} \right) \times \ln \left[\frac{|\{ j : n_{ij} > 0 \}|}{M} \right]^{-1}, \qquad (1)$$

where $m \in \{1, \ldots, M\}$ indexes the universe of documents analyzed. The first bracketed element of (1) represents the raw count of a given term in document i relative to its total across all documents. The second term consists of the log of the inverse frequency with which term j appears (at least once) across the universe (with cardinality M) of documents analyzed. By "rewarding" the frequent intra-document use of terms that are rare on the whole, the TF–IDF transformation tends to be better able to differentiate unique documents (Salton and Buckley, 1988).

In our example from the previous section, transforming the raw counts of matrix **N** into a TF-IDF matrix **T** is a relatively straightforward computational task. Consider the term "*the*", which appears twice in D2. This term appears a total of four times across all documents. And thus, in D2, the relative frequency of "the" is given by:

$$\left(\frac{n_{ij}}{\Sigma_m n_{mj}} \right) = \frac{2}{4} = 0.5$$

Additionally, the term "the" makes at least one appearance in 3 out of a total of 5 documents, and thus its document frequency is 3/5. The natural log of the inverse of the document frequency is therefore:

$$\ln \left[\frac{|\{ j : n_{ij} > 0 \}|}{M} \right]^{-1} = \ln \left(\frac{5}{3} \right) = 0.223,$$

and thus the TF-IDF value for the word "the" in D2 is:

$$t_{2,4} = \left(\frac{n_{2,4}}{\Sigma_m n_{m,4}} \right) \times \ln \left[\frac{|\{ j : n_{2,4} > 0 \}|}{5} \right]^{-1} = \left(\frac{2}{4} \right) \times \ln \left(\frac{5}{3} \right) = 0.112.$$

Applying the identical transformation to the other elements of the raw unigram matrix is given in Table 2 below. Notice from the Table that D1 and D5 have identical components, indicating that

they are substantially similar (indeed identical) to one another. In addition, D2 and D4 share many similar components, but are far from identical. D3 is the most unlike the others.

	hickory	dickory	dock	the	mouse	ran	up	clock	struck	one	down
D1	0.458	0.458	0.458	0	0	0	0	0	0	0	0
D2	0	0	0	0.112	0.458	0.458	1.609	0.458	0	0	0
D3	0	0	0	0.056	0	0	0	0.458	1.609	1.609	0
D4	0	0	0	0.056	0.458	0.458	0	0	0	0	1.609
D5	0.458	0.458	0.458	0	0	0	0	0	0	0	0

Table 2: TF-IDF matrix T

Dimensionality Reduction

Note that like the raw count matrix **N**, the TF–IDF transformed matrix **T** has a fixed point at $n_{ij} = 0$, and thus any term that appeared with a frequency of zero in the raw matrix would remain at zero after transformation. Consequently, in most "real world" applications **T** usually remains both extremely large (lots of columns) and sparse (lots of cells with 0s). The next step, then, is to employ a technique known as *singular value decomposition* that allows one to reduce the dimensionality of the data set with minimal information loss.

The concept of SVD is a fairly straightforward matrix operation. Consider our transformed TF–IDF word count matrix **T** (with elements t_{ij}). Singular value decomposition (or SVD) involves using the algebraic structure of **T** to produce synthetic variables that are designed to explain internal variation within **T**. The key algebraic relationship for accomplishing this task is given by the decomposition:

$$\mathbf{T} = \mathbf{U} * \mathbf{\Sigma} * \mathbf{V}^{\mathbf{T}},$$

where **U** is a column orthonormal basis; **V** is a row orthonormal basis; and **Σ** is the diagonal matrix of eigenvalues, where the component eigenvalues σ_{ii} are ordered from largest to smallest as one proceeds from down the diagonal, from upper left to lower right.[3]

A machine classifier typically makes use of the first *k* components of **U** to estimate the parameters of a predictive model (e.g., logit, probit), where *k* is chosen by the researcher. See, e.g., Bishop (2006); Jolliffe (2002). These eigenvectors are decreasing in strength, in that the first eigenvector explains the largest fraction of the variability in the columns of matrix **T**, the second eigenvector explains the second largest, and so on. Usually, the researcher will need to select a criterion for determining the value of *k*. One common rule of thumb is to pick the number *k* that retains some specified percentage γ of the "energy" of the data, as measured by the sum of the squares of the principal eigenvalues in the diagonal matrix **Σ**. That is, the value of *k* is the smallest value for which:

$$Energy_k = \frac{\sum_{i=1}^{k}(\sigma_{ii})^2}{\sum_i(\sigma_{ii})^2} \geq \gamma$$

[3] The SVD is done in python using Scipy's SVD function.

Leskovec et al. (2014). The subset of the columns of **U** selected becomes the variables of interest for calibrating a predictive model.

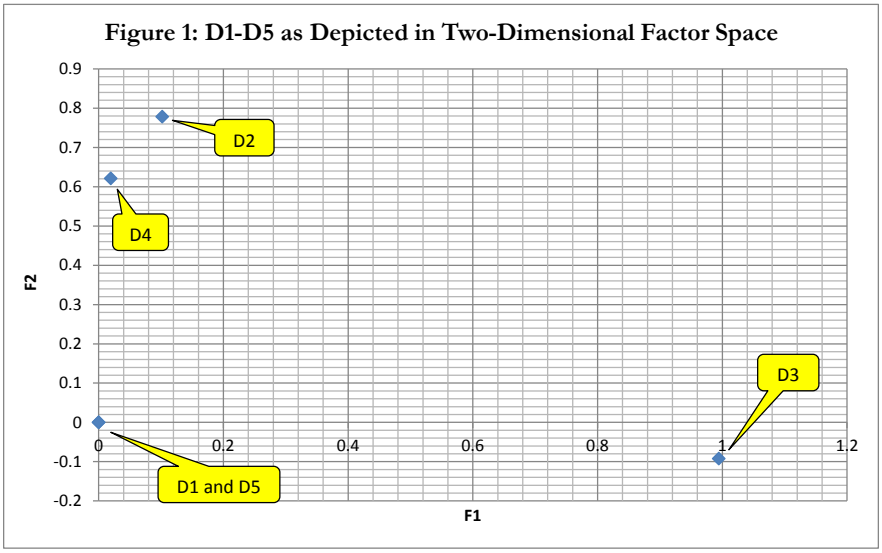
In our running example involving five simple documents, the singular value decomposition of matrix **T** yields the following set of non-zero eigenvalues: {2.327, 1.883, 1.636, 1.122}. Note that there are only four non-zero singular values, even though the example data set has five observations. This observation reflects the fact that the data set is not of full rank, since D1 and D5 have identical content. The first two eigenvalues manifest aggregate energy of approximately 63%, indicating that two factors are capable of explaining three fifths of the variation in the data. Adding a third factor would increase the energy to 84%, and adding a fourth would increase it further to 100%. For illustrative purposes, we limit our attention below to the first two eigenvalues, which have associated eigenvectors denoted as F1 and F2. The numerical values of these factors as follows:

	F1	F2
D1	0	0
D2	0.10223301	0.77838211
D3	0.99457099	-0.0921333
D4	0.01941521	0.62099336
D5	0	0

Table 3: 2-Factors

Although some information is clearly lost in reducing the dimensionality of the parsed data set from 11 columns to 2, the retained factors still manage to capture a significant amount of the variation in the documents. D1 and D5, for example, still share identical coordinates, and D2 and D4 still appear similar (albeit not identical) in both dimensions. D3 remains the most unlike the others. A scatter plot of the documents in factor space helps illustrate this point, as pictured in Figure 1.

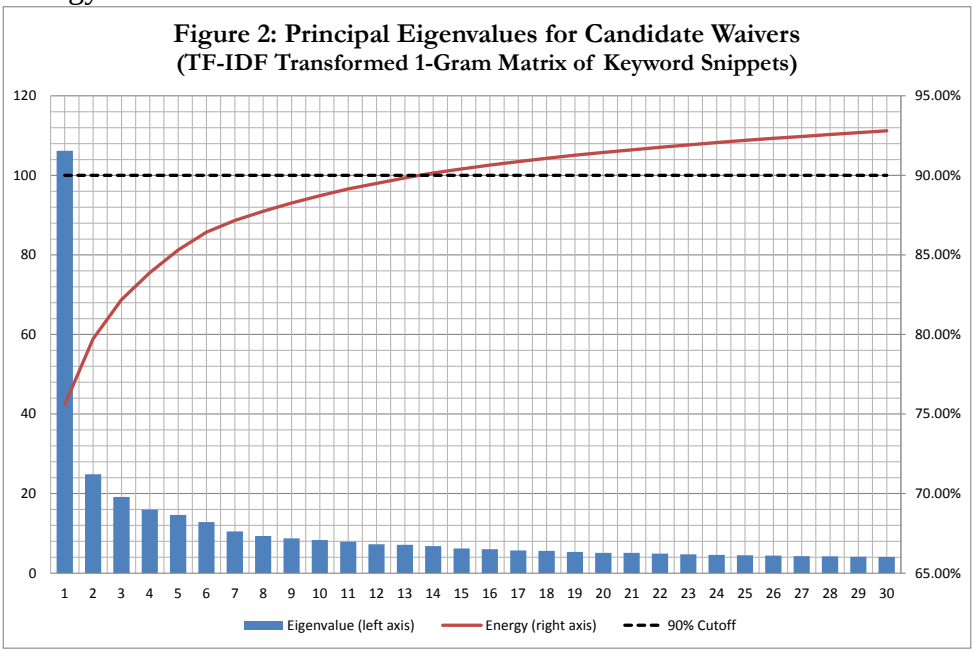
Figure 1: D1-D5 as Depicted in Two-Dimensional Factor Space



Even reduced dimensionality space, the documents effectively cluster in three areas, and are amenable to analysis as such.

Rauterberg & Talley (2017) undertake the same steps as above with approximately 10,600 snippets of candidate corporate opportunity waivers taken from public filings. The eigenvalues that emerge from that analysis are pictured in the figure below, along with their cumulative energy.

Figure 2: Principal Eigenvalues for Candidate Waivers
(TF-IDF Transformed 1-Gram Matrix of Keyword Snippets)



As can readily be seen from the figure, substantial variation in the data set can be captured with a relatively small number of factors. The number of factors used for model calibration must be selected with some care. Select too few, and the classifier will be insufficiently nuanced to be helpful. Select too many, and the researcher is sure to overfit the model, making its extrapolation to out-of-sample data unreliable. Rauterberg & Talley (2017) ultimately select the first fourteen principal eigenvectors, corresponding to a 90% energy threshold. Based on calibration tests (see below), this choice appears to work extremely well in the corporate opportunity waiver context.

Supervised Calibration, Out-of-Sample Extrapolation, and Evaluation

The process described above – i.e., summarizing latent textual data and reducing its dimensionality – are the first steps of a variety of distinct machine learning applications. What happens from there turns on the nature of the project. Broadly speaking, machine learning techniques applied to textual data tend to divide into one of two approaches: unsupervised learning and supervised learning. *Unsupervised learning* explores the structure of the data itself, with the goal of uncovering logical patterns, similarities, sequential regularities, and natural "clusters" internal to the text itself and without outside input. The task of identifying natural clusters, or "topics" associated with the data is often concomitant with (or very similar to) extracting principal factors, as described above.

While unsupervised learning techniques can be incredibly useful in uncovering patterns in data, evaluating the significance of such patterns is sometimes challenging unless one has an independent means for assessing importance and/or gravity of various types of document. This limitation is particularly salient in legal applications of text analysis, which tend to turn on

whether the language used in a contract / regulation / judicial opinion imposes a net legal burden or benefit on populations of interest, and the magnitudes of such effects. Consequently, it will often be desirable – in addition to summarizing the textual data as above – to involve "real world" human classifiers to assess the language and import of some subset of the textual data being analysed. In turn, this human-coded data can be used to "train" a predictive model to classify the universe of documents, including those that human coders have never previously assessed.

Rauterberg & Talley (2017) pursue the latter course of supervised learning. From a data set of 10,682 snippets that included candidate waivers, the authors and a team of research assistants manually coded 1,000 randomly selected snippets along over 40 dichotomous dimensions, including not only the presence/absence of a waiver disclosure (our primary topic of interest), but also the scope, reach, and location of such disclosures when they occurred. The most significant variable, of course, is whether the candidate disclosure was a *bona fide* disclosure of a waiver (in whatever form). They find approximately 62% of the hand-classified sample fits this description. The hand-coding of this sample, in turn, creates a "training" data set for a machine learning *perceptron* – an algorithm for predicting the presence or absence of a genuine disclosure in a given document.

Stripped down to its bare essentials, calibrating a perceptron reduces to estimating a qualitative regression such as:

$$\Pr\{y_i = 1\} = f(X_i, \varepsilon_i|\beta), \tag{2}$$

where y_i denotes the dependent dichotomous variable (here, the presence or absence of a waiver disclosure), f is some (potentially non-linear) function of data attributes Xi and an error term ε_i, and β is a vector of estimated coefficients. In what follows, we employ logistic regression, but a similar approach would apply to probit, linear probability, SVM estimation, and others.

Once the elements of β are estimated, it is possible to assess how well the model "fits" the patterns manifest within the sample data set. Unlike more conventional regression analysis approaches, however, where the elements of X_i are easily interpreted variables of interest, here these elements consist of the principal eigenvectors extracted from the decomposition of the text data matrix. As such, their interpretational content is recondite, and thus the interpretation of the estimated coefficients is of little interest.

Hand Coded

ML Classifier	Present	Absent	Total
Present	595	35	630
Absent	30	340	370
Total	625	375	1000

Logit Estimation

Pseudo R^2	0.7194
Log lik.	-185.65
$\chi^2(14)$	951.83***

Fit Measure (50% Classifier)

Sensitivity (True Positive Rate)	95.20%
Specificity (True Negative Rate)	90.67%
Positive predictive value	94.44%
Negative predictive value	91.89%
False Positive Rate (1 - Specificity)	9.33%
False Negative Rate (1 - Sensitivity)	4.80%
Converse False Positive	5.56%
Converse False Negative	8.11%
Correctly classified	93.50%

Table 4: Within-Sample Classification Accuracy of COW Disclosures

What *is* usually of interest, however, is whether the model does a good job in practice of classifying the textual data. Several diagnostics are helpful in measuring predictive performance, such as the rate of false positive and false negative classifications. Table 4 provides several metrics along these lines, employing a classification "assignment rule" that assigns the value of 1 whenever the estimated model probability of a COW is at least 50%. Using this assignment rule, we find a correct classification rate of 93.5% across the entire sample, and reasonably good rates of both false positive classifications (9.33%) and false negative classifications (4.8%). The underlying prediction model – built on a logistic regression – delivered strong predictive power at all conventional levels of statistical significance.

Another approach for assessing classification accuracy makes use of the (so-called) "Receiver Operating Characteristic" (or ROC) function. This function represents parametric curve that emerges as one varies the assignment rule from 0% to 100%, plotting the false positive rate (horizontal axis) against the true positive rate (vertical axis) along the way. Good classifiers will tend to "bow" towards the northwest corner of the unit square, a point that coincides with perfect classification. Consequently, the area under the curve of the ROC (sometimes called the ROC-AUC) is a common proxy for classifier performance: The closer the ROC-AUC area is to 1, the better the classifier.

Figure 3 illustrates the ROC function for our within-sample estimations. As can be seen from the Figure, the quality of the machine classifier appears exceptionally good, with a ROC-AUC measure of 0.9764.

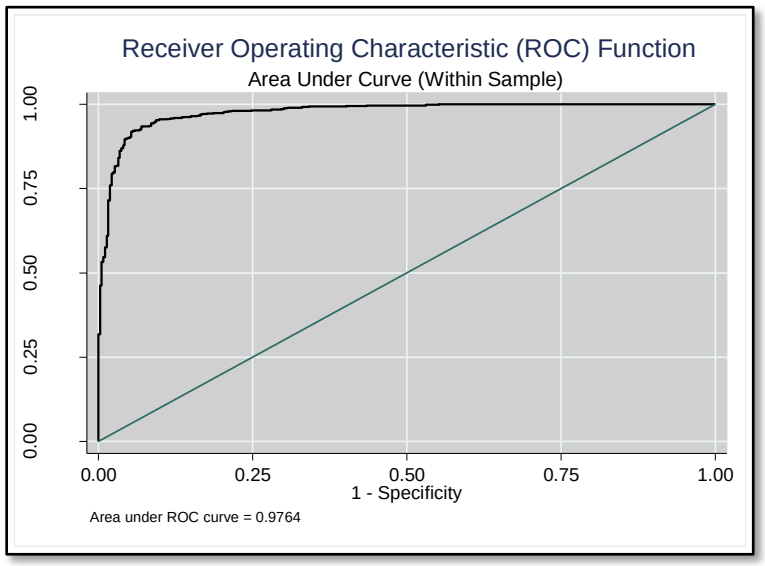


Figure 3

Although in-sample calibrations are instructive, they necessarily degrade somewhat when the predictive model is taken outside of the training sample. Nevertheless, it is predominantly out-of-sample prediction where the ML approach can be useful in economizing the time and expense of hand coding. In order to investigate our out-of-sample performance, we employed a Monte Carlo simulation approach similar to that employed in Talley & O'Kane (2012), and previously proposed by Breiman (1996) and Friedman et al. (2000; 2003). Within each iteration of the simulation, the hand-coded data were randomly segregated into two groups: A provisional

"training" dataset, consisting of roughly 75% of our observations, and a provisional "testing" dataset, consisting of the remaining 25% of the data. We then fit equation (2) to the training data, marshalling the resulting coefficient estimates to generate predictions of the presence/absence of the waiver in the testing data, and generating predictive metrics similar to those discussed above. We repeated the Monte Carlo simulation for 1,000 iterations, which produced empirical distributions for each of these metrics.

Table 5 below presents classification performance metrics from the simulations. The results are surprisingly good. Overall, while the percentage of correct classifications declined (as expected), they fell by an extremely modest one percent (from 93.5% to 92.5%), a pattern generally replicated symmetrically in both positive and negative classifications. Similarly, the area under the ROC curve declined, but again only trivially—from 0.9764 in the full sample to a mean value of 0.9727 across the Monte Carlo simulations. Moreover, the standard deviations associated with each of the MC metrics also appear notably modest – indicating precise estimation.

	MC Iter.	Mean	St. Dev.	Median	Min	Max
% Correctly Classified	1,000	92.4980	1.5123	92.4000	86.8000	96.8000
Sensitivity (50% classifier)	1,000	94.3740	1.8423	94.4828	87.3418	98.6928
Specificity (50% classifier)	1,000	89.3847	3.1513	89.5294	78.4946	98.9362
Area under ROC	1,000	0.9727	0.0086	0.9728	0.9363	0.9945

Table 5: Monte Carlo Classification Metrics, Simulated Out-of-Sample Data

A final diagnostic measure of how well the ML classifier performs is to extrapolate the calibrated within-sample model onto the full data set, thereby generating estimated probabilities of the presence/absence of a waiver disclosure over the entire universe of snippets. The frequency distribution of estimated probabilities can provide some sense of how definitively the ML classifier discriminates between positives and negatives.

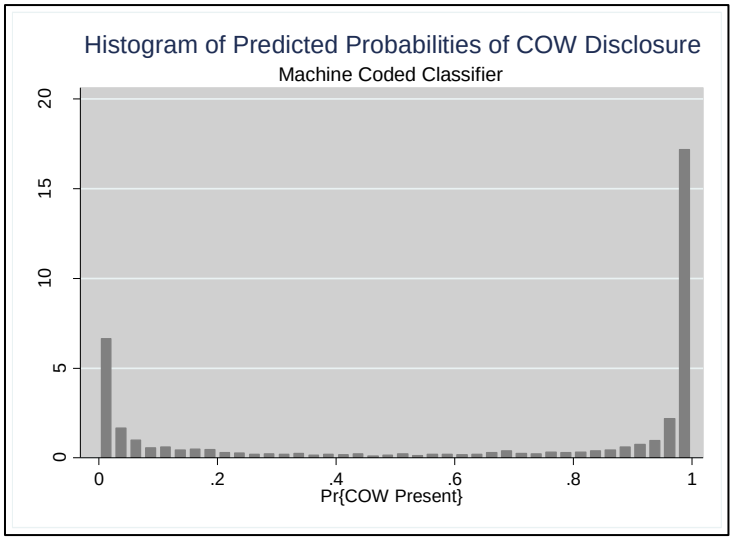


Figure 4

Figure 4 illustrates the histogram of estimated waiver probabilities extrapolated over the full data set (N=10,620). The figure does *not* correct any of the known misclassifications in the hand-coded set, and thus it accurately illustrates the noise that the ML classifier introduces. Note the strongly bimodal distribution in the empirical frequency, indicating that the ML classifier not only discriminates between likely waivers and non-waivers, but does so with some degree of statistical definitiveness.

Fortunately, the MC simulations yielded a ML classifier that performed extremely well by conventional measures. We thus deemed it unnecessary to iterate our calibrations further. That said, in many applications, the researcher must often continue to iterate on the calibration, such as by selecting a different preceptor model, adjusting the number of retained factors to optimize the out-of-sample performance, extracting 2-grams or 3-grams rather than 1-grams, or even going back to the original data set to code additional training documents by hand.

Conclusion

Once the calibrated model was extrapolated to the full data set, we proceeded with our analysis of the ML-classified waivers. That analysis can be found in the main text of Rauterberg & Talley (2017). The reader is referred there for additional discussion.

References

1. Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer Science+ Business Media LLC, New York.
2. Breiman, L. (1996), "Bagging Predictors," *Machine Learning*, 24, 123–140.
3. Friedman, J., T. Hastie, and R. Tibshirani (2000). "Additive logistic regression: a statistical view of boosting." Annals of Statistics. 28: 337-407.
4. Friedman, J., T. Hastie, & R. Tibshirani, (2003). *Elements of Statistical Learning*.
5. Goose, M., (c. 1697). "Hickory Dickory Dock," *Stories or Tales from Times Past: Tales of Mother Goose* (Charles Perrault, ed.).
6. Jolliffe, I.T. (2002), *Principal Component Analysis* (Springer-Verlag 2d ed).
7. Leskovec et al. (2014), "Mining of Massive Data Sets" (Stanford CA).
8. Rauterberg, G. & E. Talley (2017), "Contracting Out of the Fiduciary Duty of Loyalty: An Empirical Analysis of Corporate Opportunity Waivers." Forthcoming, *Columbia Law Review*, http://ssrn.com/abstract=2822248.
9. Salton, Gerard and C. Buckley, (1988), "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing & Management* 24 (5): 513–523.
10. Talley, E. & D. O'Kane (2012). "The Measure of a MAC: A Machine-Learning Protocol for Tokenizing Force Majeure Clauses in M&A Agreements, 168 *J. Inst. & Theor. Econ.* 181.