Firmly Rooted in the Data: Exploring the Use of Random Forests to Perform EDA Project Classification

Ryan Aven

November 2020

**Abstract**

This paper explores the potential application of automated classification procedures for EDA Grant Data, using a Random Forest model and Natural Language Processing techniques. By using a Term Frequency-Inverse Document Frequency weighting algorithm, the model shows significant promise in automating key areas of EDA grant review and management. The paper concludes with areas for additional projects where Machine Learning algorithms may yield significant results.

**Data used**

The data used was a comprehensive list of Fiscal Year 2020 EDA grants, as of October 9th, 2020 with an “Approved” status. This totaled 1407 awarded grants. Of the data used, the primary columns of interest were the EDA Program and Project Description columns. The EDA Program column has only 7 categories: Economic Adjustment Assistance, Planning, Public Works, Regional Innovation Strategies, Research and National Technical Assistance, Technical Assistance, and Trade Adjustment Assistance for Firms. The Project Description column is unstructured text describing the project.

**What is a Random Forest Classifier?**

The method utilized here followed a supervised Random Forest classification model, along with Natural Language Processing (covered later). Compared to other probabilistic classification algorithms, the Random Forest is considered among data scientists to be among the most powerful tools. The algorithm’s primary strengths lie in utilizing ensemble learning and resampling methods, which lend themselves to a model that is accurate and precise.

Ensemble learning refers to the use of multiple learning algorithms to obtain better predictive performance than would occur using individual predictive algorithms alone. Random Forest does this by use of multiple Decision Trees, as well as forming a “whole-model classification” based on the outcomes of individual “trees.” A key feature of Random Forest that lends to its statistical robustness is the property of Feature Randomness, where each tree has a random set of branches. Thus, this means that individual trees are not mirror image variations of one another, and some decisions are masked from individual trees. Provided a sufficiently large data set, this allows a wide array of uncorrelated sub-models, giving the “collective intelligence” of the aggregate model reduced bias and lower risk of overfitting.

|  |  |
| --- | --- |
|  |  |

Two decision trees, one with a full set of decisions, the other with decisions E and F masked. The Random Forest Model uses many variations and iterations on this process to “crowdsource” an overall model prediction.

Bagging, or bootstrapping, is a resampling method that is also used in the Random Forest Model. Using the overall sample observations, bagging draws a sample of N observations from the data with replacement. Rather than casting aside “used” observations in a discard pile, bagging “replaces” them in the available pool of observations to be redrawn for each model. This allows many more viable groups to be used in individual trees, and greatly enhances the overall precision of the model.

Together with appropriately selected variables, the features of Random Forest produce a model with high accuracy and precision through effectively relying on the wisdom of crowds. With each sub-model casting a “vote” for a respective classification, the model’s overall consensus pick is reached with typically low amounts of variance.

**So what is Natural Language Processing, anyway?**

Natural Language Processing in general refers to the idea of training a computer to understand text. It has broad application, from text translation to the algorithm underpinning search engines. In this case, it’s used to “read” the EDA project description and classify it among EDA’s project types.

But it is not necessarily easy. To turn text into meaningful insight, a certain amount of processing is done in advance. In this case, modifications were made to the project description text to assist. This includes removing single characters, special characters, converting text to lowercase, and a text normalization process called lemmatization. Lemmatization is an advanced source of word stemming, or reducing a word to its root form, and requires a special package to execute in Python.

|  |  |
| --- | --- |
| **Word** | **Lemmatized Form** |
| Drying | Dry |
| Studies | Study |
| Am, Are, Is | Be |
| Vans | Van |
| Studio’s | Studio |

Once lemmatization is applied, the text is much easier to analyze in a process known as vectorization. Vectorization works by indexing each word (e.g. assigning each lemmatized word a numerical value). For example, a sentence may look something like this:

*Raw form:*

1. The grant was awarded to the City of Dallas.
2. The grant was the large award expected.

*Lemmatized form:*

1. the grant be award to the city of dallas.
2. the grant be the large award expect

*Vectorized form:*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | the | grant | be | award | to | city | of | dallas | large | expect |
| Index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 Count | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 2 Count | 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

*Matrix form:*[2, 1, 1, 1, 1, 1, 1, 1, 0, 0  
 2, 1, 1, 1, 0, 0, 0, 0, 1, 1]

It’s important to note that in analyzing large amounts of text, the vectorized matrix could potentially contain hundreds of rows and columns, depending on the number of documents used and the breadth of vocabulary used within them. In some models, such as the “Bag of Words” method, this alone would be sufficient information to conduct textual analysis. However, words that commonly appear in analyzed text may be overweighted (e.g. assigned too much importance) and reduce the model’s ability to understand the most important words in a text classifier. It also creates a problem of sparse matrices (e.g. matrices with a lot of “0” values), reducing the computational efficiency of the model.

**Methodology**

This analysis used a Random Forest classifier, weighted with a Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a numerical statistic that determines how important a word is in a document. It is calculated as follows:

***TF-IDFt,d = Tft,d \* Idft***

*Where*

***Tft,d = (number of times a term ‘t’ appears in a document ‘d’) / (total number of terms that appear within a document)***

*And*

***Idft = log(number of documents / number of documents with term ‘t’)***

For instance, if we were to calculate TF-IDF value for the word ‘city’ in the example above would be (1/8) \* log(2) = 0.04, and the word ‘the’ would be (2/8)\*log(2/2) = 0. Since the word ‘the’ appeared in all documents, the model determined it would not be important for determining a classification and assigned it a weight of zero.

To validate the model, the data was split: 80% of data was randomly placed in a “training” subset, and the remaining 20% placed in the “test” subset. This ensured that the model would not overfit the data, and lends credibility to the model’s overall predictive ability. Furthermore, words that appeared in fewer than 4 rows and greater than 70% of the data set were ignored for purposes of this analysis.

**Results**

The model achieved an overall 91% accuracy rate among the seven EDA programs, and a weighted precision score of 91%. This means that overall, the model was right 9 out of 10 times when it made a prediction, and that the predicted number of grants in one category was close to the actual number (meaning a low false positive rate). In more common programs such as Economic Adjustment Assistance and Planning, the model was 99% and 95% accurate, and in several program areas the model achieved 100% precision.

There is reason to believe that if anything, this is a baseline figure for the true accuracy the model, as the training and test data sets were fairly small. Evidence suggests that accuracy increased with additional observations, and in certain program areas the limited size of the data set meant that relatively few observations were used to train. Increasing the data set to multiple years would almost certainly enhance the predictive capabilities of the model and lead to better results.

**Applications for future use**

The application of this type of modeling in EDA has significant potential in at least a couple of areas.

1. *Special Initiative Coding*

A Random Forest or similar classifier (potentially paired with some hard-coded rules) could assist Investment Review Committees in assigning Special Initiative Codes (SICs) to projects. It could use fields such as the Project Description, Grantee Name, Program Area and others to make “recommendations” for SICs in EDA’s grant processing systems. More advanced algorithms would allow for selective coding – e.g. only making recommendations where the model is 90% certain or more. A human reviewer would still then make any alterations and approve the SICs chosen. Overall, this could result in savings of hundreds of staff hours annually, as review committees would no longer be required to manually review and add SICs in common areas.

1. *Assessing Risk within the Revolving Loan Fund (RLF) Program*  
   A historically underleveraged dataset, the RLF Program has information about recipients’ credit, loan term, loan amounts, interest rates, geographic information, and other descriptors. Paired with economic data such as local unemployment rates or market analysis, it may be possible to assess the likelihood of default on RLF loans and conduct enhanced monitoring and evaluation of the program, as well as develop data-driven policies to program administration. This has been a longstanding practice within financial institutions, and more recently startups such as Upstart have deployed enhanced Machine Learning algorithms to better assess the likelihood of loan defaults or write offs.

**Conclusions**

Using the project description alone, the Random Forest classifier weighted by a TF-IDF Natural Language Processing technique showed overwhelming accuracy in predicting over 9 out of 10 EDA programs. More significantly, Machine Learning classification algorithms hold significant promise for several key EDA grant management processes, demonstrated by this algorithm’s success in identifying EDA project area. Enhanced by decades of additional grant records, reports, and other supplementary materials provided during grantee application and review phases, classification algorithms hold significant promise in automating much of the day-to-day grant application review.

Appendix

1. <https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/>
2. <https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76>
3. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
5. <https://machinelearningmastery.com/ensemble-methods-for-deep-learning-neural-networks/>
6. <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html>