Data Oriented Proposal Engine (DOPE)

Final Capstone Report

Produced by:

Jacob Chambon

Jack Harmon

Kalev Jaakson

Mehmet Fatih Ogut

Georgetown University

School *of* Continuing Studies

Professional Certificate – Data Science

Fall 2019 – Cohort A

Table of Contents

Table of Tables

Table of Figures

1.0 Abstract

The United States Government (USG) spends over $164B, on millions of contracts, to thousands or contractors, annually on services contracts. Government contracting is a very competitive business, and contracting organizations are required to bid (i.e., submit a proposal) for nearly all government service contracts. Set-asides ensure fair opportunities for small businesses and unique socio-economic programs. Government contracting officers prefer to use small business set-asides whenever possible to help the federal government meet its small business contracting goals.

Responding to proposals is labor intensive and requires time and resources. Any undetermined or unannounced intel for an upcoming government opportunity provides a competitive advantage for interested parties and helps allocate time and resources effectively.

2.0 Problem Statement/Hypotheses

The ability to predict if an upcoming contract opportunity will have a set-aside, and if so, what type of set-aside will provide a competitive advantage to two types of potential bidders, including:

- Organizations, typically small businesses, who are targets of a set-aside
- Organizations, typically large businesses, who want to avoid committing time and resources to an opportunity that they will not be able to compete on due to a set-aside restriction

3.0 Applications/Motivations

Set-aside restrictions are not always immediately announced and posted with upcoming opportunities since the government needs to complete a series of market research activities to ensure that there is a sufficient number of small businesses that are able to perform the work and promote competition. Being able to predict this information ahead of time will help organizations and employees, especially sales roles, prioritize and focus their business development efforts.

It is important for the predicting model to produce accurate results because incorrectly predicting a set-aside could deprioritize, or filter out, a potentially viable opportunity for an organization to pursue. In the example of a small business attempting to identify/prioritize opportunities that have historically been restricted to their set-aside, a model that predicts more false negative results than false positives can have negative consequences as the model would be deprioritizing more potentially viable opportunities. However, if the model predicts more opportunities will have the set-aside, but ultimately has no set-aside, the small business wouldn't have missed the opportunity. The same concept applies to a large business attempting to identify opportunities with no set-aside.

4.0 Tasks

The following subsections outline the team's approach and activities within each phase of the data science pipeline and perform the analysis.

4.1. Ingestion

The team leveraged data from USA Spending (usaspending.gov) to obtain information on historical contracts and the winners (i.e., awardees) of those contracts. The data elements in USA Spending fall into two categories: contract data (e.g., awarding agency, funding agency, contract value, product or services (PSC) code, type of set-aside) and awardee data (e.g., company locale, size, socio-economics).

Data was downloaded from USA Spending via a web application. The following steps were taken to filter the data to four years of historical Department of Defense (DoD) contracts and awardees:

1. Filtered on Department of Defense (DoD) contract awards from 2014-2017 and downloaded the data locally as a CSV file
2. Read the CSV files into Pandas dataframes and uploaded to a shared PostgreSQL database on the Linux Ubuntu server
3. Leveraged Pyscopg2 as a database API to read data between PostgreSQL and Python

Table 1 below provides a breakdown of the data ingested into the PostgreSQL database by year.

***Table 1**: Complete Data Download*

Year	# of rows	# of columns	Disk Space Required
2014	1,348,146	276	2.10 GB
2015	3,207,054	276	4.98 GB
2016	3,667,714	276	5.83 GB
2017	3,669,339	276	4.00 GB
Sum	**11,892,253**	**276**	**16.91 GB**

4.2. Wrangling

PostgreSQL and its PgAdmin graphical user interface (GUI) were used to consolidate and wrangle multiple years of USA Spending data. Initial filters were applied to the data set to create a representative and meaningful data set for government contractors competing on service-based contracts. There are two main categories of government contracts: product- and service-based contracts. For the purposes of this effort, the team focused on service-based contracts, such as consulting efforts. The data used for exploratory data analysis (EDA) included the following filters:

- Action (i.e., Contract) Year = 2015-2016
- Modification Number = 0; retains only original contract information and removes instances pertaining to contract modifications or options
- Primary Place of Performance = USA; retains only US-based opportunities
- Product or Service Type = Service; retains only service-based contracts

Initial munging and wrangling efforts provided a more manageable data set to work with during the following analysis phases. More importantly, it created a more targeted/focused data set with only relevant instances such as service-based, DoD contracts.

4.3. Computation

The subject of this project's EDA effort is a raw 1,000,000 row x 276 column data set. The approach for exploring the data is domain agnostic; so little to no prior knowledge of the data set and its structure is assumed. There are three main steps:

1. An initial data scrub eliminates columns of no or little value. The question being answered in this step is not which columns are important, but rather which columns are certainly not important or redundant. To that end, column headers that contain the string "code" were eliminated, because each had a non-"code"-containing counterpart whose values mapped very close to one-to-one. Other columns to eliminate were columns that could only have been added after the award winner was announced. This information will not be available in our eventual test set, so it should be eliminated. The inclusion of the strings "recipient" and "business" in the header were used to identify these targets for removal.

2. Creating a summary table of the edited raw data set. The summary table cites the number of non-nulls, the number of unique values, and the data type in each column. These facts about the rows remaining in the main data set are useful to start understanding the format and content.

3. Filtering or grouping column headers by the three columns in the summary table will facilitate inspecting smaller tables and maybe highlight similarities or additional avenues to explore. Grouping first by data type (of which there are only three, int64, float64, and object) and then eliminating columns that exceed 20% nulls produced a few insights:

- Most of the data in the float64 columns are monetary
- The object group is the largest (141 columns), which makes sense, given that our data set contains mostly categorical data
- Of the 141 object columns, 73 are binary, and most of those are similarly to the "recipient"-containing columns above, in that they could only have been added after the award winner was announced
- The remaining 68 non-binary object columns have a range of unique value counts that would make one hot encoding (OHE) for a classification problem difficult

A cumulative distribution function (CDF) of the unique value counts (ordered from largest to smallest) of one or more columns may be a useful tool in understanding how the data in a column of categorical value is distributed, allowing bucketing in a more informed way. The CDF plot is displayed in **Figure 1**.

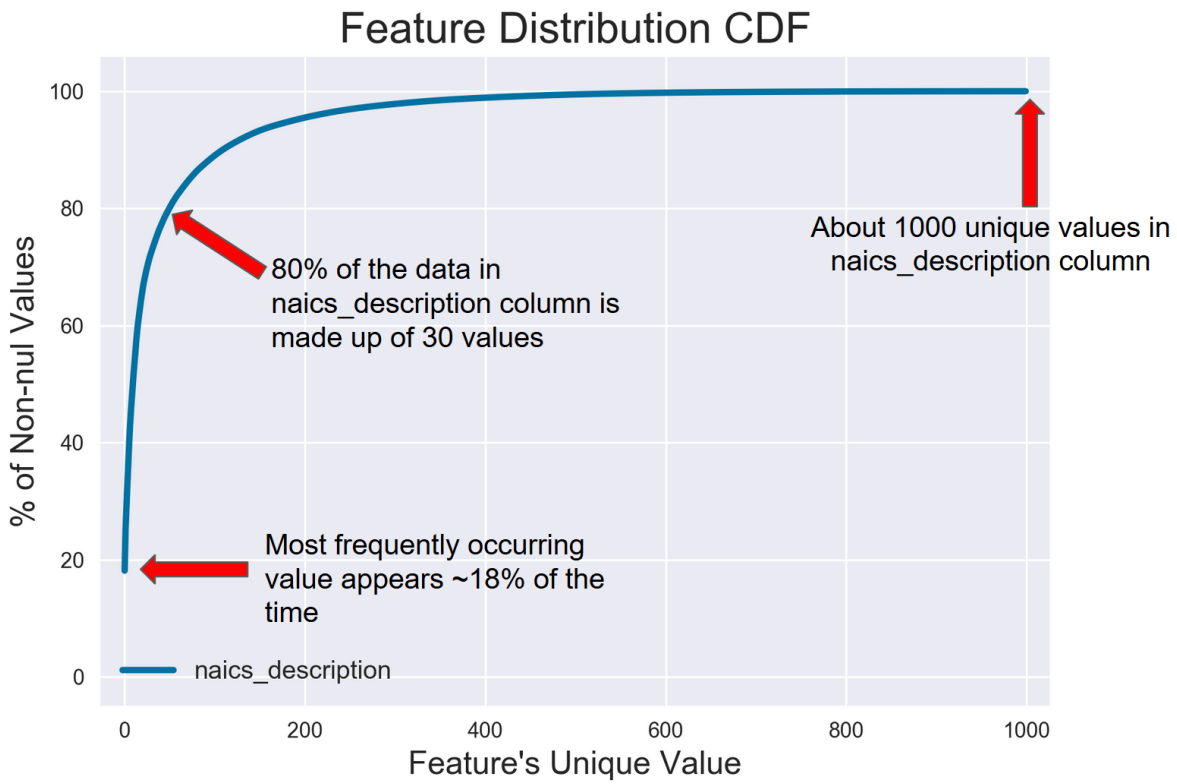

Figure 1: *Feature Distribution CDF*

4.4. Modeling

Following ingestion, wrangling, and computation, a series of models were applied to the data set to determine which machine learning model and methodology was best suited for the problem and data set. Research indicated that Random Forest is a good starting point for classification problems with high dimensional categorical data. The advantages of Random Forest include:

- Ability to run efficiently with large data sets
- Handles large amounts of input variables - need to OHE all categorical data
- Provides estimates on feature importance

A number of classification models were tested for comparison, including Random Forest, KNearest Neighbors (KNN), Naive Bayes, and Support-vector Machine (SVM). **Table 2** below shows the results of the model comparison.

Table 2: *Classification Model Comparison*

Model	Precision	Recall	F1 Score
Random Forest	0.90	0.89	0.89
SVM	0.68	0.67	0.67
KNN	0.66	0.42	0.51
Naive Bayes	0.69	0.74	0.71

The Random Forest Classifier proved to be the most effective. Therefore, the team proceeded with Random Forest and did further tuning to increase the performance of the model.

Feature importance was analyzed leveraging Scikit-learn's feature importance function to reduce features and improve model performance. After one hot encoding (OHE) the initial data set, the feature set included over three thousand features. Using the feature importance function, a pareto (or cumulative total) of importance as calculated. The features that accounted for the top 80% of importance were kept and deemed important/relevant. The resulting OHE feature set included just over 300 features, ***which is about a 90% reduction in features***. Most importantly, there was a negligible decrease in model accuracy after feature selection (***less than 1% reduction in accuracy***).

The team also noticed a class imbalance in the dataset because significantly more contracts did **not** have a set-aside than contracts that did. To account for this issue, the model included two phases. The first phase predicted whether or not the contract had a set-aside, yes or no. If the model predicted yes, than the contract was ran through the second phase to predict the actual type of set-aside.

Once the model was tuned, trained, and tested, the team applied the model to a new set of unknown 2017 contracts. The ***model predicted the type of set-aside for a sample of 2017 data at an accuracy of approximately 85%***, proving the effectiveness and value of the application for future opportunities.

4.5. Visualization

Initial modeling was performed using Random Forest. First, an N-Estimators calculation was to determine the most appropriate number of decision trees for the model. **Figure 2** shows the N-Estimators hyperparameter tuning exercise. The team used N-Estimators = 17 for the following models.

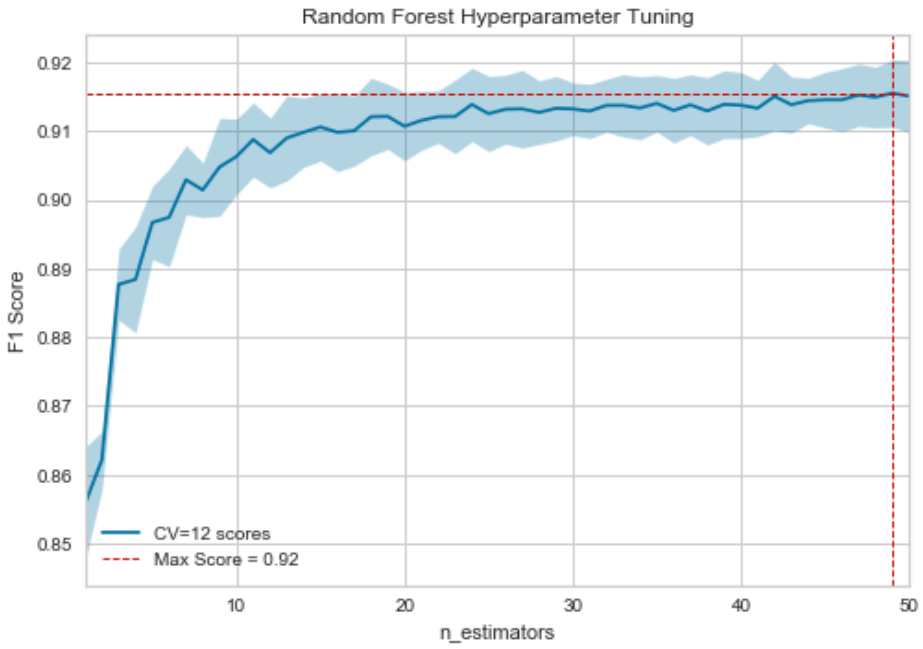

Figure 2: *Random Forest Hyperparameter Tuning (N-Estimators)*

The initial Random Forest model predicted whether or not the contract had a set-aside at approximately 90% accuracy. **Figure 3** below highlights the performance metrics (precision, recall, F1, and support) of the first and initial Random Forest model.

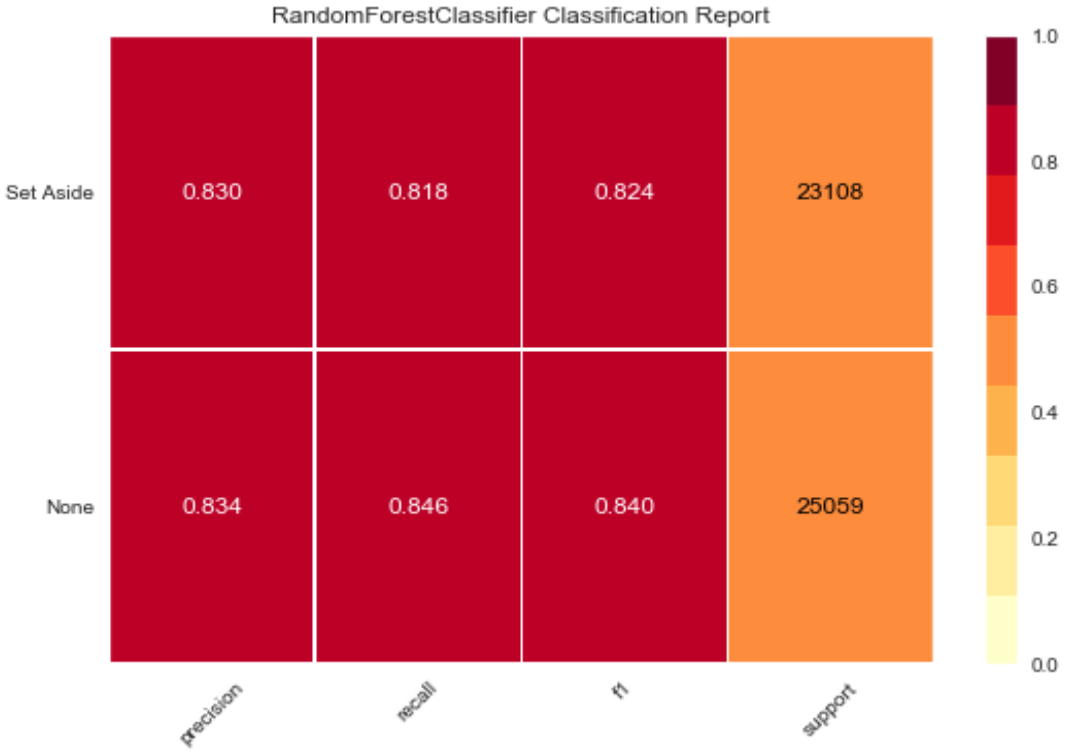

Figure 3*: Initial Random Forest Performance Metrics*

Next, the KNN, Naive Bayes, and SVM models were tested for comparison. Heat maps of the performance metrics for the KNN and Naive Bayes models are provided in **Figure 4** and **Figure 5**.

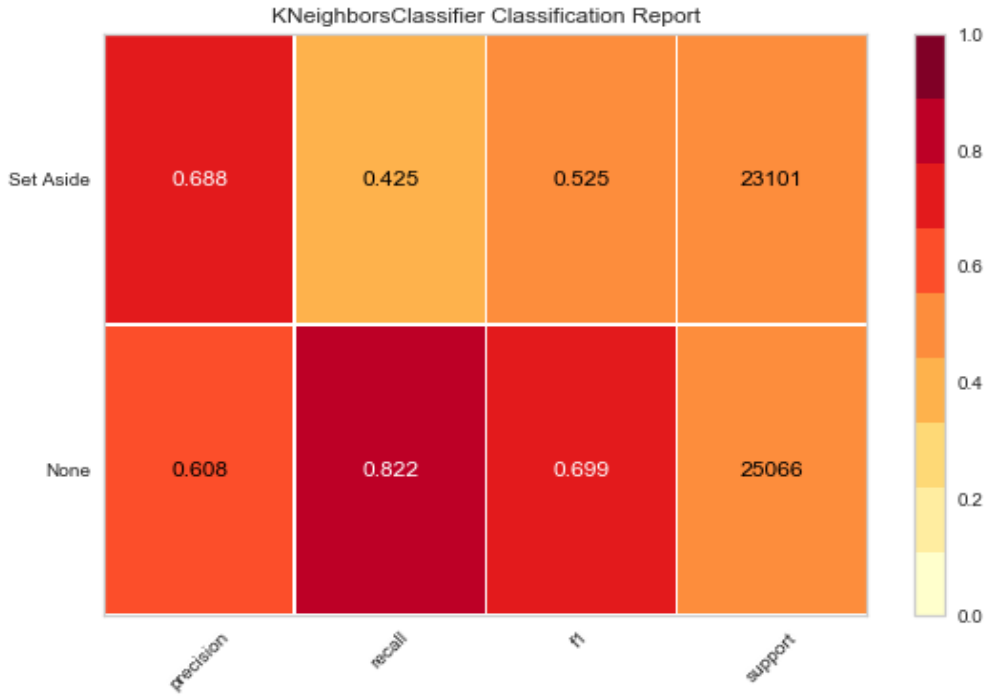

Figure 4: *KNN Performance Metrics*

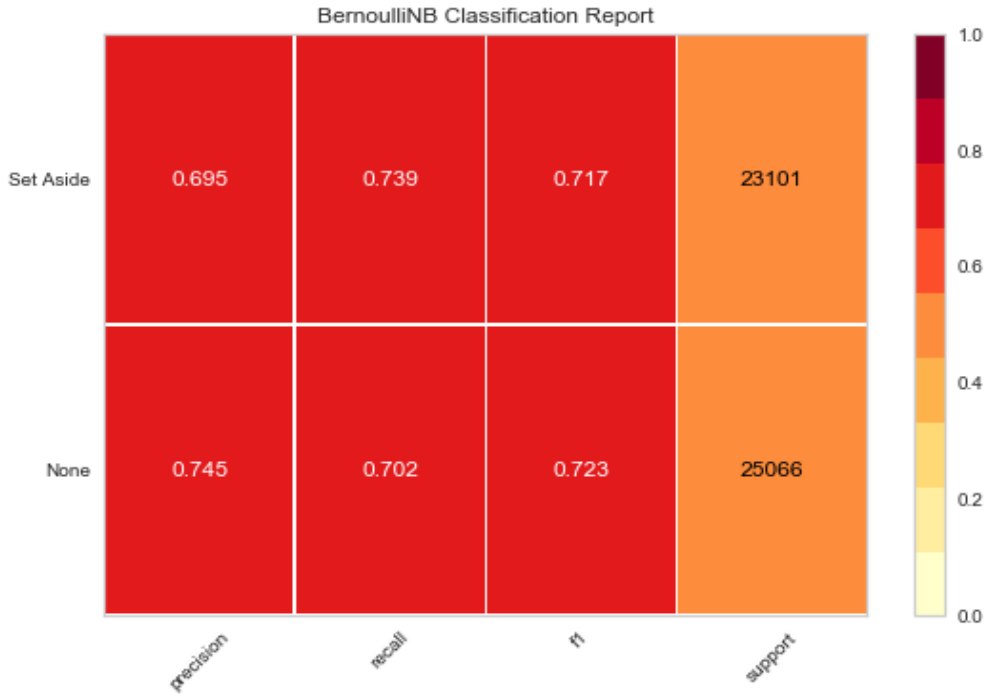

Figure 5*: Naive Bayes Performance Metrics*

As shown by the performance metrics between the various models, Random Forest was the most effective model. Once the Random Forest model was selected, feature analysis was performed as outlined in **Section 4.4**. **Figure 6** provides a sample of the top five features derived from the feature importance analysis.

Feature	Importance	Cumulative Total
contract_value	0.12	12%
dod_claimant_program_description_CONSTRUCTION	0.03	15%
naics_description_COMMERCIAL AND INSTITUTIONAL BUILDING CONSTRUCTION	0.02	17%
portfolio_group_Electronic & Communication Services	0.01	18%
portfolio_group_Facility Related Services	0.01	20%

Figure 6: *Sample of Top Five Features from Feature Importance Analysis*

Finally, the Random Forest Classifier was re-fitted with only the updated important features. **Figure 7** provides performance metrics of the final Random Forest model.

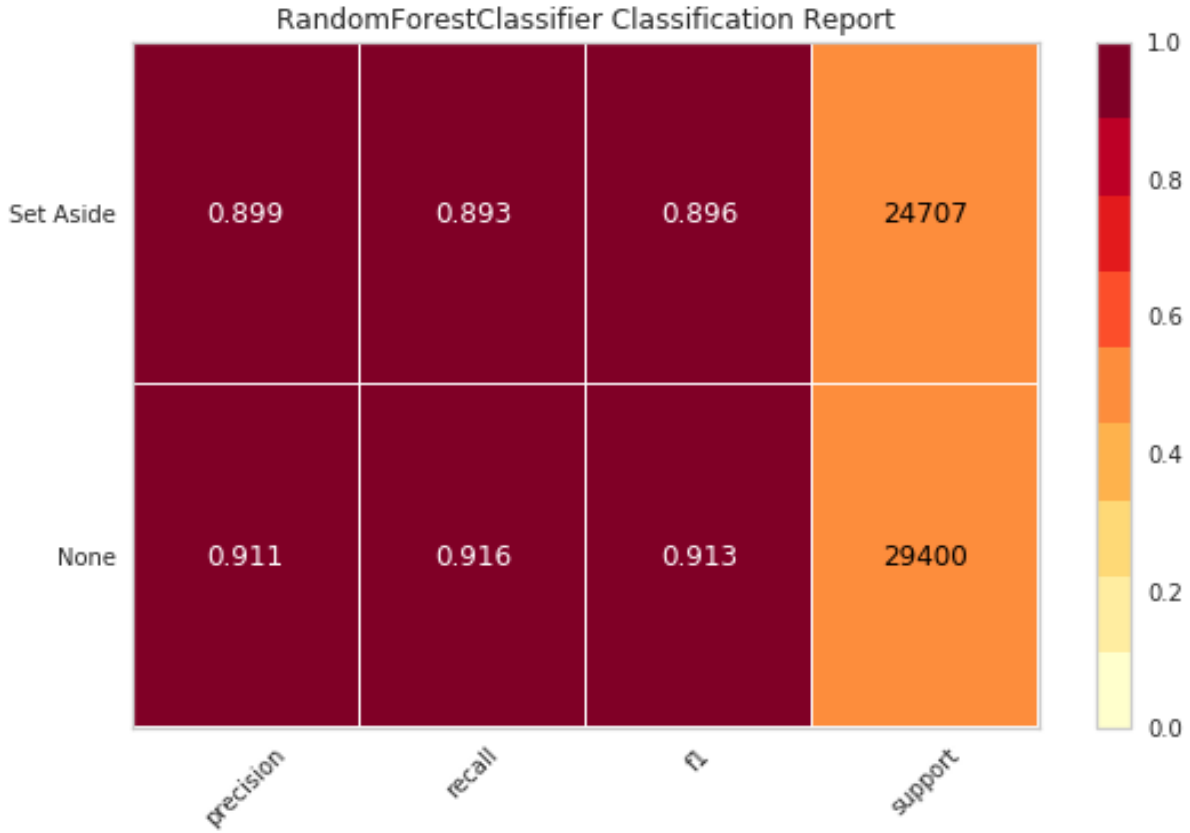

Figure 7: *Final Random Forest Performance Metrics*

As also covered in **Section 4.4**, there was a class imbalance in the data set since significantly more contracts did not have a set-aside versus contracts that did. **Figure 8** is a cumulative CDF plot identifying that the no set-asides accounted for approximately 55% of the contracts.

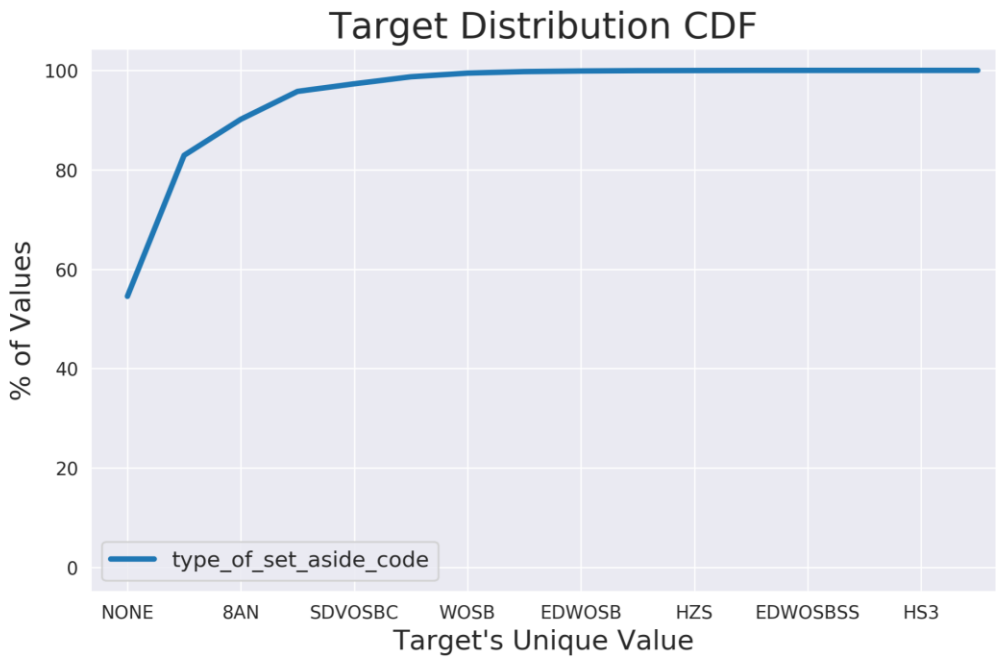

Figure 8: *Target Distribution CDF of Type of Set-Aside Codes*

After the class imbalance was handled using the approach covered in **Section 4.4** (first predicting if there is a set-aside, yes or no, then if yes, predicting the type of set-aside), the final Random Forest Classifier was tested on its ability to predict the type of set-aside. The performance metrics are summarized in **Figure 9**.

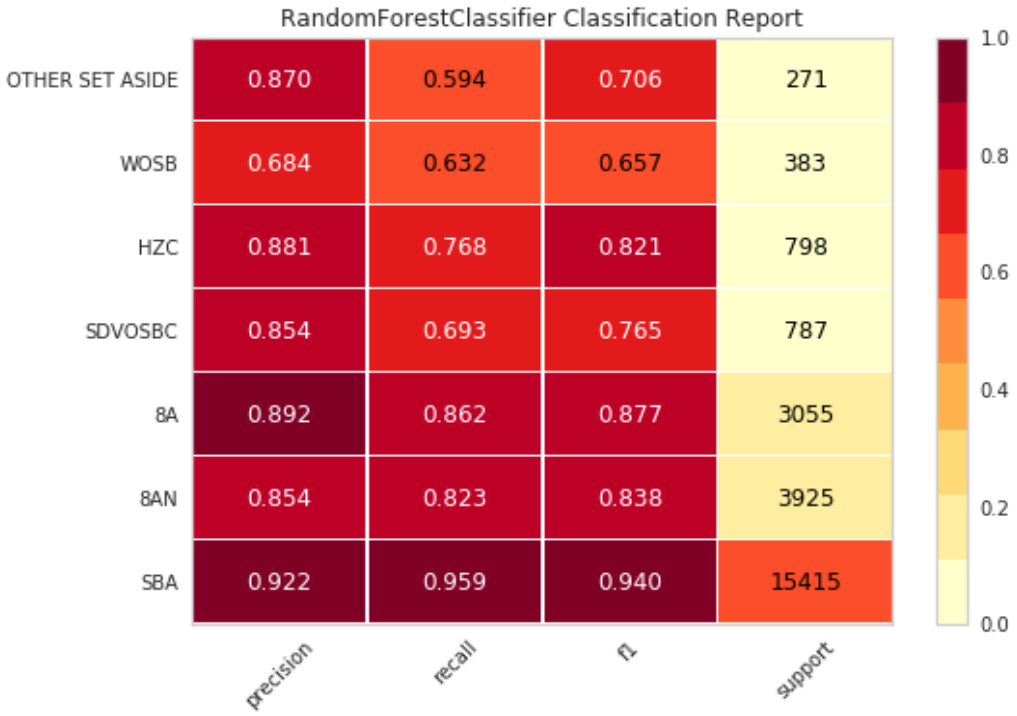

Figure 9: *Final Random Forest Performance Metrics for Predicting Type of Set-Aside*

5.0 Methodology

A collaborative working environment was set-up to increase efficiency and help maintain version control of documents and work products. The environment was hosted on a Hyper-V virtual machine. The steps for setting up the environment included:

- Installed Linux Ubuntu 18.04 Server OS hosted on a Hyper-V Virtual Machine
- Purchased a domain on GoDaddy (dopelytics.site)
- Used PostgreSQL for data storage
- Hosted Jupyterhub on the server for collaboration (dopelytics.site:8084)
- Used Bitvise Client for FTP and SSH
- Used git for version control and github as a cloud repository

The team's approach to completing this project was focused on the main activities of the data science pipeline. A high-level overview of the approach and activities is provided below:

1. ***Ingestion*** - stored data in Postgres on the Ubuntu server and used python to read and write to the database
2. ***Wrangling*** - consolidated multiple years of data into a representative data set for training and testing
3. ***Computation*** - removed non-value added features and cleaned duplictave/redundant information
4. ***Modeling*** - created random forest classifier to predict type of set-aside and performed feature analysis to improve model performance
5. ***Visualization*** - created dopelytics.site to interact with the data and predict target opportunities by set-aside and contract value

One area of opportunity for this effort is additional feature analysis to reduce the number of features even further. A reduced number of features would make the model more broadly applicable to future data sets of upcoming opportunities as contract related information, requirements, descriptions, etc. evolve.

6.0 Final Product

The final data product and GUI for this project is a Flask application hosted on the same server as the PostgreSQL database. The application has multiple pages providing background on the project, including the problem problem statement and instructions on how to utilize the tool. The 'Input page' allows for the end user to select a set-aside applicable to their organization and run a search on predicted contracts under that set aside. The user can further refine the search by a particular PSC code. The Data Oriented Proposal Engine will then query the Postgres database using the given filters and return the 10 contracts with the highest contract value.

The product is hosted on an Ubuntu 18.04 server OS on a Hyper-V virtual machine. Hyper-V is hosted on a custom built server with 64gb of RAM, 16 cores, and 2TB of dedicated memory. A static port for the server was created and assigned a domain, dopelytics.site, to the public IP address of the network. The result is an application available to the public without the need to download, configure, and run our code.

7.0 Conclusion

The Random Forest Classifier proved to be the most accurate model for predicting the type of set-asides for upcoming government opportunities. The effectiveness of the random forest model is largely due to its ability to run efficiently with large data sets and handle a large amount of input variables. Since most of the data elements in USA Spending are categorical, the team had to OHE most of inputs creating thousands of input variables.

Some of the key takeaways from this project include:

- Wrangling is difficult and takes a significant amount of time
- Feature analysis is difficult for largely categorical data sets, and feature importance/elimination is very important for making an effective model
- Using a centralized server, database, and working directory provided effective collaboration amongst team members

To continue to build out the capabilities and applications for this effort, the next steps are as follows:

- Leverage existing code to solve for additional targets of advantage that can provide competitive advantages, such as number of offers received
- Perform additional feature analysis to reduce the number of features, making the model more applicable/adaptable to evolving contract information
- Expand on the interactive features of the dopelytics.site page to provide a more exploration opportunities and insights for users

8.0 Github Location

The location on Github of the code related to the project:

https://github.com/georgetown-analytics/Data-Oriented-Proposal-Engine