Using Random Forest Classifier

to Analyze JMU Advancement (Constituent & Fundraising) Data

and Identify Potential First-Time Donors

Renee [last name redacted for publishing on BecomingADataScienctist.com blog]

ODU ECE 607 Machine Learning

Final Project

Spring 2014

Random Forest

Random Forest is the name for an "ensemble learning" method of classifying data via machine learning. It is called an ensemble learning method because it involves the automatic construction of many decision trees, each trained separately, whose results are then combined and the most common classification among all of the trees becomes the selected classification for a given data point. [1] In the early 1990s, it was discovered that combining many "weak" decision tree learners in this way resulted in a more accurate classification scheme than any one "strong" decision tree could generate independently. [2]

Random Forest can also solve regression and probability density tasks, but in this project, it was used for classification.

Problem

James Madison University stores many details about its constituents (alumni, students, parents, etc.) and their charitable giving to the University. One challenge for the Office of Annual Giving is deciding whom to target with solicitations – in other words, determining which populations are most likely to make a donation, so mailing and calling efforts can be targeted to a smaller audience for a higher return on investment.

For this project, I decided to look at one fiscal year (FY2013, July 2012 – June 2013) and compare those constituents that had been in the database prior to the fiscal year starting, but have never made a gift to the university ("Never Givers") to those that had been in the database prior to the fiscal year starting and gave their first gift in FY2013 ("First Time Donors"). Since there is a large pool of Never Givers, and it would cost a lot to contact all of them by anything other than email, it would be valuable to know who from that pool is best to target for solicitations. I hoped to identify what factors may indicate that a person is likely to give for the first time in a fiscal year.

The fields in my final data set include:

- Classification (1 = First Time Donor, 0 = Never Giver)
- Current Record Type: Alumni, Student, Parent, Friend, etc. (as of FY14)
- Preferred Class Year
- Preferred School: College of Business, College of Science & Math, etc.
- "OK to Contact" indicators: OKtoMail, OKtoEmail, OKtoCall
- Estimated Age as of FY13 (Actual age if we have birthdate. Otherwise, based on class year for alumni.)
- Assigned to Development Officer (0 = unassigned, 1 = assigned)
- 5-digit zip code (used as continuous numeric value)
- First digit of zip code, indicating region (used as category label)

- Distance from Harrisonburg, VA (location of JMU) in miles – based on current zip
- Has Preferred Address (1 = yes, 0 = no)
- Has Business Address
- Has Phone Number
- Has Email
- Count of Non-Solicitation Appeals in FY12-13 (newsletters, etc. until first gift date if donor, through FY13 if not)
- Count of Solicitation Appeals in FY12-13 (until first gift date if donor)
- Count of Non-Solicitation Appeals from organization other than Office of Annual Giving in FY12-13 (newsletters, etc. until first gift date if donor)
- Count of Solicitation Appeals from organization other than Office of Annual Giving in FY12-13 (until first gift date if donor)
- Count of Non-Solicitation Appeals in lifetime
- Count of Solicitation Appeals in lifetime
- Years since added to database (max 12 when previous system implemented)
- Years since record modified (null if modified after FY13, only have latest modification date and not historical record)
- Count of Events attended
- Recent or Current Parent
- Has Ever been a JMU Employee

Note that many of these data points are not independent, such as class year and age, which have a close relationship for alumni record types.

Some of the data is incomplete, such as appeal counts – we do not have a record of newsletters that colleges sent out without notifying us to add those to our centralized database, for instance.

Also, the data set is very imbalanced. There are over 140,000 records of non-donors, and just over 2,000 records of first-time donors in fiscal year 2013.

I attempted to use only information that would be available prior to the gift being made when possible, though values like the primary record type use the current value since it is not easy to get the historical primary record type at a given date. Though the appeal that triggered the gift (phone call from telefund, mailing from college, etc.) , and the gift amount and allocation given to would be important to those viewing the results of this study, those are not appropriate columns to pull into the dataset since they could not be used to classify first time donors vs non donors prior to the gift being made. (Though the particular appeal could be the reason a person gave for the first time.)

Preliminary Pre-Processing and Random Forest Training

I first pulled a subset of data to try out with the Random Forest package in scikit-Learn. [3] The first task was to "pre-process" the data to get it into a form that the Random Forest classifier could use.

I imported the columns from a CSV file and combine them into sample and target datasets, then started manipulating the data. I didn't want to get rid of the rows that were missing zip codes or ages, so I converted those with a missing "zip1" column into 10 (since the "real" data would have values of 1 to 9 for the first digit of the zip code). I filled in the "miles from Harrisonburg" column with 9999 if the value was missing. I converted the age to 0 if it was missing a value, which of course would skew any statistics run on that column, but I hoped would not confuse the classifier too much, and would prove more valuable than removing the rows of people without age values, which would often be non-alumni donors. The yes/no (1/0) columns I imported were OKtoMail, OKtoEmail, and OKtoCall, (thinking that if someone chose not to be contacted by JMU, that could be an indicator of not wanting to give to the university), and Assigned to Development Officer.

I then made a copy of the dataset and split it into two arrays, one for each class (never givers and first time donors), in order to calculate the results by classification. I used the scikit-Learn (skLearn) Cross Validation approach to split the full dataset into training and testing sets, reserving 25% of the data to test.

Then, I implemented the skLearn Random Forest Classifier, which turned out to be the easiest part of the process so far! I used the Random Forest "score" function to determine how well the Random Forest classified the data, and my initial result looked great: 98.6% of the data had been classified correctly!

```
#build cross-validation data sets
from sklearn.cross_validation import train_test_split
sample_train, sample_test, target_train, target_test = train_test_split(sample, target,
test_size=0.20)
#train the random forest classifier
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators = 50)
forest = forest.fit(sample_train, target_train)

#test the model on various sets
trnresult = forest.score(sample_train,target_train)
tstresult = forest.score(sample_test,target_test)
class0result = forest.score(class0data,class0target)
class1result = forest.score(class1data,class1target)
```

Then, I calculated the number of points in each class and the results of the classifier for each, and found that there were 141,504 never givers in my class 0, which had a 99.9% correct classification, and only 2,300 first time donors in my class 1, which was only classified correctly 16.0% of the time. So basically my dataset was so imbalanced, I could have just classified all data points as Never Givers and still have classified 98% correct overall!

This is when I started reading about how to treat imbalanced data sets in order to improve the results and learned that some common methods were "upsampling" the smaller class (creating copies of the records so they were more likely to appear in each iteration of the classifier training), or

"downsampling" the larger class by selecting a random subset of the class so the representation was more even for training. [4] There were other approaches, but these two appeared to be widely accepted, and I decided to try downsampling. I selected 6500 random rows from the never givers class, so it wasn't more than 3 times larger than the first time donors class, and tried the classifier again.

This time, the overall score was worse at 87.6% of points correctly classified, but the important class of First Time Donors (for which I was trying to identify traits) was now classified correctly more than half the time at 55.6%.

I then tried variations of the preprocessing such as removing the rows with null values in the age and zip code columns (which actually made the outcome slightly worse), and removing the Never Giver records which had been in our database prior to the last 2 data migrations and were more than 11 years old, which also didn't provide much of an improvement (and was still strongly imbalanced). I decided the best approach to move forward was to start bringing in more columns that I had intended to use, now that I had made a functioning classifier.

Next Round of Pre-Processing: Importing categorical data columns

The next thing I wanted to learn how to do was to use some categorical data columns in the classifier. For example, some important categorical data in JMU's database includes primary record type (whether the constituent is an Alumni, Employee, Parent, etc. – only one value per entity with a trumping order), which is stored as a string code such as 'AL'.

I learned how to use the skLearn LabelEncoder and OneHotEncoder preprocessing functions [5] to handle this categorical data. First, the LabelEncoder determines how many possible labels exist in a column (all of the record types, 10 in this case), and converts them to integer labels (0-9). Then, the OneHotEncoder function converts those labels to a sparse array with one column indicating each possible value ([1 0 0 0 0 0 0 0 0 0] for 1), then I appended that data to my data set so the columns could be used by the Random Forest Classifier.

```
#convert record type string category to int, then to 1 of k encoding, then add to existing
list
labelenc = preprocessing.LabelEncoder()
rec_num = labelenc.fit_transform(rec_type)
print(labelenc.inverse_transform(np.array([0,1,2,3,4,5,6,7,8,9])))
#convert to list of lists, need each item as list for next step to work
rec_num =  [[x] for x in rec_num]
enc = preprocessing.OneHotEncoder()
rec = enc.fit_transform(rec_num).toarray()
sample = np.append(sample,rec,1)
```

Once my classifier was able to use the Primary Record Type and Preferred School categorical data, as well as additional imported yes/no columns Number of Solicitations (lifetime), Number of Non-Solicitation Appeals (lifetime), Number of Solicitations (FY12-13), Number of Non-Solicitation Appeals

(FY12-13), Has Preferred Address, Has Business Address, Has Phone Number, Has Email, the performance improved drastically.

Here is a sample of a data row for a Never Giver:
[48 0 22655 55.7 9 6 23 15 5.49 1 0 1 1 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.0 0.0 0.0] 0

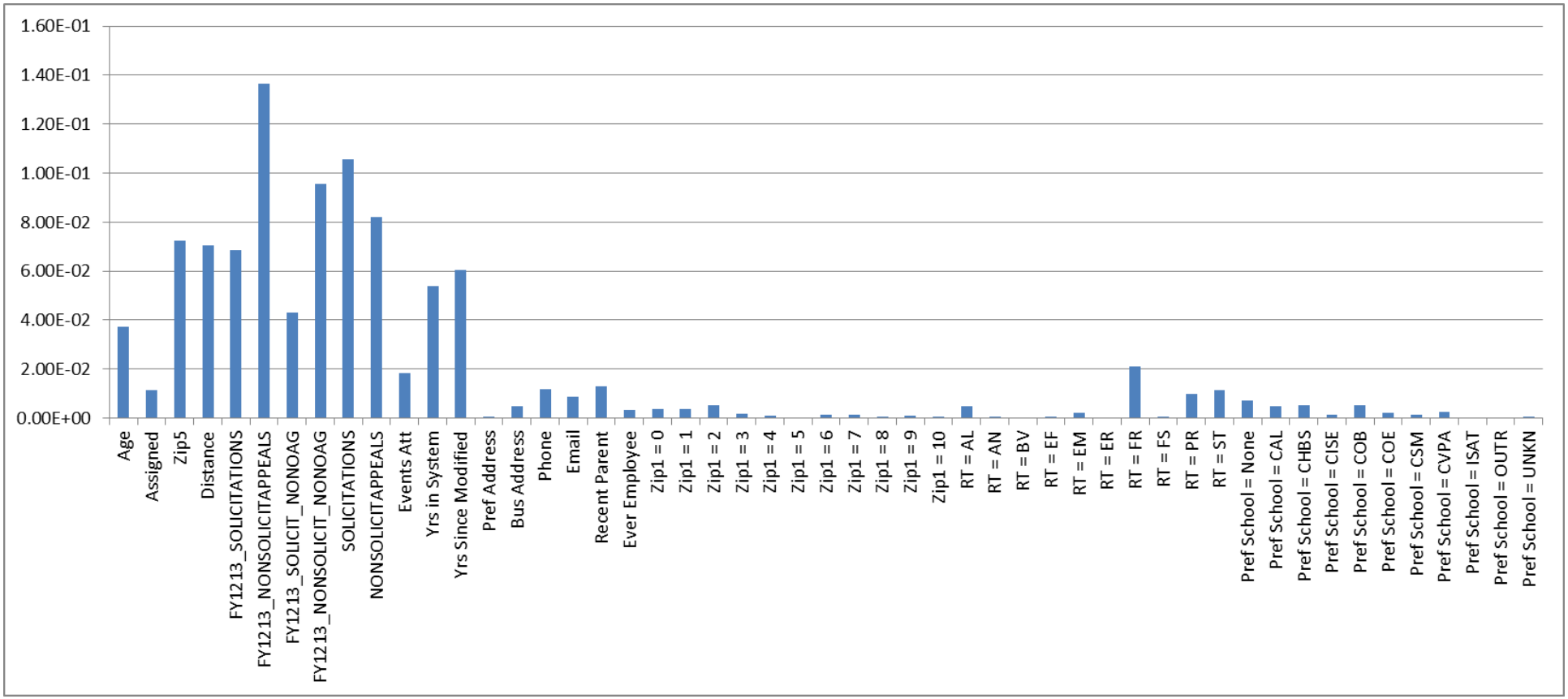
And a First Time Donor:
[29 0 23517 174.89 10 12 35 49 8.14 1 0 0 1 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 1.0 0.0 0.0 0.0] 1

I was then able to leave in the imbalanced large number of never givers, only removing the records that had been in the database more than 11 years prior to FY13 (still keeping over 106,000 never givers and only about 2,000 first time donors), and the classifier was now classifying 87.9% of the first time donors correctly, and 99.9% of the points overall.

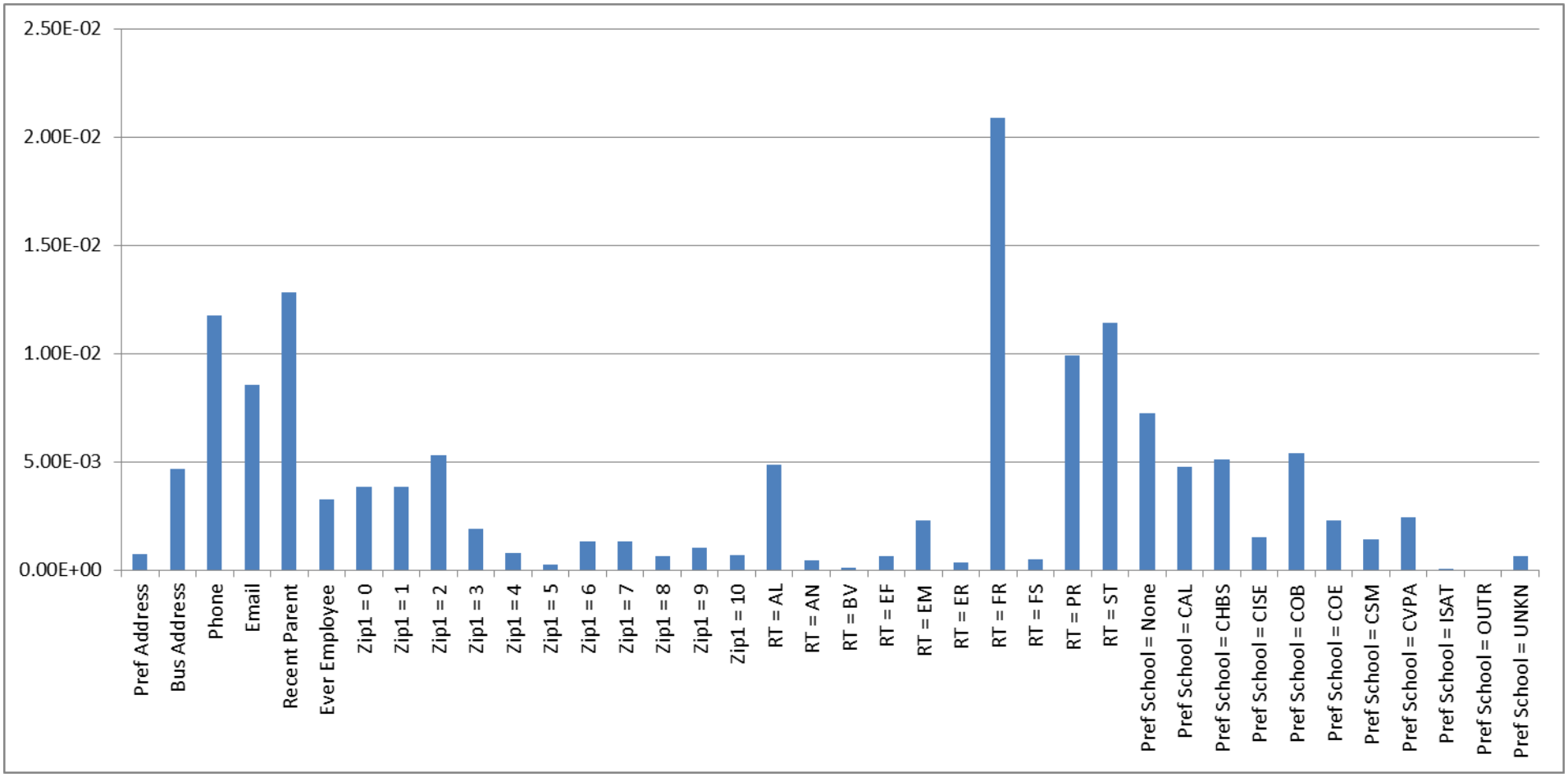
I used the feature_importance option in skLearn Random Forest to output the feature weights, and found that at this point, the number of Solicitations, the distance from JMU, and the length of time the record had been in the database were the strongest differentiators between the classes.

This led me to remove the OKtoMail, OKtoCall, and OKtoEmail columns, because their importances were negligible. My guess is that so many people in both classes have all of these values set to “Yes” that it wasn’t an effective differentiator.

The remaining feature importances are shown in the following figures.



Feature importance after importing categorical columns. The FY12-13 Solicitation/Appeal counts stood out as strongest differentiators, while the categorical data on zip1 (region), record type (alumni, parent, etc.), and college categories had seemingly little effect.



"Zooming in" on just the categorical data, it appears that "Record Type = Friend" has the strongest impact. That makes sense because people that make a gift to the university and are not already in the database and don't have a relationship to the university such as Alumni or Parent are entered as "Friend", so it makes sense that many first-time donors would have that label, and few non-donors would.

However, after seeing that solicitations (contacts with a constituent that involve asking for a gift) and appeals (non-solicitation contacts) may have such a strong weight in differentiating donors from non-donors, I had a few thoughts. At first, it made sense because people you don't ask for gifts tend not to give, and those you stay in touch with the most in general are most likely to make a donation. Additionally, those that we don't have good contact records for, or are long-time never givers would receive fewer solicitations and appeals. I also wondered if it could be that we are just already good at selecting groups to solicit, therefore generating a higher solicitation count for those that were good prospects for giving.

Then I realized that I was counting all solicitations and appeals in fiscal years 2012 and 2013, and looking at people that gave for the first time in FY 2013. Therefore, it was possible I was counting appeals that occurred after the first gift had been made. When a person makes a gift, they are likely to receive thank-you notes and follow-up solicitations, therefore increasing their appeal count beyond those that do not give, so the classifier could be separating the classes based on appeal data from contacts occurring after the gift was made. I decided to modify my dataset to only look at appeals that occurred during this time frame, but before the first gift date, in order to more accurately reflect what happened that differentiates non-donors from first time donors at the time the decision to give is made.

I also tried varying the number of estimators in the Random Forest. The default is 10, and I saw as many as 500 being used in some examples. Increasing from 10 to 50 improved the output slightly, but upping the number of estimators from 50 to 500 estimators actually decreased correct classification of first time donors from 86.3% to 85.7% (may not be significant, but it didn't increase), so I left it at 50 estimators to reduce run time.

I also want to mention that when I test the results, I output the training score and testing score from the split dataset, then test on the entire Class 0 and Class 1 datasets to see how good the trained Random Forest is at identifying points that belong in each class. (This was especially important because of the imbalanced dataset.)

An example output result at this point is:

```
The training score is: 0.999746

The testing score is: 0.987576

The class 0 test score is: 0.999718

The class 1 test score is: 0.871555
```

I also wanted to note a lesson I learned in the process of doing this project, which is to be suspicious of results that are too good. At one point, my results were showing 100% correct classification on both classes, but when I applied it to another set of data, I was getting 100% of one class tested correct and

0% of the other! I checked the field importances and saw that one solicitation count column had an importance vastly outweighing every other column. It turned out that there was an error in my database query, and where there should have been a null count of solicitations, it was returning a number over 80000 in a column that otherwise had numbers less than 20. So, the Random Forest was using that one value to split between the two classes. Of course, that didn't generalize well when applied to the other dataset that did not have the same issue!

Modifying columns to generate better results

Because of the significant improvement in the last round of training, I figured that the way to continue improving the classifier was to improve the imported data, so I modified the solicitation columns to only look at solicitations and appeals prior to the first gift, I had just added current parent, employee, years since record/contact modified, and event participations, and I additionally added a count of appeals that did not come from the Office of Annual Giving (OAG). I was going to add a column showing the count of appeals from the alum's preferred college, but not all of the codes were simple to cross-reference in our data, so I substituted "non-OAG" data for "college-specific" data. In reality, many of the college solicitations are sent by OAG, but this was as close as I could get given the time constraints.

Some features that looked like strong differentiators at this point were:

1. Non-Solicitation Appeals in FY12-13
2. Solicitations (lifetime)
3. Non-OAG Non-Solicitation Appeals in FY12-13
4. Non-Solicitation Appeals (lifetime)
5. Zip5
6. Distance from Harrisonburg
7. Solicitations (FY12-13)
8. Years Since Record or Contact Info Last Modified
9. Years in System
10. Non-OAG Solicitations (FY12-13)

Seemingly Weak features included:

- Assigned to Development Officer
- Events Attended
- Region (Zip1)
- Preferred College
- Record Type

Most of these are categorical, so maybe seem weak in comparison to single numeric attributes, but when looked at individually (see "zoomed in" figure on p.8 above), the strongest are:

1. Record Type Friend (makes sense since added as friend if donor not already constituent)
2. Recent/Current Parent
3. Has Phone Number
4. Record Type Student
5. Record Type Parent
6. Has Email Address

Of the college categories, the College of Business was the largest indicator, which makes sense since they receive more gifts than the other colleges, and the College of Health and Behavioral Studies was second. Of the zip code categories, "2" was strongest, which is primarily in Virginia.

Output:

```
Shape:
(103789, 51) (103789,)
(101757, 51) (2032, 51)

The training score is: 1.000000
The testing score is: 0.997350
The class 0 test score is: 0.999980
The class 1 test score is: 0.973917

Feature Importances:
[  2.21005462e-02   1.53765510e-02   2.69308555e-02   3.44631443e-02
   4.35548755e-02   4.45949194e-02   1.10124040e-02   2.85798050e-02
   4.35287677e-02   4.12931743e-02   1.93608431e-02   5.10320161e-01
   4.68130304e-02   1.33038503e-03   4.11625739e-03   1.51935600e-02
   1.39101257e-02   5.77075576e-03   2.43150687e-03   1.39943292e-03
   1.35425163e-03   4.16851231e-03   8.58521635e-04   3.05408303e-04
   1.04694681e-04   3.72944525e-04   7.88004306e-04   5.06285769e-04
   4.64961785e-04   1.12023298e-03   4.70034945e-03   3.72626794e-04
   9.37895234e-05   4.60990141e-03   2.73700700e-03   2.83141224e-05
   7.96505112e-03   1.00878383e-02   7.20140007e-03   2.05314198e-03
   5.19320022e-03   2.47167306e-03   2.24840186e-03   8.70878554e-04
   3.87191185e-03   1.16339285e-03   1.00228745e-03   1.02290313e-03
   0.00000000e+00   7.52457074e-06   1.73487592e-04]
```

Comparing to Other Classifiers

Using skLearn's "Extra Trees Classifier" [6] variation on Random Forest increased the correct classification of first time donors to 87.5%. It is described in the documentation as follows:

"This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting."

I also wanted to compare to Support Vector Machine (SVM) classification, but the number of columns I was using made that infeasible due to runtime. So, I narrowed the dataset down to the most important 18 columns, and the results were as follows:

Using SVM:

The training score is: 0.989377
The testing score is: 0.980249
The class 0 test score is: 0.999941
The class 1 test score is: 0.367126

Using Random Forest on the same 18-column dataset:

The training score is: 1.000000
The testing score is: 0.997399
The class 0 test score is: 0.999971
The class 1 test score is: 0.974902

So clearly, the Random Forest was a better classifier for this data, able to classify over 97% of the first time donors correctly vs 37% for the SVM, and the Random Forest training ran in about 1/10 of the time.

Testing Trained Model as a Predictor on FY14 Data

Though the Random Forest had a good result and could classify close to 100% of the data correctly for FY13, I wondered whether the trained model could be used to predict who would give for the first time the next year in FY14.

So, I trained the model using FY13 data and tested it on FY14 data. The data is not exactly comparable, partially because we are not yet done with Fiscal Year 2014 (our fiscal year goes through June), and also because some of the columns are using current data in both datasets (such as primary record type), so there isn't a "one year before" predictor value in some cases.

The results of applying the model to this year's data are:

The training score is: 1.000000
The testing score is: 0.997302
The 2013 class 0 test score is: 0.999961
The 2013 class 1 test score is: 0.974409

The 2014 testing score is: 0.829842
The 2014 class 0 test score is: 0.834407
The 2014 class 1 test score is: 0.670847

So the Random Forest classifier trained on FY2013 data was able to correctly classify 97.4% of the 2013 first time donors as first time donors, and when applied to FY2014 data, was able to correctly classify 67.1% of the first time donors and 83.4% of the non-donors so far this fiscal year.

The next task is to find out what the differentiators are (beyond just which columns are important to the classifier, but which values it is using to predict who will become a first time donor) and determine what data, if any, can be used to make decisions about segmenting out a population from the database for solicitation next year.

Future Research

Given time, I would learn how to identify how the Random Forest is classifying each data point (i.e. what is the resulting Decision Tree). I would also like to create more visualizations to better understand the data, the relationships between columns, and the results.

A change I would make to this dataset is to remove the record type “Friend” for anyone that was not in the database prior to making a gift. Those records turn out not to be relevant to decision-making for solicitations, because if a person’s record does not exist in the database prior to making their first gift, we cannot impact their likelihood of giving.

A similar exercise could be completed comparing the population left out of this one: people that have made gifts before. The classes could then be people who gave last year and gave again this year, vs people who gave last year and did not give this year. It would be interesting to see which features were important to the Random Forest classifier in that case.

Summary

In summary, the Random Forest Classifier worked very well and was able to correctly classify over 97% of First Time Donors and 99.9% of Never Givers based on the dataset I used. I would like to spend more time better understanding how it is dividing up the classes and tweaking the dataset, both to enable JMU to utilize what the classifier learned to segment solicitations in the future, and to make this model a better predictor of who will become a donor in future years. I learned a lot during the two weeks of working on this final project, and I hope to complete more Machine Learning projects with this data in the future.

Works Cited

[1] "Random Forest," [Online]. Available: http://en.wikipedia.org/wiki/Random_forest. [Accessed 01 05 2014].

[2] "Introduction to Random Forests," CIP Labs, [Online]. Available: http://www.cip-labs.net/2013/01/17/introduction-to-random-forests/. [Accessed 01 05 2014].

[3] "Random Forest Classifier," SciKit-Learn, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 27 04 2014].

[4] F. Provost, "Machine Learning from Imbalanced Data Sets 101," New York University, [Online]. Available: https://archive.nyu.edu/bitstream/2451/27763/2/CPP-02-00.pdf.

[5] "Preprocessing," SciKit-Learn, [Online]. Available: http://scikit-learn.org/stable/modules/preprocessing.html. [Accessed 28 04 2014].

[6] "Extra Trees Classifier," SciKit-Learn, [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html. [Accessed 09 05 2014].