I apply machine learning (ML) techniques to Snowplow web event data to understand how variation in marketing site experiences might correlate to customer conversion. Edit

customer-conversion machine-learning marketing-site rbf-kernel snowplow svm Manage topics

100 commits 1 branch 0 releases 1 contributor

b-knight committed on GitHub Update README.md Latest commit d294ac9 2 hours ago

Data	Dropped redundant .csvs	23 hours ago
Images	Added logistic regression	11 hours ago
Proposal	Added F2 equation	2 days ago
.gitignore	Corrected images	5 days ago
Notebook 1 - Data Munging.ipynb	Added optimized RBF kernel	5 days ago
Notebook 2 - Exploratory Analysis.ipynb	Dropped redundant .csvs	23 hours ago
Notebook 3 - KNN (Baseline).ipynb	Added logistic regression	11 hours ago
Notebook 4 - SVM with RBF Kernel.ipynb	Added logistic regression	11 hours ago
Notebook 5 - Linear SVM.ipynb	Added logistic regression	11 hours ago
Notebook 6 - Logistic Regression.ipynb	Added logistic regression	11 hours ago
README.md	Update README.md	2 hours ago
visuals.py	Initial Analysis	14 days ago
visuals.pyc	Initial Analysis	14 days ago

README.md

Understanding Customer Conversion with Snowplow Web Event Tracking

Benjamin S. Knight, January 27th 2017

Project Overview

Here I apply machine learning techniques to Snowplow web event data to infer whether trial account holders will become paying customers based on their history of visiting the marketing site. By predicting which trial account holders have the greatest likelihood of adding a credit card and converting to paying customers, we can more efficiently deploy scarce Sales Department resources.

Snowplow is a web event tracker capable of handling tens of millions of events per day. The Snowplow data contains far more detail than the MSNBC.com Anonymous Web Data Set hosted by the University of California, Irvine’s Machine Learning Repository. At the same time, we do not have access to demographic data as was the case with the Event Recommendation

Engine Challenge hosted by Kaggle. Given the origin of the data, there is no industry-standard benchmark for model performance. Rather, assessing baseline feasibility is a key objective of this project.

Problem Statement

To what extent can we infer a visitor’s likelihood of becoming a paying customer based upon that visitor’s activity history on the company marketing site? We are essentially confronted with a binary classification problem. Will the trial account in question add a credit card (cc_date_added IS NOT NULL ‘yes’/‘no’)? This labeling information is contained in the ‘cc’ column within the file ‘munged df.csv.’

Metrics

As we discuss later, the data is highly imbalanced (successful customer conversions average 6%). Thus, we are effectively searching a haystack for rare, but exceeedingly valuable needles. In more technical terms, we want to maximize recall as our first priority. Selecting the model that maximizes precision is a subsequent priority. To this end, our primary metric is the F2 score shown below.

$$F_\beta = \frac{(1+\beta^2) * \text{True Positive}}{(1+\beta^2) * \text{True Positive} + \beta^2 * \text{False Negative} + \text{False Positive}}$$

$$\text{where } \beta = 2$$

The F2 score is derived from the F1 score by setting the weight of the \beta parameter to 2, effectively increasing the penalty for false negatives. While the F2 score is the arbiter for ultimate model selection, we also use precision-recall curves to clarify model performance. We have opted for precision-recall curves as opposed to the more conventional receiver operating characteristic (ROC) curve due to the highly imbalanced nature of the data (Saito, 2016).

Data Preprocessing

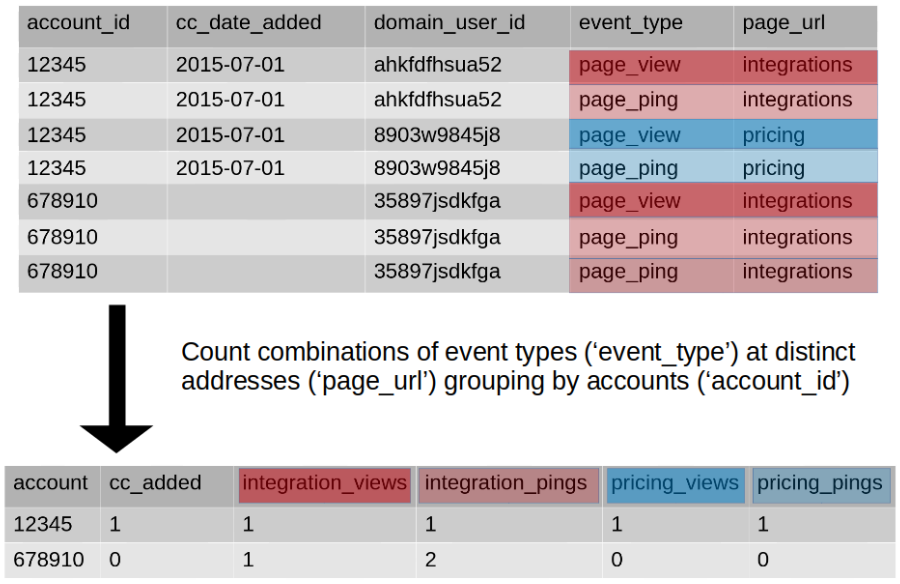
The raw Snowplow data available is approximately 15 gigabytes spanning over 300 variables and tens of millions of events from November 2015 to January 2017. When we omit fields that are not in active use, are redundant, contain personal identifiable information (P.I.I.), or which cannot have any conceivable baring on customer conversion, then we are left with 14.6 million events spread across 22 variables.

Snowplow Variable Name	Snowplow Variable Description
event_id	The unique Snowplow event identifier
account_id	The account number if an account is associated with the domain userid
reg_date	The date an account was registered
cc_date_added	The date a credit card was added
collector_tstamp	The timestamp (in UTC) when the Snowplow collector first recorded the event
domain_userid	This corresponds to a Snowplow cookie and will tend to correspond to a single internet device
domain_sessionidx	The number of sessions to date that the domain userid has been tracked
domain_sessionid	The unique identifier for the Snowplow cookie/session
event_name	The type of event recorded
geo_country	The ISO 3166-1 code for the country that the visitor’s IP address is located
geo_region_name	The ISO-3166-2 code for country region that the visitor’s IP address is in
geo_city	The city the visitor’s IP address is in
page_url	The page URL
page_referrer	The URL of the referrer (previous page)

mkt_medium	The type of traffic source (e.g. ’cpc’, ’affiliate’, ’organic’, ’social’)
mkt_source	The company / website where the traffic came from (e.g. ’Google’, ’Facebook’)
se_category	The event type
se_action	The action performed / event name (e.g. ’add-to-basket’, ’play-video’)
br_name	The name of the visitor’s browser
os_name	The name of the vistor’s operating system
os_timezone	The client’s operating system timezone
dvce_ismobile	Is the device mobile? (1 = ’yes’)

I use the phrase 'variable' as opposed to 'feature', since this dataset will need to undergo substantial transformation before we can employ any supervised learning technique. Each row has an 'event_id' along with an 'event_name' and a ‘page url.’ The event id is the row’s unique identifier, the event name is the type of event, and the page url is the URL within the marketing site where the event took place.

The distillation of the raw data into a transformed feature set with labels is handled by the iPython notebook 'Notebook 1 - Data Munging.' In transforming the data, we will need to create features by creating combinations of event types and distinct URLs, and counting the number of occurrences while grouping on accounts. For instance, if ‘.../pay-ment plan.com’ is a frequent page url, then the number of page views on payment plan.com would be one feature, the number of page pings would be another, as would the number of web forms submitted, and so forth. Given that there are six distinct event types and dozens of URLs within the marketing site, then the feature space quickly expands to encompass hundreds of features. This feature space will only widen as we add additional variables to the mix including geo region, number of visitors per account, and so forth.



account_id	cc_date_added	domain_user_id	event_type	page_url
12345	2015-07-01	ahkfdfhsua52	page_view	integrations
12345	2015-07-01	ahkfdfhsua52	page_ping	integrations
12345	2015-07-01	8903w9845j8	page_view	pricing
12345	2015-07-01	8903w9845j8	page_ping	pricing
678910		35897jsdkfga	page_view	integrations
678910		35897jsdkfga	page_ping	integrations
678910		35897jsdkfga	page_ping	integrations

Count combinations of event types (‘event_type’) at distinct addresses (‘page_url’) grouping by accounts (‘account_id’)

account	cc_added	integration_views	integration_pings	pricing_views	pricing_pings
12345	1	1	1	1	1
678910	0	1	2	0	0

With the raw data transformed, our observations are no longer individual events but indivual accounts spanning the period November 2015 to January 2017. Our data set has 16,607 accounts and 581 features. 290 of these represent counts of various combinations of web events and URLs grouped by account. Next there are two aggregated features - the total number of distinct cookies associated with the account, and the sum total of all Internet sessions linked to that account.There are also 151 features that represent counts of page view events linked to IP addresses within a certain country (e.g. a count of page views from China, a count of page views from France, and so forth).

46 of the features represent counts of page views coming from a specific marketing medium ('mkt medium'). Recall that 'mkt medium' is the type of traffic. Examples include ‘partner link,' 'adroll,' or 'appstore.' The ‘mkt medium’ subset of features is followed by 86 features that correspond to Snowplow’s 'mkt source' field. 'mkt source' designates the company / website where the traffic came from. Examples from this subset of the feature space include counts of page views from Google.com

('mkt source google com') and Squarespace ('mkt source square'). There are two additional feature:'mobile pageviews' and 'non-mobile pageviews' that represent counts of page views that took place on mobile versus non mobile devices. I have also included an additional feature derived from these two - the share of page views that took place on a mobile device.

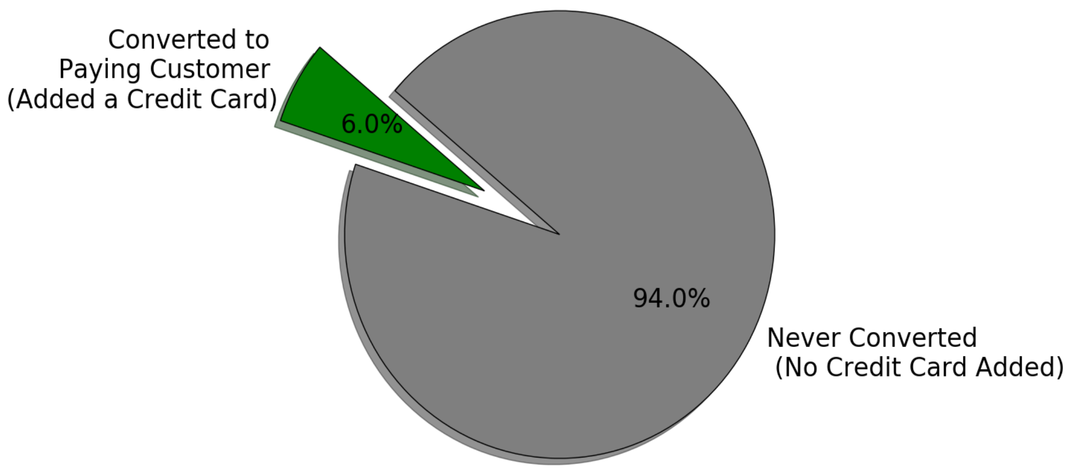
With the aggregations completed, we then take the transformed data and drop all features that are uniformly zero for all observations. Finally, we scale the features using robust scaling.

It bears noting that 'br_name' (the name of the visitor's browser), 'os_name' (the name of the vistor's operating system), and 'os_timezone' (the client's operating system timezone) were not included in the ultimate version of the transformed data. The transformed variables of 'br_name' and 'os_name' were used initially. However, their incorporation added +40 features to the already expansive feature space resulting in inferior performance and so were subsequently dropped.

Data Exploration

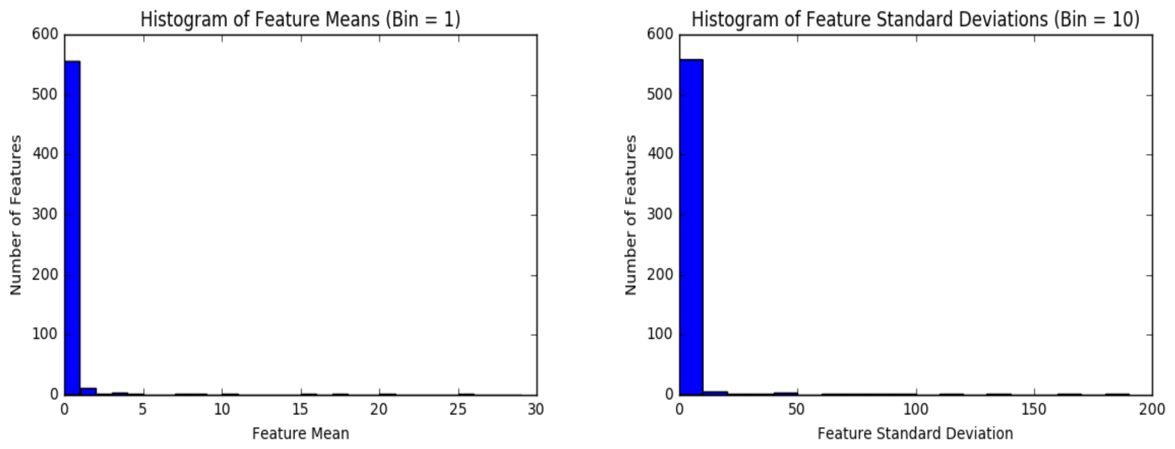
Exploring the transformed data, two features quickly become apparent. First, we can see that the data is highly imbalanced. Only approximately 6% of the labeled accounts show a succesful conversion to paying customer.

Summary Statistics: Distribution of Labels (16,607 Observations)



The second feature of note is that in addition to our feature space being wide with over 500 features, the features themselves are fairly sparse as the histograms below make clear. This is to be expected. The Snowpow features are highly specific. Examples include counts of certain types of events localized within Bangladesh, or the number page views associated with a bit of on-line content that was only made briefly available. As a result, the majority of features are extremely sparse.

Summary Statistics: Means and Standard Deviations of Spare Feature Space (581 Features)



Benchmark

How do we know if our ultimate model is any good? To establish a baseline of model performance, I implement a K-Nearest Neighbors model within the iPython notebook 'Notebook 3 - KNN (Baseline).' In the same manner as the subsequent model selection, I allocate 90% of the data for training (14,946 observations) and 10% for model testing (1,661 observations). Given the binary nature of the label, I specify the model as having 2 neighbors. I run the resulting model on the test data using 100-fold cross validation. Averaging the 100 resultant F2 scores, we thus establish a benchmark model performance of F2 = 0.04.

Algorithms and Techniques

We start our analysis with establishing a benchmark using K-Nearest Neighbors (KNN) before moving on to more sophisticated algorithms. Like KNN, logistic regression is computationally inexpensive - a definite strength given the size the data set (n = 16,607). In addition, logistic regression is uniquely well-suited to the binary nature of the outcome variable. The second algorithm selected is linear Support Vector Machines (SVM). As with the other algorithms, linear SVM is relatively inexpensive. Linear SVM is also well-suited for the high dimensionality of our data set (581 features). The finel algorithm used is SVM with a RBF kernel. SVM + RBF models tend to be perform well with binary data (Wainer, 2016), but are far less expensive than random forest models.

Implementation

The models were run using a 90%:10% split between training (14,946 observations) and testing (1,661 observations). All models were first run using their default setting, while SVM + RBF, linear SVM, and logistic regression were run a second time after tuning the hyper-parameters with Bayesian optimization (see below). Mean F2 scores, recall, and precison were derived from 100-fold cross validation of the testing data.

Refinement

In theory, we should be able to improve upon the baseline models by tuning the models' hyper-parameters. Our primary hyper-parameters of interest are C and gamma for the SVM + RBF model, and just C for the linear SVM model. Recall that C is the penalty parameter - how much we penalize our model for incorrect classifications. A higher C value holds the promise of greater accuracy, but at the risk of overfitting. The selection of the the gamma hyper-parameter determines the variance of the distributions generated by the RBF kernel, with a large gamma tending to lead to higher bias but lower variance.

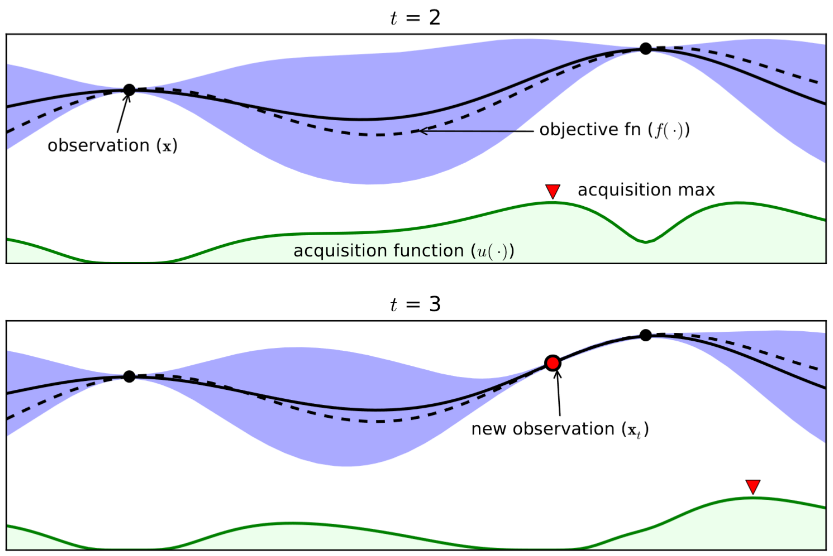
The C and gamma hyper-parameters can vary by several orders of magnitude, so finding the optimal configuration is no small task. Employing a grid search can be computationally expensive - almost prohibitatively expensive without parallel computing resources. Fortunately, we do not have to exhaustively scan the hyper-parameter space. Rather, we can use Bayesian optimization to find the optima within the hyper-parameter space using surprisingly few iterations.

Here I am indebted to Fernando Nogueira and his development of the BayesianOptimization for Python. By means of this package, we are able to scan the hyper-parameter space of the SVM + RBF kernel model for suitable values of C and gammma within the range 0.0001 to 1,000. We are able to scan for suitable values for C within the linear SVM model in similiar fashion.

The below figure illustrates the second and third iterations of this process in a hypothetical unidimensional space - for instance, the hyper-parameter C. Thus, the horizontal axis represents the individual values of C while the horizontal axis represents the metric that we are trying to optimize - in this case, the F2 score.

The true distribution of F1 scores is represented by the dashed line, but in reality is unknown. The dots represent derived F2 scores. The continuous line represents the inferred distribution of F2 score. The blue areas represent aa 95% confidence interval for the inferred distribution, or in other words, represent areas of potential information gain.

An Acquisition Function Combing a Unidimensional Space for Two Iterations



The Bayesian optimizer resolves the perennial dilemma between exploration and optimization by use of an acquisition function, shown above in green. The red triangle denotes the global maximum of the acquisition function, with the subsequent iteration deriving the F2 score for that value of C. Note how the acquisition function derives high value from regions of relatively low information (the exploration impetus), yet achieves even greater values when in the vicinity of known maxima of the inferred distribution (the optimization impetus).

For the purposes of Bayesian optimization, we used 20-fold cross validation with a custom scoring function to maximize the F2 scores. The optimization yielded values of C = 998 and gamma = 0.2 for the SVM + RBF model, C = 335 for the linear SVM model, and C = 585 for the logistic regression model.

Results

The optimal model in terms of F2 score, recall, but also precision was the linear SVM model with hyper-parameter tuning via Bayesian optimization. The linear SVM model was so successful that the non-optimized version was the second best performing model. The linear SVM model achieved a mean F2 score of 0.25 versus 0.04 for the benchmark KNN model. For recall, the linear SVM achieved a mean score of 0.33. In other words, the model sucessfully found a third of the valuable needles within our haystack. The model also achieved a mean precision of 0.14 - effectively tying with the hyper-parameter tuned logistic regression model. To put this in context, a sales representative engaged in blind guessing which acccounts would convert to paying customers would be hard pressed to be accurate more than 6% of the time (the rate of customer conversion).

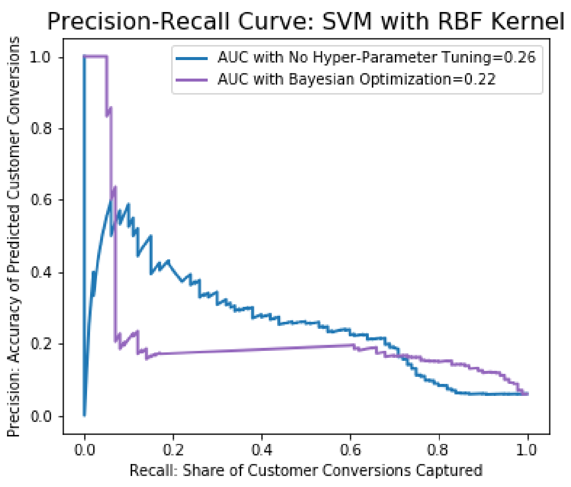
Results: Comparision of Performance Metrics Averaged from 100-Fold Cross Validation

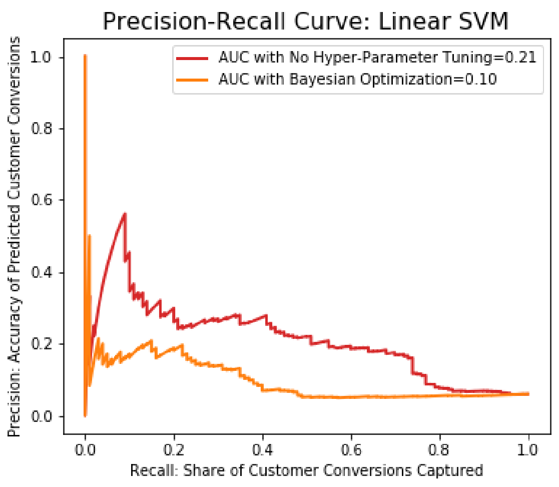
Model Used	F2 Score	Recall	Precision
K-Nearest Neighbors (Baseline)	0.04	0.04	0.04
Logistic Regression	0.16	0.19	0.13
Logistic Regression with Hyper-Parameter Tuning	0.15	0.18	0.14
Support Vector Machines with RBF Kernel	0.00	0.00	0.00
Support Vector Machines with RBF Kernel and Hyper-Parameter Tuning	0.03	0.03	0.03
Linear Support Vector Machines	0.16	0.20	0.13
Linear Support Vector Machines with Hyper-Parameter Tuning	**0.25**	**0.33**	**0.14**

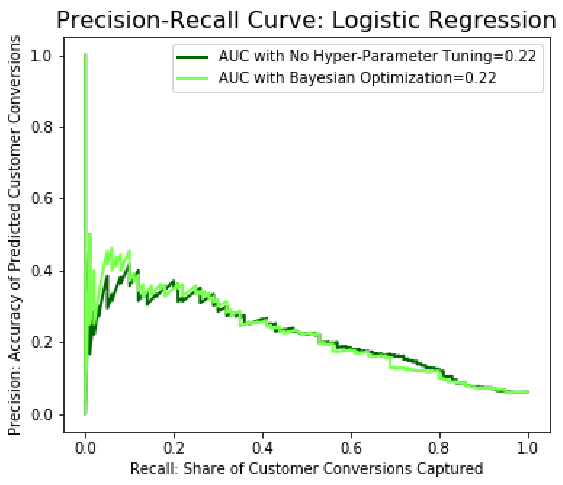
It is striking how the AUC scores for the precision-recall curves imply a very different performance ranking than what the F2 scores report. Looking to the figures below, we can see that the linear SVM with hyper-parameter tuning actually has the lowest AUC of any of the curves (AUC = 0.10). These AUC scores are based upon average precision as opposed to recall.

However, it is recall, not precision, that is our priority here. Nevertheless, it is worthing bearing in mind that more often than not, the price for greater recall is precision and vice versa.

Results: SVM with RBF Kernel and Linear SVM (Default Hyper-Parameter Settings)







Conclusion

This project sought to differentiate soon-to-be paying customers from non-paying account-holders based solely on their activity history on the marketing site. This is a tall order, and while the linear SVM did achieve a F2 score of 0.25 (a more than five-fold improvement vis-avis the KNN benchmark) in practical terms the model is not yet successful enough to be used as part of the sales process. Should the company see a drastic influx of new leads, then our linear SVM model can conceivably be used to prioritize sales outreach. However, in the mean time we should re-examine the model for areas of improvement.

A major challange of this project is the high dimensionality of the data combined with the sparse feature space. Dimensionality reduction via principal component analysis (PCA) was attempted, but yielded inferior model performance and was discontinued. Nevertheless, dimensionality reduction could be used to open up an array of robut models such as random forests. A possible alternative to PCA is linear discriminant analysis (LDA). Unlike PCA, LDA emphasizes discrimination between classes. Moreover, given the large number of observations available to us, PCA is generally less likely to perform well relative to LDA (Martinez and Kak, 2001).

LDA coupled with random forest modeling is one potential area to pursue. Another, more radical aproach would be to reconceive the problem not as one of supervised classification, but rather outlier detection. Given that the accounts of interest only make up 6% of the data set, there is an argument to be made that such observations are in fact - outliers. In pratical terms, we could leverage Sci-Kit Learn's One Class SVM funcationality. In this fashion, future extensions might take a factorial approach, varying between random forest versus One Class SVM and LDA-reduced data versus no dimensionality reduction.