Parse.ly Analytics Product Usage and Churn Prediction

Andrew Gerlach
Department of Linguistics
University of Colorado Boulder
`andrew.gerlach@colorado.edu`

Annelise Lynch
Department of Applied Mathematics
University of Colorado Boulder
`annelise.lynch@colorado.edu`

Dmitri Tarasov
Department of Computer Science
University of Colorado Boulder
`dmitri.tarasov@colorado.edu`

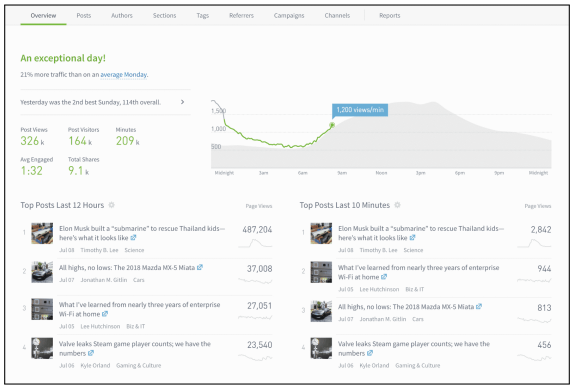
Abstract

Measuring usage of an application provides insight into what a user does with the product. However, it can be challenging to find the metrics that can predict user behavior, if a user is deriving sufficient value from a product, and if a product is robust enough to continue to attract new users. This paper discusses the analysis of churn, return user prediction data, using Parse.ly's content analytics software. After applying several machine learning algorithms to the data, it was discovered that the data showed little correlation between the usage metrics, user types, size of the customer, and user churn.

1 Introduction

Adding new customers and reducing the loss of current customers, also known as customer churn, is paramount to businesses whose product is a web-based application. These companies strive to have a better understanding of how their product is used by tracking several product use metrics and using these data to improve their product to keep current users engaged, and to attract new users to the product.

Parse.ly Analytics Figure 1 is a content analytics platform for brands, publishers, and digital media with about 55,000 unique users. The company uses its tracker on its own analytics product to track how its clients use the product. The tracker looks at which pages in the dashboard are being looked at, how long the pages are looked at, how many sessions per day, per week, and per month a user has, the size of a user's company, and if a user receives recurring reports about their own data, and many others.



An exceptional day!

21% more traffic than on an average Monday.

Yesterday was the 2nd best Sunday, 114th overall.

Post Views	Post Visitors	Minutes
326 k	164 k	209 k

Avg Engaged	Total Shares
1:32	9.1 k

Top Posts Last 12 Hours

		Page Views
1	Elon Musk built a "submarine" to rescue Thailand kids—here's what it looks like Jul 08 Timothy B. Lee Science	487,204
2	All highs, no lows: The 2018 Mazda MX-5 Miata Jul 07 Jonathan M. Gitlin Cars	37,008
3	What I've learned from nearly three years of enterprise Wi-Fi at home Jul 05 Lee Hutchinson Biz & IT	27,051
4	Valve leaks Steam game player counts; we have the numbers Jul 06 Kyle Orland Gaming & Culture	23,540

Top Posts Last 10 Minutes

		Page Views
1	Elon Musk built a "submarine" to rescue Thailand kids—here's what it looks like Jul 08 Timothy B. Lee Science	2,842
2	What I've learned from nearly three years of enterprise Wi-Fi at home Jul 05 Lee Hutchinson Biz & IT	944
3	All highs, no lows: The 2018 Mazda MX-5 Miata Jul 07 Jonathan M. Gitlin Cars	813
4	Valve leaks Steam game player counts; we have the numbers Jul 06 Kyle Orland Gaming & Culture	456

Figure 1: A snapshot of Parse.ly Analytics

The goal of this project was to classify a returning visitor based on their common behaviors and attributes. This enables Parse.ly to use the categories' characteristics to find ways to increase user activity. Each data example had 23 associated features. Dimensionality reduction using t-SNE and UMAP was applied to reduce the number of features with the intention of finding the most useful features for classification by another model, such as SVM, Decision Trees, and k-Nearest Neighbors.

2 Dataset and Features

The Parse.ly usage dataset existed in 5 different databases/data sources, and multiple tables. The full data transformation was completed using `SQL` and is fully documented in the file `data_etl_csci5622_project.sql`. The final result of the ETL and data manipulation process was data aggregated into 3 different tables to use for algorithm development: `Visitors per month, Visitors per week, Visitors per day`.

Originally this project was scoped as an unsupervised learning project to identify potential classifications such as super users, active users, and infrequent users. It was determined that no distinct clusters existed, and the project was changed into a supervised learning project to predict whether a user returned or did not return within a given time frame.

The data had a total of 23 unique features, which are detailed below in Table 2. The predictor was a field called `did_return_visit`, which was a binary field with 1 indicating that a user returned the following time period (month, week, or day) and 0 indicating that a user did not return the following time period (month, week, day).

Feature Name	Description
`is_admin`	Binary field if a user is an admin of the customer organization
`is_superuser`	Binary field if a user is a superuser of the organization
`is_staff`	Binary field if a user is an active staff within the organization
`apikey_muvs`	apikey (customer) monthly unique visitors, an indicator of how large a customer is by how much traffic they receive on the internet.
`months_since_user_signup`	The months since the user registered for the product
`total_*_views` (17 fields)	The number of views a user had of a specific part of the product within the specified time frame. Examples include: `total_overview_views, total_author_views, total_referrerviews`

Table 1: List of features

3 Methods

3.1 Feature Engineering

The following feature selection algorithms were used to reduce dimensionality:

- **sklearn's Recursive Feature Elimination:** A feature selection method that removes the weakest features by recursively eliminating a very small amount of features over many loops. (1)

- **sklearn's feature selection SelectKBest:** Selects top k features with the highest scores. The default score function ANOVA F-Value was used. (2)

- **sklearn's ensemble ExtraTreesClassifier feature importance:** Fits randomized trees to sub-samples of the data. The criteria for splitting, and thus feature selection, was the Gini impurity. This class has the attribute `feature_importances_`, a scored list of features. (3)

The method used to determine feature relevance was voting. Any feature that did not receive a vote from any of the algorithms was excluded as seen in Figure 2

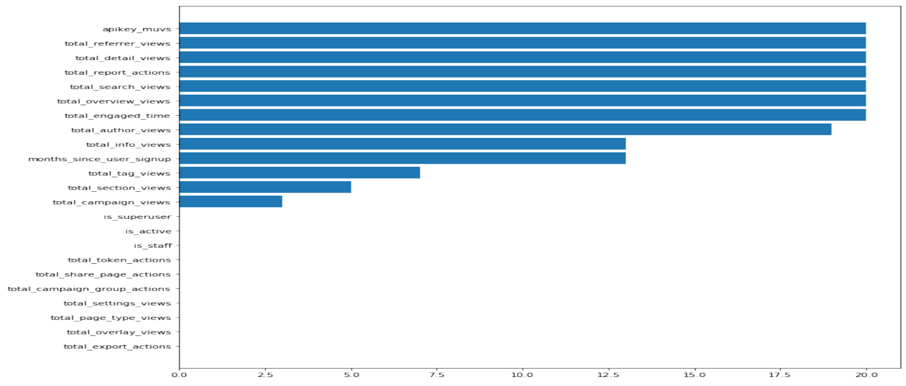


Figure 2: Final feature engineering voting results

3.2 Initial Algorithm Development

Given the sparse user activity, it was hypothesized that combining engagement features could result in a more defined decision boundary. t-sne, PCA, and UMAP were used to cluster the data in a lower-dimensional space as a first step in using other classifiers. No patterns or conclusions were drawn from the PCA as it was not effective in separating the returning and non-returning users from the feature clusters.

It was also discovered that the dataset was imbalanced, with between 75-80% of users returning. This is excellent for Parse.ly, but when sampling the data and not accounting for the imbalanced dataset, it was found that models would only predict the majority. This yielded a predictable accuracy of 75-80%. While these findings have high accuracy and precision, they are misleading because of the imbalanced data and are represented below.

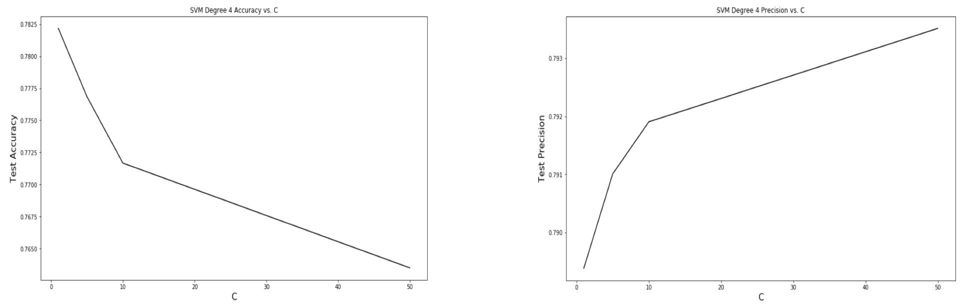


Figure 3: Misleading findings using unbalanced dataset

3.2.1 t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a dimensionality-reducing algorithm used for data visualization that uses the similarity between data points and clusters them in such a way that the more similar data are, the closer they will be in the lower-dimensional space. It starts by building a distribution between data points based on similarity then creates a low-dimensional distribution that tries to match the high-dimensional distribution (5).

openTSNE was implemented using several values for the hyperparameters. ‘n_components‘: determines the number of dimensions in the lower-dimensional embedding space, and ‘perplexity’: a continuous value of nearest neighbors that the distribution will try to maintain for each data point. There was not a notable separation of the data for any combination of perplexity and dimension as displayed in the following graphs with varying perplexities.

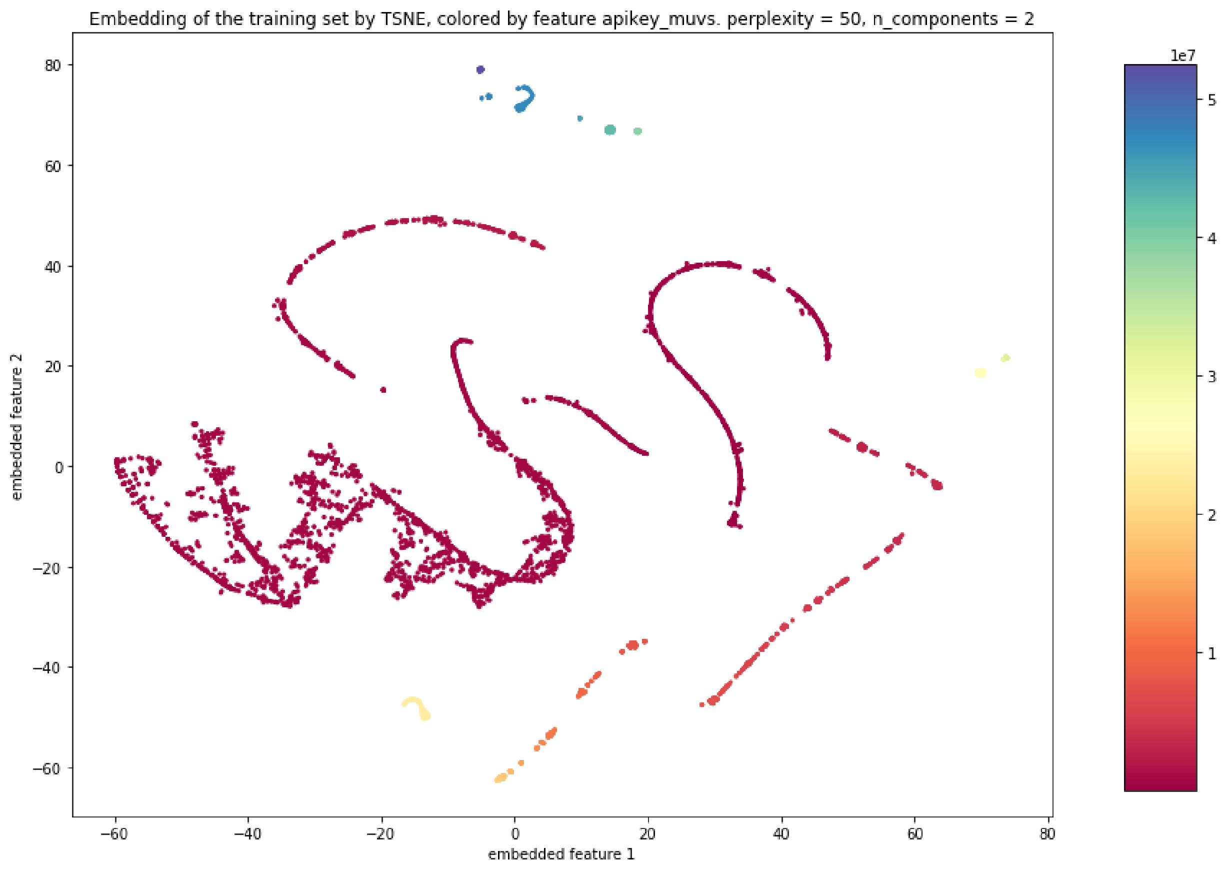


Figure 4: t-SNE Pairwise Projection of Customer Monthly Unique Visitors

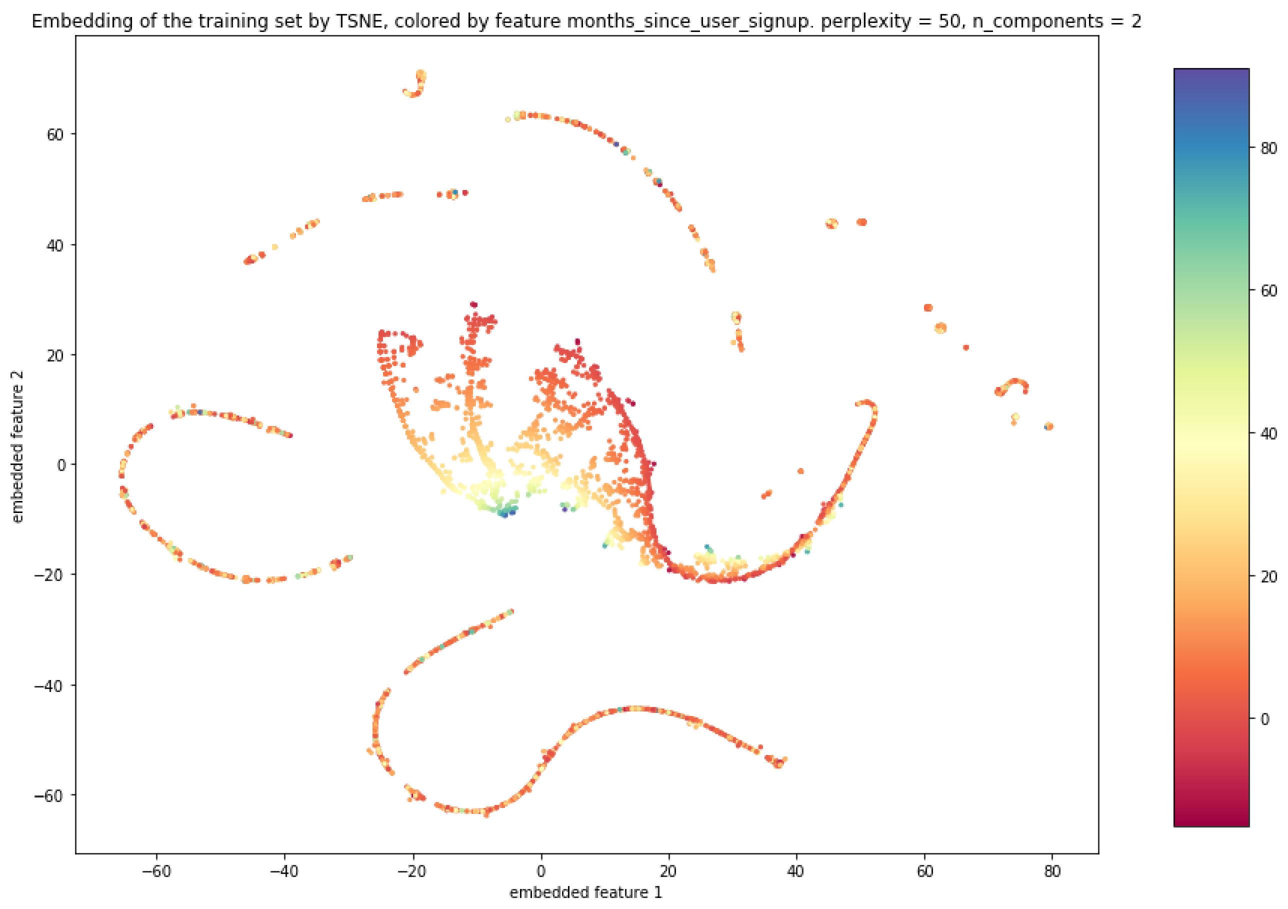


Figure 5: t-SNE Pairwise Projection of Months Since the User Signed Up

The t-sne shows that the feature space of high `apikey_muvs` and length of time using the service collapse into one sub-region.

3.2.2 UMAP

Uniform Manifold Approximation and Projection is the second dimensionality-reducing algorithm attempted. UMAP builds an adjacency matrix where edge weights represent the likelihood of two points being connected. Connected-ness is determined by extending a radius from each data point to its n-th nearest neighbor. The dimensionality is then reduced using stochastic gradient descent to find the best low-dimensional embedding. UMAP is different from t-SNE in that it preserves the original structure of the high-dimensional data (4).

As seen by the figures, both of the dimensionality reductions were able to localize the features of high usage but it was not indicative of churn.

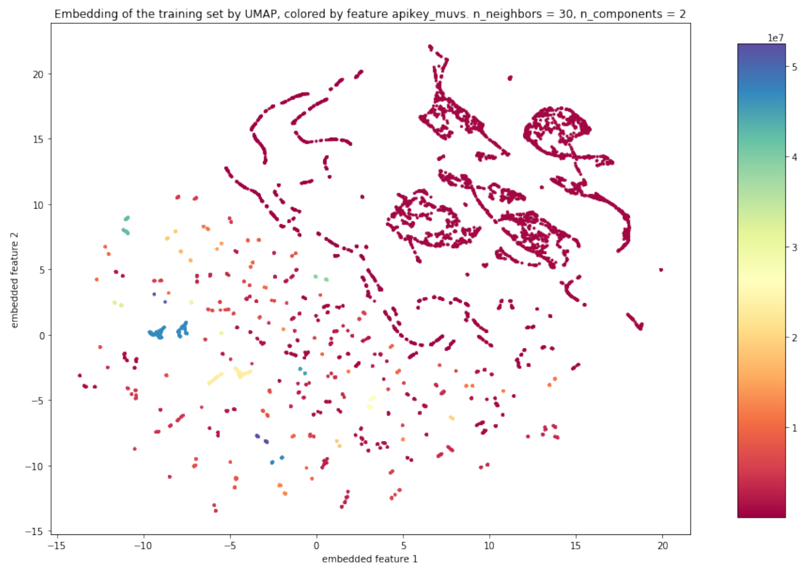


Figure 6: UMAP Pairwise Projection of Customer Monthly Unique Visitors

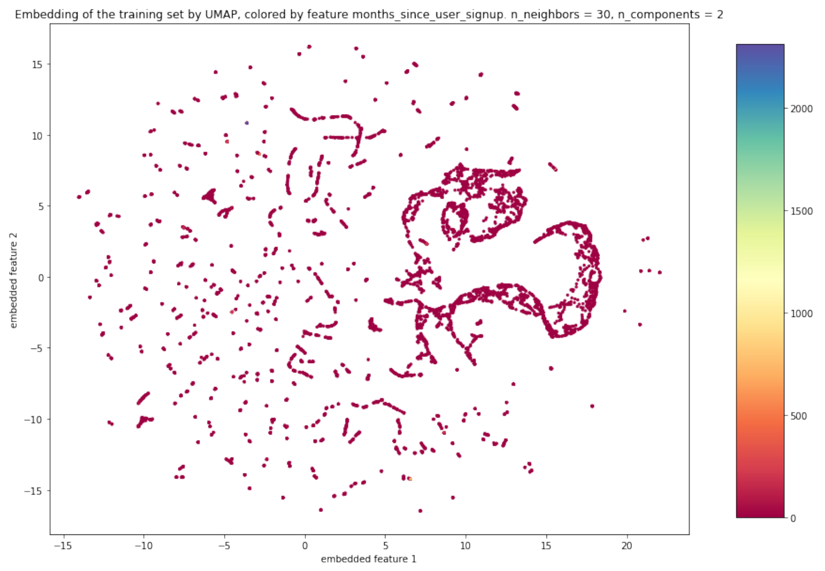


Figure 7: UMAP Pairwise Projection of Months Since the User Signed Up

3.3 Post-Processing Classifiers

With the data embedded in the lower-dimensional space, SVM, *k*-Nearest Neighbors, and Decision Tree algorithms were ran on the transformed data to get a classification.

3.3.1 SVM

Support Vector Machines is a classification algorithm that aims to build a separating hyperplane that maximizes the distance (margin) between the two classifications. SVMs work best on data that is separable. When data is not separable, the data can be transformed by kernel function that changes the orientation of the data to make it separated. scklearn's implementation of SVM was used for classification. The hyperparameters tuned for better performance were 'kernel': linear, polynomial (with which different degrees of polynomial were tried), and RBF (Gaussian), and several values for 'C', which penalizes misclassification.

3.3.2 *k*-Nearest Neighbors

Because t-SNE and UMAP both use the notion of neighbors in some way it was hypothesized that classifying the data in the lower-dimensional data with the *k*-Nearest Neighbors algorithm in tandem with these two dimension-reducing algorithms would yield more accurate results. This is done by using the transformations created by t-SNE and UMAP to transform new data and place it in the embedding space to get classifications based on the number of nearest neighbors. The parameter we tuned was 'n_neighbors', the number of neighbors to consider when classifying.

3.3.3 Decision Tree Classifiers

AdaBoost, Adaptive Boost, uses several weak learners and weights them by their classification error rate. The only requirement for a learner to be 'weak' is that it does better than chance. The weak learner in this case is a set of decision stumps, decision trees that only choose one feature. The classifications by a given stump is weighted based on its error rate. High error increases weights of classifiers and lower error will reduce the weight. We used this as another means to find which features are impacting classification the most. The weighted classifiers then combine to make one classifier.

A full decision tree was also attempted. The features were split based on gini importance. The information gain from the binary features were:

Feature	Information Gain
total search view	0.544
total referrer view	0.302
total report action	0.079
total detail view	0.055
total author view	0.016
total tag view	0.003
total section view	0
total overview view	0
total info view	0

Table 2: Gini Importance for Most Relevant Features

3.4 Final Algorithm Development

During algorithm development it was discovered that a feature that corresponded to the user returning was mistakenly used. By using UMAP, SVM, and KNN, this resulted in an accuracy of 0.95, which was concerning and led to the discovery of this error. SVM + KNN had an accuracy of 0.81. After the correction was made, The accuracy and precision of the model reduced to the near baseline results of 0.56 and 0.53 respectively.

4 Results and Discussion

4.1 Error Reporting

A large Type I error rate (false positive) was a common finding in this data set. These are incorrect predictions that users returned when they did not return. An example of this ratio and high Type I errors can be seen in the confusion matrix using validation data below.

After performing dimensionality reduction on the dataset using PCA, the data can still be interpreted with scales of usage on the reduced features. A plot of one of the downstream classifications for three embedded features is shown in Figure 8.

Model	Accuracy	Precision	Recall
Baseline (PCA + SVM)	0.589	0.238	0.772
UMAP and SVM	0.563	0.541	0.708
UMAP and KNN	0.543	0.527	0.533
UMAP and Decision Tree	0.561	0.540	0.571
t-SNE and Decision Tree	0.583	0.720	0.307
t-SNE and SVM	0.565	0.427	0.613
t-SNE and KNN	0.515	0.515	0.497
Autoencoder and Decision Tree	0.584	0.561	0.745
Autoencoder and SVM	0.522	0.216	0.548
Random Forest	0.564	0.538	0.566
Decision Tree	0.607	0.523	0.523
Adaboost	0.543	0.550	0.550

Table 3: Error reporting on test data

	predicted positive	predicted negative
true positive	1,698	125
true negative	1,353	424

Table 4: Confusion matrix using validation data for PCA and SVM

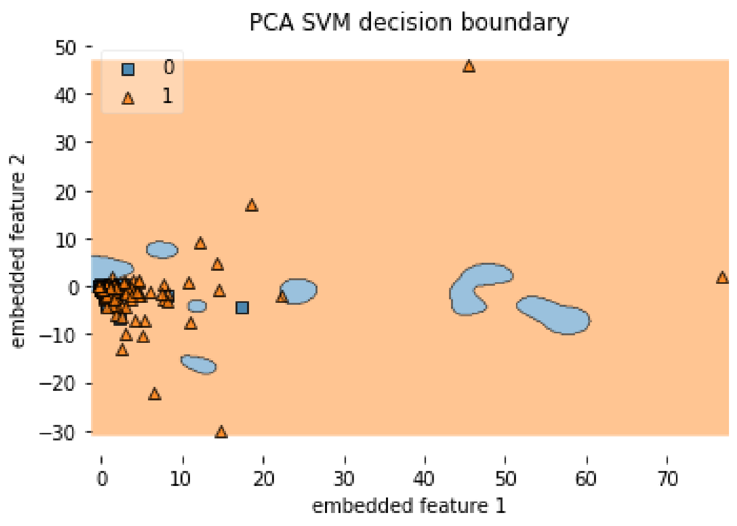


Figure 8: PCA SVM classification on a pair of features.

Further investigation of the high Type I error rate can be see in Figure 8. The decision boundary shows that low usage corresponds to the user leaving. However, the infrequent users are too embedded with the churned users to make any significant progress. The other dimensionality-reduction techniques also did not make a distinguishable change of the cluster as the two categories remained joined throughout the manifold of UMAP and t-SNE.

The decision boundary for t-SNE is seen in Figure 9. The infrequent users do not have any distinguishable feature from the churned users.

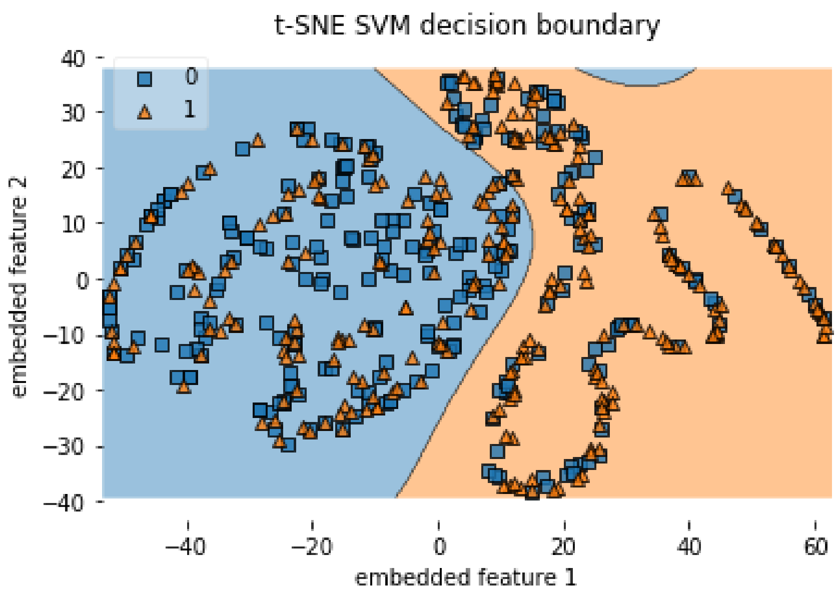


Figure 9: t-SNE SVM classification on a pair of features.

4.2 Conclusion and Recommended Next Steps for Parse.ly

After exhausting many options, it was concluded that there is little correlation between the usage of the product and churn. Although, by looking at the t-SNE and UMAP maps, it can be seen that the predictors are able to distinguish users that use the product constantly. The usage does not correspond to churn as predicted by the decision boundaries. Overall, it is a difficult task to quantify human behavior as a means to understand and predict future human behavior, especially in a contained environment such as Parse.ly's content analytics tool.

The dataset from Parse.ly is large, clean and complete. The following are recommendations for future endeavors with this data:

- Predict user behavior from previous user behavior: Instead of predicting churn, it would be beneficial to predict the behavior that a user will perform in the product. This will potentially give insights into types of users, common trends of usage, and metrics on how to appeal to and train those types of users.

- Include user roles in the data set or anything more specific about the user: The dataset does not contain the role of the user (journalist, analyst, marketing, etc). This would likely be beneficial in both predicting churn and predicting user behavior.

- Aggregate the dataset by customer/company instead of by user with more company-specific features. The dataset only contains 1 feature specific to the company. Instead of predicting user churn, aggregating by company would provide a broader predictor of total revenue churn. This would require more company-specific features.

References

[1] The scikit-yb developers *Recursive Feature Elimination* `https://www.scikit-yb.org/en/latest/api/model_selection/rfecv.html`

[2] scikit-learn developers *sklearn.feature_selection.SelectKBest* `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html`

[3] scikit-learn developers *sklearn.ensemble.ExtraTreesClassifier* `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html`

[4] Leland McInnes, John Healy, James Melville *UMAP: Uniform ManifoldApproximation and Projection forDimension Reduction* `https://arxiv.org/pdf/1802.03426.pdf`

[5] Laurens van der Maaten *Visualizing Data using t-SNE* `http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf`