Final Project Report for Stats 170B, Spring 2021

Project Title:

Uncovering Patterns of Paciolan's

Custom Text Fields Usage for Client Tagging

Student Names:
Yihan Wang, 28632594, yihaw13@uci.edu

Sirui Hu, 48235202, siruih1@uci.edu

Yadi Yang, 26936863, yadiy1@uci.edu

GitHub repository: https://github.com/siruih1/stats170ab

1 Introduction and Problem Statement

Paciolan is a company aiming to serve business clients like university sports and venues (hereinafter called "clients"), and provide their clients a better platform to serve their patrons and donors (hereinafter called "users"). Our project aimed to derive business insights from five custom fields (tags, keywords, motives, user-defined fields, donor categories) assigned by clients to the users. However, after performing a series of clustering analyses on all custom fields, only one of them, which is the "**tags**", is informative enough for us to dig out more useful patterns about user behaviors. Hence, exploring "custom fields" now narrows down to "tags". Questions we can address are "How do clients use tags to group users? How do clients name the tags?" To answer these questions, our primary approach is to find out the similarities among the users that have been assigned similar tags. By processing in a machine learning pipeline, we will be able to gain some useful insights. We are expecting to find out the reasons for the tags, and possibly unify the tags for better classifications.

The data we use are all provided by the company and already shared with us through snowflake. We have access to a great number of data related to user accounts, transactions, and clients' information. Therefore, we need to understand the data structure of Paciolan and select the datasets that could be helpful in our later analysis. According to Paciolan, an organization is one hierarchy level below a client, meaning a client can have one or more organizations. However, for those data we can access, there is only one organization per client, each of which has a unique id. We decided to use *org_id* to represent the client in our later analysis.

2 Related Work

We were trying to cluster the tags and to find out some patterns about the frequency of some tags. When dealing with the vectorization of the tags, we performed **TF-IDF (term frequency-inverse document frequency)**[1]-based **cosine similarity** to convert the string stored inside the query with a numeric value representing the degree of similarity to other vectorized strings. **TF-IDF** and **cosine similarity** are widely applied in machine learning algorithms for **Natural Language Processing (NLP)**, as many NLP-related

[1] Spark Jones, K. (1972), "A Statistical Interpretation of Term Specificity and Its Application in Retrieval", Journal of Documentation, Vol. 28 No. 1, pp. 11-21. https://doi.org/10.1108/eb026526

types of research have shown that if the representations of an entity are sufficiently close in a textual sense then they can be captured using the cosine similarity metric.[2] Also, **cosine similarity** has proven to be a robust metric for scoring the similarity between two strings.[3]

Market basket analysis often uses the **association rule** to determine a client's shopping preferences. If a customer wants to buy bread and eggs, he/she also tends to buy milk. One thing to clarify is that the rules do not extract an individual's preference but rather find out relationships between sets of elements of every distinct transaction. The implication here is **the co-occurrence but not causality.** For a given rule, the itemset is the list of all items in the antecedent and the consequent.

FP Growth is an algorithm to implement the association rule faster than the basic Apriori algorithm. It scans the database twice and uses a tree structure (**FP-tree**) to store all the information. This is efficient in processing bulk data as it will only generate the frequent itemsets according to the minimum support defined by the user and the "count accumulation and prex path count adjustment, which are usually much less costly than candidate generation and pattern-matching operations performed in most Apriori-like algorithms."[4]

3 Data Sets

3.1 Datasets Introduction

Since the raw data stored in the Snowflake schema was originally in JSON format, we only selected the variables that we are interested in in each dataset.

Accounts

account_id	org_id	keywords	userdefinedfields	tags	preferredemail	preferredcontactmethod	account_name
138788	MSSTATE	NaN	NaN	NaN	{'mr.erise@gmail.com'}	NaN	{'Erise Wilson'}
106626	MSSTATE	{'PRITCC', '0'}	NaN	{'NONSTMNOND17'}	NaN	{'N'}	{'Pritchard Engineering'}
119515	MSSTATE	{'P6625156412', 'DAVIDJ', 'PJW6412@GMAIL.COM'}	NaN	{'NONSTMNOND17'}	{'jw6412@gmail.com'}	{'N'}	{'James Davidson'}
137104	MSSTATE	NaN	NaN	NaN	{'richlopez70@gmail.com'}	NaN	{'Rich Lopez'}
132816	MSSTATE	NaN	NaN	NaN	{'john.harden@gmail.com'}	NaN	{'John Harden'}

Table 3.1.1. A sample of the raw Accounts Data

Interpretation

Each row is a user account record of a donor or patron (account_id) stored by an organization (org_id). This cleaned version stores all the keywords, user-defined fields, tags, and other data through a set and dictionary that have been assigned to an account between 2018 and 2021. Even though user-defined fields are stored as a dictionary, we will use the unique keys of each row for our later analysis. We kept the preferred email because, in line 3, the organization sometimes stores the user's email address under keywords.

---

[2] N. Koudas, A. Marathe, and D. Srivastava. Flexible StringMatching Against Large Databases in Practice. In VLDB,pages 1078–1086, 2004.

[3] Tata, Sandeep, and Jignesh M. Patel. "Estimating the Selectivity of tf-idf based Cosine Similarity Predicates." SIGMOD Record, vol. 36, no. 2, 2007, p. 7. ACM Digital Library, https://dl.acm.org/doi/10.1145/1328854.1328855.

[4] Han, Jiawei, et al. "Mining Frequent Patterns without Candidate Generation." *Association for Computing Machinery*, vol. 29, no. 2, June 2000, pp. 1–12, http://www.cse.msu.edu/~cse960/Papers/MineFeqPatteren-HPY-SIGMOD2000.pdf.

Ticket Seasons

	activity	org_id	last_datetime	name	season	size
0	VB	BAYLOR	2013-02-26 09:15:15	2012 Acrobatics & Tumbling	AT12	M
1	VB	BAYLOR	2013-04-30 15:20:03	Acrobatics & Tumbling 2013	AT13	T
2	VB	BAYLOR	2014-01-23 14:03:05	Acrobatics & Tumbling 2014	AT14	T
3	VB	BAYLOR	2015-01-27 08:17:17	Acrobatics & Tumbling 2015	AT15	T
4	VB	BAYLOR	2017-01-09 11:29:14	2016 Acrobatics and Tumbling	AT16	T

Table 3.1.2. A sample of the raw Ticket Season Events

Interpretation

Ticket Season Data shows what season each organization has and the size of the season on a yearly basis. For example, Baylor University has been having Acrobatics & Tumbling season every year since 2012. The size of the season changed from medium to titanic from 2012 to 2013 as well.

3.2 Data Cleaning and Preprocessing

We joined the organization-related datasets by a common primary key "organization id": *org_id*.

3.3 Data Visualization

PCA

In fact, initially, we also applied the dimensionality reduction method PCA to help reduce a certain number of features both organization-wide and account-wide. However, after we decided to shift our focus to TF-IDF and cosine similarity analysis, our PCA results can be regarded as a supplement as the PCA outputs indicate large variances. More details are included in the Appendix.

Word Cloud

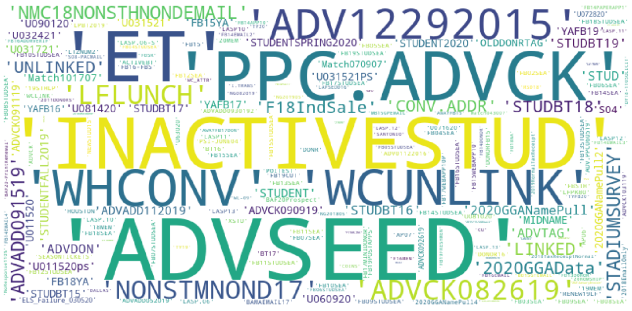


Fig 3.3. Word cloud of the tags with high frequency

Interpretation

We generated a word cloud for tags that are assigned to each account. The frequency of each tag is based on the number of unique user accounts being assigned these tags by organizations. There are in total 134,480 unique tags in Fig.3.3. Based on these plots we get to know what are the ones that are used most frequently by Paciolan's clients.

4 Overall Technical Approach

4.1 Data Processing Pipeline Overview

A 3-phase pipeline is designed and has been used throughout the project. The first phase is to clean and organize the raw data extracted from the Snowflake data warehouse provided by Paciolan. After obtaining the clean data, we move forward to the modeling and predicting process, which is the second phase of the pipeline. We mainly used unsupervised learning methods such as **PCA** to group similar custom fields. Text preprocessing methods such as **TF-IDF vectorization** and **cosine similarity** in **NLP** are used for quantifying the text-based custom fields. Later, we applied Logistic Regression as well as Random Forest models to test out the reason for the tagging patterns.

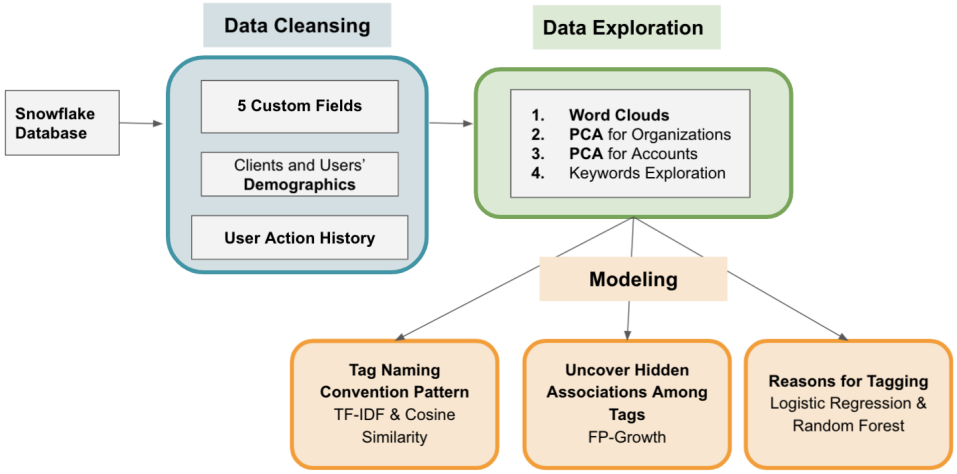


Fig 4.1. A Python Data Analysis Pipeline

4.2 Data Query and Extraction

Our sponsor Paciolan uses Snowflake to store and share the data online. Snowflake is a third-party cloud computing-based data warehousing platform. We can use SQL to easily query the data stored on that.

There are two types of features for our project, **user demographics data**, and **action data**. The **user demographics data** includes the information of donor membership, allocation of donations, ticket data, and user emails. The **Action data** includes the information of donation history, ticket purchasing history, and all transaction history. The response variables for our project are the *tags*. They are created by each organization with their own rules.

4.3 Data Management and Processing

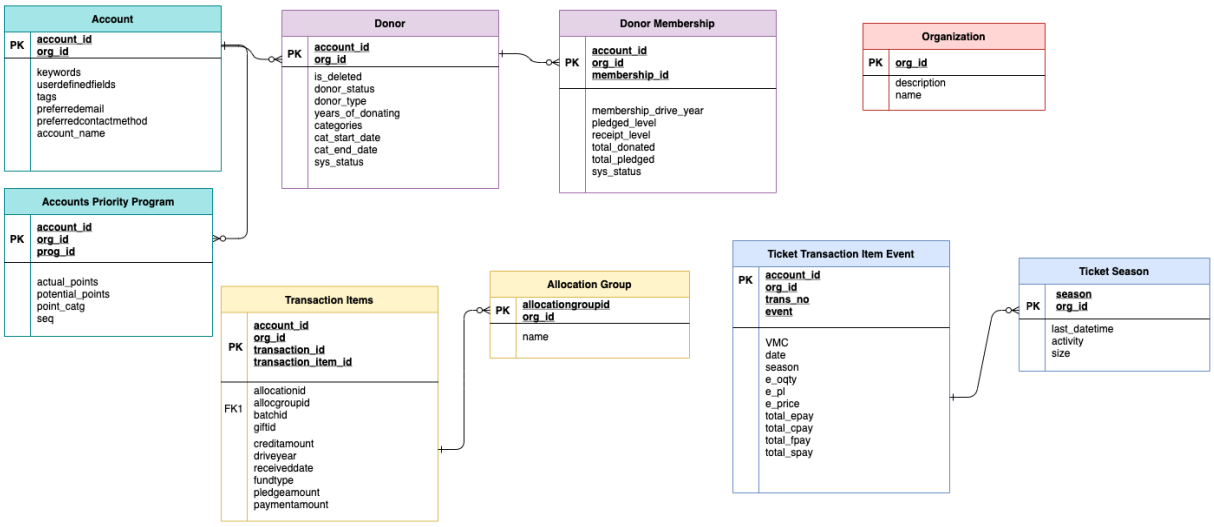


Fig 4.3. ER Diagram that shows the relationships between the cleaned datasets

After extracting the data from Snowflake and processing them using our Python data pipeline in Jupyter Notebook, we uploaded and stored the cleaned version to the **Amazon S3 bucket** (Simple Storage Service). It is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface.

Fig. 4.3 shows a complete schema of the cleaned data we plan to use in the project. The five colors each represent Account, Donation, Transaction, Ticketing, and Organization-related datasets. The number of variables we selected here has already been narrowed down by the time we extracted them from the data warehouse. Since the organization id appears in each table, we will analyze the organization level. The second common primary key is account_id, which is the foundation of our dataset structure since most of the data is collected individually based on the distinct account info.

4.4 Text Vectorization and Cosine Similarity Analysis

As we stated in the "Related Work" section, **TF-IDF** and **cosine similarity** are the fundamental mechanisms we used for vectorizing and quantifying the string-like user custom fields into numeric scores. This made more sense to the analysis as the quantitative values can be fed into the models easily. We performed this transformation in Python using the NLP tools **feature_extraction.text.TfidfVectorizer** and **metrics.pairwise.cosine_similarity** provided by the **Scikit-learn** library. Details and references about these tools are listed in the "Software" section.

We first used the preset parameters to call the **TfidfVectorizer** function, which will return us the vectorized features that the tf-idf model considered important in the given string lists. By feeding the results gained from the previous step into the cosine similarity function, we can get a correlation matrix of the similarities between each pair of the strings that fall between 0 and 1.

4.5 Tags Co-Occurrence Frequency

We applied the **Association Rule** to find frequent itemsets. In our scenario, things stored in the itemset would be the tags from each organization. The frequency of a pair of tags that appears within the

collection unit was counted. The higher the frequency, the more chance we see the pair of tags appear in the same account. That means, we can more easily find those accounts that share similar characteristics. We can recommend tags to an account according to what tags it already has.

4.6 Reasons for Tagging

After performing some calculations and aggregations, we noticed that more than 900k accounts have one or more tags containing sports and year. Therefore, we believe that these tags are related to seasonal ticket selling. We tried to reason the tags that have been labeled on more than 100 unique accounts since they are used more frequently and should be more likely to have impacts on the tagging patterns. The visualization detail can be found in section 6.1.

We labeled each account of patrons and donors as 0 or 1 depending on whether the account is labeled a certain tag by the organization/client. We used the total dollar spent and the total quantity of tickets bought by the account on each seasonal ticket from 2018 to 2020.

Table 4.6 shows the sample data we used to reason tag "fbbb." The column "tag" represents whether the account is labeled with the selected tag "fbbb." The columns "total_epay_[season]" represents the total amount of dollars an account has spent on the season, whereas the pattern "e_oqty_[season]" represents the total number of seasonal tickets the account has bought.

We filtered out the seasons that have been bought by less than 1000 accounts. The reason is that if there are too few people buying the tickets, it means the ticket itself is not so popular and not contributing to the model. And for each column we used for logistic regression, we should avoid collinearity by eliminating the columns with only zeros. For this tagging table, there used to be 96 columns related to seasonal ticketing but we used that method to shrunk the number of columns to only 17.

We selected **logistic regression** and **random forest** as the models to reason the tagging with ticket transaction data. Logistic regression has a great advantage in explainability and simplicity but it assumes little or no multicollinearity and the linearity between independent variables and log odds. Random forest works with non-linear data and has the feature importance attribute with no formal assumptions.

We have also tried SVM using linear kernel because it is effective in high dimensional spaces and is able to provide feature weights. However, it took us hours to train a model for one tag which is too time-consuming and costly in comparison to logistic regression and random forest. Therefore, we decided to drop it from our model selection. Naive Bayes assumes the features to be independent which is not the case in our sample data because the number of tickets and dollars spent should be highly associated with each season.

We used 5-fold cross-validation. The average sensitivity, average specificity, average F1 score and average AUC score are used to validate the usefulness of the model. For logistic regression, we also used pseudo-r-squared as a metric. A sample result can be found in Section 6.

account_id	org_id	tag	e_oqty_B17	total_epay_B17	e_oqty_B18	total_epay_B18	e_oqty_B19	total_epay_B19	e_oqty_VB17	...	e_oqty_VB18
2.400000e+01	USC	False	283.0	3629.0	192.0	1609.0	70.0	0.0	0.0	...	50.0
5.100000e+01	USC	False	152.0	0.0	269.0	6702.0	292.0	0.0	0.0	...	91.0
7.710000e+02	USC	False	210.0	0.0	1693.0	10.0	612.0	0.0	0.0	...	0.0
9.540000e+02	USC	False	4.0	288.0	0.0	0.0	0.0	0.0	0.0	...	0.0
9.690000e+02	USC	False	6.0	0.0	0.0	0.0	14.0	0.0	0.0	...	0.0

Table 4.6. Sample table to build models for tag "fbbb"

5 Software

Name	Description
Snowflake. connector	A Python package that connects to Snowflake
boto3	AWS SDK for Python
pyspark	A package for Apache Spark in Python that is good at working with large datasets and has a machine learning library containing FP-Growth.
scipy	A python scientific computation library that contains models for optimization etc. (dendrogram: a visualization model showing how each cluster is composed; linkage: a model for creating a distance matrix; square form: a model for converting a vector-form distance vector to a square-form distance matrix, and vice-versa)
sklearn	A python library that provides many unsupervised and supervised machine learning algorithms. (TfidfVectorizer: a model for converting raw files to a matrix of TF-IDF features; cosine_similarity: a model for computing cosine similarity between two samples; standardScaler: a model that standardizes features by removing the mean and scaling to unit variance; PCA(Principal Component Analysis): a machine learning algorithm that projects the data to a lower-dimensional space.)

6 Experiments and Evaluation

6.1 Text Vectorization and Cosine Similarity Analysis

We applied text vectorization and cosine similarity on tags. To avoid the unique tags, we also filtered out the tags that contain a long string of numbers. We finally used 2790 unique tags to compute the tf-idf and cosine similarity. A dendrogram of 50 randomly selected tags is drawn in Fig 6.1.1 to show the network of tags built by cosine similarity. The distance between each tag pair in the dendrogram is the cosine similarity score. The pairs of tags that are placed closer and linked at the right-hand side of the graph have higher cosine similarity.

The outputs we got from the matrix are extremely important for the later part of our analysis, as they provided us a glance at how those similar tags were clustered. This will save us a lot of time by selecting some common naming conventions.

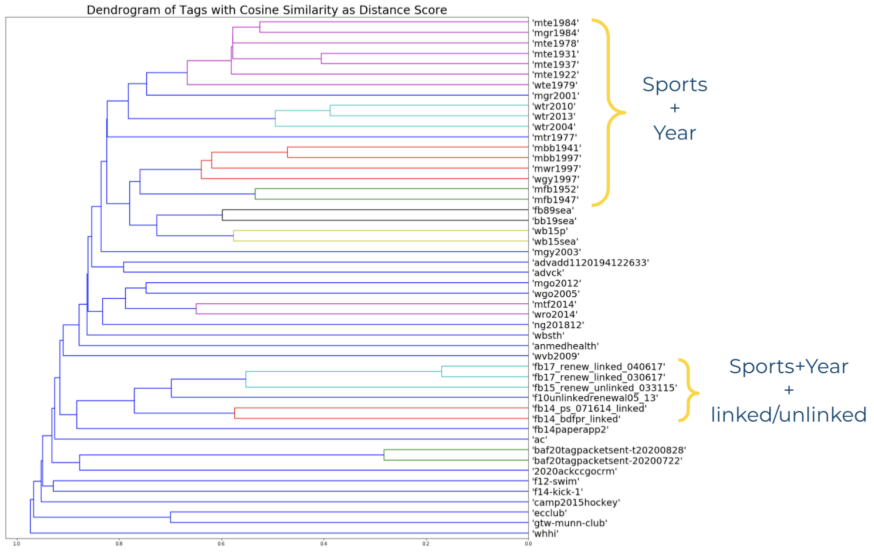


Fig 6.1.1 Dendrogram of tags with cosine similarity as distance score

It is also worth tracing these tags back to their origins, which is the account data. Below in Fig 6.1.2 is a summary of how many accounts in total were tagged by each subgroup of tags. It could help dig out potential user interests or characteristics. For example, the x-axis represents the top 15 popular tagging patterns, while the y-axis provides an approximate number of accounts that are using those tags. We can utilize this information along with the clustering graph to obtain some ideas about what the tagging preferences are for each organization. From Fig 6.1.2, we can see that over a million users were tagged with a string like "advcheck", which suggests that these users have been actively purchasing tickets or some related event items. Another commonly used tagging pattern is the one with the second-highest count, the "sports+year+season" tags. This pattern suggests that over 0.9M users watch the games repeatedly for multiple sports seasons.

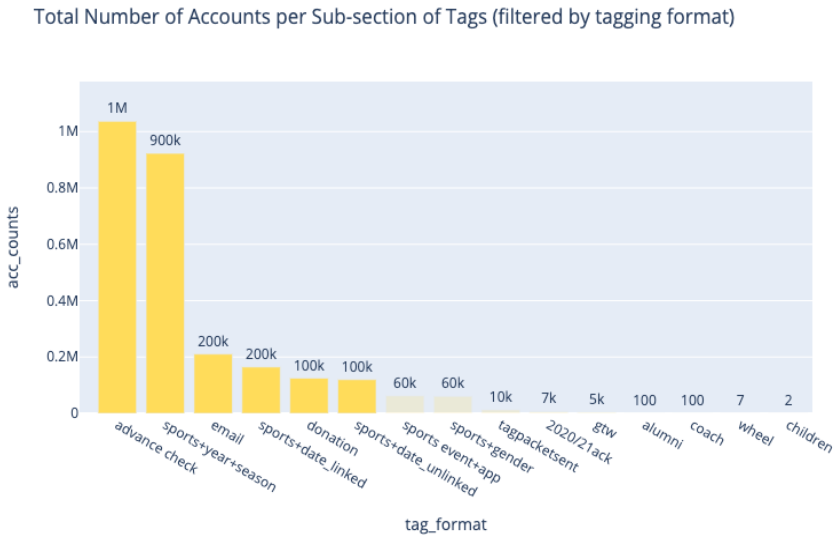


Fig 6.1.2 Bar plot: total number of accounts per sub-section of tags

6.2 Visualize The Tags' Co-Occurrence Frequency

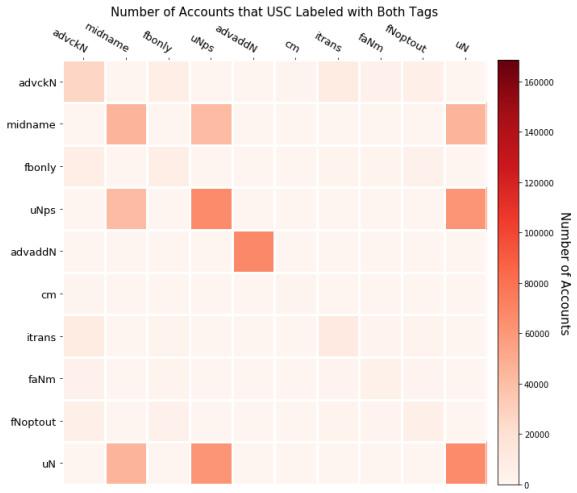


Fig 6.2. Bar plot: total number of accounts per sub-section of tags

We learned from the idea of the FP-Growth algorithm and based on that we built the heat map [Fig 6.2] to visualize how many customer accounts were labeled both tags by one selected organization. The darker the color, the more frequently two tags are assigned together. In this way, we are more able to see the

hidden relationship among different tags and build a simple rule-based tag recommendation system based on that.

6.3 Reasons for Tagging

We used cross-validation to confirm the usability of the models. Below is the logistic regression and random forest result to predict the tagging of two tags "fbonly" and "fbbb" from USC. There are in total 17 tag patterns in USC but these two show the best result among all the tags. They both have very high average sensitivity and average specificity to measure the model performance. Because the number of accounts with no tags is a lot which causes the most specificity to be above 0.95, we mainly look at average sensitivity. The ROC curves for both logistic regression

By looking at the feature importance graph for "fbbb" generated from random forest, we can tell that the ticket sellings regarding seasons "B19", "B18", and "F19" contribute the most to the model. Then we possibly conclude that the ticket sellings regarding the above seasons are related to the tagging of "fbbb" by USC.

The high performance of the model indicates that these tags are highly associated with the seasonal ticket selling action whereas the tags with low performance indicate little or no direct association between ticket buying action and the tagged patrons and donors' group.

Model	Sample Tag	Sensitivity	Specificity	F1 Score	AUC Score	Pseudo R-squared
Logistic Regression	fbonly	0.870	0.988	0.782	0.991	0.682
	fbbb	0.793	0.999	0.722	0.996	0.744
Random Forest	fbonly	0.887	0.998	0.915	0.996	-
	fbbb	0.936	0.999	0.943	0.999	-

Table 6.3. Modeling Result for two USC tags

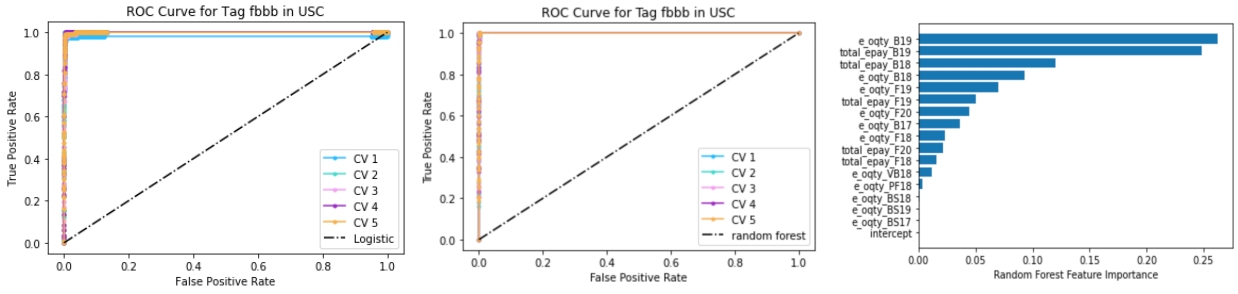


Fig 6.3. ROC Curves and Feature Importance Ranking for Tag "fbbb" in USC.
Left: Logistic Regression ROC curve. **Middle:** Random Forest ROC curve.
Right: Random Forest Feature Importance

7 Notebook Description

Our project workflow is extracting raw data from Snowflake using SQL script, uploading them to S3, reorganizing them, and saving cleaned versions locally. The whole process includes data retrieving, data exploration, and modeling. Each phase has been divided into several parts of subtasks in the Python Jupyter Notebook.

8 Members Participation

Task	Subtask	Sirui	Yihan	Yadi
Data Retrieve	-	33%	33%	33%
Data Exploration	Word Clouds	33%	33%	33%
	Keywords Exploration	50%	25%	25%
	PCA for Accounts	5%	80%	15%
	PCA for Organizations	33%	33%	33%
Data Modeling	Tag Naming Convention Pattern	33%	33%	33%
	Uncover Hidden Associations Among Tags	70%	15%	15%
	Reasons for Tagging	40%	30%	30%

There are several tasks and results that are shown in the Appendix but not in the report on previous pages. There are also many different tryouts along with the projects but we did not put those in considering the limited space. Our participation took those efforts into consideration as well.

9 Discussion and Conclusion

- From this project, we learned how to use Snowflake to retrieve data, store and access large data on AWS S3 Bucket. We also practiced some NLP Preprocess techniques. At the end of the project, we came up with three methods to derive three different insights from the tag data.

- We found it harder than we thought before that the real datasets are very messy and unorganized, but we have resolved it smoothly with the support of our sponsor.

- Unlike finishing a course project, we became more creative and open-minded in solving the open-ended questions and always came up with new ideas and approaches about the data.

- If we were in a research lab, we would try to build a tag recommendation system by combining the three techniques. This system will be able to recommend tags to Paciolan's clients so that to help these organizations and universities group their patrons and donors. By doing this, human labor is saved a lot and there will not be any discrepancies among people who assign tags.

Appendix

1 Additional Relevant Datasets

Ticket Transaction Item Events

	account_id	org_id	trans_no	vmc	date	event	season	e_oqty	e_pl	e_price	total_epay	total_cpay	total_fpay	total_spay
0	2282.0	BAYLOR	32193	2	2018-07-09	F04	F18	0	7.0	58.33	0.0	0.0	0.0	0
1	2282.0	BAYLOR	32193	2	2018-07-09	F05	F18	0	7.0	58.33	0.0	0.0	0.0	0
2	2282.0	BAYLOR	32193	2	2018-07-09	F06	F18	0	7.0	58.35	0.0	0.0	0.0	0
3	2282.0	BAYLOR	32193	3	2018-07-09	F01P	F18	0	2.0	75.00	0.0	0.0	0.0	0
4	2282.0	BAYLOR	32193	3	2018-07-09	F02P	F18	0	2.0	75.00	0.0	0.0	0.0	0

Appx 1. A sample of the raw Ticket Transaction Item Events

Interpretation

This dataset stores detailed transaction data related to event tickets. It contains information regarding the basic information of the ticket being sold: quantity, price, price level, and which event and season it belongs to. This dataset also shows the day of the transaction and the total payment.

Transaction Items

	org_id	account_id	allocation_id	allocationgroup_id	transaction_id	channel_id	receiveddate	fundtype	pledgeamount	paymentamount
0	TAM	10039	PWFBEC	12MF	18375	TAM	2016-06-09 05:00:00	C	0	0
1	TAM	14100	WFBEC	12MF	18603	TAM	2016-08-03 05:00:00	C	0	0
2	TAM	15698457	KFC-ADA_NUAfromNUB	KFC	18355	TAM	2016-06-03 05:00:00	C	0	0
3	CLEMSON	174062	ISF	AF	2964011	EV_CLEMSON	2021-04-09 23:03:52	S	45000	0
4	FSU	974798	BAF	BAF	1379631	EV_FSUSE	2021-04-10 03:24:05	S	0	130000

Appx 2. A sample of the raw Transaction Items Data

Interpretation

The Transaction Item dataset stores the transaction information between organizations and people that are merchandised on Paciolan. There are almost 11 million rows in this dataset. From this dataset, we can get how much money one person donated to an organization each year and where the money is allocated from the allocation id.

Donors

	org_id	account_id	is_deleted	donor_type	donor_status	years_of_donating
0	NEBRASKA	634566	False	Individual	Prospect	0.0
1	ARKANSAS	443476	False	Individual	Active	0.0
2	ARKANSAS	443495	False	Individual	Active	0.0
3	ARKANSAS	443579	False	Individual	Active	0.0
4	ARKANSAS	443554	False	Individual	Active	0.0

Appx 3. A sample of the raw Donors Data

Interpretation

Donors data stores all the accounts that are associated with a "donor" role. Since we are trying to find out the association between the keywords assigned to the users and their characteristics, we here assumed that the strong indicator (i.e. years_of_donating) could have some potential relationship to the keywords since fundraising service has been counting as a significant business branch of Paciolan.

2 PCA

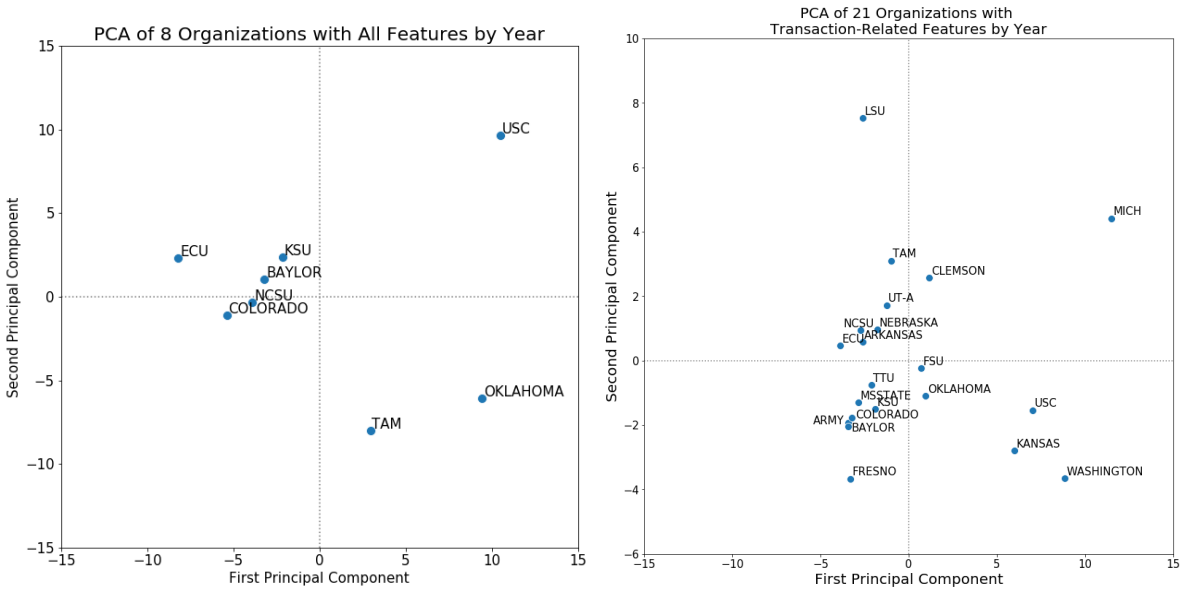
Get three aggregated datasets

For three transaction-related datasets (donor membership dataset, transaction items dataset, and ticket item event dataset), we first grouped each dataset by *org_id, account_id,* and *year*. Therefore, we got the total amount each donor/patron paid for each organization per year. Then, we grouped by *org_id* and *year* and ended up with a set of aggregated variables calculated from the previous step. To summarize, we have about 36 variables for the donation data; 49 variables for the transaction data; 42 variables for the ticket data. These variables were calculated by the 7 aggregate factors: sum, median, mean, min, max, variance, and size. They represented the aggregated amounts that each organization received per year.

Get one dataset with all features (inner join the three above)

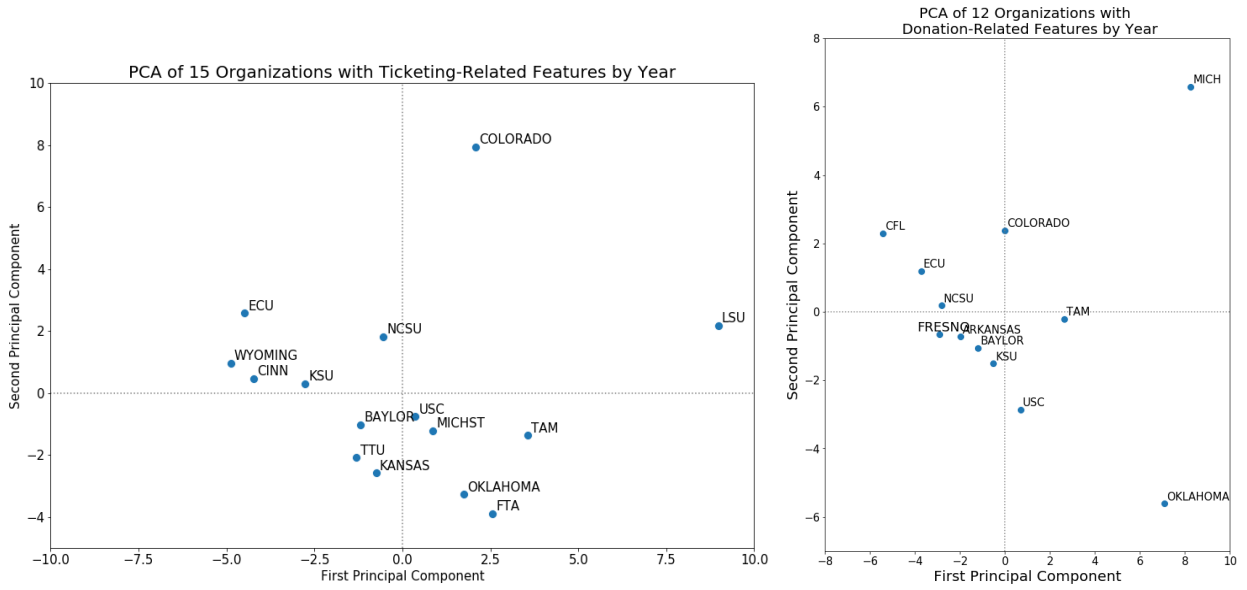
Each dataset contains the number of active accounts that have at least one transaction activity with an organization each year. Finally, we inner join all three datasets using the *org_id* so that we only kept the organizations that have all three relevant information, and ended up with a new dataset with 120 variables in total for all features data.

We used these four generated datasets to create four PCA plots as shown below. Appx 4.A is for all features data, and 8 organizations are in common in all three datasets. Appx 4.B is for transaction data. Appx 5.C and Appx 5.D are for organizations and donor membership.



A B

Appx 4. PCA plots of all features and transaction-related features of organizations (A & B)



C D

Appx 5. PCA plots of ticketing-related and donation-related features of organizations (C & D)

Interpretation

PCA plots show clusters of samples based on their similarity. We can easily see that some of them are clustered together and some of them become outliers. For example, in the second plot [Appx 4.B] about the transaction items dataset, MICH and LSU are obvious outliers, and we can see that they are not similar to any other organizations. This finding is proven in Appx 5.C that LSU is still an outlier. BAYLOR, NCSU, and KSU are always in one cluster in all four plots, meaning that they share some similar characteristics, and we can analyze them together.

After we checked the PCA loadings, we found out the total transaction amount and the total payment to large season events are key factors in PC 1 and PC 2. Large season events are those events that have "L" in the size column in the ticket season data [Appx 4]. PCA has helped us reduce the dimension of feature space in a large dataset, although there is not a very clear separation between organizations, we will stick with the 8 organizations in Appx 4.A and see if there are any trends of the user-defined fields that match the suggestions of these PCA results.

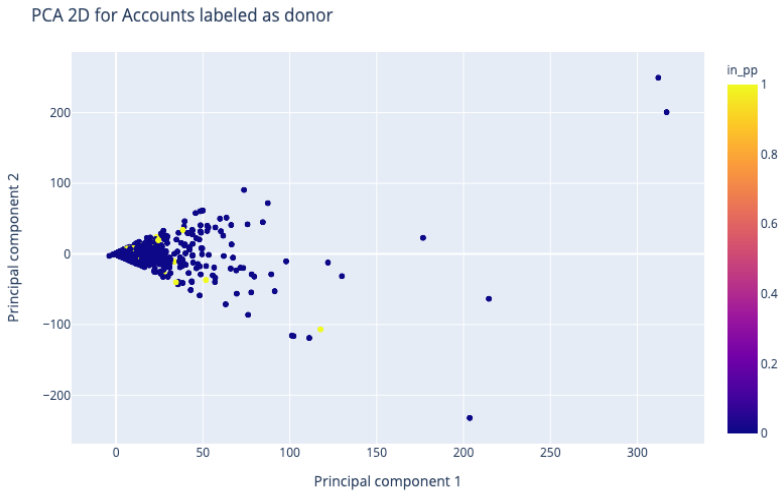
PCA of Accounts Data with Transaction-related Features

Besides testing out organization-wide features, we also tried reducing dimensionality account-wide. As we have noticed, different organizations apparently have diverging transaction/priority points policies, so doing the account-wide PCA was only an experiment to confirm our assumption that there could be very large variance between accounts. We first generated a PCA plot with those accounts enrolled in the priority program and we labeled them with colors representing different organizations. Looking at Appx. 6, we can see that there's apparently a huge variance between organizations, and, within-organization there's a large variance, too. This trumpet-shaped PCA pattern indicates that the features we selected (all transactional-related) have even large variance account-wide. The uncertainty here is whether this pattern could result in some tagging patterns.

variable	PC1	PC2
total_donated_2016	0.377000	-0.198740
total_donated_2017	0.382853	-0.202273
total_donated_2018	0.394821	-0.038577
total_donated_2019	0.170442	0.453306
total_donated_2020	0.155881	0.466829
total_pledged_2016	0.346490	-0.187793
total_pledged_2017	0.382842	-0.202271
total_pledged_2018	0.391791	-0.038840
total_pledged_2019	0.193065	0.400151
total_pledged_2020	0.157814	0.471441
total_paymentamount	0.129238	0.123235
total_pledgedamount	0.040629	0.102633
total_creditamount	0.037489	0.094325

Appx 6. **Left:** PCA of accounts enrolled in a priority program
Right: the feature loadings of PCs

Additionally, we tested whether or not donor status and enrollment of priority programs (PP) have any impact on each other. We labeled each donor account with colors either in bright yellow (not in PP) or in navy blue (in PP). Appx.7 below shows that users who are donors tend not to be enrolled in a PP (only a few are, most of them belong to MICH, TAM, and OKLAHOMA). Again, a trumpet-shaped PCA suggests that there's a large variance, but this time the pattern looks neater, and most scatters gathered around the origin. In this case, we realized that it is not suitable to consider PP data in our analysis as only a few organizations contain such information, which is imbalanced.



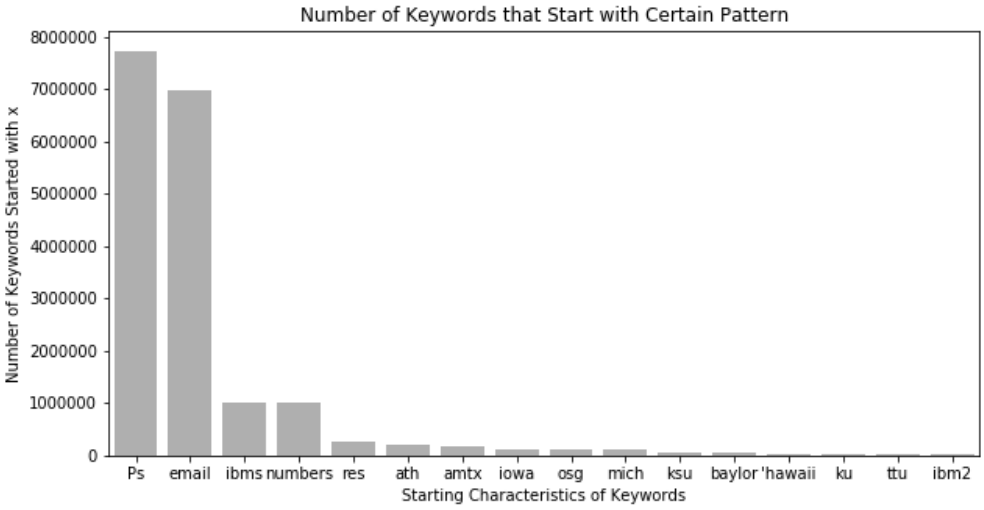
variable	PC1	PC2
total_donated_2016	0.313631	-0.376983
total_donated_2017	0.317302	-0.377242
total_donated_2018	0.307132	0.097672
total_donated_2019	0.330281	0.142659
total_donated_2020	0.244053	0.311015
total_pledged_2016	0.301958	-0.368917
total_pledged_2017	0.318676	-0.377335
total_pledged_2018	0.091598	0.071045
total_pledged_2019	0.319648	0.293707
total_pledged_2020	0.280779	0.370293
total_paymentamount	0.267788	0.174656
total_pledgedamount	0.236515	0.198756
total_creditamount	0.167907	0.109692

Appx 7. **Left:** PCA of donor accounts labeling with PP enrollment status
Right: the feature loadings of PCs

3 Keywords Exploration

As we explored the keywords, we found there's a great number of them that start with a specific pattern and end with numbers. As shown in Appx 8, more than 7 million keywords start with the character "P" and end with digits like "P92018", and "P000000." Besides keywords in such patterns, there are also a

large number of them that are alphanumeric but only the ending two characters are different from each other. For example, we saw several keywords look like “dahlbk”, “dahlel”, “dahli”, “dahlet”, “dahlic”, “dahlbn”, etc.



Appx 8. Number of keywords that start with a specific pattern