Rosetta Stone Action Plan and Supporting Information

MGSC 410

December 11, 2020

Written By:

Trent Commins, Kayelin Santa Elena, Nate Everett, Chris Kitahara,

Haley Noorani, Ayra Tusneem

Table of Contents

Action Plan

1. Identify goals for analysis
    a. Determine the most valuable subscribers
    b. Understanding the subscriber segments present in the database
    c. Identify the most likely subscribers who could be sold additional products or services
    d. Identify the subscriber profile of those not continuing with their usage of the product and identify the barriers to deeper subscriber engagement where possible
    e. Relevant business opportunities
2. Data Preparation
    a. N/A's, missing data
    b. Data Manipulation – Creating any necessary columns (ex: creating uniformed currency column)
    c. Normalization, feature scaling
3. Data Exploration
    a. Graphical and statistical models of exploration
4. Building Models
    a. Predict features based on certain features of data that is relevant for respective goals
5. Analyze Results from Models
    a. Analyzing results in order to gain insight into goals at hand
    b. Draw conclusions in order to present relevant business opportunities

Detailed Analytical Plan and Findings

Part 1

The main factor in determining the value of a subscriber is the subscription type of the user. Lifetime subscriptions, on average, bring in $200 while limited subscriptions average $50. When examined by user type this finding was upheld (appendix 1.1). However, there was an anomaly when examining user platforms, there are no lifetime subscribers on the app (appendix 1.2). In predicting the revenue from each user these same variables of the subscription platform and the type of subscription, lifetime or limited, also proved to be among the most important (appendix 1.3).

As subscription type and platform proved to be two of the most important features in predicting revenue generated from a user, further exploration was performed. Users who were engaged with emails, opening more than 20% of emails received, made up a great portion of lifetime subscribers for both consumers and non-consumers. Consumers that were limited time subscribers were mostly not engaged, however the majority of non-consumers who were limited time subscribers were never sent emails (appendix 1.4). This under reach of users classified as "other" represents one of the greatest shortcomings made evident in the data. This shortcoming is exacerbated by the fact that non-consumers favor web services, which leads to higher conversion to paying users for both consumers and non-consumers. Understanding non-consumers and web users became a major part of our analysis and the basis for some of our recommendations.

Part 2

Understanding the subscriber segments present in the database is imperative to data science. It gave a lot of information that became useful for moving forward with our analysis of data. We felt that looking into the language would give a comprehensive understanding of the data. Utilizing PivotTables and conditional formatting, we found:

Top Languages Count (Appendix 2.11, 2.12, 2.13):

	All	Lifetime Subscription	Limited Subscription
1.	ESP	ALL	ESP
2.	FRA	ESP	FRA
3.	ALL	FRA	ENG
4.	ENG	TGL	ITA
5.	ITA	ENG	DEU

Top Languages Per Country (Appendix 2.14, 2.15, 2.16):

	Europe	Other	US/Canada
1.	FRA	ESP	ESP
2.	ESC	ALL	FRA
3.	EBR	FRA	ALL
4.	DEU	EBR	ENG
5.	ITA	ITA	ITA

Top Languages based on the sum of Dollar_val and average of sub_len_mnth (Appendix 2.11):

	Sum of Dollar_val	Average of sub_len_mnth
1.	ALL	ALL
2.	ESP	PAS
3.	FRA	TGL
4.	ENG	LAT
5.	DEU	SVE

Overall, the language Spanish (ESP) was the most popular language and the top 5 were the same across all clusters excluding the average of subscription lengths (sub_len_mnth) which had vastly different results. When looking at the Country variable with subsets of the type of user (User.Type) and the type of subscription (Subscription.Type) and comparing it to the average dollar value (Dollar_val) and count of Country. Based on the table the "Other" region listed under "Country" purchases about twice as much as the other regions despite constituting about ⅓ of the dataset (Appendix 2.17).

We suggest that Rosetta Stone focus on analyzing the "other" category across the dataset. For Country, encompassing every country besides the United States, Canada, and Europe into

one category does not provide any valuable information. If anything, they need to add Asia as a category. China and India have about 4 billion people across both countries, they should not be encompassed as "other" when they have a big population and promise for growth. In addition, the "other" category in "User.Type." Businesses and schools are clientele with very different wants and needs and should not be grouped together either.

In addition to using PivotTables and conditional formatting, we used the K-Prototype clustering algorithm to create clusters. K-Prototype clustering is a combination of K-Means and K-Modes that can be applied to both numerical and categorical data. Using the K-Prototype clustering algorithm, we created six different clusters:

Cluster 1 - Out of all six, cluster 1 is by far the largest cluster (Appendix 2.1). The user type majority of cluster 1 is "other users (business/homeschool)," are platform "unknown" users, and come from "other" country (Appendix 2.2, Appendix 2.3, Appendix 2.6). Also, this is the first initial purchase of Rosetta Stone for most users in cluster 1 (Appendix 2.9, Appendix 2.10). Most users in cluster 1 are not subscribed to receive emails (Appendix 2.4) and compared to the average number of emails sent, many users did not open the emails (Appendix 2.7). Considering that cluster 1 has an extremely low number of app users and that they are not subscribed to receive emails, it's no surprise that most of the users do not have their push notifications on (Appendix 2.8).

Cluster 2 - Cluster 2 is the only cluster entirely made up of platform "app" users (Appendix 2.3). Despite being only made up of platform "app" users, many of the users do not have their push notifications on (Appendix 2.8). The user type majority of cluster 2 is "consumer" and come from the countries US and Canada while some are from Europe. (Appendix 2.2, Appendix 2.6). This is the first initial purchase of Rosetta Stone for most users in cluster 2 (Appendix 2.9, Appendix 2.10). There are some users that opted out to receive emails, but the majority of users in cluster 2 are subscribed to receive emails (Appendix 2.4). However, compared to the average number of emails sent, many users did not open the emails (Appendix 2.7). Because these users have opted in to receive emails, they are being sent a ton of emails. This is an issue because if not many users are opening them, that means the emails being sent are going straight to the spam folder and never reach the user.

Cluster 3 - Cluster 3 is the smallest cluster by far, out of the 6, (Appendix 2.1) so was disregarded for most analysis. The user type majority of cluster 2 is "consumer" and come from Europe. (Appendix 2.2, Appendix 2.6). The cluster is made up mostly of platform "app" users, but they have a very small quantity of users who have the push notifications on (Appendix 2.8). Most users in cluster 3 are not subscribed to receive emails (Appendix 2.4) and compared to the average number of emails sent, many users did not open the emails (Appendix 2.7). Unlike all of the other clusters, the majority of users have renewed their subscription, and this is not their first initial purchase (Appendix 2.9).

Cluster 4 - Opposite of cluster 2, cluster 4 is entirely made up of platform "web" users (Appendix 2.3). Because it is only made up of platform "web" users, there is a good number of users who have opted in to receive emails (Appendix 2.7). Most users in this cluster are from the US and Canada, but there is a good amount from Europe. Surprisingly, cluster 4 has the highest amount of push notification users (Appendix 2.7). For most users in cluster 4 this is their first initial purchase of Rosetta Stone (Appendix 2.9, Appendix 2.10). Like cluster 2, compared to the average number of emails sent for users in cluster 4, many users did not open the emails

(Appendix 2.7). Because these users have opted in to receive emails, they are being sent a ton of emails. This is an issue because if not many users are opening them, that means the emails being sent are going straight to the spam folder and never reach the user.

Cluster 5 - Majority of users in cluster 5 are from the US and Canada (Appendix 2.6). Similarly to clusters 3 and 6, the majority of cluster 5 is made up of "consumer" user type and are majority platform "app" users. (Appendix 2.2, Appendix 2.3). While there is a sizable number of users who have not subscribed to receive emails, there are still more users who have opted in to receive them. Despite having almost as many opted-in email subscribers as clusters 2 and 4, cluster 5 receives way less emails (Appendix 2.7). We would expect them to have opened more emails since they are receiving less and not being spammed, but cluster 5 opened less emails than clusters 2 and 4.

Cluster 6 - Cluster 6 has the same amount of initial purchase and renewal purchase users and almost the same amount of opted-in email subscribers and non-opted-in email subscribers (Appendix 2.9, Appendix 2.4). Most users in this cluster are from Europe but there is also a large amount from "other" and the US/Canada. Similar to clusters 3 and 5, the majority of cluster 6 is made up of "consumer" user type and are majority platform "app" users. (Appendix 2.2, Appendix 2.3). Also, many of the users do not have their push notifications on (Appendix 2.8). Very similar to cluster 1, and cluster 4, compared to the average number of emails sent, many users did not open the emails (Appendix 2.7)

[Please refer to Appendix 2.18 and Appendix 2.19 for the following paragraph.] The most important findings using K-Prototype clustering algorithm came from clusters 1-2, and 4-6. We deemed cluster 3 to be irrelevant as it only accounted for 0.14% of the dataset. In cluster 1 the average dollar value per use is $65.91 with the highest proportion of user type "other" and lead platform "unknown" users at 94%. Cluster 2 has the highest average dollar value per user at $65.63 with the highest proportion of email subscribers at 71% and the highest email open percentage at 26%. Cluster 4 (labeled cluster 3) has a high average dollar value per use at $65.63 with the highest lead platform "web" users at 100%. Cluster 5 (labeled cluster 4) has the lowest average dollar value per use at $2.94 with the lowest email open percentage at 17%, which is 3% below the 20% threshold. Cluster 6 (labeled cluster 5) has a very low average dollar value per user at $10.42 with a low proportion of email subscribers at 55%.

Part 3

Determining which clients can be sold additional goods is useful for all aspects of the business, but it is most helpful in determining where to spend money on marketing and advertising. There is no singular metric that can determine who is most likely to buy a product versus another customer, but by pulling in data from various aspects, even those on free trial or demo users can give specific insight towards eager customers. The variables in the data that immediately stuck out to us were regarding the notification systems e.g., emails sent, emails clicked, emails opened, etc. We intuitively believed that users that were both email subscribers and had push notifications on would be most likely to engage with the content on their devices, and in turn be more likely to purchase additional goods or services.

To test this intuition, we deployed firstly a multinomial logistic regression model that aimed at predicting a feature we engineered. The feature was a combination of the following variables: email subscriber, push notifications, open count, send count, click count, unique click

count, unique open count. Depending on the aggregate from the normalized values of the previous variables, this new variable could take on a 1 representing little to no engagement with notifications through 5 representing very active engagement with notifications. This model attempted to classify this new variable and resulted very poorly. The variable did *not* have enough significance relative to the other variables and was therefore discarded from further analysis.

Following this model, we used a new binary logistic regression model to predict if a user had push notifications on or off. We decided to predict push notifications as consumers with push notifications on, on average paid almost double the amount than those who did not receive notifications (Appendix 3.1). The variables we included to determine whether a user had push notifications on were language, country, user type, preferred platform, subscription length, subscription type, whether they used demos or free trials, had auto-renew on, and another variable that we created, Dollar_val. Dollar_val is a column that we engineered to sum all money spent by each user converted to USD. Our research and visualizations displayed that consumers with push notifications on had spent the most by far, almost twice the number of other users. Additionally, we noted that customers with push notifications opened emails and clicked on notifications much more often than those who did not have push notifications (Appendix 3.2).

This model was successful in predicting users with push notifications or not for the same reasons we had intuited. With over 95% accuracy we were confident that these users would be correctly classified and could be used as a target for marketing and advertisements. In order to validate the results of this model we tested an additional random forest classifier to predict the same variable. The classifier had an equal amount of predictive power as the logistic classifier as verified by its area under the curve of the receiver operator function. We believe these results can translate into business decisions about allocating the correct number of resources to the correct customers. Customers who are actively engaging with the content will see more advertisements and in turn, Rosetta Stone can expect a greater return on their MROI (marketing return on investment).

Part 4

In examining individuals who did not continue their usage of Rosetta Stone's services and thus did not become paying users, user type and user activities both proved to have predictive power in determining who would become a paying user and who would only use the free services. Variables regarding user types were the most predictive. Users classified as 'other,' made up of commercial and homeschooling users, are most likely to be paying users. This is likely due to the nature of their purchase being used for business or formal education, while consumers use Rosetta Stone on a more discretionary basis. Users accessing through the web rather than the app are also more likely to be paying users, part of this is since almost all non-consumer users use the web service while almost as many consumers use the app as use the website. However, exploring this further shows that even within the consumers segment of the user base, vastly more individuals who use the website purchased services than those who use the app (appendix 4.1). Examining user activities predictors are the same between consumer and non-consumer users. Engagement with email is the most poignant factor. Users with higher levels of engagement (measured at opening more than 20% of received emails, appendix 4.2) over users who did not receive or open emails. A user being an email subscriber or free-trial user

was shown to hold predictive power. The variable ‘Demo User’ did show a positive correlation with usage which is likely due to the fact that the demo is app based and app-based users are far less likely to purchase anything, explaining why a user who did use the demo pointed to them being less likely to make a purchase. This assumption was confirmed when examining just app-based users, specifically looking at an app-based user being more likely to make a purchase, thought only slightly.

We suggest that commercial and homeschooling users become the main target of Rosetta Stone. The nature of their use makes them much less cyclical than consumers, however they are an under reached segment. They are more likely to be unreached by email and far less likely to have a known lead. Beyond non-consumers, users that highly engaged consumers or otherwise are more likely to be paying so should be targeted. The total count of recorded activities was also somewhat relevant to predicting whether a user would become a paying one, though it was not terribly important. So, we recommend targeting users that are engaging emails by opening over 20% of received emails.

Analyzing patterns of lifetime users prior to the lifetime subscription can help in predicting lifetime users, thus more analysis should be conducted on patterns with lifetime users (Appendix 1.2). Doing so can identify the patterns lifetime users tend to display and specific targeting can be done.

Barriers to higher engagement are low engagement with emails and the discretionary nature of the purchases form consumers. App based conversion from non-paying to paying user is another standout feature of data exploration and modeling. Further analysis should be done to find faults within the app in order to make improvements to increase engagement (Appendix 4.3). It could be flaws in the apps themselves, of which there are many, but it could also be part of Rosetta stones strategy to target web-based users as to avoid paying charges to Apple or Google for using their app stores.

Part 5

Three core business applications were found in our analysis, one for operations, one for data management, and the last for marketing. For the operations segment we suggest analysis and restructuring of the mobile apps offered. They are far less profitable than web services and are almost completely unutilized by non-consumer users. A strategy to either push app users to the web or increase conversion of free users to premium is needed. Data management regarding countries and user types is another area of concern. Without data on a third of the user's locations sufficient analysis of this metric is not possible. Additionally, non-consumers, classified as ‘other’ make up home schooled and corporate clients, however they are combined into one field. Users typed as ‘other’ proved to be one of the most valuable segments and without better understanding of this segment they will remain underreached. Marketing teams will also be able to use a deeper understanding of non-consumer users to prepare better advertisements and reach these users, which have largely not been received emails. Marketing should also avoid users with low engagement (under 20% of sent emails opened) as they are less likely to be paying users. Email subscription also has proven to be valuable so incentives to increase email subscription should also be undertaken

Division of Labor

Section 1: All Members

- We met as a group and discussed our findings to come to a consensus for section 1.
- Performed modeling to identify key features of the most valuable users
    - Random forest proved to be the most effective
    - Linear regression and elastic net models were also constructed but did not yield the same quality results as the random forest
- Visualizations to determine relation of key variables to the rest of our data

Section 2: Kayelin and Haley

Kayelin Santa Elena

- Data exploration
    - Explored
        - Dimensions of each dataset
        - Unique variables in all categorical data in the subscribers dataset
        - Summary statistics for app users dataset
        - How many missing variables in each dataset
    - Created
        - Visuals and graphs using geom_bar, geom_histogram, and geom_point
- to further explore the data
- Data prep
    - Subscribers dataset
    - Created a subset of the subscribers dataset containing 20% of the actual dataset
        - Used this subset to run and test my model before using all 40,000 data points
    - Variable manipulation
        - Created dummy variables for all binary data
        - One hot encoding for all nominal data
            - Factor one hot dummy variables to work with model
- Modeling
    - Check for optimal number of clusters using elbow method
        - Optimal number was 6 clusters
    - Use function kproto() from "clustMixType" package to create 6 clusters
        - Average model run time was 7 minutes
    - Put variable's cluster ID into own column in the dataset
- Graphs and Analysis
    - Save final dataframe to excel using "writeexl" package for further analysis
    - Used geom_bar, geom_point, and facet_wrap to create graphs and analyze clusters
    - Used excel to create graphs and analyze clusters

Haley Noorani

1. First, we analyzed the data and tried to gain an understanding of the data by looking at it in Excel and through some exploratory analytics
2. After we received my section for the project, we looked again at the data set and tried to think of some ideas that would be beneficial to Rosetta Stone
    a. We decided to focus on the language variable in the data set and really get a complete understanding of the variable and comparing it to other variables
3. First, we wanted to make a clustering model for the purchase amount that the user paid for and compare it to how long the user was on the platform. We also wanted to compare it to language but that turned out to not be possible with the methods we wanted to use which is a Gaussian Model which hard assigns data points to a cluster. While the results were good, the visualizations were not good and did not offer anything vital.
4. We then decided to go back to the dataset and look at the breakdown of how many observations there were for each language. With a dataset this big, it was easier to make a Pivot Table in Excel than make graphs and use conditional formatting to identify the top and bottom results in order to see the results.
5. We then added the comparison to a variable that the group created called Dollar_val which converted all of the observations under the Currency variable. Under this column was the sum of Dollar_val which summed the amount of money paid during this period that was paid under the breakdown of per language.
6. The next variable we added to compare language to was sub_len_mnth. We decided to use the average amount of this variable in order to get a better gage of how long a customer is using the program to learn a language. Do some languages takes longer to learn than others? This was when the data started to look a little skewed based on the variable Subscription.Type due to the option to have a lifetime membership that lasts until 2099. As a result, we added a filter to the table to be able to remove the lifetime observations to see the true average amount of time that a person was subscribed to a particular language.
7. While we also compared language and sum of Dollar_val, we also decided to compare it to the average amount of Dollar_val to see how much people spend per language on average.
8. To get more out of the one Pivot Table, we added a filture for the Country variable to see which languages are more popular in each region based on all of the variables we already assigned to compare against language. We also added a filter for User.Type to be able to see which languages are more popular in consumers compared to other.
9. We then decided to take language out of the picture and make a new Pivot Table comparing Country to another variable to gain a better understanding of each geographical section represented in the dataset.
    a. We compared Country to average Dollar_val to see which section gave the most money to Rosetta Stone.
    b. We also added another comparison to see the count breakdown per country.
10. We added subsections to the Country variable of User.Type and Subscription.Type

Section 3: Chris and Nate

Chris Kitahara

- Initial Data Exploration
    - Created graphs and visualizations of raw data
    - Identified unnecessary or irrelevant variables
    - Analyzed simple graphs to better understand the data
- Data Prep
    - Clean data
        - Delete unnecessary columns
        - Replaced values of 'NA' with a mean value
        - Utilize new column of Dollar_val to simplify currencies
- In-Depth Analysis
    - Create visualizations with new cleaned data and assessed correlations
    - Analyze and apply results of models
    - Identify key features that maximize potential profit
        - Gauge most profitable customers based on logistic regression and random forest modeling

Nate Everett

- Data Preparation
    - Prepared data through a pipeline of normalization, label encoding, and interpolation
    - Employed feature engineering by creating new variables with the information gathered from certain features
- Data Exploration
    - Investigated the distribution of features of interest and their correlations with other similar features using correlation matrices and graphs
    - Used both graphical and statistical methods of data exploration to gather insight into the data shapes
- Model Building and In-Depth Analysis
    - Created both multinomial and logistical regression models to predict features based on certain features of the data
    - Utilized K-Means Clustering to evaluate the aggregation of data points with one another
    - Validated results from regression models using random forest decision tree
    - Evaluated modeling results using AUC, ROC, and confusion matrix performance metrics

Section 4: Trent and Ayra

Ayra Tusneem

- Data Preparation
    - Prepared data through normalization, label encoding, and feature scaling.
    - Created new features where needed (dummy variables, averages, etc)

- Data Exploration
    - Created graphs to gauge relationships amongst variables and gather initial insight
    - Divulged into correlations amongst features based on VIF scores and used methods such as pandas profiling, etc to understand data
- Model Building
    - Created linear regression, random forest, KNN, Gaussian model, and decision tree models to predict features based on data exploration
    - Evaluated the model's results using AUC, ROC, and accuracy performance metrics
        - Low accuracy performance with Gaussian model
        - High accuracy performance with linear regression, random forest, KNN, and decision tree
        - Used linear regression and decision tree results for analysis
    - Analyzed results with Trent to gage insight on the barriers to subscription and identifying user profile

Trent Commins

- Data preparation
    - Converted app activity data set into one showing counts of each activity by user ID, a form more compatible with the rest of our data
    - Indexed currencies to U.S. Dollar for analysis
    - Identified and fixed discrepancy in recording of app payments
    - Examined percent of emails opened by users and its relation to paying and not paying users
        - Found that 20% was a good cutoff line between the two
        - Created feature to measure email engagement
- Exploration
    - Created graphs establishing relationships between features both derived and original
- Model building
    - Part 1
        - Built random forest, linear regression, and elastic net models to determine most valuable subscriber
            - Evaluated on R2, RMSE basis
    - Part 4
        - Build logistic regression, naive bays and random forest models to explore what users were likely to be paying users
            - Logistic regression yielded the best results
                - Evaluated on accuracy and AUC of the ROC

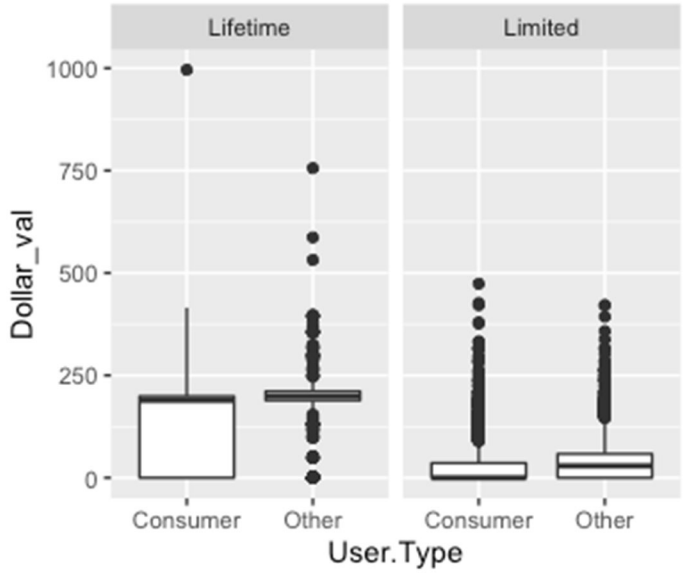
Section 5 (and presentation): All

- Met as a team to discuss findings and business application
- Determined three main areas of focus, as previously discussed
- Established a narrative for the presentation
- Collaborated to convert rough findings and visualizations into presentation

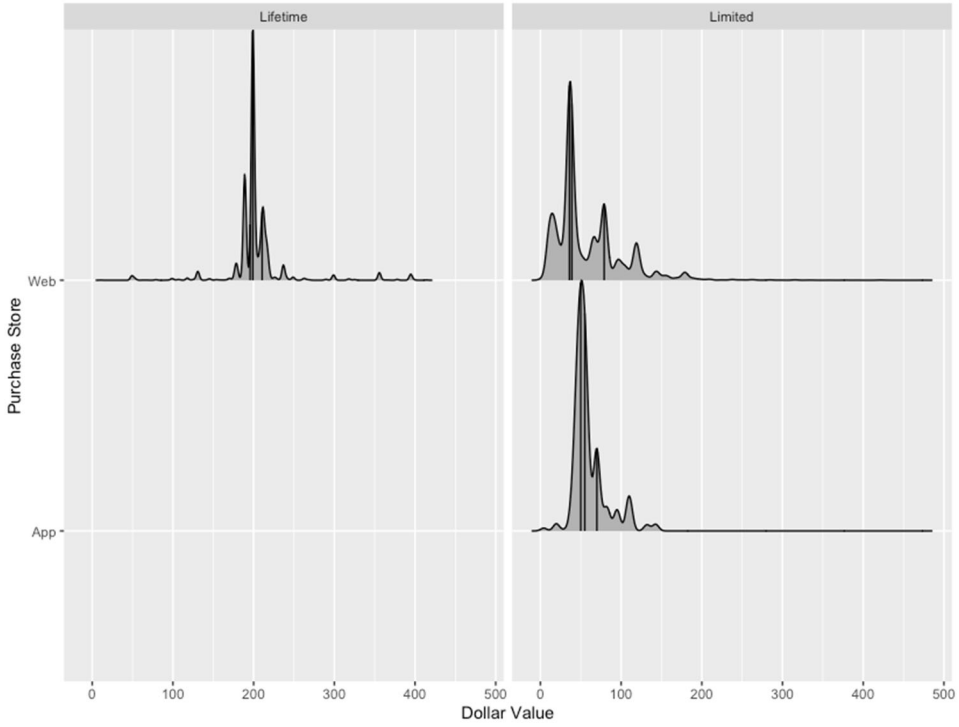
Code and Other Supporting Information

Appendix

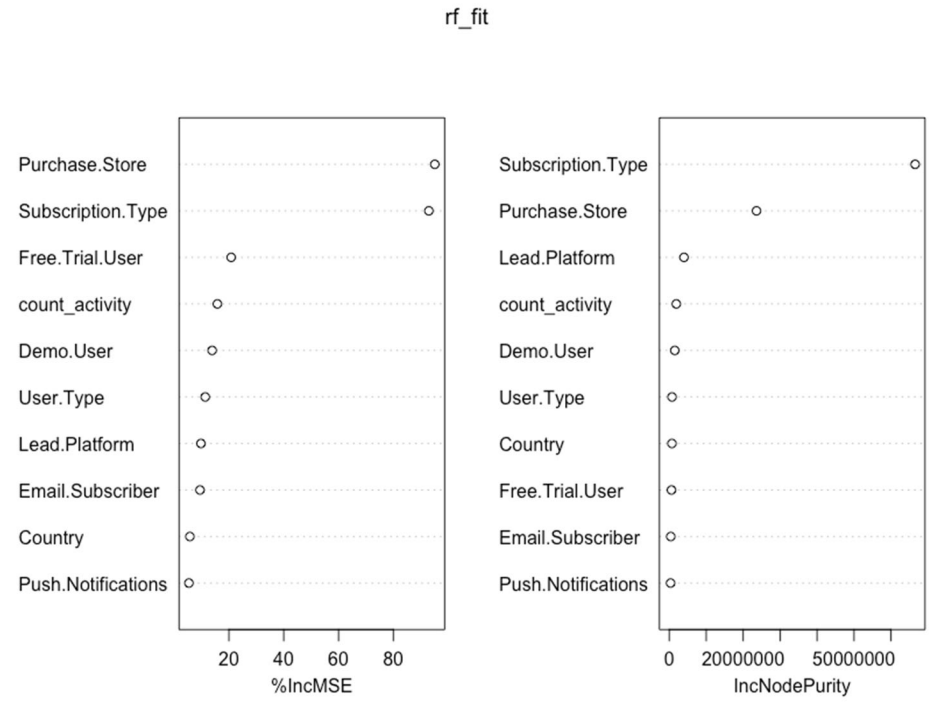
Appendix 1.1



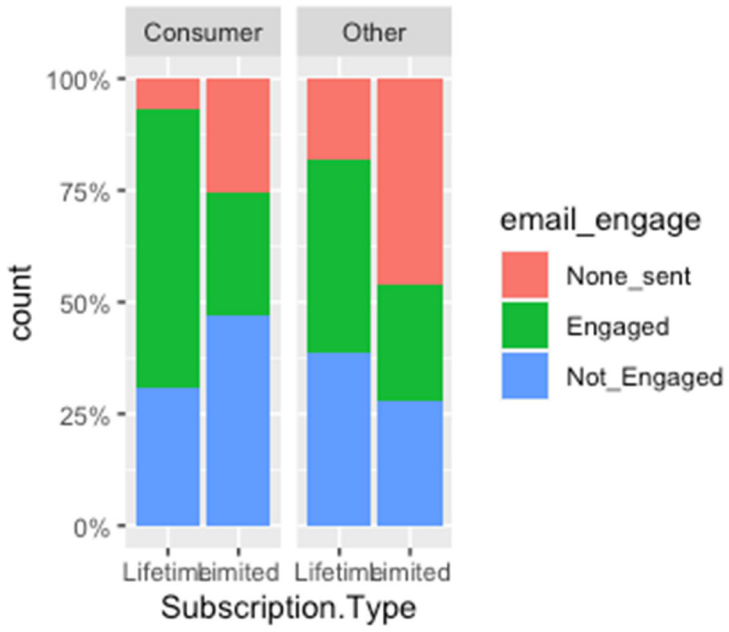
Appendix 1.2



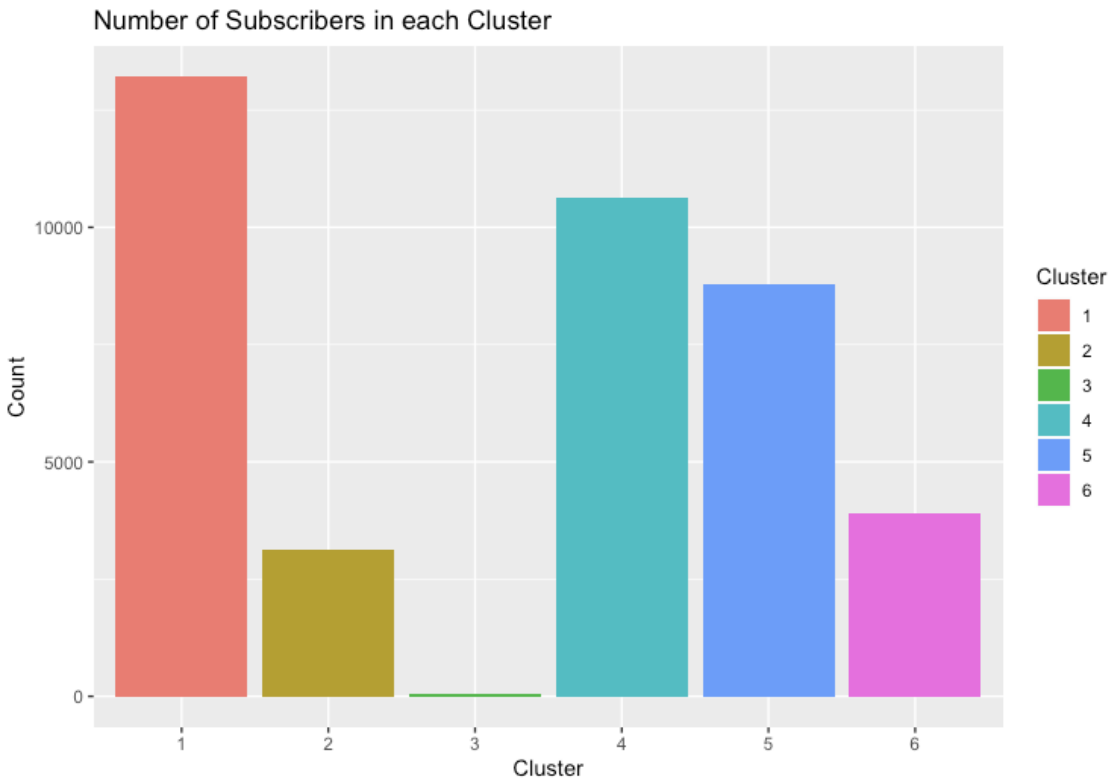
Appendix 1.3



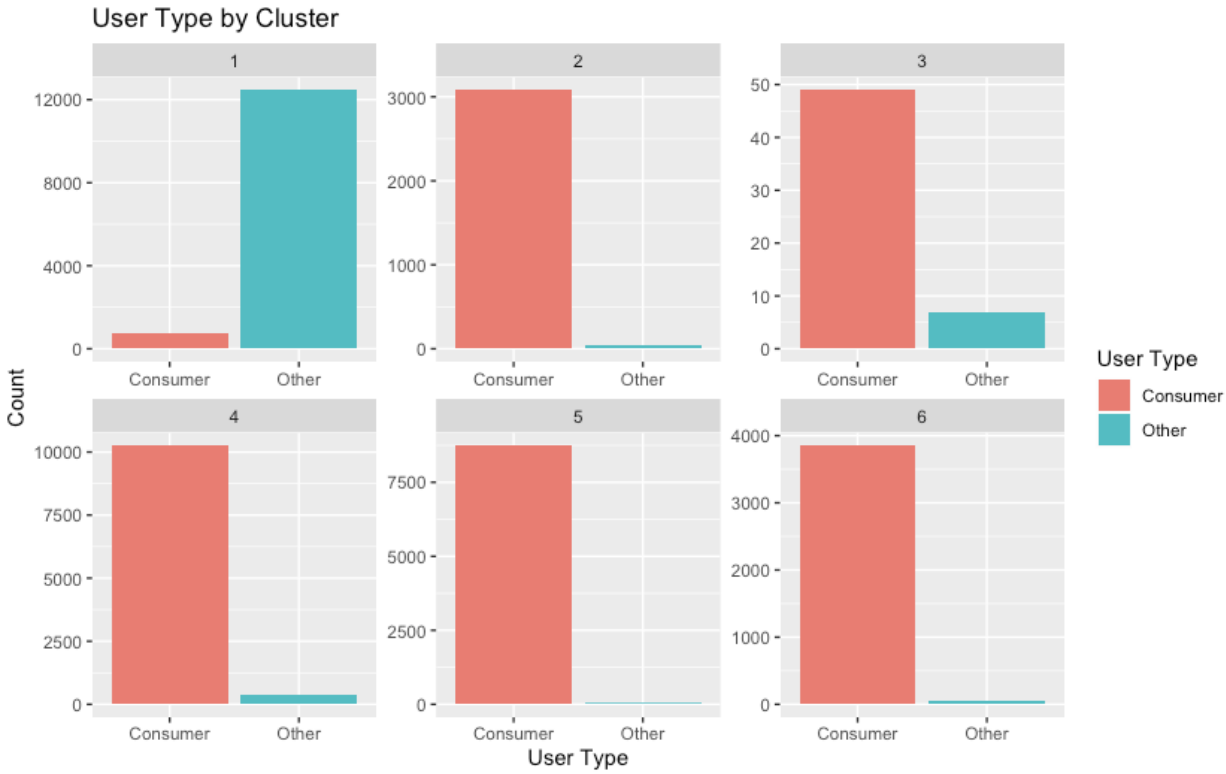
Appendix 1.4



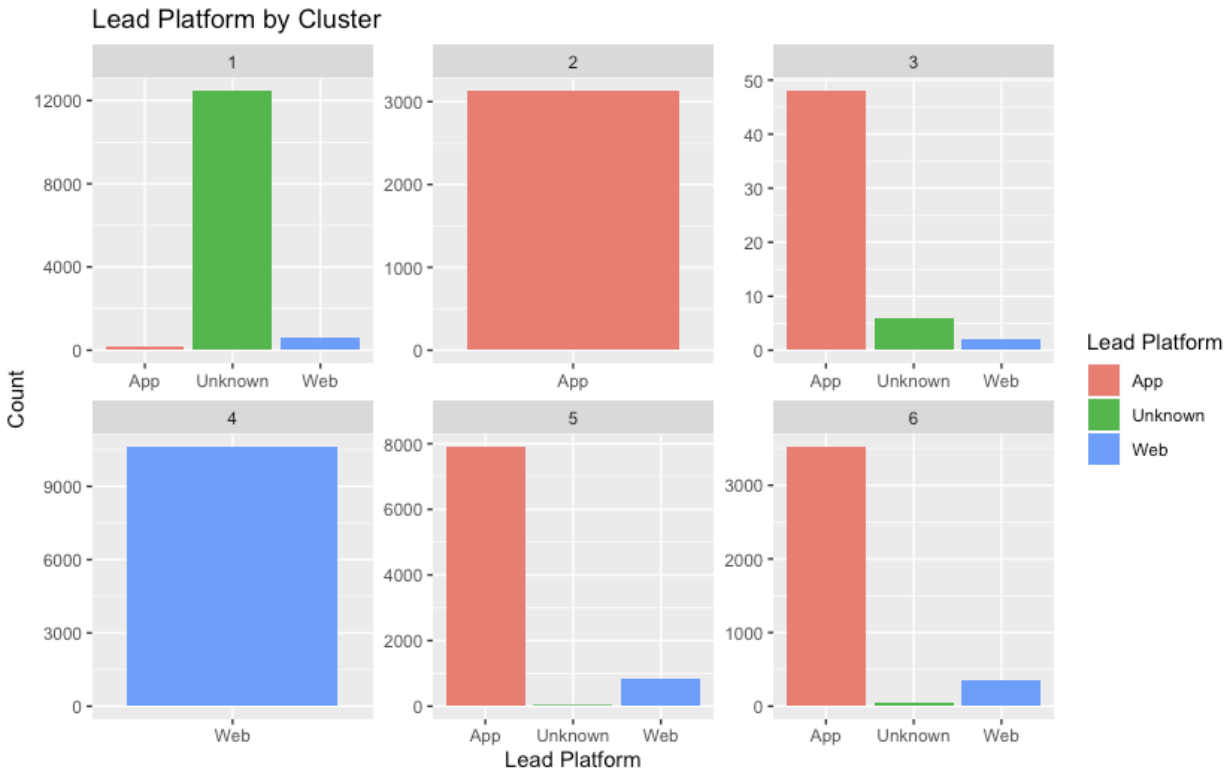
Appendix 2.1



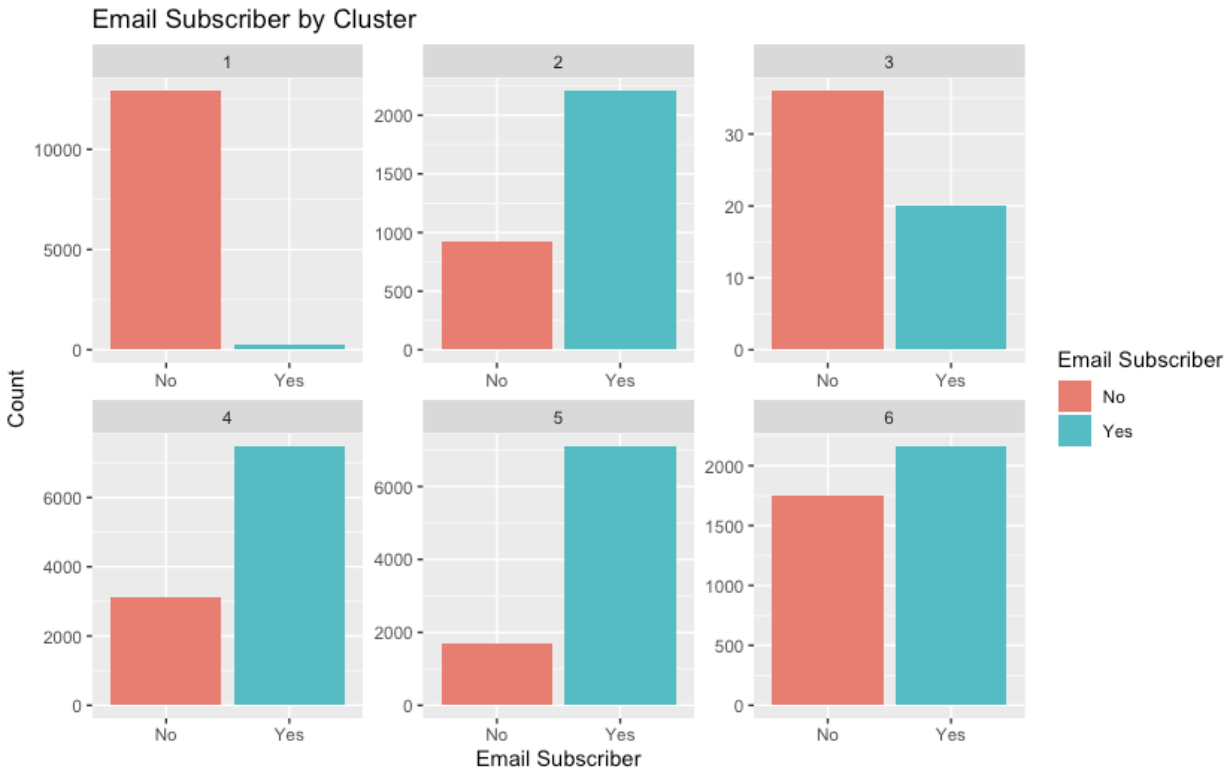
Appendix 2.2



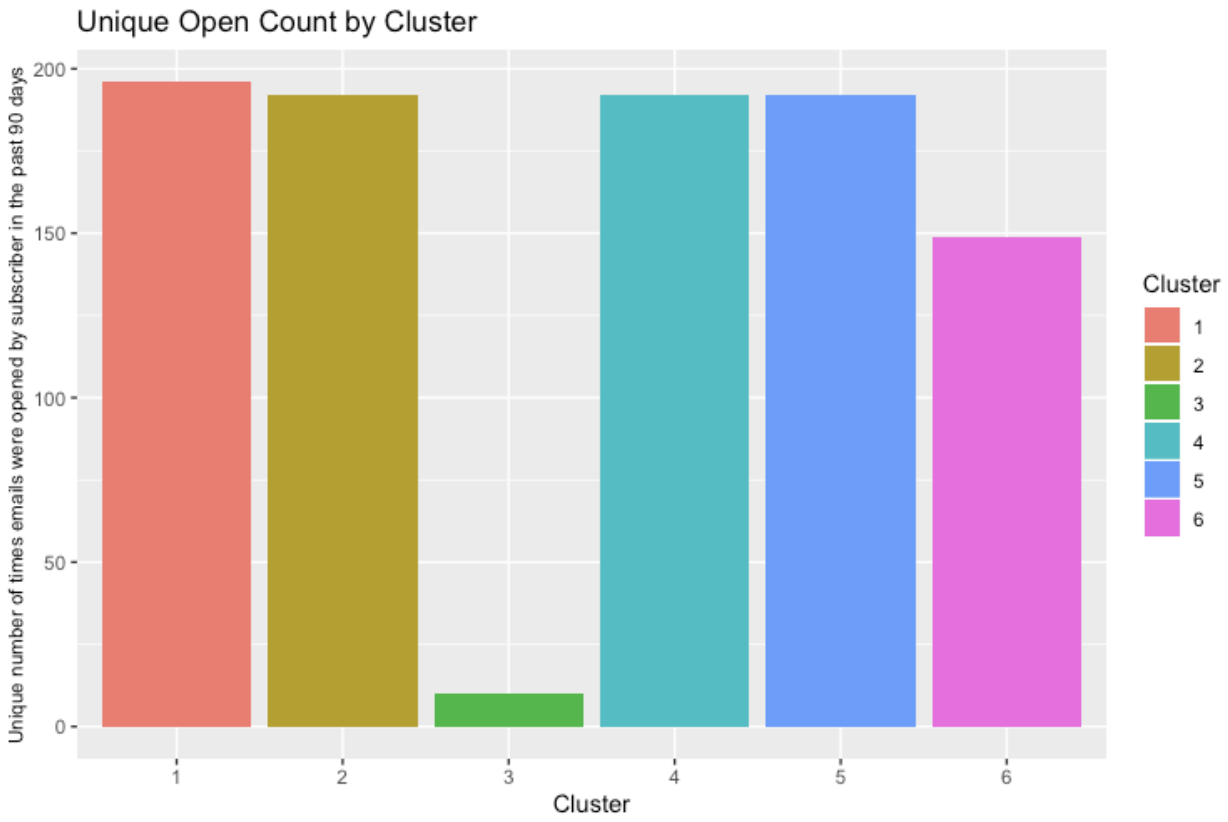
Appendix 2.3


Lead Platform by Cluster
1
2
3
4
5
6
Count
Lead Platform
App
Unknown
Web
Lead Platform

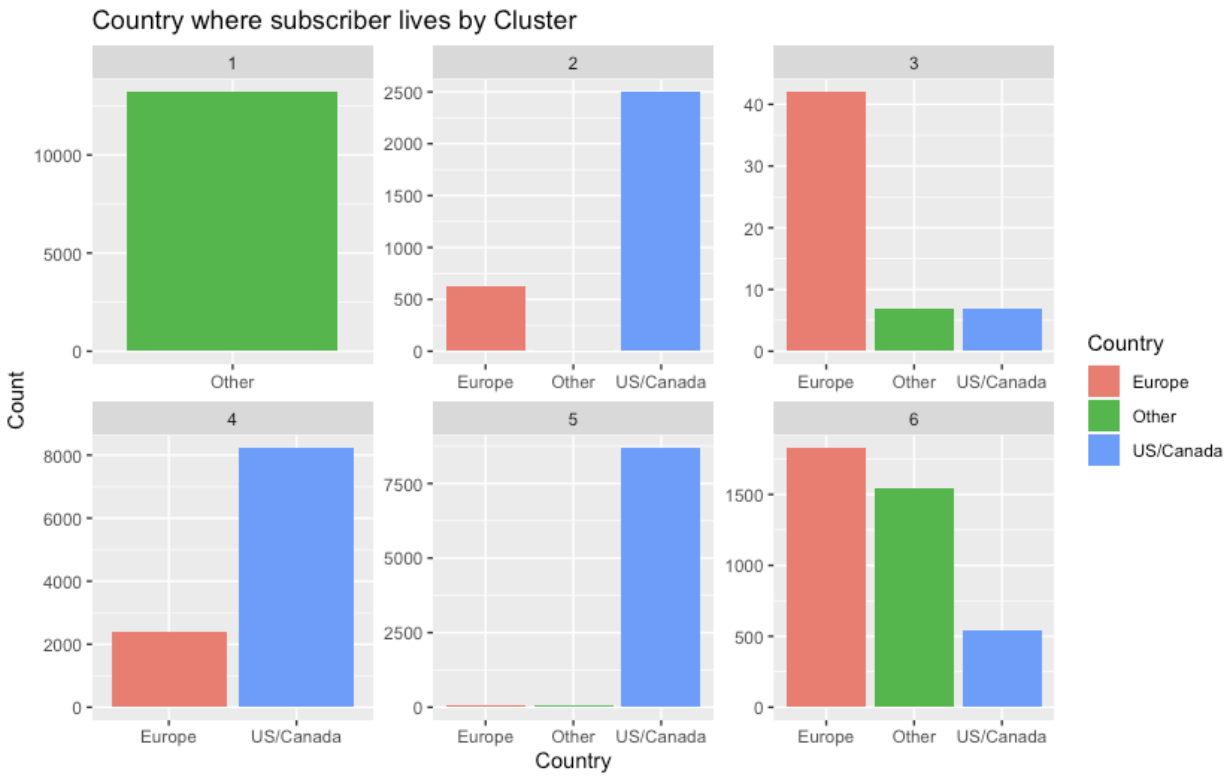
Appendix 2.4


Email Subscriber by Cluster
1
2
3
4
5
6
Count
Email Subscriber
No
Yes
Email Subscriber

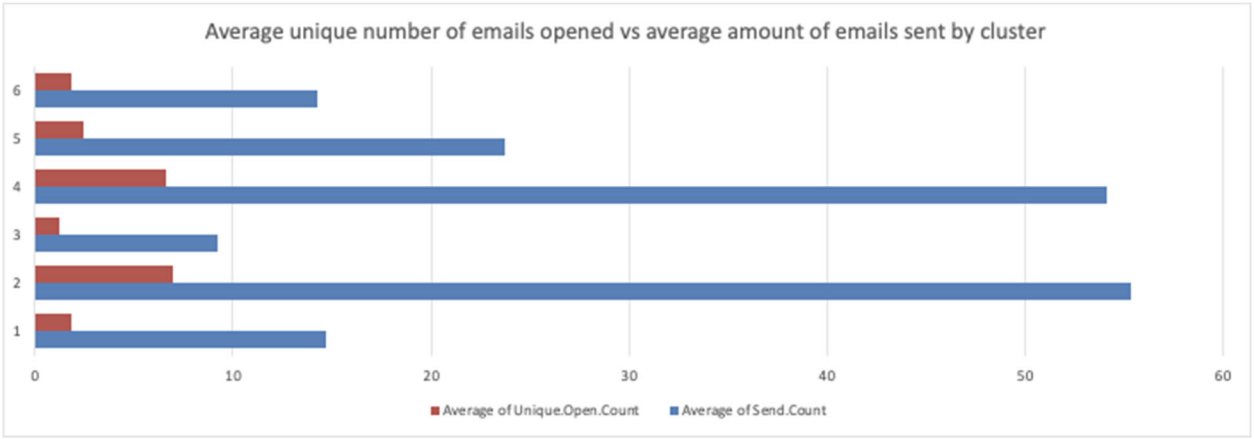
Appendix 2.5



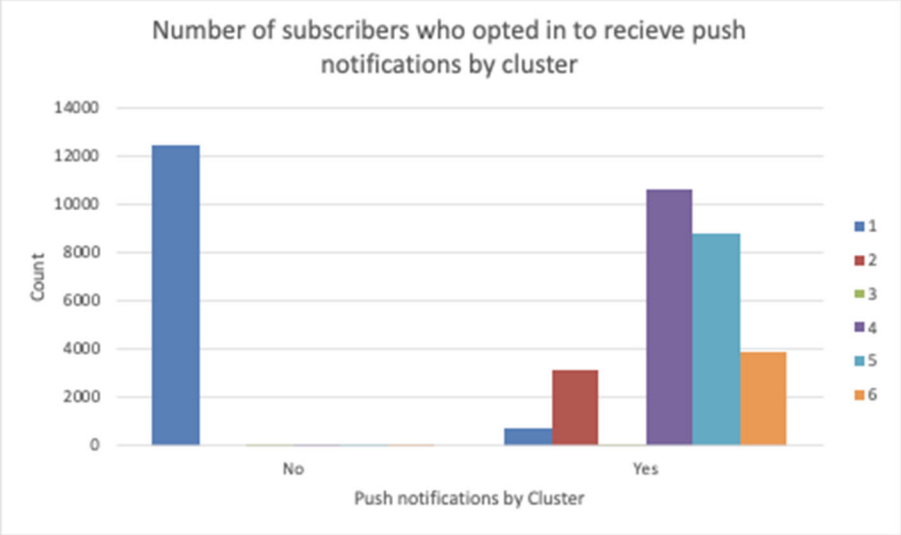
Appendix 2.6



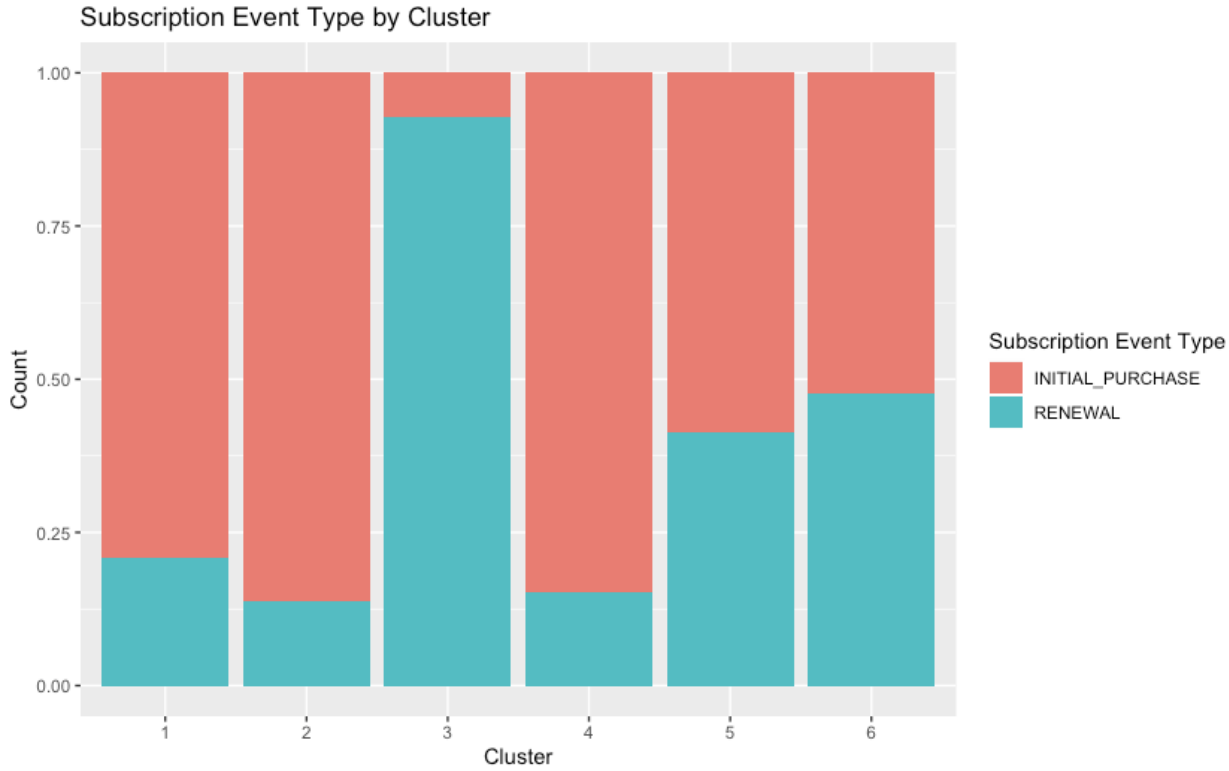
Appendix 2.7



Appendix 2.8



Appendix 2.9



Appendix 2.10

Appendix 2.11 All Subscription Type, All Country, All User.Type

	A	B	C	D	E
1	Subscription.Type	(All)			
2	Country	(All)			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	6272	$1,021,075.22	850.2217053	162.7989823
7	ARA	656	$18,509.87	60.44871516	28.21626381
8	CHI	924	$24,106.44	70.32502319	26.08922414
9	DAR	27	$0.00	8.195767196	0
10	DEU	1978	$55,131.59	38.51498628	27.87239351
11	EBR	1235	$41,305.50	51.42845575	33.44575198
12	ENG	3350	$59,767.35	33.39405117	17.84100082
13	ESC	1715	$50,703.35	52.70147855	29.56463749
14	ESP	9805	$229,762.51	40.89900561	23.43319885
15	FAR	196	$5,200.01	44.81377551	26.53065714
16	FRA	4474	$110,804.09	46.80171946	24.76622446
17	GLE	109	$3,471.40	102.9023591	31.84772325
18	GRK	468	$11,669.10	46.51900183	24.93396394
19	HEB	376	$9,127.95	44.27203647	24.27645665
20	HIN	187	$5,696.57	68.40813598	30.46296791
21	IND	19	$0.00	7.54887218	0
22	ITA	2444	$54,537.44	41.42409984	22.31482889
23	JPN	1281	$34,182.39	60.1155905	26.68414465
24	KIS	51	$0.00	9.542717087	0
25	KOR	548	$15,556.05	58.39494265	28.38695346
26	LAT	133	$7,848.70	14.09854995	59.01278195
27	NED	305	$7,653.85	66.17587822	25.09460041
28	PAS	2	$0.00	14.71428571	0
29	POL	237	$5,673.73	34.82263412	23.93978531
30	POR	621	$15,705.70	50.03312629	25.2909879
31	RUS	792	$19,937.13	57.03828463	25.17314199
32	SVE	410	$25,399.25	190.4372822	61.94939073
33	TGL	722	$36,534.12	235.2036506	50.60127074
34	TUR	184	$5,194.92	47.92643634	28.23325
35	URD	23	$0.00	9.347826087	0
36	VIE	159	$4,311.75	34.46203953	27.11792726
37	**Grand Total**	**39703**	**$1,878,866.00**	**177.3388859**	**47.32302336**

Appendix 2.12 Lifetime subscription, All Country, All User.Type

	A	B	C	D	E
1	Subscription.Type	Lifetime			
2	Country	(All)			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	5170	$935,996.43	1028.157709	181.0437971
7	ARA	33	$3,063.31	1034.374459	92.82757576
8	CHI	55	$3,383.56	1035.938961	61.51927273
9	DEU	57	$4,723.46	1034.119048	82.8677193
10	EBR	50	$3,830.39	1034.288571	76.6078
11	ENG	82	$1,484.17	1035.679007	18.09959512
12	ESC	72	$4,036.00	1035.84375	56.05555556
13	ESP	309	$21,568.98	1034.370897	69.80252427
14	FAR	7	$447.00	1030.061224	63.85714286
15	FRA	166	$13,044.39	1035.892427	78.58066265
16	GLE	10	$516.56	1034.567857	51.656
17	GRK	17	$1,478.06	1034.628151	86.94470749
18	HEB	13	$491.32	1036.002747	37.79384615
19	HIN	11	$1,283.03	1039.058442	116.6390909
20	ITA	78	$4,757.67	1034.966575	60.99576923
21	JPN	65	$5,716.51	1035.864835	87.94630769
22	KOR	27	$2,504.97	1034.464286	92.77666667
23	NED	17	$391.72	1035.758403	23.04235294
24	POL	6	$0.00	1041.863095	0
25	POR	25	$1,870.18	1033.431429	74.8072
26	RUS	37	$1,823.96	1034.647683	49.29616486
27	SVE	71	$8,446.62	1034.071429	118.9665437
28	TGL	156	$15,060.72	1035.222985	96.54307692
29	TUR	7	$636.00	1035.142857	90.85714286
30	VIE	4	$451.47	1036.616071	112.8675
31	**Grand Total**	**6545**	**$1,037,006.48**	**1029.602063**	**158.4425486**

Appendix 2.13 Limited subscription, All Country, All User.Type

	A	B	C	D	E
1	Subscription.Type	Limited			
2	Country	(All)			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	1102	$85,078.79	15.44027094	77.20397985
7	ARA	623	$15,446.56	8.86035313	24.79383477
8	CHI	869	$20,722.88	9.210216998	23.846816
9	DAR	27	$0.00	8.195767196	0
10	DEU	1921	$50,408.13	8.973376961	26.24056968
11	EBR	1185	$37,475.11	9.957564798	31.62456852
12	ENG	3268	$58,283.19	8.244918255	17.83451222
13	ESC	1643	$46,667.35	9.617946266	28.40374515
14	ESP	9496	$208,193.53	8.571413528	21.92434022
15	FAR	189	$4,753.01	8.323129252	25.14819471
16	FRA	4308	$97,759.70	8.689124884	22.69259476
17	GLE	99	$2,954.84	8.794733045	29.84688721
18	GRK	451	$10,191.04	9.273202407	22.59653014
19	HEB	363	$8,636.63	8.755509642	23.79236281
20	HIN	176	$4,413.55	7.742491883	25.07696023
21	IND	19	$0.00	7.54887218	0
22	ITA	2366	$49,779.77	8.6699523	21.03963306
23	JPN	1216	$28,465.88	7.957941729	23.40944022
24	KIS	51	$0.00	9.542717087	0
25	KOR	521	$13,051.08	7.811694543	25.05005854
26	LAT	133	$7,848.70	14.09854995	59.01278195
27	NED	288	$7,262.13	8.943576389	25.21574002
28	PAS	2	$0.00	14.71428571	0
29	POL	231	$5,673.73	8.665739023	24.56159791
30	POR	596	$13,835.52	8.783197507	23.21396558
31	RUS	755	$18,113.17	9.128949858	23.99095412
32	SVE	339	$16,952.63	13.7469448	50.00774513
33	TGL	566	$21,473.40	14.70362191	37.93886479
34	TUR	177	$4,558.92	8.883979015	25.75659887
35	URD	23	$0.00	9.347826087	0
36	VIE	155	$3,860.28	8.6	24.90503506
37	**Grand Total**	**33158**	**$841,859.52**	**9.11213842**	**25.38933337**
38					

Appendix 2.14 All Subscription Type, Europe, All User.Type

	A	B	C	D	E
1	Subscription.Type	(All)			
2	Country	Europe			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	393	$56,968.66	617.8857688	144.9584198
7	ARA	132	$4,301.30	103.4066558	32.585625
8	CHI	116	$3,188.22	35.93318966	27.48462931
9	DAR	1	$0.00	3.285714286	0
10	DEU	401	$12,359.50	27.20858568	30.82170599
11	EBR	585	$21,249.31	59.45744811	36.32360564
12	ENG	308	$6,851.06	38.85899814	22.24370726
13	ESC	627	$20,055.17	54.14194577	31.98592338
14	ESP	271	$6,374.09	35.32551397	23.52062989
15	FAR	29	$756.69	8.07635468	26.09284138
16	FRA	664	$19,059.28	55.75419535	28.70373579
17	GLE	17	$773.96	135.262605	45.52708235
18	GRK	112	$3,479.61	57.13552296	31.06792197
19	HEB	41	$876.63	34.75	21.3812122
20	HIN	33	$1,024.07	103.288961	31.0323
21	IND	1	$0.00	6.5	0
22	ITA	396	$10,370.36	48.46103896	26.18777433
23	JPN	139	$5,257.96	74.68550874	37.82701727
24	KIS	3	$0.00	13.01190476	0
25	KOR	72	$3,243.23	92.86904762	45.04485694
26	LAT	2	$0.00	7.75	0
27	NED	104	$2,851.20	48.19848901	27.41537622
28	POL	55	$1,356.27	26.81363636	24.65942182
29	POR	72	$2,530.66	24.37053571	35.14806389
30	RUS	140	$4,677.22	60.79132653	33.40867857
31	SVE	120	$8,376.68	208.5696429	69.8056775
32	TGL	32	$2,288.19	111.6160714	71.50601875
33	TUR	44	$1,291.77	10.64123377	29.3584
34	URD	1	$0.00	3.214285714	0
35	VIE	25	$685.99	10.29571429	27.439424
36	**Grand Total**	**4936**	**$200,247.07**	**100.2877836**	**40.56869335**

Appendix 2.15 All Subscription Type, Other, All User.Type

	A	B	C	D	E
1	Subscription.Type	(All)			
2	Country	Other			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	2625	$479,574.45	886.9741633	182.6950277
7	ARA	253	$8,458.92	52.80688876	33.43445257
8	CHI	359	$12,428.62	63.68095901	34.62009926
9	DAR	12	$0.00	6.726190476	0
10	DEU	732	$23,439.36	37.99565769	32.02098156
11	EBR	617	$19,341.75	46.15408659	31.34805737
12	ENG	1579	$29,027.10	33.98586357	18.38322005
13	ESC	690	$23,894.68	45.68074534	34.62996844
14	ESP	2883	$106,063.41	46.28654427	36.78925225
15	FAR	77	$2,690.91	36.24165121	34.94683636
16	FRA	1643	$50,629.39	45.61427267	30.81521273
17	GLE	34	$1,271.35	67.30357143	37.39268824
18	GRK	166	$5,484.92	46.0408778	33.04167366
19	HEB	157	$4,748.61	41.4733849	30.24591083
20	HIN	71	$2,004.34	36.53923541	28.23012817
21	IND	11	$0.00	7.405844156	0
22	ITA	776	$23,153.42	41.67258837	29.83687539
23	JPN	533	$17,187.98	57.84876709	32.24762439
24	KIS	20	$0.00	8.071428571	0
25	KOR	213	$7,139.06	55.35429242	33.5167172
26	LAT	75	$5,177.30	13.96428571	69.03066667
27	NED	107	$3,546.05	105.231976	33.14069159
28	PAS	2	$0.00	14.71428571	0
29	POL	99	$2,950.27	40.22113997	29.80065776
30	POR	249	$7,380.65	49.50874928	29.64117562
31	RUS	263	$7,664.16	47.51887561	29.1413057
32	SVE	145	$11,888.89	168.5258621	81.99233724
33	TGL	257	$18,400.08	255.8158699	71.59563346
34	TUR	80	$2,702.96	46.22633929	33.7870325
35	URD	8	$0.00	6.616071429	0
36	VIE	63	$2,272.83	39.81405896	36.07666032
37	**Grand Total**	**14799**	**$878,521.46**	**199.2760997**	**59.36356903**
38					

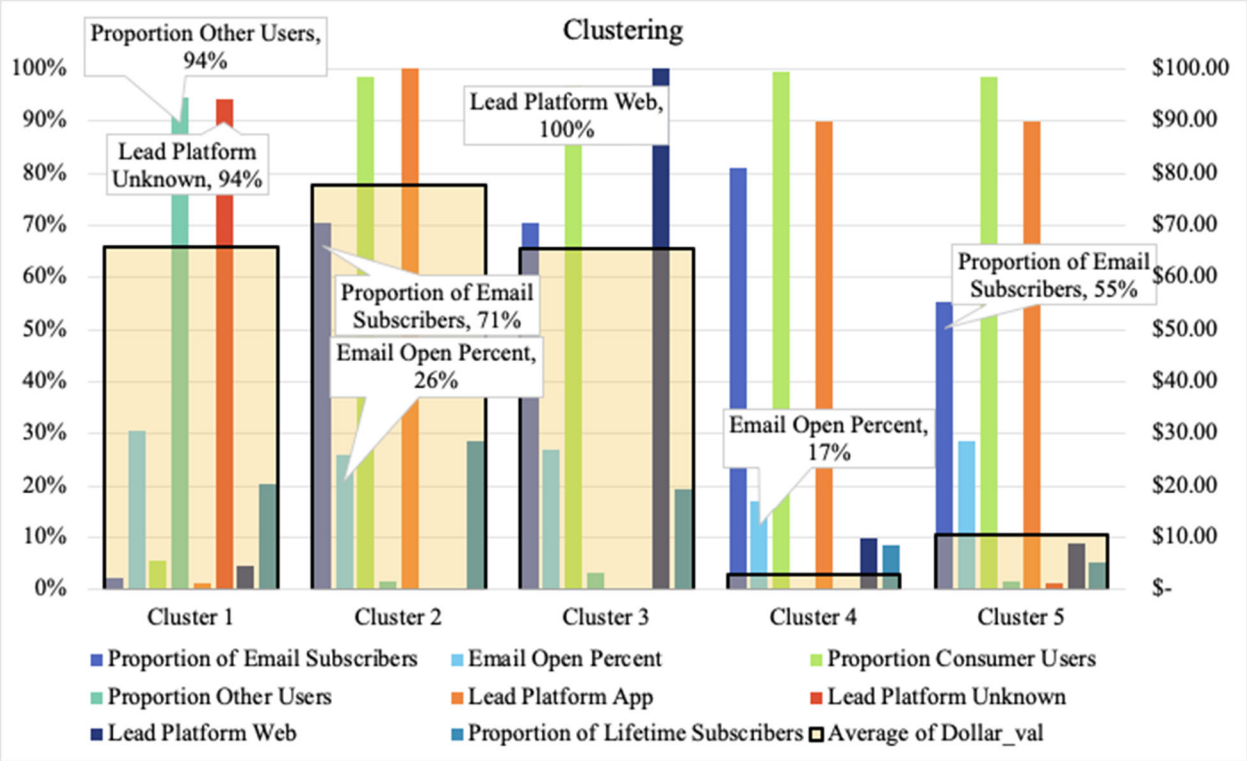
Appendix 2.16 All Subscription Type, US/Canada, All User.Type

	A	B	C	D	E
1	Subscription.Type	(All)			
2	Country	US/Canada			
3	User.Type	(All)			
4					
5	**Row Labels**	**Count of Language**	**Sum of Dollar_val**	**Average of sub_len_mnth**	**Average of Dollar_val2**
6	ALL	3254	$484,532.11	848.6337585	148.9035372
7	ARA	271	$5,749.65	46.65880337	21.21642089
8	CHI	449	$8,489.61	84.52251034	18.90781841
9	DAR	14	$0.00	9.806122449	0
10	DEU	845	$19,332.73	44.33038884	22.87897249
11	EBR	33	$714.44	7.711038961	21.64978788
12	ENG	1463	$23,889.19	31.60479934	16.32890393
13	ESC	398	$6,753.50	62.60382268	16.96859573
14	ESP	6651	$117,325.01	38.7907726	17.64020595
15	FAR	90	$1,752.41	63.98531746	19.47122222
16	FRA	2167	$41,115.41	44.95886347	18.97342553
17	GLE	58	$1,426.09	114.2857143	24.58775921
18	GRK	190	$2,704.57	40.67857143	14.23457913
19	HEB	178	$3,502.71	48.93378812	19.67814607
20	HIN	83	$2,668.17	81.80120482	32.14662651
21	IND	7	$0.00	7.923469388	0
22	ITA	1272	$21,013.67	39.08176101	16.52017915
23	JPN	609	$11,736.45	58.7740441	19.27167504
24	KIS	28	$0.00	10.22193878	0
25	KOR	263	$5,173.76	51.4197447	19.67209138
26	LAT	56	$2,671.40	14.50510204	47.70357143
27	NED	94	$1,256.60	41.60828267	13.36808511
28	POL	83	$1,367.20	33.69061962	16.47223855
29	POR	300	$5,794.39	56.62738095	19.31463385
30	RUS	389	$7,595.75	62.12357694	19.52634977
31	SVE	145	$5,133.68	197.3426108	35.40468966
32	TGL	433	$15,845.85	232.1031013	36.59548978
33	TUR	60	$1,200.19	77.53571429	20.00309667
34	URD	14	$0.00	11.34693878	0
35	VIE	71	$1,352.94	38.222334	19.05542583
36	**Grand Total**	**19968**	**$800,097.47**	**180.1271177**	**40.06898377**

Appendix 2.17

Row Labels	Average of Dollar_val	Count of Country
Europe	40.56869335	4936
Consumer	40.40516242	4907
Lifetime	136.1233012	430
Limited	31.21177406	4477
Other	68.23925517	29
Other	59.36356903	14799
Consumer	24.24782597	2241
Lifetime	120.8086975	238
Limited	12.77429256	2003
Other	65.630043	12558
US/Canada	40.06898377	19968
Consumer	39.34126509	19586
Lifetime	144.7051048	3291
Limited	18.06158441	16295
Other	77.38075916	382
Grand Total	47.32302336	39703

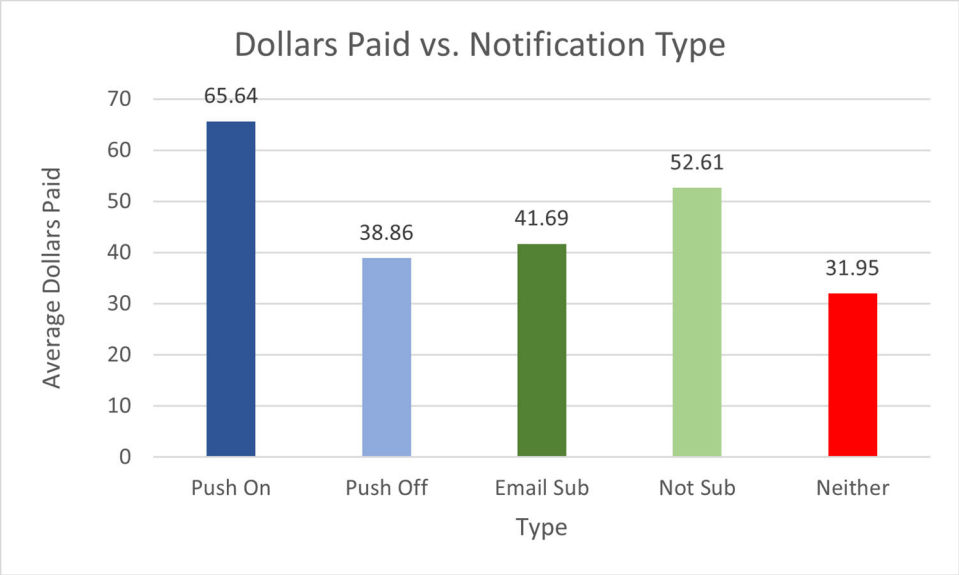
Appendix 2.18



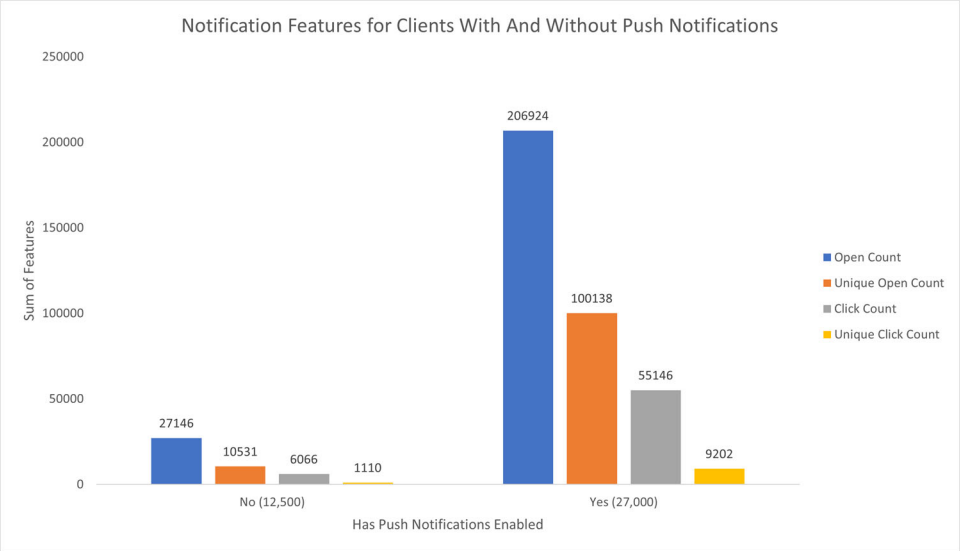
Appendix 2.19

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Proportion of Email Subscribers	2%	71%	70%	81%	55%
Email Open Percent	31%	26%	27%	17%	29%
Proportion Consumer Users	6%	99%	97%	100%	99%
Proportion Other Users	94%	1%	3%	0%	1%
Lead Platform App	1%	100%	0%	90%	90%
Lead Platform Unknown	94%	0%	0%	0%	1%
Lead Platform Web	5%	0%	100%	10%	9%
Proportion of Lifetime Subscribers	20%	29%	19%	8%	5%
Average of Dollar_val	**$ 65.91**	**$ 77.69**	**$ 65.63**	**$ 2.94**	**$ 10.42**

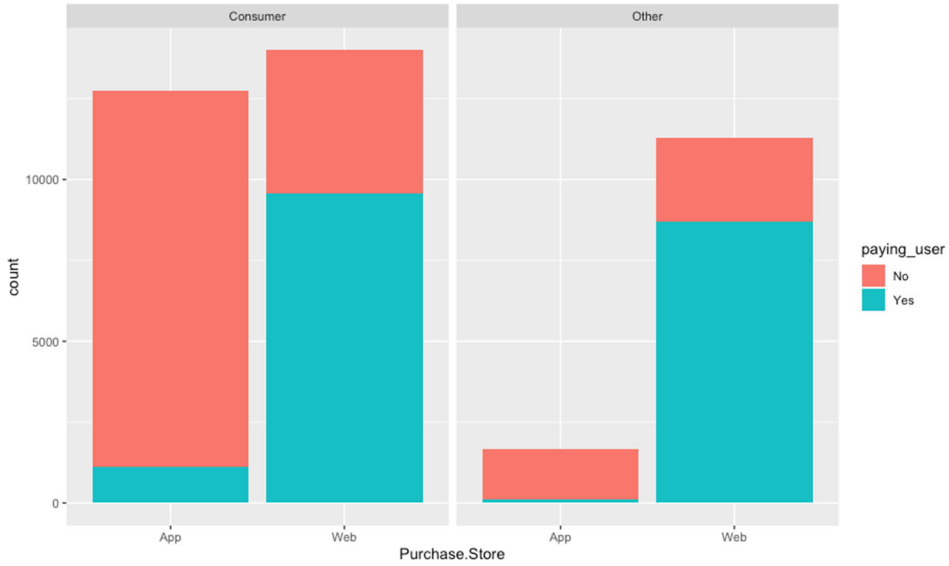
Appendix 3.1



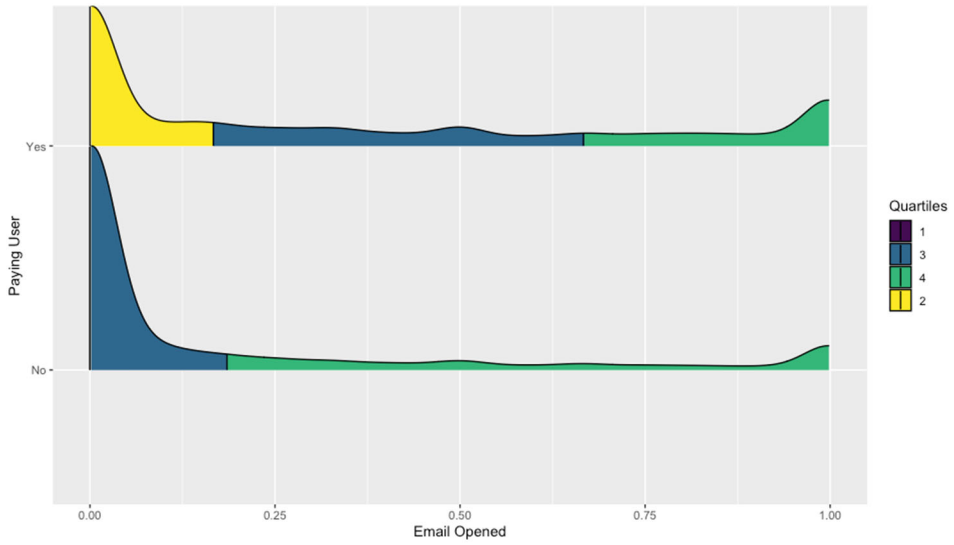
Appendix 3.2



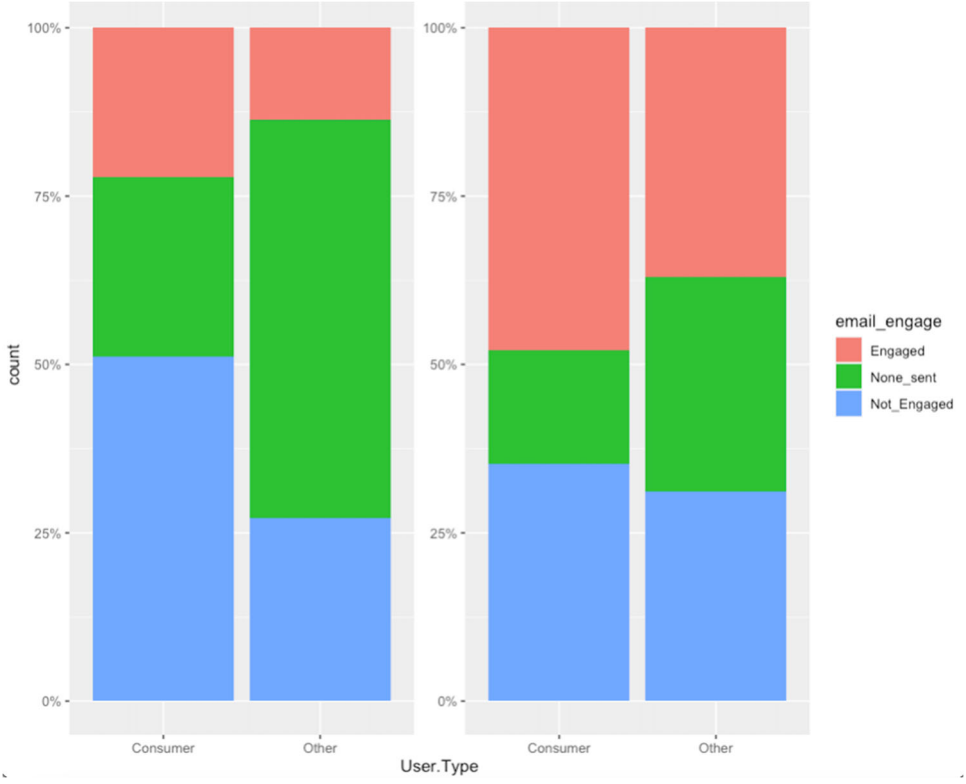
Appendix 4.1



Appendix 4.2



Appendix 4.3



Supporting Code

```
# Kayelin Santa Elena, Haley Noorani
# MGSC 410
# Final Project

###############---------- Clustering ----------###############

# import packages
library(forcats)
library(zoo)
library(factoextra)
library(cluster)
library(scales)
library(mondate)
library(RColorBrewer)
library(ggplot2)
library(dplyr)
library(tictoc)

# import new data
subs <- read.csv("/Users/kksizzle/Desktop/MGSC 410/final project/Rosetta (1).csv")

# train and test subset data for model testing
# set.seed(310)
# subs_indx <-  sample(1:nrow(subs), 0.5*nrow(subs), replace=FALSE)
# subs_subset <- subs[subs_indx,]

# entire dataset
subs_indx <- subs
subs_subset <- subs

#---------- Rearange Data ----------#

# dummy variables
subs_subset$Subscription.Type <- ifelse(subs_subset$Subscription.Type == 'Limited', 1, 0)
subs_subset$Subscription.Event.Type <- ifelse(subs_subset$Subscription.Event.Type == '
INITIAL_PURCHASE', 1, 0)
subs_subset$Purchase.Store <- ifelse(subs_subset$Purchase.Store == 'Web', 1, 0)
subs_subset$Demo.User <- ifelse(subs_subset$Demo.User == 'No', 1, 0)
subs_subset$Free.Trial.User <- ifelse(subs_subset$Free.Trial.User == 'No', 1, 0)
subs_subset$Auto.Renew <- ifelse(subs_subset$Auto.Renew == 'No', 1, 0)
subs_subset$User.Type <- ifelse(subs_subset$User.Type == 'Consumer', 1, 0)
subs_subset$Email.Subscriber <- ifelse(subs_subset$Email.Subscriber == 'No', 1, 0)
subs_subset$Push.Notifications <- ifelse(subs_subset$Push.Notifications == 'No', 1, 0)
```

```
# one hot encoding
onehotdf <- subs_subset[ , c("ID","Currency","Country","Lead.Platform")] # new df for
categorical vars

subs_subset1 <- select(subs_subset,-c("Language","Currency","Country","Lead.Platform",
                                      "Subscription.Start.Date","Subscription.Expiration",
                                      "Free.Trial.Start.Date", "Free.Trial.Expiration")) # new df for w/out cat.
vars. + sub/trial dates cols

# one-hot coding
library(caret)
dummies <- dummyVars(" ~ .", onehotdf)
onehotdfdummies <- data.frame(predict(dummies, newdata = onehotdf))


subs_subset2  <- data.frame(subs_subset1,onehotdfdummies) # combine new encoded vars
with the df
subs_subset2 <- select(subs_subset2, -ID) # drop ID column, not needed in cluster

#factor one-hot dummies {so that it works with kproto()}
require(plyr)
subs_subset2[,18:27] <- colwise(as.factor)(subs_subset2[,18:27])


#---------------- K prototype clustering ----------------#

#install.packages("clustMixType")
library(clustMixType)

# Check for  optimal number of clusters
tic() #timer
wss<-vector()
for (i in 2:15){ wss[i] <- sum(kproto(subs_subset2, i,na.rm = FALSE)$withinss)}

par(mfrow=c(1,1))
plot(1:15, wss, type="b", xlab="Number of Clusters",
    ylab="Within groups sum of squares",
    main="Assessing the Optimal Number of Clusters with the Elbow Method",
    pch=20, cex=2)

# apply k-prototyps
kpres <- kproto(subs_subset2, 6, na.rm = FALSE)
subs_subset2$cluster = kpres$cluster  # add cluster column to df
toc() #timer
```

```
#------new df to make graphs and plots-----#

# use this when using train and test subset
# new_subs <- subs[subs_indx,]
# df <- select(new_subs, -c("ID","Subscription.Start.Date","Subscription.Expiration",
#                           "Free.Trial.Start.Date", "Free.Trial.Expiration"))

df <- select(subs, -c("ID","Subscription.Start.Date","Subscription.Expiration",
                      "Free.Trial.Start.Date", "Free.Trial.Expiration"))

df$cluster = kpres$cluster # add cluster column to df

# save df to excel
# install.packages("writexl")
# library(writexl)
# write_xlsx(df,"/Users/kksizzle/Desktop/MGSC 410/final project/clusters.xlsx")

#------Graphs-----#

# df <- read.csv("/Users/kksizzle/Desktop/MGSC 410/final project/clusters.csv")

ggplot(df, aes(x = User.Type, fill = User.Type))+
  geom_bar(position = "dodge") +
  facet_wrap(~cluster, scale = 'free') +
  labs(x = "User Type", y = "Count",
       title = "User Type by Cluster", fill = "User Type")

ggplot(df, aes(Lead.Platform, fill = Lead.Platform)) +
  geom_bar(position = "dodge") +
  facet_wrap(~cluster, scale = 'free') +
  labs(x = "Lead Platform", y = "Count",
       title = "Lead Platform by Cluster", fill = "Lead Platform")

ggplot(df, aes(Email.Subscriber, fill = Email.Subscriber)) +
  geom_bar(position = "dodge") +
  facet_wrap(~cluster, scale = 'free') +
  labs(x = "Email Subscriber", y = "Count",
       title = "Email Subscriber by Cluster", fill = "Email Subscriber")

ggplot(df, aes(y = Open.Count, x = factor(cluster), fill = factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Cluster", y = "Total number of times emails were opened by subscriber in the past 90
days",
```

```
      title = "Open Count by Cluster", fill = "Cluster") +
  theme(axis.title.y = element_text(size = 8.75))

ggplot(df, aes(y = Send.Count, x = factor(cluster), fill = factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Cluster", y = "Number of emails sent to subscriber in the past 90 days",
      title = "Send Count by Cluster", fill = "Cluster") +
  theme(axis.title.y = element_text(size = 8.75))

ggplot(df, aes(y = Unique.Open.Count, x = factor(cluster), fill = factor(cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Cluster", y = "Unique number of times emails were opened by subscriber in the past
90 days",
      title = "Unique Open Count by Cluster", fill = "Cluster") +
  theme(axis.title.y = element_text(size = 8.75))

ggplot(df, aes(Country, fill = Country)) +
  geom_bar(position = "dodge") +
  facet_wrap(~cluster, scale = 'free') +
  labs(x = "Country", y = "Count",
      title = "Country where subscriber lives by Cluster", fill = "Country")

ggplot(df, aes(x = factor(cluster), fill = Subscription.Event.Type))+
  geom_bar(position = 'fill') +
  labs(x = "Cluster", y = "Count",
      title = "Subscription Event Type by Cluster",
      fill = "Subscription Event Type")

ggplot(df, aes(User.Type, fill = Subscription.Event.Type))+
  geom_bar(position = "dodge") +
  facet_wrap(~cluster, scale = 'free') +
  labs(x = "User Type", y = "Count",
      title = "User Type and Subscription Event Type by Cluster",
      fill = "Subscription Event Type")

ggplot(df, aes(x = factor(cluster), fill = factor(cluster)))+
  geom_bar(stat = "count")  +
  labs(x = "Cluster", y = "Count",
      title = "Number of Subscribers in each Cluster",
      fill = "Cluster")
```

Final Project

December 10, 2020

[ ]:
```
#MGSC 410 Final Project
#Haley Noorani and Kayelin Santa Elena
```

[1]:
```
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np
from plotnine import *

from sklearn.preprocessing import StandardScaler #Z-score variables

# from sklearn.model_selection import train_test_split # simple TT split cv
# from sklearn.model_selection import KFold # k-fold cv
# from sklearn.model_selection import LeaveOneOut #LOO cv
# from sklearn.model_selection import cross_val_score # cross validation metrics
# from sklearn.model_selection import cross_val_predict # cross validation␣
 ↪metrics

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

from sklearn.metrics import silhouette_score
from kmodes.kprototypes import KPrototypes


%matplotlib inline
```

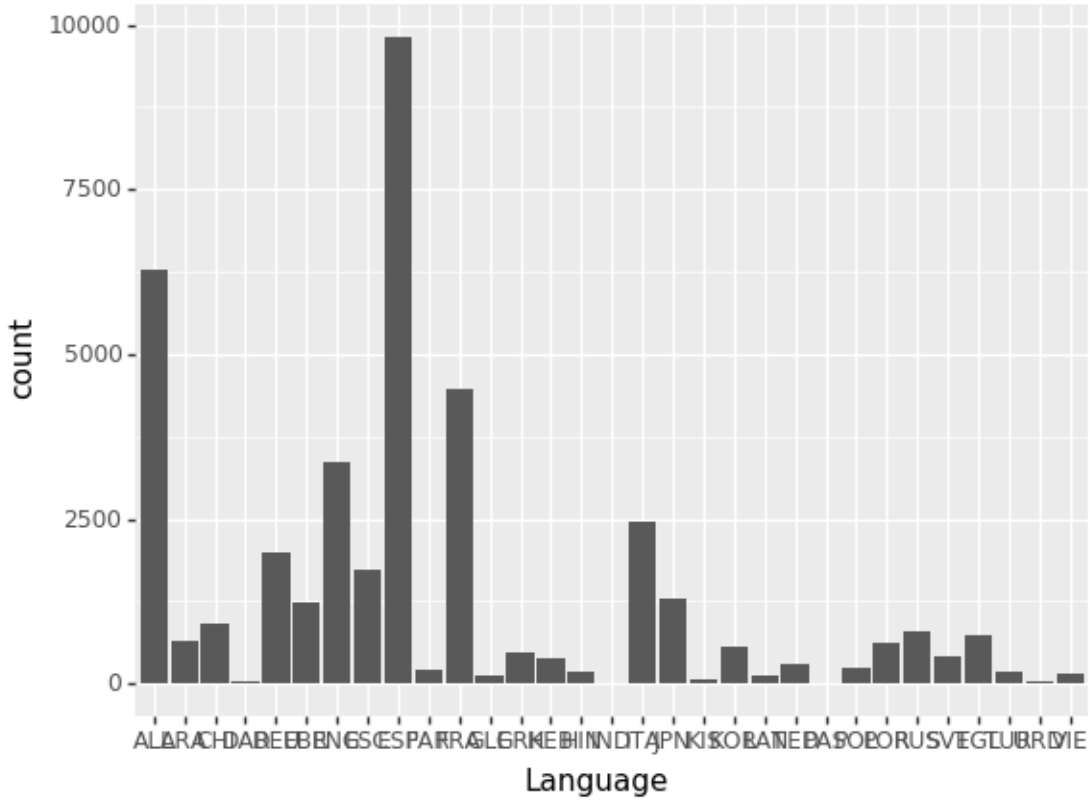
[9]:
```
rosetta=pd.read_csv("Rosetta 1.csv")
rosetta=rosetta.fillna(0)
rosetta=rosetta.replace(np.nan,0)
print(rosetta)
```

```
          ID Language Subscription.Type Subscription.Event.Type  \
0          2      EBR           Limited        INITIAL_PURCHASE
1          3      ESP           Limited        INITIAL_PURCHASE
2          6      ESP           Limited        INITIAL_PURCHASE
```

```
3          7      ESP           Limited                 RENEWAL
4          8      DEU           Limited        INITIAL_PURCHASE
…      …      …                 …                       …
39698  38266      FRA           Limited                 RENEWAL
39699  38655      GRK           Limited        INITIAL_PURCHASE
39700  39099      DEU           Limited        INITIAL_PURCHASE
39701  39504      RUS           Limited        INITIAL_PURCHASE
39702  39894      FRA           Limited        INITIAL_PURCHASE

      Purchase.Store  Purchase.Amount Currency Subscription.Start.Date  \
0                Web            39.00      USD               11/28/2019
1                Web             0.00      USD               12/31/2018
2                Web            38.34      USD                8/23/2019
3                Web            79.00      USD                7/21/2019
4                Web            38.40      USD                 3/7/2020
…              …                …      …                       …
39698            App      29990000.00      GBP                 6/6/2019
39699            App      41490000.00      GBP                2/21/2019
39700            App      99990000.00      GBP                12/3/2018
39701            App      42990000.00      GBP                 2/9/2020
39702            App      41490000.00      GBP                3/31/2020

      Subscription.Expiration Demo.User  … Lead.Platform Email.Subscriber  \
0                   2/28/2020        No  …           Web               No
1                  12/31/2019        No  …           Web              Yes
2                  11/23/2019       Yes  …           App              Yes
3                  11/11/2019       Yes  …           App              Yes
4                    6/7/2020       Yes  …           App              Yes
…                       …       …  …           …               …
39698                6/7/2020        No  …       Unknown               No
39699               6/25/2020        No  …       Unknown               No
39700               12/3/2019       Yes  …           App               No
39701                5/9/2020        No  …       Unknown               No
39702                7/5/2020        No  …           Web              Yes

      Push.Notifications Send.Count Open.Count Click.Count Unique.Open.Count  \
0                    Yes        4.0        3.0         0.0                1.0
1                    Yes        1.0        0.0         0.0                0.0
2                    Yes      162.0        1.0         0.0                1.0
3                    Yes        2.0        0.0         0.0                0.0
4                    Yes       25.0       17.0         4.0                7.0
…                  …         …          …           …                …
39698                 No        0.0        0.0         0.0                0.0
39699                 No        0.0        0.0         0.0                0.0
39700                Yes        0.0        0.0         0.0                0.0
39701                 No       27.0        0.0         0.0                0.0
39702                Yes        1.0        0.0         0.0                0.0
```

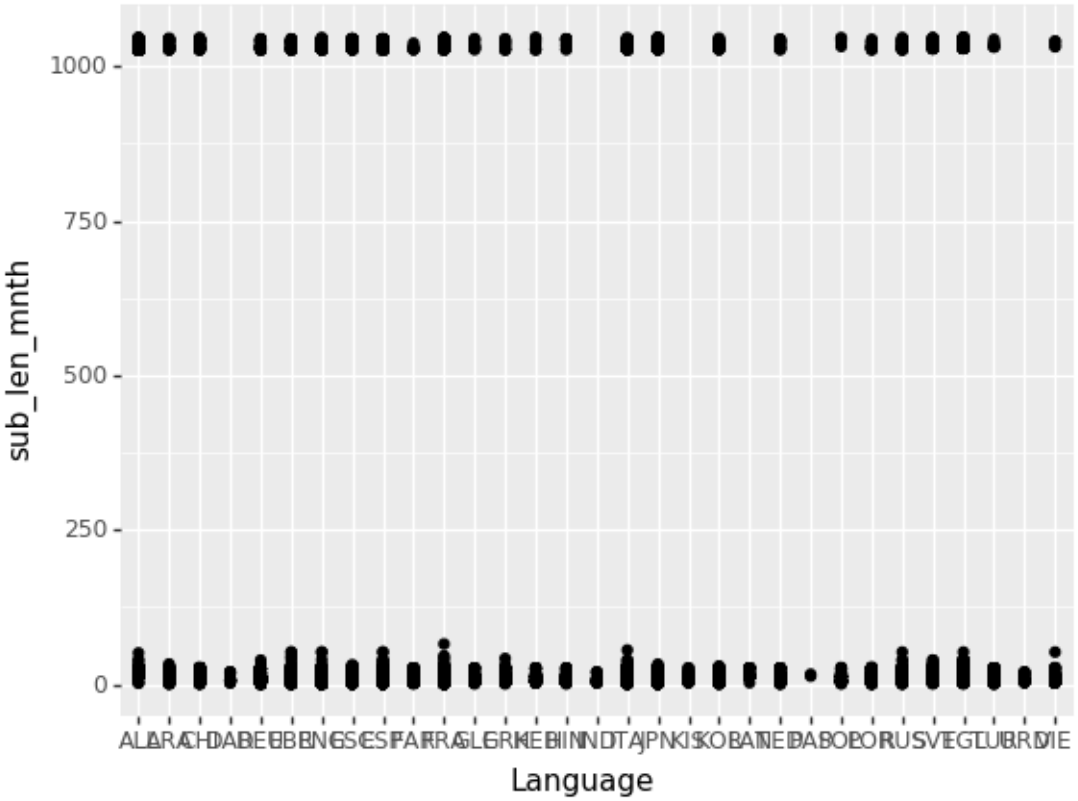
```
      Unique.Click.Count  Dollar_val  sub_len_mnth
0                    0.0     39.0000      3.285714
1                    0.0      0.0000     13.035714
2                    0.0     38.3400      3.285714
3                    0.0     79.0000      4.035714
4                    2.0     38.4000      3.285714
…                      …           …             …
39698                0.0     39.5868     13.107143
39699                0.0     54.7668     17.500000
39700                0.0    131.9868     13.035714
39701                0.0     56.7468      3.214286
39702                0.0     54.7668      3.428571

[39703 rows x 26 columns]
```

[5]:
```
(ggplot(rosetta, aes("Language"))+geom_bar())
```



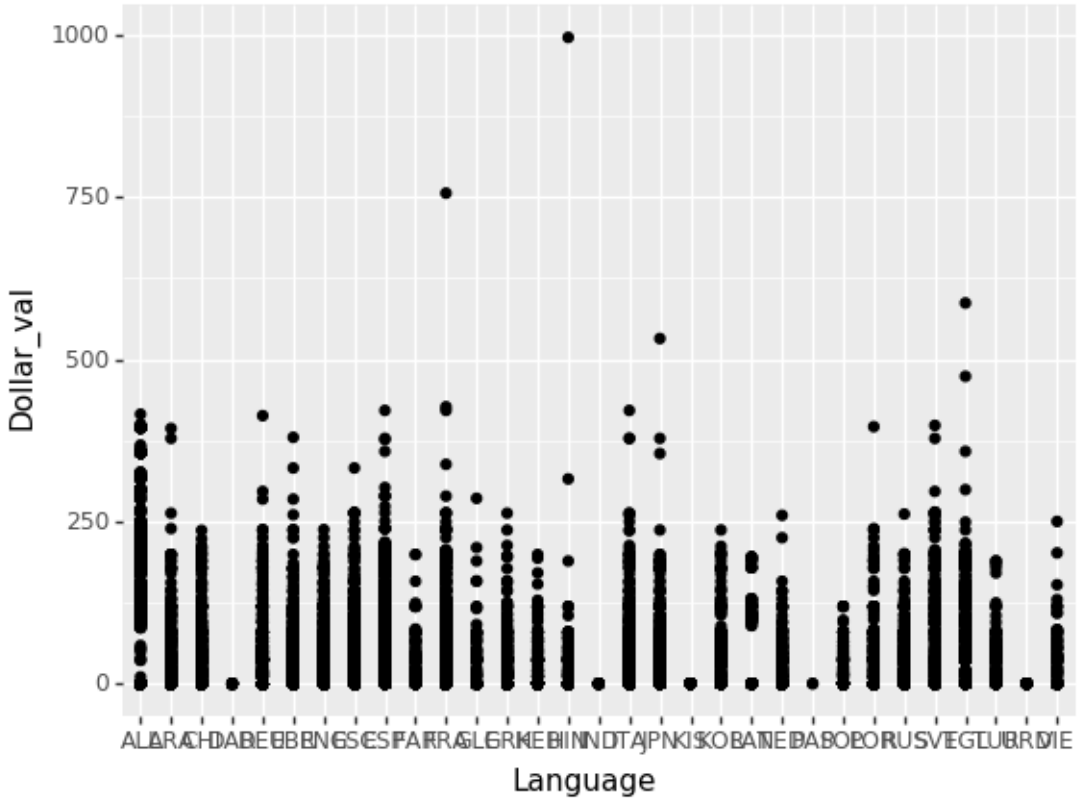
[5]: <ggplot: (-9223371889821377172)>

[8]:
```
(ggplot(rosetta,aes("Language", "sub_len_mnth"))+geom_point())
```



[8]: <ggplot: (-9223371889821375816)>

[9]:
```
(ggplot(rosetta,aes("Language", "Dollar_val"))+geom_point())
```



[9]:
```
<ggplot: (-9223371889821267360)>
```

[10]:
```
features=["Dollar_val", "sub_len_mnth","Send.Count", "Open.Count","Unique.Click.
↪Count"]
X=rosetta[features]
z=StandardScaler()
X[features]=z.fit_transform
```

[41]:
```
EM=GaussianMixture(n_components=4)
EM.fit(X)
```

[41]:
```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
                means_init=None, n_components=4, n_init=1, precisions_init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose=0,
                verbose_interval=10, warm_start=False, weights_init=None)
```

[42]:
```
cluster=EM.predict(X)

cluster
```

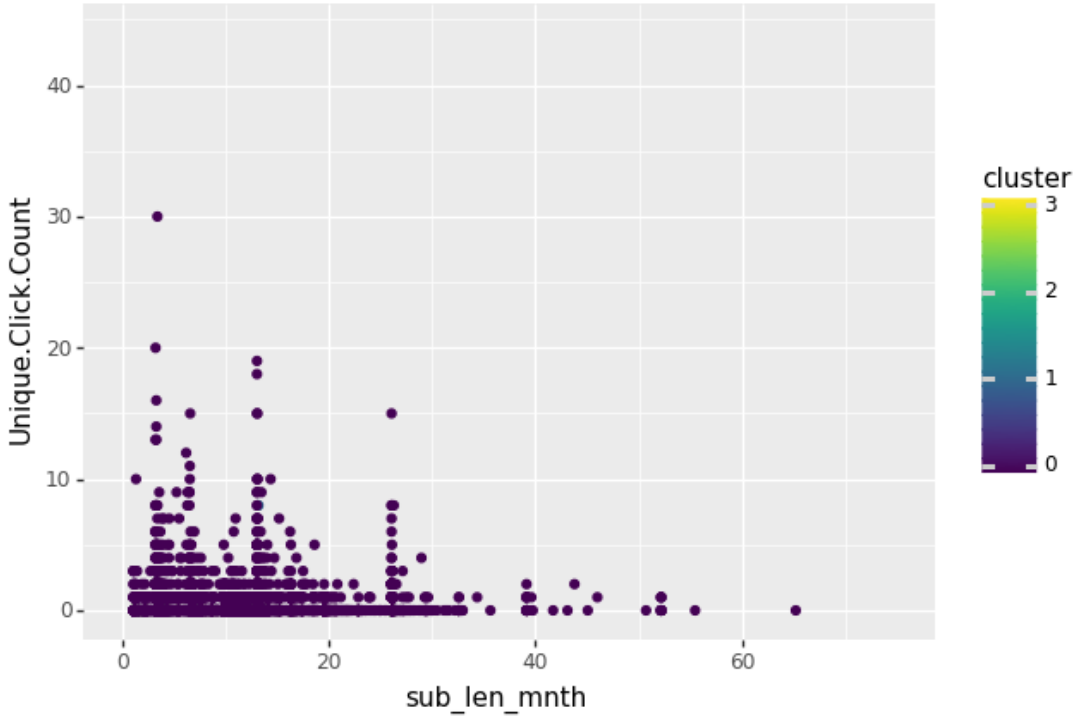
[42]:
```
array([0, 0, 0, …, 0, 0, 0], dtype=int64)
```

[43]:
```
silhouette_score(X, cluster)
```

[43]:
```
0.8836717366696815
```

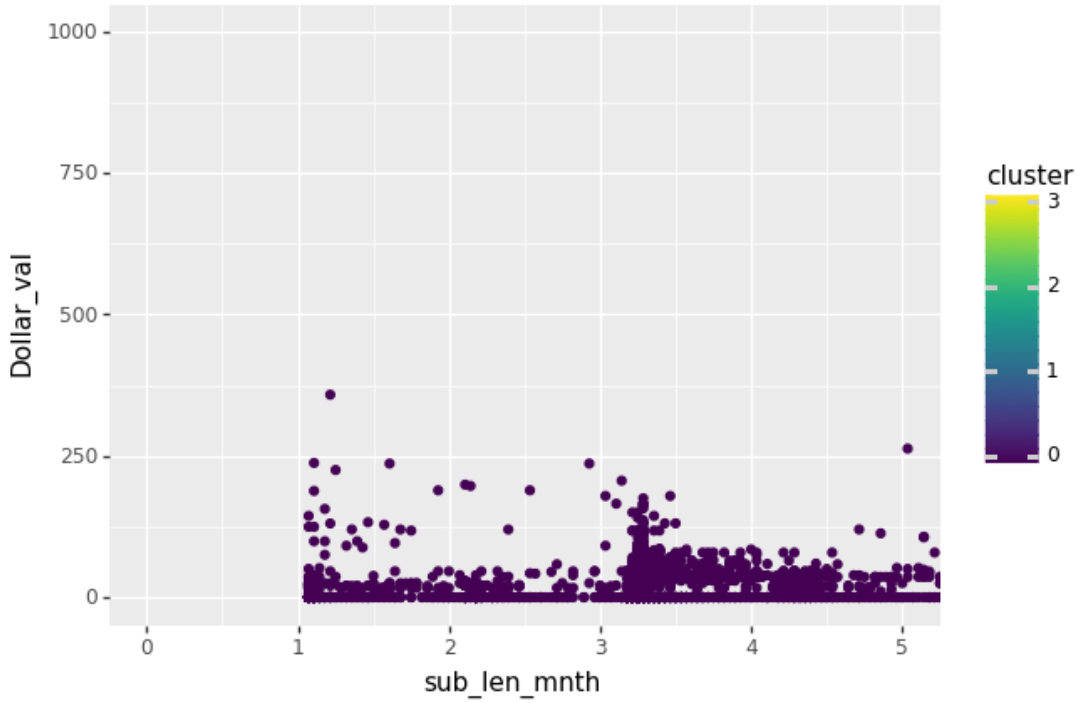
[44]:
```
X["cluster"]=cluster
```

[49]:
```
(ggplot(X, aes(x="sub_len_mnth", y="Unique.Click.Count",␣
↪color="cluster"))+geom_point()+coord_cartesian(xlim=(0,75)))
```



[49]:
```
<ggplot: (-9223371915859000040)>
```

[51]:
```
(ggplot(X, aes(x="sub_len_mnth", y="Dollar_val",␣
↪color="cluster"))+geom_point()+coord_cartesian(xlim=(0,5)))
```



[51]:
```
<ggplot: (-9223371915865779168)>
```

[57]:
```
features=["Dollar_val", "sub_len_mnth","Send.Count", "Open.Count","Unique.Click.
↪Count"]
X=rosetta[features]
z=StandardScaler()
X[features]=z.fit_transform(X)
```

[66]:
```
EM=GaussianMixture(n_components=3)

EM.fit(X)
```

[66]:
```
GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
                means_init=None, n_components=3, n_init=1, precisions_init=None,
                random_state=None, reg_covar=1e-06, tol=0.001, verbose=0,
                verbose_interval=10, warm_start=False, weights_init=None)
```

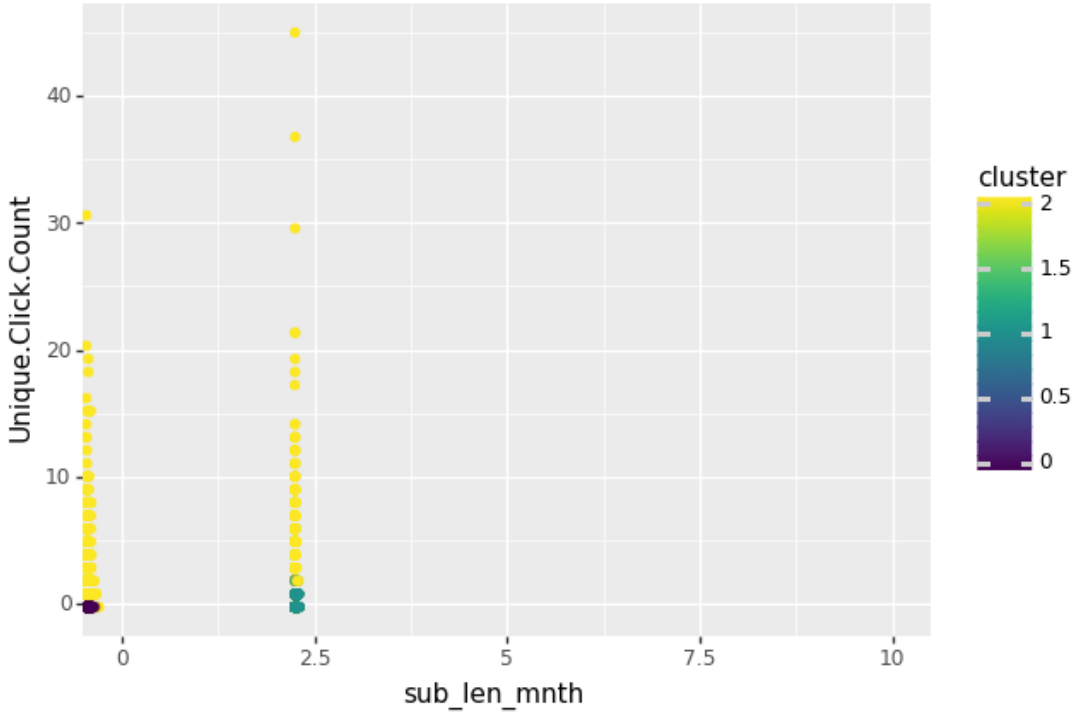
[67]:
```
cluster=EM.predict(X)

cluster
```

[67]:
```
array([0, 0, 2, …, 0, 0, 0], dtype=int64)
```

[68]:
```
silhouette_score(X, cluster)
```

[68]: 0.5410042391040515

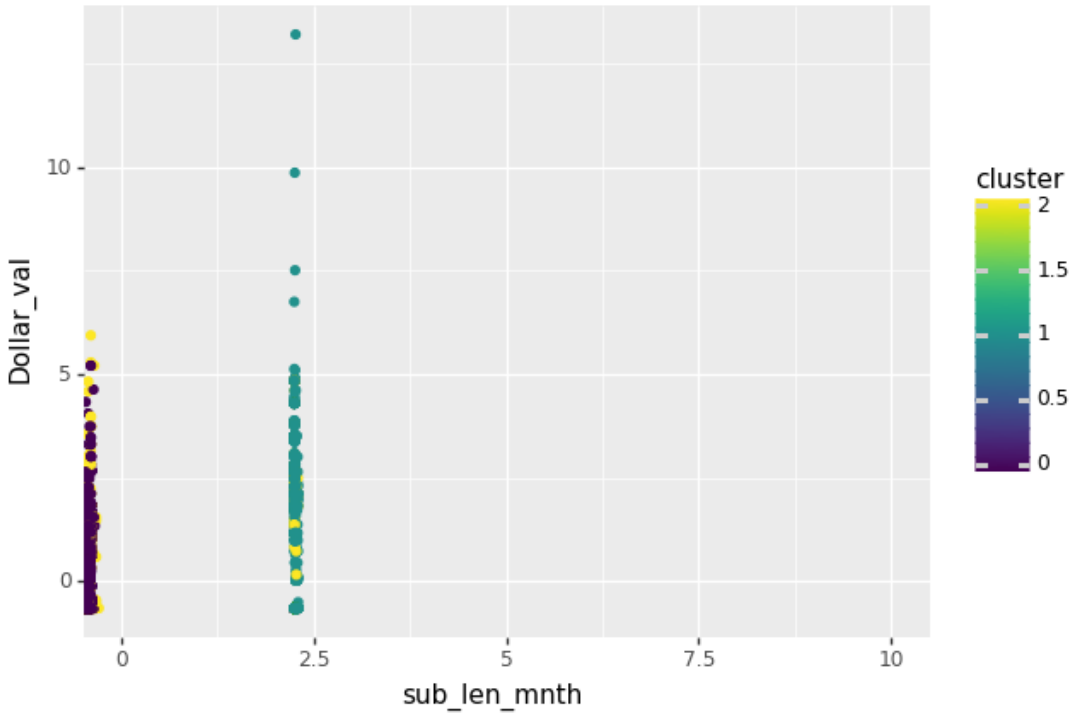
[69]:
```
(ggplot(X, aes(x="sub_len_mnth", y="Unique.Click.Count",␣
↪color="cluster"))+geom_point()+coord_cartesian(xlim=(0,10)))
```



[69]: <ggplot: (-9223371915862810884)>

[ ]:
```
(ggplot(X, aes(x="sub_len_mnth", y="Unique.Click.Count",␣
↪color="cluster"))+geom_point()+coord_cartesian(xlim=(0,75)))
```

[70]:
```
(ggplot(X, aes(x="sub_len_mnth", y="Dollar_val",␣
↪color="cluster"))+geom_point()+coord_cartesian(xlim=(0,10)))
```



[70]: <ggplot: (-9223371915854823096)>

```
#Trent Commins Python Code, cleaning of app activity data to be used in R code
import numpy as np
import pandas as pd

app_df = pd.read_csv("/Users/trentcommins/Documents/school/Chapman/2020-
21/Fall/MGSC410/Homework/Final/Dataset/App_activity.csv")
app_df.columns = ['ID', 'Sess_platform','App_activity','Date']

# ID, main_platform, count_app_lunch, count_start, count_complete, count_onboarding,
count_other
#create list of IDs
def new_df(app_data):
    id_list = pd.DataFrame(data=np.unique(app_data['ID']))
    return id_list

#Count of each type of activity for each ID
def count_ID(app_data,id_data): # call to create list of IDs
#Empty lists to be appended into
    count_app_launch = []
    count_start = []
    count_complete = []
    count_onboarding = []
    count_other = []

    #Iterate through df by ID create count of each attribute and append into empty list
    for i in range(1,(len(id_data)+1)):
        is_id_i = app_data[(app_data['ID']==i)]
        val_launch = (is_id_i.App_activity == 'App Launch').sum()
        val_start = (is_id_i.App_activity == 'Start').sum()
        val_complete = (is_id_i.App_activity == 'Complete').sum()
        val_onboarding = (is_id_i.App_activity == 'Onboarding content').sum()
        val_other = (is_id_i.App_activity == 'Other').sum()
        count_app_launch.append(val_launch)
        count_start.append(val_start)
        count_complete.append(val_complete)
        count_onboarding.append(val_onboarding)
        count_other.append(val_other)

    df2 = id_data
    #rename
    df2['count_app_launch'] = count_app_launch
    df2['count_start'] = count_start
    df2['count_complete'] = count_complete
    df2['count_onboarding'] = count_onboarding
```

```
    df2['count_other'] = count_other


    return df2


idlst = new_df(app_df)
final = count_ID(app_df,idlst)
final.columns = ['ID','count_app_launch','count_start','count_complete',
                 'count_onboarding','count_other']

#export to csv
final.to_csv('app_data.csv')


#Trent Commins R Code, cleaning exploration and engineering, Part 1 and Part 4 modeling

library(tidyverse)
library(ggplot2)
library(devtools)
library(glmnet)
library(glmnetUtils)
library(dplyr)
library(caret)
library(plotROC)
library(olsrr)
library(MASS)
library(ggridges)
library(scales)
library(randomForest)
library(randomForestExplainer)
## Cleaning ##
df_raw <- read.csv("/Users/trentcommins/Documents/school/Chapman/2020-
21/Fall/MGSC410/Homework/Final/Dataset/Subscriber_Information.csv")
#factorize data
df_raw <- df_raw %>% mutate(Currency = factor(Currency),
                            Purchase.Store = factor(Purchase.Store),
                            Language = factor(Language),
                            Subscription.Type = factor(Subscription.Type),
                            Subscription.Event.Type = factor(Subscription.Event.Type),
                            Demo.User = factor(Demo.User),
                            Free.Trial.User  = factor(Free.Trial.User),
                            Auto.Renew = factor(Auto.Renew),
                            Country = factor(Country),
```

```
                    User.Type = factor(User.Type),
                    Lead.Platform = factor(Lead.Platform),
                    Email.Subscriber = factor(Email.Subscriber),
                    Push.Notifications = factor(Push.Notifications),
                    Send.Count = as.numeric(Send.Count),
                    Open.Count = as.numeric(Open.Count),
                    Click.Count = as.numeric(Click.Count),
                    Unique.Open.Count = as.numeric(Unique.Open.Count),
                    Unique.Click.Count = as.numeric(Unique.Click.Count),
                    Subscription.Start.Date = as.Date(Subscription.Start.Date, "%m/%d/%Y"),
                    Subscription.Expiration = as.Date(Subscription.Expiration, "%m/%d/%Y"))

#convert currency to USD
keep_money <- c("USD","NULL", "EUR","GBP")
df_raw <- df_raw[df_raw$Currency %in% keep_money,]
df_raw$Currency = droplevels(df_raw$Currency)
USD_df <- subset(df_raw, Currency =="USD")
USD_df <- USD_df %>% mutate(Purchase.Amount = as.numeric(Purchase.Amount),
                    Dollar_val = Purchase.Amount)
NULL_df <- subset(df_raw, Currency =="NULL")
NULL_df <- NULL_df %>% mutate(Purchase.Amount = as.numeric(Purchase.Amount),
                     Dollar_val = 0)
# GBP 1.32 : USD 1
# EUR 1.19: USD 1
EUR_df <- subset(df_raw, Currency =="EUR")
EUR_df <- EUR_df %>% mutate(Purchase.Amount = as.numeric(Purchase.Amount),
                    Dollar_val = Purchase.Amount*1.19)
GBP_df <- subset(df_raw, Currency =="GBP")
GBP_df <- GBP_df %>% mutate(Purchase.Amount = as.numeric(Purchase.Amount),
                    Dollar_val = Purchase.Amount*1.32)
df2 <- bind_rows(USD_df,NULL_df,EUR_df,GBP_df)

#App payments were recorded incorrectly, subset and adjust
web_df <- subset(df2, Purchase.Store =="Web")
app_df <- subset(df2, Purchase.Store =="App")
app_df <- app_df %>% mutate(Dollar_val = Dollar_val/1000000)
df_working <- bind_rows(web_df,app_df)
df_working <- df_working %>% mutate(sub_len_mnth =
(as.numeric(difftime(Subscription.Expiration, Subscription.Start.Date), units="weeks")/4))
df_working <- df_working[!is.na(df_working$Dollar_val), ]

#Dataframe from Python cleaning
df_glance <- read.csv("/Users/trentcommins/Documents/school/Chapman/2020-
21/Fall/MGSC410/Homework/Final/Dataset/app_data.csv")
```

```
df_glance <- subset(df_glance, select = -c(X))

df <- df_working %>% mutate(paying_user = factor(ifelse(Dollar_val == 0,"No","Yes")))
df <- subset(df, Auto.Renew !="NULL")

df$Auto.Renew = droplevels(df$Auto.Renew)

#create feature to examine email engagement
open_pct_df <- df %>% drop_na(Send.Count)
open_pct_df <- open_pct_df %>% mutate(open_pct = Open.Count/Send.Count)
open_pct_df <- open_pct_df %>% mutate(email_engage = factor(ifelse(open_pct >=
0.10,"Engaged","Not_Engaged")))
no_open_pct_df<- subset(df, is.na(Send.Count))
no_open_pct_df <- no_open_pct_df %>% mutate(email_engage = factor("None_sent"))

df2 <- bind_rows(no_open_pct_df,open_pct_df)

df2 <-df2 %>% inner_join(df_glance)
df2 <- df2 %>% mutate(count_activity = count_app_launch + count_app_launch +
count_complete + count_onboarding+ count_other)

#Clean dataframe
write.csv(df2,"/Users/trentcommins/Documents/school/Chapman/2020-
21/Fall/MGSC410/Homework/Final/Dataset/mod_ready.csv", row.names = FALSE)




## Section 4, Logistic Regression ##
df_class <- df_working
#remove redundant and irrelevant data
df_model <- subset(df_class, select = -
c(ID,Language,Subscription.Type,Subscription.Event.Type,Purchase.Amount,Currency,Subscripti
on.Start.Date,

Subscription.Expiration,Free.Trial.Start.Date,Free.Trial.Expiration,Auto.Renew,Send.Count,Ope
n.Count,

Click.Count,Unique.Open.Count,Unique.Click.Count,Dollar_val,sub_len_mnth,open_pct,count_
app_launch,
                                count_start,count_complete,count_onboarding,count_other))
trainSize <- 0.75
train_idx <-sample(1:nrow(df_model), size =floor(nrow(df_model)*trainSize))
df_class_train <- df_model%>% slice(train_idx)
```

```
df_class_test <- df_model%>% slice(-train_idx)

#preliminary Logistic regression
logit <- glm(paying_user ~.,
          data = df_class_train,
          family = binomial)
summary(logit)

#predict into the training data
scores_logit <- data.frame(
  scores = predict(logit,
                newdata = df_class_test,
                type = "response"),
  df_class_test)
scores_logit_CutOff <- scores_logit %>% mutate(CutOff05 =
                                    ifelse(scores > 0.5,1,0))
#area under the curve
Test_ROC <- ggplot(scores_logit_CutOff,
                aes(m = scores,
                   d = paying_user)) +
  geom_roc(labelsize = 3.5,
         cutoffs.at =
           c(.99,.9,.7,.5,.3,.1,0))
Test_ROC
calc_auc(Test_ROC) #find AUC
table(scores_logit$paying_user,
     scores_logit_CutOff$CutOff05)

#logistic regression using most relevant features
logit_best <- glm(paying_user ~ email_engage + Demo.User + Email.Subscriber + Free.Trial.User
+ Purchase.Store,
               data = df_class_train,
               family = binomial)
summary(logit_best)

#predict into the training data
scores_logit_best <- data.frame(
  scores = predict(logit_best,
                newdata = df_class_test,
                type = "response"),
  df_class_test)
scores_logit_best_CutOff <- scores_logit_best %>% mutate(CutOff05 =
                                        ifelse(scores > 0.5,1,0))
Test_best_ROC <- ggplot(scores_logit_best_CutOff,
```

```
                  aes(m = scores, #preds
                      d = paying_user)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at =
             c(.99,.9,.7,.5,.3,.1,0))
Test_best_ROC
calc_auc(Test_best_ROC) #find AUC
table(scores_logit_best$paying_user,
      scores_logit_best_CutOff$CutOff05)

#Final logistic regression to find the importance of user activity and attributes, regardless of
purchase store
logit_activity <- glm(paying_user ~ email_engage + Demo.User + Email.Subscriber +
Free.Trial.User,
                data = df_class_train,
                family = binomial)
summary(logit_activity)

#predict into the training data
scores_logit_activity <- data.frame(
  scores = predict(logit_activity,
                   newdata = df_class_test,
                   type = "response"),
  df_class_test)
scores_logit_activity_CutOff <- scores_logit_activity %>% mutate(CutOff05 =
                                          ifelse(scores > 0.5,1,0))
Test_activity_ROC <- ggplot(scores_logit_activity_CutOff,
                  aes(m = scores, #preds
                      d = paying_user)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at =
             c(.99,.9,.7,.5,.3,.1,0))
Test_activity_ROC
calc_auc(Test_best_ROC) #find AUC
table(scores_logit_activity$paying_user,
      scores_logit_activity_CutOff$CutOff05)




## Section 1, Linear Regression and Random Forest##
#linear regression
lm1 <- lm(Dollar_val ~.,
          data = df_class_train)
```

```
#predict into the training data
scores_lm1 <- data.frame(
  scores = predict(lm1,
                   newdata= df_class_test,
                   na.rm = TRUE,
                   type = "response"),
  df_class_test
)

ggplot(data = scores_lm1 )+
  geom_point(mapping = aes(x = scores, y = Dollar_val))+
  geom_abline(color = "red")

summary(lm1)


#Random Forest
RMSE <-function(t, p) {
  sqrt(sum(((t-p)^2))*(1/nrow(t)))
}

df_paying <- subset(df_class, Dollar_val <= 500)
df_paying <- subset(df_paying, Dollar_val > 0.5)

trainSize <- 0.75
train_idx_paying <-sample(1:nrow(df_paying), size =floor(nrow(df_paying)*trainSize))

df_paying_train <- df_paying%>% slice(train_idx)
df_paying_test <- df_paying%>% slice(-train_idx)

rf_fit<- randomForest(Dollar_val ~ .,
                      data=df_paying_train,
                      type = regression,
                      mtry = 3,
                      ntree = 300,
                      na.rm = TRUE,
                      importance = TRUE,
                      localImp = TRUE)
rf_fit
varImpPlot(rf_fit)
plot(rf_fit)

scores_rf1_test <- data.frame(
```

```
  scores = predict(rf_fit,
                   newdata= df_paying_test,
                   na.rm = TRUE,
                   type = "response"),
  df_paying_test
)
ggplot(data = scores_rf1_test )+
  geom_point(mapping = aes(x = scores, y = Dollar_val))+
  geom_abline(color = "red")
# Predict limited or lifetime subscription

df_logit <- subset(df_class, select = -
c(ID,Language,Subscription.Type,Subscription.Event.Type,Purchase.Amount,Currency,Subscripti
on.Start.Date,

Subscription.Expiration,Free.Trial.Start.Date,Free.Trial.Expiration,Auto.Renew,Send.Count,Ope
n.Count,

Click.Count,Unique.Open.Count,Unique.Click.Count,Dollar_val,sub_len_mnth,open_pct,count_
app_launch,
                                count_start,count_complete,count_onboarding,count_other))
df_logit <- subset(df_logit, paying_user ="Yes")

train_idx_logit <-sample(1:nrow(df_model), size =floor(nrow(df_logit)*trainSize))

df_logit_train <- df_logit%>% slice(train_idx_logit)
df_logit_test <- df_logit%>% slice(-train_idx_logit)

names(df_class_test)
logit_lf <- glm(User.Type ~.,
                data = df_logit_train,
                family = binomial)

summary(logit_lf)

scores_logit_lf <- data.frame(
  scores = predict(logit_lf,
                   newdata = df_class_test,
                   type = "response"),
  df_logit_test)

scores_lf_CutOff <- scores_logit_lf %>% mutate(CutOff05 =
                                     ifelse(scores > 0.5,1,0))
```

```
Test_ROC_lf <- ggplot(scores_lf_CutOff,
                      aes(m = scores, #preds
                          d = paying_user)) +
  geom_roc(labelsize = 3.5,
           cutoffs.at =
             c(.99,.9,.7,.5,.3,.1,0))

Test_ROC_lf

calc_auc(Test_ROC_lf) #find AUC

table(scores_logit$paying_user,
      scores_logit_CutOff$CutOff05)



## Exploration and Visualization ##
df_lifetime <- subset(df, sub_len_mnth >= 500)
df_lifetime <- subset(df_lifetime, Dollar_val <= 500)
df_lifetime <- subset(df_lifetime, Dollar_val > 0.5)

#Viz for determining cut off point for email engagement
ggplot(open_pct_df, aes(x = open_pct, y = paying_user, fill = factor(stat(quantile)))) +
  stat_density_ridges(
    geom = "density_ridges_gradient", calc_ecdf = TRUE,
    scale = 1,
    quantiles = 4, quantile_lines = TRUE) +
  scale_x_continuous(limits = c(0, 1)) +
  labs(x = "Email Opened",
       y = "Paying User") +
  scale_fill_viridis_d(name = "Quartiles")

#email engagement by paying user and user type
ggplot(data = df) +
  geom_bar(mapping = aes(x=User.Type,fill=email_engage),position = "fill")+
  facet_wrap(~paying_user, scale = "free")+
  scale_y_continuous(labels = scales::percent)

#Lead Platform by purchase store and user type
ggplot(data = df) +
  geom_bar(mapping = aes(x=Purchase.Store,fill=Lead.Platform),position = "fill")+
  facet_wrap(~User.Type)+
  scale_y_continuous(labels = scales::percent)
```

```
#Attributes of lifetime verse limited subscription, by email engagement and user type
ggplot(data = df) +
  geom_bar(mapping = aes(x=User.Type,fill=email_engage),position = "fill")+
  facet_wrap(~Subscription.Type, scale = "free")+
  scale_y_continuous(labels = scales::percent)
```

Ayra_Final

December 10, 2020

[ ]:
```
from pandas_profiling import ProfileReport
#Standard libraries for data analysis:

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy.stats import norm, skew
from scipy import stats
import statsmodels.api as sm
# sklearn modules for data preprocessing:
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
#sklearn modules for Model Selection:
from sklearn import svm, tree, linear_model, neighbors
from sklearn import naive_bayes, ensemble, discriminant_analysis,␣
 ↪gaussian_process
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
#from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
#sklearn modules for Model Evaluation & Improvement:

from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import f1_score, precision_score, recall_score, fbeta_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
```

```
from sklearn.model_selection import KFold
from sklearn import feature_selection
from sklearn import model_selection
from sklearn import metrics
from sklearn.metrics import classification_report, precision_recall_curve
from sklearn.metrics import auc, roc_auc_score, roc_curve
from sklearn.metrics import make_scorer, recall_score, log_loss
from sklearn.metrics import average_precision_score
#Standard libraries for data visualization:
import seaborn as sn
from matplotlib import pyplot
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import matplotlib
%matplotlib inline
color = sn.color_palette()
import matplotlib.ticker as mtick
from IPython.display import display
pd.options.display.max_columns = None
from pandas.plotting import scatter_matrix
from sklearn.metrics import roc_curve
#Miscellaneous Utilitiy Libraries:

import random
import os
import re
import sys
import timeit
import string
import time
from datetime import datetime
from time import time
from dateutil.parser import parse
import joblib

plt.style.use('ggplot')
%matplotlib inline

subinfo = pd.read_csv('subinfo.csv')
activity = pd.read_csv('activity.csv')
```

[ ]:
```
df_subinfo = pd.DataFrame(data = subinfo)
df_activity = pd.DataFrame(data = activity)
df_subinfo.head()
```

[ ]:
```
df_activity.columns = df_activity.columns.str.replace(' ', '_')
df_subinfo.columns = df_subinfo.columns.str.replace(' ', '_')
```

[ ]:
```
df_activity.head()
```

[ ]:
```
df_subinfo.head()
```

[ ]:
```
#df_subinfo['Auto_Renew'] = df_subinfo['Auto_Renew'].convert_dtypes()
#df_subinfo['Auto_Renew'] = pd.to_numeric(df_subinfo.Auto_Renew,␣
 ↪errors='coerce')
```

[ ]:
```
df_subinfo.Auto_Renew.count()
print (df_subinfo.dtypes)
```

[ ]:
```
profile = ProfileReport(df_subinfo, minimal=True)
profile
```

[ ]:
```
df_subinfo.isna().any()
```

[ ]:
```
df_subinfo = df_subinfo.fillna(df_subinfo.mean())
```

[ ]:
```
df_subinfo.isna().any()
```

[ ]:
```
def impute_nan_most_frequent_category(DataFrame,ColName):
    # .mode()[0] - gives first category name
     most_frequent_category=DataFrame[ColName].mode()

    # replace nan values with most occured category
     DataFrame[ColName + "_Imputed"] = DataFrame[ColName]
     DataFrame[ColName + "_Imputed"].fillna(most_frequent_category,inplace=True)

#2. Call function to impute most occured category
for i in ['Currency']:
    impute_nan_most_frequent_category(df_subinfo, i)

# Display imputed result
df_subinfo[['Currency','Currency_Imputed', 'Auto_Renew']].head(10)
```

[ ]:
```
#3. Drop actual columns
df_subinfo = df_subinfo.drop(['Currency'], axis = 1)
df_subinfo['Currency'] = df_subinfo['Currency_Imputed']
df_subinfo = df_subinfo.drop(['Currency_Imputed'], axis = 1)
```

[ ]:
```
df_subinfo.head()
```

[ ]:
```
df_subinfo.isna().any()
```

[ ]:
```
df_subinfo.head()
```

[ ]:
```
df_subinfo['Subscription_Start_Date'] = pd.
 ↪to_datetime(df_subinfo['Subscription_Start_Date'], errors='coerce')
df_subinfo['Subscription_Expiration'] = pd.
 ↪to_datetime(df_subinfo['Subscription_Expiration'], errors='coerce')
df_subinfo['Free_Trial_Start_Date'] = pd.
 ↪to_datetime(df_subinfo['Free_Trial_Start_Date'], errors='coerce')
df_subinfo['Free_Trial_Expiration'] = pd.
 ↪to_datetime(df_subinfo['Free_Trial_Expiration'], errors='coerce')
df_subinfo['Free_Trial_Expiration'] = df_subinfo['Free_Trial_Expiration'].dt.
 ↪strftime('%d.%m.%Y').replace('NaT',0)
df_subinfo['Free_Trial_Start_Date'] = df_subinfo['Free_Trial_Start_Date'].dt.
 ↪strftime('%d.%m.%Y').replace('NaT',0)


df_subinfo['Subscription_Start_Date'] = df_subinfo['Subscription_Start_Date'].
 ↪dt.strftime("%Y%m%d").astype(int)
df_subinfo['Subscription_Expiration'] = df_subinfo['Subscription_Expiration'].
 ↪dt.strftime("%Y%m%d").astype(int)
df_subinfo['Free_Trial_Start_Date'] = df_subinfo['Free_Trial_Start_Date'].
 ↪astype(int, errors='ignore')
df_subinfo['Free_Trial_Expiration'] = df_subinfo['Free_Trial_Expiration'].
 ↪astype(int, errors='ignore')
```

[ ]:
```
#label encoding
df_subinfo_copy = df_subinfo.copy()
le = LabelEncoder()
# Label Encoding will be used for columns with 2 or less unique values
le_count = 0
for col in df_subinfo_copy.columns[1:]:
    if df_subinfo_copy[col].dtype == 'object':
        if len(list(df_subinfo_copy[col].unique())) <= 2:
            le.fit(df_subinfo_copy[col])
            df_subinfo_copy[col] = le.transform(df_subinfo_copy[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
print(le)
```

[ ]:
```
df_subinfo_copy.head()
```

[ ]:
```
dataset2 = df_subinfo[['Purchase_Amount',
'Send_Count', 'Open_Count','Click_Count',
'Unique_Click_Count', 'Unique_Open_Count', 'Dollar_val',
'sub_len_mnth']]

#Histogram:
```

```
fig = plt.figure(figsize=(15, 12))
plt.suptitle('Histograms of Numerical␣
 ↪Columns\n',horizontalalignment="center",fontstyle = "normal", fontsize = 24,␣
 ↪fontfamily = "sans-serif")
for i in range(dataset2.shape[1]):
    plt.subplot(6, 3, i + 1)
    f = plt.gca()
    f.set_title(dataset2.columns.values[i])
vals = np.size(dataset2.iloc[:, i].unique())
if vals >= 100:
    vals = 100

plt.hist(dataset2.iloc[:, i], bins=vals, color = '#ec838a')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
```

[ ]:
```
df_subinfo['Purchase_Amount'] = df_subinfo['Purchase_Amount'].apply(np.int64)
df_subinfo['Send_Count'] = df_subinfo['Send_Count'].apply(np.int64)
df_subinfo['Open_Count'] = df_subinfo['Open_Count'].apply(np.int64)
df_subinfo['Click_Count'] = df_subinfo['Click_Count'].apply(np.int64)
df_subinfo['Unique_Open_Count'] = df_subinfo['Unique_Open_Count'].apply(np.
 ↪int64)
df_subinfo['Unique_Click_Count'] = df_subinfo['Unique_Click_Count'].apply(np.
 ↪int64)
df_subinfo['Dollar_val'] = df_subinfo['Dollar_val'].apply(np.int64)
df_subinfo['sub_len_mnth'] = df_subinfo['sub_len_mnth'].apply(np.int64)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Subscription_Type"]]
sectors = contract_split .groupby ("Subscription_Type")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Customers by␣
 ↪Subscription Type',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('\n Contract Type',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Contract Type \n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
```

```
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)


add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Currency"]]
sectors = contract_split .groupby ("Currency")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Location of␣
 ↪Purchase Based on Currency',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Currency',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Currency \n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)
```

```
#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Subscription_Event_Type"]]
sectors = contract_split .groupby ("Subscription_Event_Type")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Subscription␣
↪Type',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Currency',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Subscription Type \n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Purchase_Store"]]
sectors = contract_split .groupby ("Purchase_Store")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
```

```
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Web vs App',legend␣
↪=True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Currency',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Web vs App \n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Language"]]
sectors = contract_split .groupby ("Language")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Language␣
↪Purchased',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Currency',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
```

```
plt.title('Langauge Purchased',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
 ↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Country"]]
sectors = contract_split .groupby ("Country")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Customers by␣
 ↪Country',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Country',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Customers by Country',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
```

```
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
 ↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Push_Notifications"]]
sectors = contract_split .groupby ("Push_Notifications")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'User Type',legend␣
 ↪=True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Push Notifications',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Push Notifications',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
 ↪points", ha='center', va=va)
```

```
add_value_labels(ax)
```

[ ]:
```
contract_split = df_subinfo[[ "ID", "Email_Subscriber"]]
sectors = contract_split .groupby ("Email_Subscriber")
contract_split = pd.DataFrame(sectors["ID"].count())
contract_split.rename(columns={'ID':'No. of customers'}, inplace=True)
ax =  contract_split[["No. of customers"]].plot.bar(title = 'Email␣
↪Subscriber',legend =True, table = False,
grid = False,  subplots = False,figsize =(12, 7), color ='#ec838a',
fontsize = 15, stacked=False)
plt.ylabel('No. of Customers\n',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.xlabel('Email Subscriber',
horizontalalignment="center",fontstyle = "normal",
fontsize = "large", fontfamily = "sans-serif")
plt.title('Email Subscriber',
horizontalalignment="center",fontstyle = "normal",
fontsize = "22", fontfamily = "sans-serif")
plt.legend()
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
x_labels = np.array(contract_split[["No. of customers"]])
def add_value_labels(ax, spacing=5):
    for rect in ax.patches:
        y_value = rect.get_height()
        x_value = rect.get_x() + rect.get_width() / 2
        space = spacing
        va = 'bottom'
        if y_value < 0:
            space *= -1
            va = 'top'
        label = "{:.0f}".format(y_value)

#ax.annotate(label,(x_value, y_value),xytext=(0, space), textcoords="offset␣
↪points", ha='center', va=va)

add_value_labels(ax)
```

[ ]:
```
#label encoded

services= ['Subscription_Type', 'Subscription_Event_Type', 'Purchase_Store',
           'Demo_User', 'Free_Trial_User', 'User_Type', 'Email_Subscriber',
           'Push_Notifications']
fig, axes = plt.subplots(nrows = 3,ncols = 3,
figsize = (15,12))
```

```
for i, item in enumerate(services):
    if i < 3:
        ax = df_subinfo[item].value_counts().plot(
        kind = 'bar',ax=axes[i,0],
        rot = 0, color ='#f3babc' )

    elif i >=3 and i < 6:
        ax = df_subinfo[item].value_counts().plot(
        kind = 'bar',ax=axes[i-3,1],
        rot = 0,color ='#9b9c9a')

    elif i < 9:
        ax = df_subinfo[item].value_counts().plot(
        kind = 'bar',ax=axes[i-6,2],rot = 0,
        color = '#ec838a')
    ax.set_title(item)
```

[ ]:
```
#Correlations for Subscription Type

correlations = df_subinfo_copy.corrwith(df_subinfo_copy.Subscription_Event_Type)
correlations = correlations[correlations!=1]
positive_correlations = correlations[
correlations >0].sort_values(ascending = False)
negative_correlations =correlations[
correlations<0].sort_values(ascending = False)
print('Most Positive Correlations: \n', positive_correlations)
print('\nMost Negative Correlations: \n', negative_correlations)
```

[ ]:
```
#Correlation for Lifetime and Limited

correlations = df_subinfo_copy.corrwith(df_subinfo_copy.Subscription_Type)
correlations = correlations[correlations!=1]
positive_correlations = correlations[
correlations >0].sort_values(ascending = False)
negative_correlations =correlations[
correlations<0].sort_values(ascending = False)
print('Most Positive Correlations: \n', positive_correlations)
print('\nMost Negative Correlations: \n', negative_correlations)
```

[ ]:
```
#Correlation for User Type, Individual vs Other

correlations = df_subinfo_copy.corrwith(df_subinfo_copy.Purchase_Store)
correlations = correlations[correlations!=1]
positive_correlations = correlations[
correlations >0].sort_values(ascending = False)
negative_correlations =correlations[
correlations<0].sort_values(ascending = False)
```

```
print('Most Positive Correlations: \n', positive_correlations)
print('\nMost Negative Correlations: \n', negative_correlations)
```

[ ]:
```
#Correlation for Demo Type

correlations = df_subinfo_copy.corrwith(df_subinfo_copy.Demo_User)
correlations = correlations[correlations!=1]
positive_correlations = correlations[
correlations >0].sort_values(ascending = False)
negative_correlations =correlations[
correlations<0].sort_values(ascending = False)
print('Most Positive Correlations: \n', positive_correlations)
print('\nMost Negative Correlations: \n', negative_correlations)
```

[ ]:
```
#Set and compute the Correlation Matrix:
sn.set(style="white")
corr = df_subinfo_copy.corr()

#Generate a mask for the upper triangle:
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

#Set up the matplotlib figure and a diverging colormap:
f, ax = plt.subplots(figsize=(18, 15))
cmap = sn.diverging_palette(220, 10, as_cmap=True)

#Draw the heatmap with the mask and correct aspect ratio:
sn.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

[ ]:
```
# Calculating VIF - VIF determines the strength of the
#correlation of a variable with a group of other independent variables in a␣
 ↪dataset

def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]

    return(vif)

dataset = df_subinfo_copy[['Subscription_Type', 'Subscription_Event_Type',␣
 ↪'Demo_User', 'Purchase_Store', 'Free_Trial_User', 'User_Type',␣
 ↪'Email_Subscriber', 'Push_Notifications', 'Send_Count', 'Open_Count',␣
 ↪'Dollar_val', 'sub_len_mnth']]
calc_vif(dataset)
```

[ ]:
```
identity = df_subinfo_copy["ID"]
df_subinfo_copy = df_subinfo_copy.drop(columns="ID")
#Convert rest of categorical variable into dummy:
df_subinfo_copy = pd.get_dummies(df_subinfo_copy)
#Rejoin userid to dataset:
df_subinfo_copy = pd.concat([df_subinfo_copy, identity], axis = 1)
```

[ ]:
```
response = df_subinfo_copy['User_Type']
df_subinfo_copy = df_subinfo_copy.drop(columns="User_Type")
df_subinfo_copy = df_subinfo_copy.drop(columns='Subscription_Start_Date')
df_subinfo_copy = df_subinfo_copy.drop(columns='Subscription_Expiration')
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(df_subinfo_copy,␣
 ↪response,stratify=response, test_size = 0.4)

print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)


#remove ID
train_identity = X_train['ID']
X_train = X_train.drop(columns = ['ID'])

test_identity = X_test['ID']
X_test = X_test.drop(columns = ['ID'])
```

[ ]:
```
#feature scaling
sc_X = StandardScaler()
X_train2 = pd.DataFrame(sc_X.fit_transform(X_train))
X_train2.columns = X_train.columns.values
X_train2.index = X_train.index.values
X_train = X_train2
X_test2 = pd.DataFrame(sc_X.transform(X_test))
X_test2.columns = X_test.columns.values
X_test2.index = X_test.index.values
X_test = X_test2
```

[ ]:
```
#Using Accuracy and ROC AUC Mean Metrics


models = []
models.append(('Logistic Regression', LogisticRegression(solver='liblinear',␣
 ↪class_weight='balanced')))
models.append(('KNN', KNeighborsClassifier(n_neighbors = 5, metric =␣
 ↪'minkowski', p = 2)))
```

```
models.append(('Gaussian NB', GaussianNB()))
models.append(('Decision Tree Classifier',DecisionTreeClassifier(criterion =␣
 ↪'entropy')))
models.append(('Random Forest', RandomForestClassifier(n_estimators=100,␣
 ↪criterion = 'entropy')))

#Evaluating Model Results:

acc_results = []
auc_results = []
names = []
# set table to table to populate with performance results
col = ['Algorithm', 'ROC AUC Mean', 'ROC AUC STD',
       'Accuracy Mean', 'Accuracy STD']

model_results = pd.DataFrame(columns=col)
i = 0
# evaluate each model using k-fold cross-validation
for name, model in models:
    kfold = model_selection.KFold(
        n_splits=10)  # 10-fold cross-validation

    cv_acc_results = model_selection.cross_val_score(  # accuracy scoring
        model, X_train, y_train, cv=kfold, scoring='accuracy')

    cv_auc_results = model_selection.cross_val_score(  # roc_auc scoring
        model, X_train, y_train, cv=kfold, scoring='roc_auc')

    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)
    names.append(name)
    model_results.loc[i] = [name,
                            round(cv_auc_results.mean()*100, 2),
                            round(cv_auc_results.std()*100, 2),
                            round(cv_acc_results.mean()*100, 2),
                            round(cv_acc_results.std()*100, 2)
                            ]
    i += 1

model_results.sort_values(by=['ROC AUC Mean'], ascending=False)
```

[ ]:
```
fig = plt.figure(figsize=(15, 7))
ax = fig.add_subplot(111)
plt.boxplot(acc_results)
ax.set_xticklabels(names)
```

```
#plt.ylabel('ROC AUC Score\n',horizontalalignment="center",fontstyle =␣
 ↪"normal", fontsize = "large", fontfamily = "sans-serif")
#plt.xlabel('\n Baseline Classification␣
 ↪Algorithms\n',horizontalalignment="center",fontstyle = "normal", fontsize =␣
 ↪"large", fontfamily = "sans-serif")
plt.title('Accuracy Score Comparison \n',horizontalalignment="center",␣
 ↪fontstyle = "normal", fontsize = "22", fontfamily = "sans-serif")
#plt.legend(loc='top right', fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")


plt.show()
```

[ ]:
```
classifier = RandomForestClassifier(n_estimators=100, criterion = 'entropy')
classifier.fit(X_train, y_train)


# Predict the Test set results

y_pred = classifier.predict(X_test)


#Evaluate Model Results on Test Set:

acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

results = pd.DataFrame([['Logistic Regression', acc, prec, rec, f1, f2]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1␣
 ↪Score', 'F2 Score'])

print (results)
```

[ ]:
```
cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index = (0, 1), columns = (0, 1))
plt.figure(figsize = (28,20))


fig, ax = plt.subplots()
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, fmt='g'#,cmap="YlGnBu"
           )
```

```
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.tight_layout()
plt.title('Confusion matrix\n', y=1.1)
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
ax.xaxis.set_label_position("top")
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')
```

[ ]:
```
classifier.fit(X_train, y_train)
probs = classifier.predict_proba(X_test)
probs = probs[:, 1]
classifier_roc_auc = accuracy_score(y_test, y_pred )


rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, classifier.
 ↪predict_proba(X_test)[:,1])
plt.figure(figsize=(14, 6))

# Plot Logistic Regression ROC
plt.plot(rf_fpr, rf_tpr, label='Decision Tree (area = %0.2f)' %␣
 ↪classifier_roc_auc)
# Plot Base Rate ROC
plt.plot([0,1], [0,1],label='Base Rate' 'k--')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])



plt.ylabel('True Positive Rate \n',horizontalalignment="center",fontstyle =␣
 ↪"normal", fontsize = "medium", fontfamily = "sans-serif")
plt.xlabel('\nFalse Positive Rate \n',horizontalalignment="center",fontstyle =␣
 ↪"normal", fontsize = "medium", fontfamily = "sans-serif")
plt.title('ROC Graph \n',horizontalalignment="center", fontstyle = "normal",␣
 ↪fontsize = "22", fontfamily = "sans-serif")
plt.legend(loc="lower right", fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")



plt.show()
```

[ ]:
```
classifier = LogisticRegression(random_state = 0, penalty = 'l2')
classifier.fit(X_train, y_train)


# Predict the Test set results

y_pred = classifier.predict(X_test)


#Evaluate Model Results on Test Set:

acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)

results = pd.DataFrame([['Logistic Regression', acc, prec, rec, f1, f2]],
               columns = ['Model', 'Accuracy', 'Precision', 'Recall', 'F1␣
 ↪Score', 'F2 Score'])

print (results)
```

[ ]:
```
cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index = (0, 1), columns = (0, 1))
plt.figure(figsize = (28,20))


fig, ax = plt.subplots()
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, fmt='g'#,cmap="YlGnBu"
           )
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.tight_layout()
plt.title('Confusion matrix\n', y=1.1)
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
ax.xaxis.set_label_position("top")
plt.ylabel('Actual label\n')
plt.xlabel('Predicted label\n')
```

[ ]:
```
classifier.fit(X_train, y_train)
probs = classifier.predict_proba(X_test)
probs = probs[:, 1]
classifier_roc_auc = accuracy_score(y_test, y_pred )
```

```
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, classifier.
 ↪predict_proba(X_test)[:,1])
plt.figure(figsize=(14, 6))

# Plot Logistic Regression ROC
plt.plot(rf_fpr, rf_tpr, label='Logistic Regression (area = %0.2f)' %␣
 ↪classifier_roc_auc)
# Plot Base Rate ROC
plt.plot([0,1], [0,1],label='Base Rate' 'k--')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])



plt.ylabel('True Positive Rate \n',horizontalalignment="center",fontstyle =␣
 ↪"normal", fontsize = "medium", fontfamily = "sans-serif")
plt.xlabel('\nFalse Positive Rate \n',horizontalalignment="center",fontstyle =␣
 ↪"normal", fontsize = "medium", fontfamily = "sans-serif")
plt.title('ROC Graph \n',horizontalalignment="center", fontstyle = "normal",␣
 ↪fontsize = "22", fontfamily = "sans-serif")
plt.legend(loc="lower right", fontsize = "medium")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")



plt.show()
```

[ ]:
```
feature_importances = pd.concat([pd.DataFrame(df_subinfo_copy.columns, columns␣
 ↪= ["features"]),
           pd.DataFrame(np.transpose(classifier.coef_), columns = ["coef"])
           ],axis = 1)

feature_importances.sort_values("coef", ascending = False)
```

[ ]:
```

```

```
# Read in the data
app_activity <- read.csv('app_activity.csv')
sub_data <- read.csv('rosetta.csv')
sub_data <- data.frame(sub_data)

# Dimensionality and Shape of the Data
dim(app_activity)
dim(sub_data)
head(sub_data)



################################
# Data Preparation
################################

# Dropping unncessary columns
sub_data <- within(sub_data, rm('Purchase.Amount'))



# Label Encoding
library(superml)
lbl <- LabelEncoder$new()
for (column in names(sub_data)){
  if (is.character(sub_data[,column])){
    sub_data[,column] <- lbl$fit_transform(sub_data[,column])
  }
}
head(sub_data)



# Interpolation
library(imputeTS)
for (column in names(sub_data)){
  sub_data[,column] <- na.mean(sub_data[,column])
}
head(sub_data, 30)



# Normalize Non-Labeled Numeric Columns
nums <- c('Send.Count', 'Open.Count', 'Click.Count',
          'Unique.Open.Count', 'Unique.Click.Count',
          'Push.Notifications','Email.Subscriber')
cols_to_norm <- sub_data[,nums]

cols_to_norm <- scale(cols_to_norm)
head(cols_to_norm)

for(col in cols_to_norm){
  cols_to_norm[,col] <- (cols_to_norm[,col]-min(cols_to_norm[,col]))/
    (max(cols_to_norm[,col])-min(cols_to_norm[,col]))
```

```
}
sub_data[,nums] <- cols_to_norm
head(cols_to_norm)



# Feature Engineering
notification_feature <- rowSums(sub_data[,nums])
sub_data['Sum.Notify'] <- notification_feature
xs <- quantile(sub_data$Sum.Notify, c(0,1/4,1/2,3/4,1))
sub_data$Notifications <- as.numeric(cut(sub_data$Sum.Notify,
                              breaks = xs,
                              labels = c(1,2,3,4)))
sub_data$Notifications <- na_mean(sub_data$Notifications, option='median')
active <- rowSums(sub_data[,c("Email.Subscriber","Push.Notifications")])
sub_data$Active <- active







################################
# Data Visualization
################################
library(ggplot2)
library(reshape2)

# Dollar Value Means
pn1 <- sub_data[sub_data$Push.Notifications == 1,]
pn0 <- sub_data[sub_data$Push.Notifications == 0,]
Push_On <- mean(pn1$Dollar_val)
Push_Off <- mean(pn0$Dollar_val)

es1 <- sub_data[sub_data$Email.Subscriber == 1,]
es0 <- sub_data[sub_data$Email.Subscriber == 0,]
Email_Sub <- mean(es1$Dollar_val)
No_Sub <- mean(es0$Dollar_val)

nei <- sub_data[sub_data$Email.Subscriber == 0,]
nei <- nei[nei$Push.Notifications == 0,]
neither <- mean(nei$Dollar_val)
means <- table(Push_Off, Push_On, Email_Sub, No_Sub, neither)



###################################
# In-Depth Analysis and Modeling
###################################
library(corrplot)
C <- cor(sub_data)
corrplot(C, method='circle')

# Modeling
```

```
set.seed(410)
train_idx <- sample(1:nrow(sub_data), size=0.7*nrow(sub_data))
sub_train <- sub_data[train_idx,]
sub_test <- sub_data[-train_idx,]



# Singular Logistic Regression
y <- as.factor(sub_train$Push.Notifications)
log_1 <- glm(y ~ Language  + Subscription.Event.Type
             + Demo.User + Free.Trial.User + Auto.Renew + Country
             + User.Type + Lead.Platform + sub_len_mnth + Dollar_val,
             data=sub_train, family='binomial')

preds <- predict(log_1, newdata=sub_test, type='response')
preds <- ifelse(preds > 0.5, 1,0)
tab <- table(preds, sub_test$Push.Notifications)
tab



# Multi-Logistic Regression
library(caret)
require(nnet)
predictor <- as.factor(sub_train$Notifications)
multi <- multinom(predictor ~ Language  + Subscription.Event.Type
                  + Demo.User + Free.Trial.User + Auto.Renew + Country
                  + User.Type + Lead.Platform + sub_len_mnth + Dollar_val
                  , data= sub_train)
summary(multi)
exp(coef(multi))
preds <- predict(multi, newdata=sub_train, 'class')
tab <- table(preds,sub_train$Notifications)
round((sum(diag(tab))/sum(tab))*100,2)



# K - Means
library(factoextra)
km <- kmeans(sub_train, centers = 4, nstart=25)
avgs <- km$centers


# Random Forest
library(randomForest)
rf <- randomForest(y ~ Language  + Subscription.Event.Type
                   + Demo.User + Free.Trial.User + Auto.Renew + Country
                   + User.Type + Lead.Platform + sub_len_mnth + Dollar_val,
                   data= sub_train, ntree=100, mtry= 5, importance=TRUE)
rf
```

1 Set working directory/Read in the data

getwd() setwd("C:/Users/Chris/Desktop/MGSC410/finalproj") getwd()

sub_data <- read.csv ("Rosetta.csv")

delete column

sub_data <- within(sub_data, rm('Purchase.Amount'))

label encoding

library(superml) lbl <- LabelEncoder$new() for (column in names(sub_data)){ if (is.character(sub_data[,column])){ sub_data[,column] <- lbl$fit_transform(sub_data[,column]) } } head(sub_data)

Interpolation

library(imputeTS) for (column in names(sub_data)){ sub_data[,column] <- na.mean(sub_data[,column]) } head(sub_data, 30)

Normalize Non-Labeled Numeric Columns

nums <- c('Send.Count', 'Open.Count', 'Click.Count', 'Unique.Open.Count', 'Unique.Click.Count', 'Push.Notifications','Email.Subscriber') cols_to_norm <- sub_data[,nums] cols_to_norm <- scale(cols_to_norm)

for (col in cols_to_norm){ cols_to_norm[,col] <- (cols_to_norm[,col]-min(cols_to_norm[,col]))/ (max(cols_to_norm[,col])-min(cols_to_norm[,col])) } sub_data[,nums] <- cols_to_norm

#

Initial Exploratory Analysis

#

library(corrplot)6 C <- cor(sub_data) corrplot(C, method='circle')

Modeling

set.seed(410) train_idx <- sample(1:nrow(sub_data), size=0.7*nrow(sub_data)) sub_train <- sub_data[train_idx,] sub_test <- sub_data[-train_idx,]

Multi-Logistic Regression

library(caret) require(nnet) predictor <- as.factor(sub_train$Notifications) multi <- multinom(predictor ~ Language + Subscription.Event.Type + Demo.User + Free.Trial.User + Auto.Renew + Country + User.Type + Lead.Platform + sub_len_mnth + Dollar_val , data= sub_train) summary(multi) exp(coef(multi)) preds <- predict(multi, newdata=sub_test, 'class') tab <- table(preds,sub_test$Notifications) round((sum(diag(tab))/sum(tab))100,2)

K - Means

library(factoextra) km <- kmeans(sub_train, centers = 4, nstart=25) avgs <- km$centers avgs

Split data test/train

set.seed(410) train_idx <- sample(1:nrow(sub_data), size=0.7*nrow(sub_data)) sub_train <- sub_data[train_idx,] sub_test <- sub_data[-train_idx,]

#

Visualizations

#

FACETED BAR PLOT FOR PUSH NOTIFICATIONS

```
sumdol <- sub_data$Dollar_val

vis1 <- ggplot(data = sub_data, aes(x = Push.Notifications, y = Dollar_val)) + geom_bar(stat = "identity", color = "darkblue") + facet_wrap(vars(Push.Notifications), nrow = 1, ncol = 2) + labs(title = "Dollar Value With/Without Push Notifs", x = "Push Notifs (0=OFF, 1=ON ", y = "Sum Dollar Value") vis1

vis12<- ggplot(data = sub_data, aes(x = ))
```

FACETED BAR PLOT FOR EMAIL SUBSCRIPTIONS

```
vis2 <- ggplot(data = sub_data, aes(x = Email.Subscriber, y = Dollar_val)) + geom_bar(stat = "identity", color = "darkblue") + facet_wrap(vars(Email.Subscriber), nrow = 1, ncol = 2) + labs(title = "Dollar Value With/Without E-mail Sub", x = "Email Sub/No Sub", y = "Sum Dollar Value") vis2
```

BOXPLOT NOTIFICATIONS

```
click_send <- sub_data[,c('Send.Count', 'Open.Count', 'Click.Count', 'Unique.Open.Count', 'Unique.Click.Count', 'Push.Notifications')]

click_send_PN <- click_send[click_send$Push.Notifications == 1,] click_send_NPN <- click_send[click_send$Push.Notifications == 0,]

vis3 <- ggplot(data = sub_data, aes(x = click_send, y = click_send_NPN)) + geom_boxplot() vis3
```