Rahul Singh

Google Analytics Ecommerce Prediction Model: Final Report

I. Problem Statement

Google Analytics is a web analytics service that provides statistics and basic analytical tools for search engine optimization (SEO) and marketing purposes. On google ecommerce platform there is a daily traffic of users crawling through various products, google analytics can capture user behavior and log it on GCP Big query platform. We have a year worth of data that is a live connection capturing key insight, what can we conclude from the data. Many large companies have critical problems regarding user retention; they utilize marketing strategy as promotions, targeted ads, and discounts to keep that rate of retention high. This process is called Churning, where we determine based on user action/data whether that individual has churned away from the platform or is a returning customer. I decided to work on creating a prediction model to capture user churn rate as it can be adapted to help many companies with this recurring problem. It can be used by companies who want to keep their customers and note which user to target for a possible return to their platform via marketing or other strategies.

II. Data Import/Cleaning/Wrangling

The complete dataset was found on Google Cloud Platform under google ecommerce analytics table on Big Query. The data was found at this link:

https://console.cloud.google.com/bigquery?project=capstone-334200&ws=!1m5!1m4!4m3!1sbigquery-public-

data!2sgoogle_analytics_sample!3sga_sessions_20170801 and it resides on a Cloud SQL server that need to be extracted. Initially I reviewed the Schema that was provided on the cloud platform and realized due to resource limitation, I will need to run this on Google Collab. Big Query has a limit to amount of data you can extract per request(1GB) in a JSON format, So data was divide by month and extracted to personal Google Drive. With 90365 rows and 50 columns this data set fall under big data and requires big tools (Apache Spark) to run. After running a spark instance, we imported the "dirty" JSON file and started to clean each field to appropriate columns with labels per schema. Post exploding the data into columns, non-useful columns were dropped labeled under variable "droplist". There were multiple fields that required expression replacement that had unnecessary variables such as "[({, www, whitespace or com" , this action was taken to reduce data size to optimize computation rate. Once all data was cleaning it was ready for exploration, data was saved under parquet file as all big data files are easily retractable in that format compared to CSV, XSLX or Pandas data frame.

III. Exploratory Data Analysis

After the majority of data has been structured into respective fields, the features are evaluated for an occurring null value. Appropriate empty fields were replaced with zero that required to bridge the gap such as timeOnSite, newVisits, pageviews, referral path, server, and domain lookup time. These fields are critical to the project as they provide user behavior. I divide the fields between Categorical and Numerical variables to extract any key insights before moving forward. I started by observing trends in the user's choice of medium for interaction with the platform. This is represented by using various count plots as shown below.

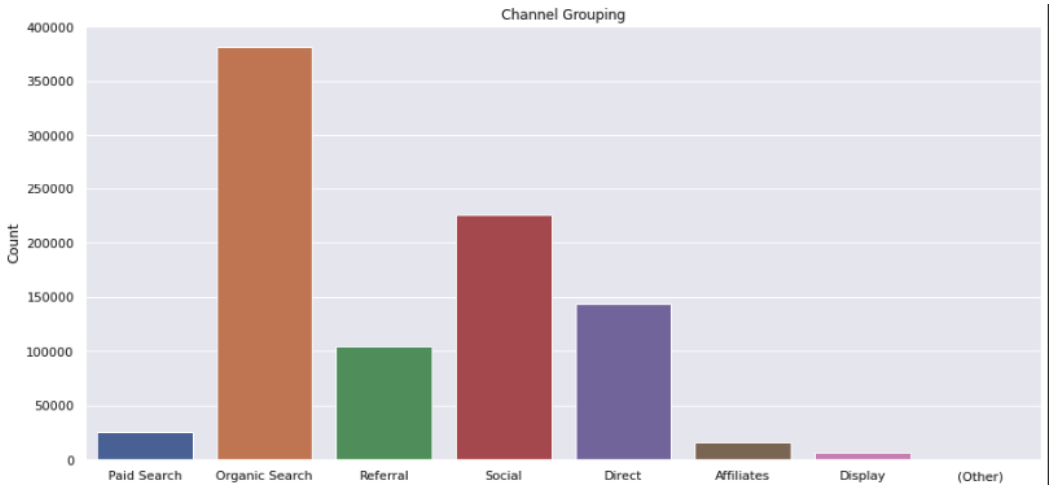

Figure 1: Channel Grouping Count Plot

One of the critical components that need to be evaluated was how user traffic is directed to the platform. After running a count on the channel grouping field that shows each user's way of reaching the platform, I saw Organic search to be most dominant followed by social and direct. Most users that interacted with the site said that most of them were utilizing it on desktop and had a quick view on each product line shown by figure below.

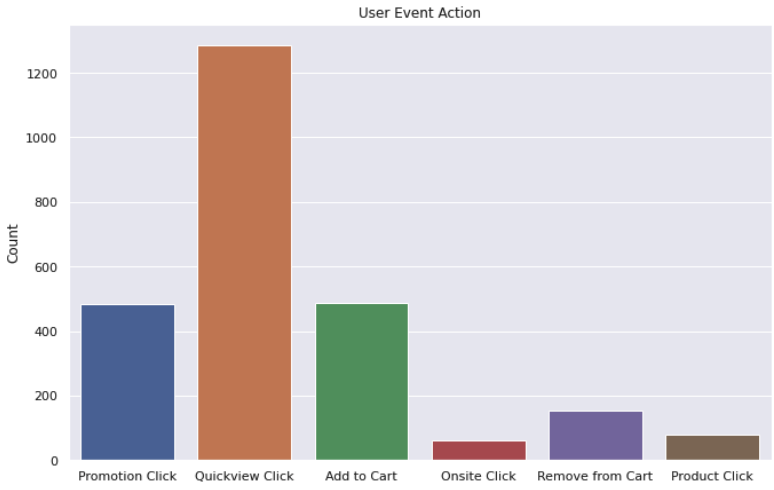

Figure 2: User Event Action Count Plot

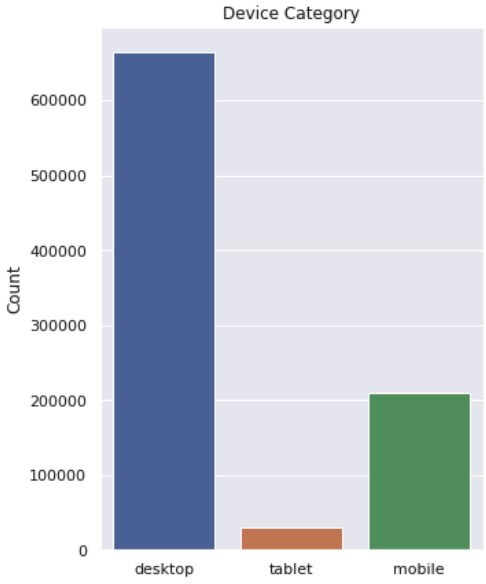

Figure 3: Device Category Count Plot

IV. Feature Engineering

My goal with feature engineering was to reduce data size and create features that show user behavior in quantified vector form to be utilized in our machine learning model. I first converted all data to respective data types and filled in missing data as conversion from unstructured JSON format. First critical step was to create a window function that will "look back 30 days" and "look ahead 30 days", we achieve this by creating a partition on fullVisitorID and orderby our date. This was a needed step as these window functions will guide user behavior with the platform in that time slot. Some of the key features we created were avg session,browstime, serverdelay time, avgdomain time and avg amount spent. These were aggregated from original data over the 30 day window. We had few categorical data that needed to be encoded into numerical values for our ML model, this was achieved by doing Hot Encoding with boolean values. Next step was to define a field labeled Churn, this will be our output variable to input data vector. Churn field is calculated based on aggregate value visitfrequency drive from visitnumber over past 30 day window if the user has returned he will

be labeled as 0 (active user) and if there has been no user return then 1 (user has churned). Once all new aggregated fields were created and evaluated for accuracy, we stored the file under a finaldf.parquet and moved forward to modeling our data.

V. Modeling/Machine Learning

After understanding a user behavior based on features, I went on to the model and machine learning stage. I performed a 70/30 split train/test split on the dataset, where X vector excluded our output variable 'Churn_vf30d' and Y column was set as our output variable. I started with a standard random forest model giving a n_estimator of 60 and max_dept of 4, then we fit the split data on our model and run to get model accuracy. I received a model accuracy of 91.8% and printed out a confusion matrix to investigate, we knew the model was overfitting our data. Then a parameter grid search was run for hyperparameter tuning result yield near similar parameter concluding to a 90% model accuracy.

When we run a predict function the model will yield 1 if greater than 0.50 or 0. What the model was doing was quantifying everything to 1 as our probability predicted function value was off. We ran a test to pick a predicted value with the highest response rate as shown below.

	Tier	Observations	Min-Prob	Mean-Prob	Max-Prob	Responders	Non-Responders	Response-Rate(%)	Cum-Response(%)	Cum-Non-Response(%)	KS-Score
0	1	24866	0.956997	0.958080	0.960987	24494	372	98.503981	10.82	1.67	9.15
1	2	24867	0.956997	0.956997	0.956997	24548	319	98.717175	21.66	3.11	18.55
2	3	24867	0.952481	0.953999	0.956997	24381	486	98.045603	32.43	5.29	27.14
3	4	24867	0.950764	0.951506	0.952481	24086	781	96.859291	43.07	8.80	34.27
4	5	24867	0.949844	0.950672	0.950764	24102	765	96.923634	53.71	12.24	41.47
5	6	24867	0.945972	0.947784	0.949844	23727	1140	95.415611	64.19	17.36	46.83
6	7	24867	0.936157	0.942052	0.945972	23098	1769	92.886154	74.39	25.31	49.08
7	8	24867	0.907874	0.920683	0.936157	22223	2644	89.367435	84.21	37.20	47.01
8	9	24867	0.844435	0.882020	0.907874	20241	4626	81.397032	93.15	57.99	35.16
9	10	24867	0.114378	0.642490	0.844435	15519	9348	62.408011	100.00	100.00	0.00

Figure 4: Predicted value table for response rate

We concluded that a 0.54 predicted cutoff will provide a more accurate model for future data. Once we replaced the value and ran roc_auc_score it should the model accuracy to be 81.8% which is more accurate as it's not overfitting our datasets.

VI. Conclusion and Future Direction

This random forest model provides a great prediction model for user behavior and helps companies investigate user retention needs. For future use cases I would like to try different models such as SVM or K-means clustering to see if a more realistic model can be created for production deployment.