Google Analytics Customer Revenue Prediction

Rahul Vaswani

Introduction

Google Merchandise Store (also known as GStore) has given a customer dataset in order to predict revenue per customer. This is a form of a customer lifetime value model (CLV). Essentially, this project's aim is to use machine learning, specifically regression, in order to

predict how much a customer may spend in the store given customer attributes and segments. There are many methods to do CLV models and this project closely followed the method that was explained here, with the slight caveat of instead of doing a multi-classification problem, a regression was performed. By using a slight deviation of recency, frequency, and monetary clusters, our goal is to implement a continuous numerical output of a customers lifetime value.

We will use customer transaction data as an input to predict total customer spend within a certain window via clustering and regression techniques. This helps answer the question: *Can we accurately predict how much a customer will spend, thus allowing marketers to target high spending customers with more advertisements?*

We will ultimately use a Root Mean Squared Error metric to score how well our model performs.

Exploratory Data Analysis

The dataset that was used was provided by Google to Kaggle and was in a json format. There was some preprocessing code needed to clean the data up, but after cleaning, the data had 29 columns and 903653 rows.

The schema of the raw file looks like the following:

- fullVisitorId- A unique identifier for each user of the Google Merchandise Store.
- channelGrouping - The channel via which the user came to the Store.
- date - The date on which the user visited the Store.
- device - The specifications for the device used to access the Store.
- geoNetwork - This section contains information about the geography of the user.
- socialEngagementType - Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- trafficSource - This section contains information about the Traffic Source from which the session originated.
- visitId - An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.
- visitNumber - The session number for this user. If this is the first session, then this is set to 1.
- visitStartTime - The timestamp (expressed as POSIX time).
- hits - This row and nested fields are populated for any and all types of hits. Provides a record of all page visits.
- customDimensions - This section contains any user-level or session-level custom dimensions that are set for a session. This is a repeated field and has an entry for each dimension that is set.
- totals - This set of columns mostly includes high-level aggregate data.

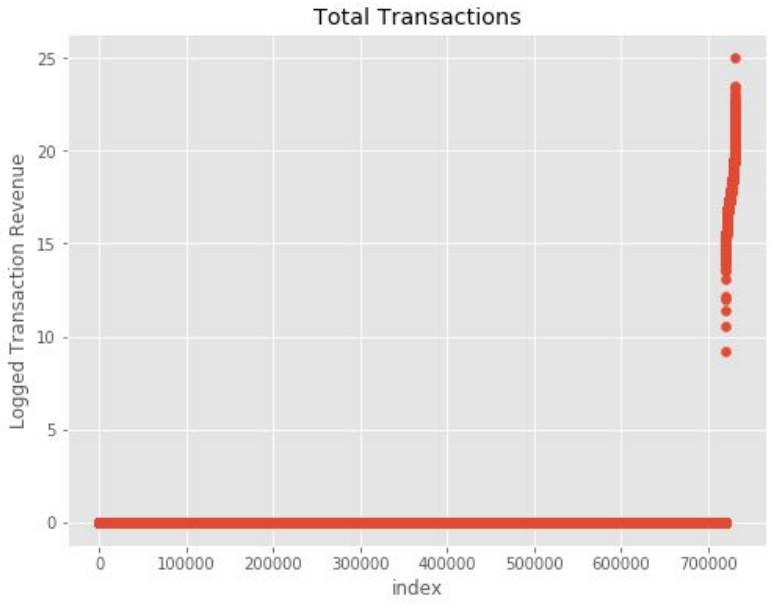
There are a few JSON formatted fields that are expanded on in the EDA. Thus, note that any feature that has a period in it is from an expanded JSON field.

From the columns above, we have interest in channelGrouping, device.deviceCategory, device.operatingSystem, totals.hits, totals.pagviews, and totals.transactionRevenue.

Of these columns, the only column that had null values was the outcome; however, this is expected and since some visitors do not buy anything. I decided to keep the missing values in for feature engineering purposes, but I will zero fill when I begin to train the models.

Outcome: totals.transactionRevenue

TransactionRevenue is the total transaction revenue expressed as a value multiplied by 10^6. Since these values are quite big and we are ultimately want to check our score with the logged value (discussed in Model and Findings sections), we plot the log of the values:

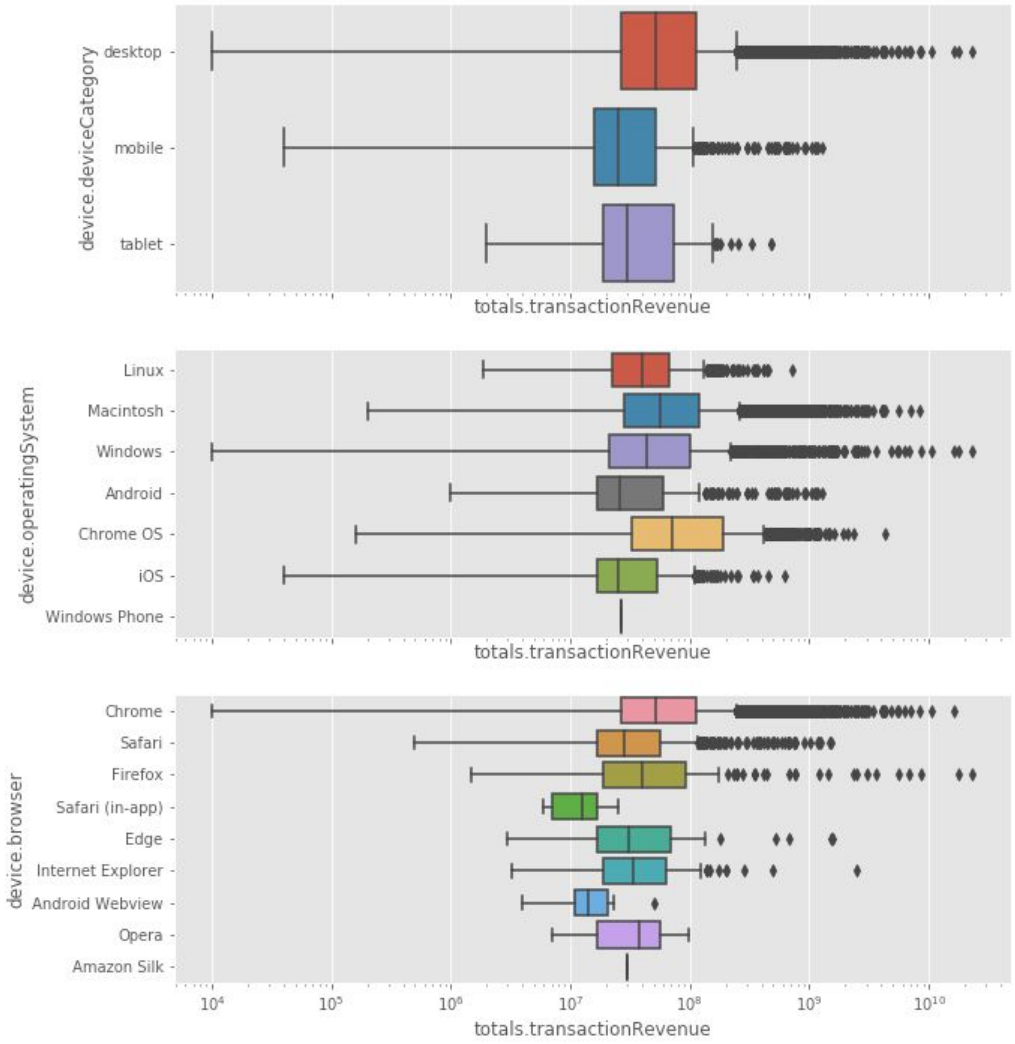


We see from the graph above that there is some sort of Pareto principle in effect going on. It seems that of all the observations, almost all of the transactions are coming from around 20% of observations. This is interesting because from a business standpoint, we should try to pinpoint these users in order to target ads toward them.

Control Features

Device Category, Operating System, and Browser

We next explore the device features in order to see if they will be influential in our model. The rationale behind the device features is the belief that Android or Chrome users might be more willing to buy Google Store products rather than Apple or Windows users. Furthermore, another belief is that most purchases would come from desktop or tablets rather than mobiles.



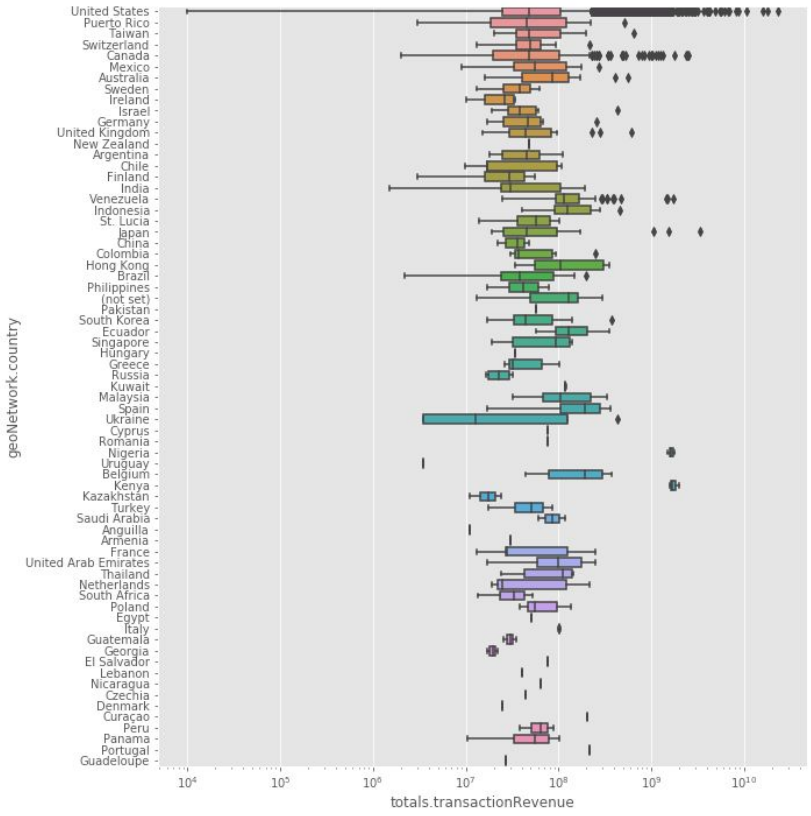
We see from the boxplots above that a few of our hypotheses seem justified (at least from the data perspective). Desktop users tend to have the most transactions, and while mobile users have high transactions, tablet users have a higher median transactions. Looking at operating system; however, shows that Chrome OS users tend to buy more on median; however, MacOS users are not that far behind. Interestingly, we see that Android and iOS users have almost the same median transactions. Unsurprisingly Chrome seems to be the browser of choice for

purchases (though with the largest browser market cap as of June 2020, it's expected); however, Firefox seems to be the second favorite browser of choice for users.

Of the three device categories that were explored, we will continue with device category and device operating system in our model since there were less categories in those groups.

Country

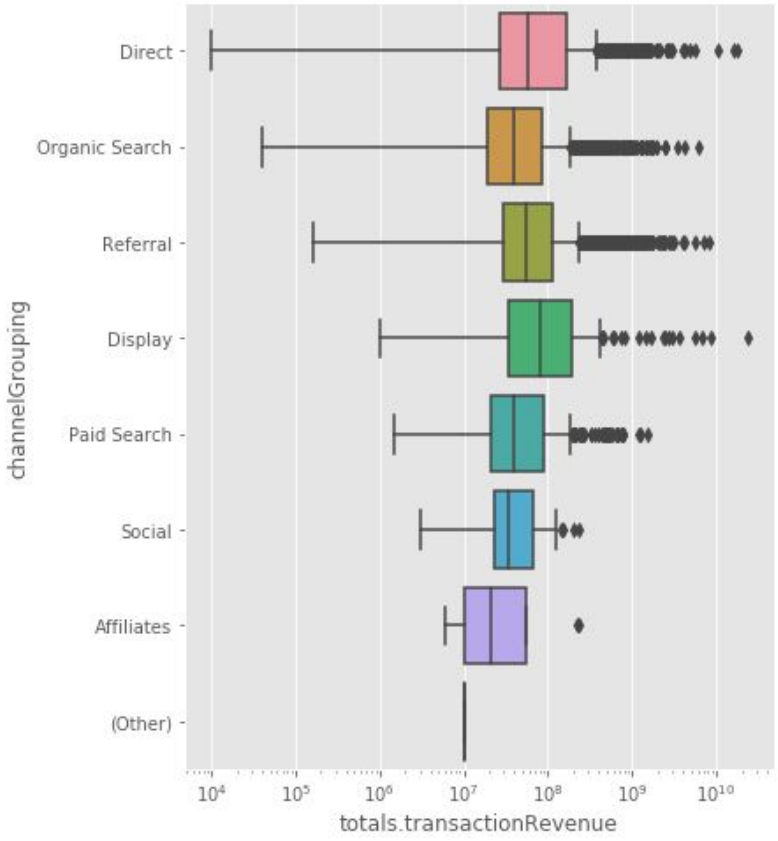
We next explored if countries played a significant role in transactions. The hypothesis here is that a lot of users from the US will buy goods; however, we will see very few purchases from other countries.



While the graph above is a bit hard to read, it does show that my assumption that many countries will have no purchases is false. From the boxplot above, we can see that there are clearly countries with very similar patterns in transaction revenue and thus one idea that could be useful here is to explore clustering the countries together (data driven) and seeing what the clusters look like.

Channel Grouping

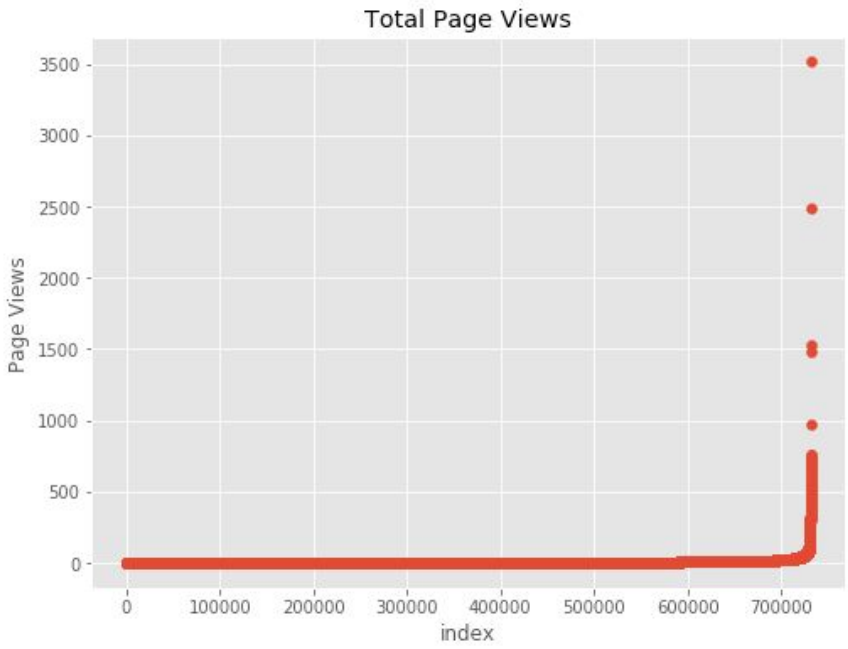
Channel grouping is an interesting feature because it shows how users ultimately got to the Google Store website. They could have directly typed in the address (Direct) or they could have searched for the store on a web search (Organic Search). What I wanted to explore was whether an advertisement (Display) is influential on a user purchasing a product.



We see that Display has the highest median transaction (which helps support my hypothesis). Interestingly, Direct has a quite high median transaction which means that many users went directly to the Google Store in order to make a purchase. This data would be very interesting to explore if we were to conduct a multi-touch attribution model.

Indirect Outcome: Page Views

The basis for our customer lifetime value model is using a Revenue, Frequency, and Monetary clustering technique in order to segment customers; however, the problem with this approach is that it is useful for inferential purposes and not predictive purposes. This is due to us not know how much a customer will spend in the future or even for customers who have not spent anything yet (some customers visit the Google Store numerous times before making a purchase). With this in mind, we needed to find an indirect outcome that we could use in lieu of transactions, and ultimately we chose to explore page views.



We see from the chart above that we can still see the slight Pareto effect that we saw with revenue; however, we are not able to use a precursor to outcome to help predict outcome (users must first visit a webpage before making a purchase [where purchases are not always guaranteed]).

Now that we have our set of features chosen, we need to do some feature engineering.

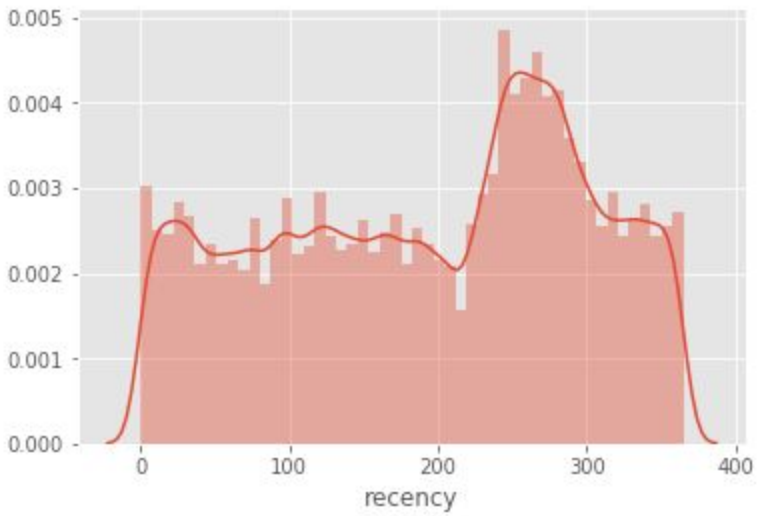
Feature Engineering

The basis of our CLV model is the idea of RFM and the need to calculate recency, frequency, and monetary clusters for each customer. As discussed in the EDA section we will not use transactions to generate these clusters; rather, we will use page views.

In order to generate these clusters, we use k-means clustering from scikit-learn to find the ideal cluster number for each of these features.

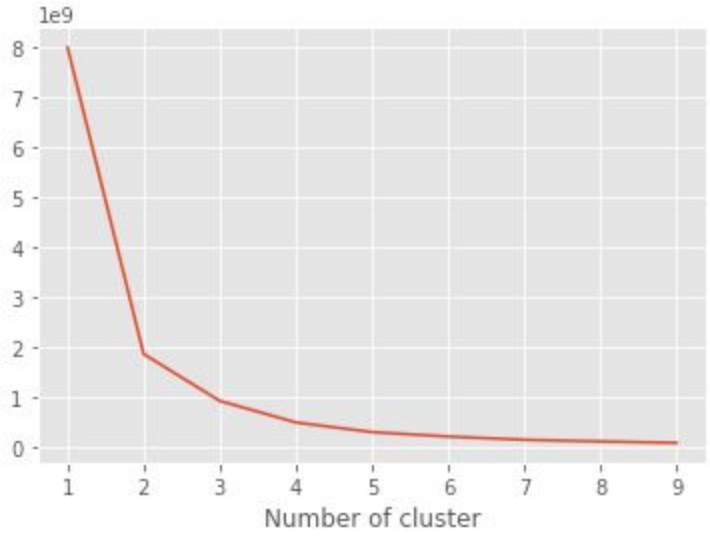
Recency

The first feature that was calculated was the recency metric. This metric allows us to group customers into clusters based on how recent the customer's visit is at a specific observation from the last visit recorded in the system. We first calculate the recency metric by taking the maximum date and subtracting each observation date in order to get the day difference. This allows us to make a few observations about the data:



From the above histogram we see that the majority of purchases took place earlier in the year (higher number means further away from the max). We note that the data ranged from August 1, 2016 to August 1, 2017, and from the histogram above, we see that a lot of the purchases took place from October 2016 to January 2017, which makes a lot of sense considering this would fall under the holiday period.

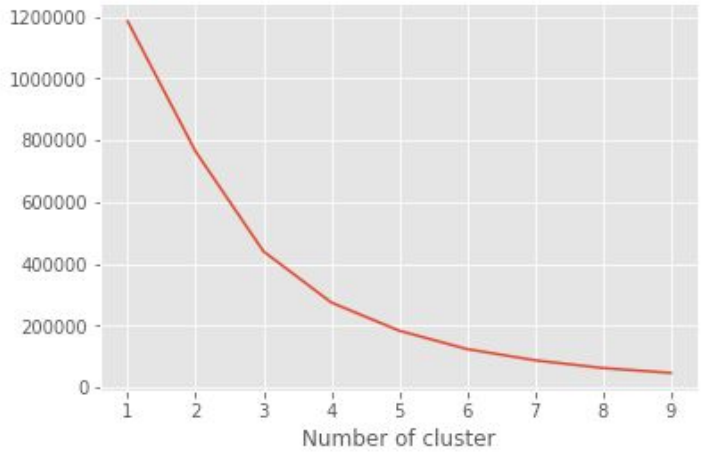
Now that we have our Recency metric, we can calculate the appropriate number of clusters based off of the explanatory power of each cluster (the inertia attribute of the k-means object):



From the graph above, we see that the elbow was between 2 and 4. Ultimately, I settled on 4 clusters since I wanted more variability in the data.

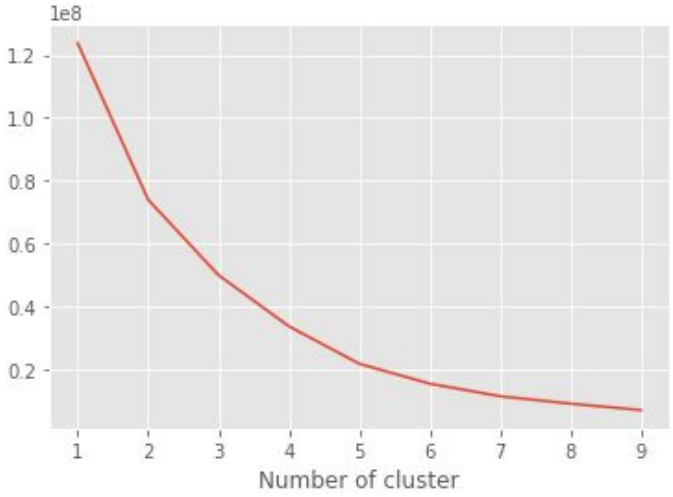
Frequency

Similar to our Recency metric, the Frequency metric tells us how often a customer visited the Google Store. This allows us to distinguish frequent customers versus one time shoppers or visitors. This metric was easily calculated by just taking the count of visits grouped by each visitor. Once that metric had been calculated, we clustered the data based on a graph similar to the one above; however, there was no clear elbow, so I settled on 5 clusters.



Page Views

As discussed earlier, we cannot use a monetary feature since that is our outcome; thus, we choose to use an indirect outcome, in this case page views. As examined earlier, page views share a similar trend as transactions. We calculate page views by summing all page views for a single user and then clustering the data. In this case, I settled on four clusters.



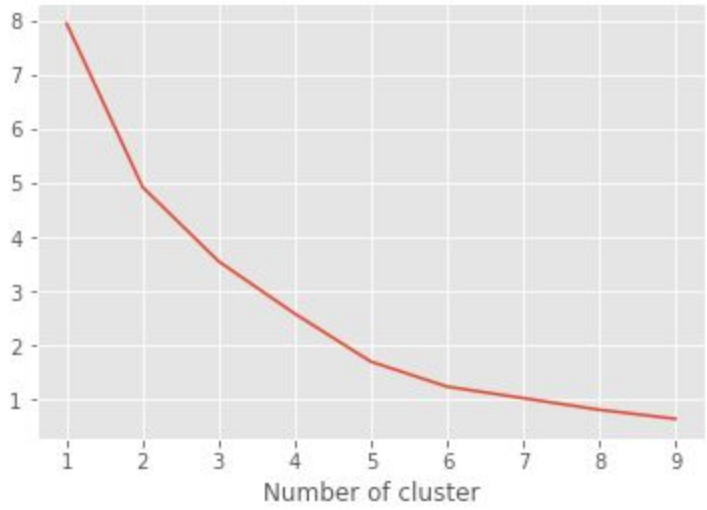
Country

We saw in the EDA that country could play an important role in figuring out where customers were coming from; however, since this is a categorical variable, we would need to figure out a way to deal with it.

The first thought was to one hot encode all the countries and plug that into the model. Naturally, this caused our data to blow up in width and caused endpoint and training errors with sagemaker.sklearn. Thus, I needed to find a better way to deal with countries. I could have done a Principle Component Analysis on the one hot encoded features; however, I thought that this seemed to be a bit overkill and somewhat pointless.

My next thought was to label encode each country. This was easy enough to do -- there is already a method in sklearn that allows us to do this; however, the issue with this is that label encoding creates some form of ordinality. Thus if the US is 1 and Canada is 100, we would be saying that Canada is better than the US. Nevertheless, trying this showed me that fewer features was the key to getting my estimator to train and deploy.

I finally settled on letting the data speak for itself. I would cluster the countries based off of page views attributes and then one hot encode those clusters.



The graph showed me that 3-5 clusters would be ideal, thus wanting more variability I chose 5. Unlike the other RFM metrics, I was very careful to check what was included in each cluster, and to no surprise, I saw the US in its own cluster, and Canada, the UK, and a few other obvious countries clustered together. This reassured me that the clusters aligned with my own beliefs about the data.

Finally with the features engineered, we could begin the modeling and deployment phase.

Modeling and Deployment

Pre Model

Before the data was modeled, I still had to transform the categorical features as well as the clustered features. As discussed earlier, I played around with Label Encoding the features; however, this was dangerous due to the model picking up any false ordinality in the features, thus I decided to settle on One Hot Encoding and clustering. One Hot Encoding these features would give a dummy variable for each of the categories in a specific feature. Pandas allowed this to be done fairly easily with pd.get_dummies() and the final dataset that the model was trained on had 117 features.

Baseline Model: Linear Regression

Ultimately, we want to predict how much a customer will spend in the Google Store based on their attributes. Therefore, we needed to apply a regression model and what better model to baseline our results with than a Ordinary Least Squares model. The Linear Regression model chosen did not have any penalization terms, thus it was not a Ridge or Lasso regression and it took the features in.

Instead of using Sagemaker's LinearLearner method, I decided to stay in familiar territory and use sagemaker.sklearn.estimator with a seperate train.py script that called sklearn's LinearRegression.

Final Model: Random Forest Regressor

Although my proposal suggested using XGBoost, I ultimately decided to stick to a bagging methodology, something in the sklearn library, and also something I could easily explain. Thus I decided to use a Random Forest Regressor as the final model. For the Random Forest model, I did not do any automated hyperparameter tuning, since I wanted to keep this model as simple as possible and the hyperparameters that I set values for were n_estimators, max_depth, min_samples_leaf, and min_samples_split since all of these I can explain clearly what they are doing and why they were chosen. I manually cycled between 50, 100, and 200 estimators and a depth of 5, 10, 20, and 50. The results of these hyperparameter scores were not logged, since once I settled on my final values, I decided to adjust my features and log those results instead. The model that I settled on was a Random Forest Regressor with 100 trees, each with a depth of 20, with a minimum of 5 leaves and a minimum split of 5.

Once again, sagemaker.sklearn.estimator was used on a train-rf.py script that called sklearn's RandomForestRegressor method.

Model Iterations

I iterated through the models many times with different feature settings in order to achieve a result that I was happy with. I scored the models using Root Mean Squared Error. RMSE was calculated by taking the square root of the mean_squared_error method of the sklearn library.

The below table shows the result of my models:

	Model	Description	Score
1	Linear Regression	Unaggregated user names. Thus multiple predictions for the same person.	9.398186193093983
1	Random Forest		4.3491594153608135
2	Linear Regression	Aggregated usernames. One prediction per user. agg_query = {'totals.transactionRevenue' : 'sum', 'country_cluster' : 'max', 'totals.hits' : 'sum', 'recency_cluster' : 'max', 'frequency_cluster' : 'max', 'pg_views_cluster' : 'max' } Mode for all other features	8.875329144575948
2	Random Forest		4.1387954942244525
3	Linear Regression	Split transaction users and non-transaction users based on the same agg query as 2.	8.937465817423477
3	Random Forest		4.187379214579118
4	Linear Regression	Final model. Removed VisitorId (accidentally left in)	1.8073501522850421
4	Random Forest		1.7398227244278657

As seen in the table above, I went through a lot of iterations of both models (there were more that actually had failed endpoints and predictions); however, I am happy with my final score. There were a few mishaps in the model building process, namely, model runs 2 and 3 left a primary key of VisitorId in the model, thus the scores were quite low; however, after fixing the mistake, the model score improved dramatically.

Feature Importances

Below are the features that were most impactful in the final model:

('totals.hits', 0.634136923774189),
('country_cluster_4', 0.11431282157527779),
('channelGrouping_Referral', 0.02986123495764036),
('frequency_cluster_4', 0.025602525107169892),
('device.operatingSystem_Macintosh', 0.021660522780089626),
('recency_cluster_2', 0.020850627188186362),
('recency_cluster_1', 0.02042571089392118),
('recency_cluster_3', 0.01837760324185198),
('pg_views_cluster_1', 0.016289693018715915),
('frequency_cluster_1', 0.015779600294175123),
('channelGrouping_Organic Search', 0.012590486923947219),
('channelGrouping_Direct', 0.01190426263737624),
('device.operatingSystem_Windows', 0.010056289595634572),
('device.deviceCategory_mobile', 0.009481584478209232),
('device.operatingSystem_Chrome OS', 0.006768179969865032),
('device.operatingSystem_Linux', 0.00641365688998192),
('channelGrouping_Paid Search', 0.003859589498247557),
('device.operatingSystem_iOS', 0.0036168347843567213),
('device.operatingSystem_Android', 0.002833936945679645),
("channelGrouping_['Direct' 'Referral']", 0.0025641964152681492),
('device.deviceCategory_tablet', 0.002112146711973257),
('country_cluster_1', 0.0017210168913030523),
("channelGrouping_['Organic Search' 'Referral']", 0.0016437856676118365),
('country_cluster_3', 0.001214689772355175),
('channelGrouping_Display', 0.001099968976326094),
('channelGrouping_Social', 0.0010277992575109353),
("channelGrouping_['Paid Search' 'Referral']", 7.637325207621112e-06),
('channelGrouping_Affiliates', 7.265458491840136e-06),
("channelGrouping_['Display' 'Paid Search']", 5.41076953721966e-06),

As expected, total.hits is at the top since ultimately to buy a good you have to visit the site. Country cluster 4 is the United States, which also makes sense since the US was in it's own cluster and had the highest transactions. Referrals were surprising since I expected direct to be higher than referrals.

Interestingly, we see that the RFM clusters that we created in our feature engineering portion had high feature importance which reassures me that a good methodology was chosen to create this CLV model.

Final Model Score and Potential Room for Improvement

From the table above, we see that my final model score resulted in an RMSE of 1.74. While this is still high, I am satisfied with the score since it is a result of numerous iterations and testing. I could have probably scored better using an XGBoost model; however, if a business user asked me what was going on under the hood of an XGBoost model, I would have to research more rather than being able to tell him/her immediately what was going on under the hood. I could have also included more numeric features; however, I felt that I would be p-hacking my original hypothesis.

Learnings

One of my biggest frustrations during this exercise was trying to debug why I was getting an endpoint error or a training error. The training errors were easier to debug since I could at least follow some of the Traceback errors; however the deployment errors were more ambiguous. Ultimately, I discovered the majority of the errors were due to the batch size of data getting passed to the endpoint. I discovered that the wider my data was, the less amount of observations I could pass into the endpoint.

Conclusion

Ultimately, we sought to build a model to predict Customer Lifetime Value, and while we settled on an imperfect model, it is a model that I can thoroughly explain to shareholders if they require. This capstone project also introduced the overall concept of CLV modeling as well as RFM clustering which I found thought provoking and insightful.

While the final model score is respectable, I was able to answer my research question from the EDA itself. Recall that we tried to answer the question: *Can we accurately predict how much a customer will spend, thus allowing marketers to target high spending customers with more advertisements?* Our EDA displayed a form of the Pareto Principle, showing us that 20% of the customers resulted in 80% of the purchases. Thus if marketers wanted to target specific people, they should focus more on retargeting ads than prospective ads. In fact, by using the RFM clusters, I have essentially segmented out users which can be used by marketers.

If I were to take this project further and into production, I would want to create a dashboard of all customers and their estimated lifetime value. I would also want to create a simple tool that sales and marketing teams can use to input a prospective customer's data and see whether or not it is worth the investment to pursue the potential client.