INSY 5377-002

Google Analytics Project Report

Summer 2022

Kamarga, Christian

Shukla, Ruchi

Van Keer, Bart

Table of Contents

1. Introduction

Businesses need analytics tools and data to reveal insight that can be used towards optimizing their online site and understanding their customers.

In this presentation, we will observe the web analytics data of a Google Merchandise Store (called GStore) that sells Google themed merchandise and paraphernalia. The store's online site attracts visitors from various locations throughout the world.

https://shop.googlemerchandisestore.com/

2. Data Description & Source

GStore's customer dataset was downloaded from Kaggle:

https://www.kaggle.com/competitions/ga-customer-revenue-prediction/data?select=train.csv

In context, Kaggle and Google Cloud partnered with RStudio, a developer of many free and open source tools for business analytics. Together, they hosted an official competition with this dataset to predict revenue per customer.

There were roughly a million records (or rows) in the Spreadsheet. The visits range from Q3 2016 through Q2 2017.

3. Variables

- **channelGrouping** - The referent channel or source that explains how the user visited the Store.
- **date** - The date on which the user visited the Store.
- **device** - The specifications for the device used to access the Store.
- **fullVisitorId**- A unique identifier for each user of the Google Merchandise Store.
- **geoNetwork** - This section contains information about the geography of the user.
- **hits** - This row and nested fields are populated for any and all types of hits. Provides a record of all page visits.
- **socialEngagementType** - Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- **totals** - This set of columns mostly includes high-level aggregate data.
- **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).

4. Research Questions

Descriptive Analysis:

- Which independent variables influence the dependent variable the most?
- From where (geographic) are visitors coming? Based on website behaviors, are certain places more important to us than others?
- What devices, browser, and operating system do visitors prefer?
- What are the number of total hits, pageviews, and conversion rate?
- What is the likelihood a visitor will see our website just once? What percentage of visitors return?
- Does one traffic source category dominate disproportionally?
- What is the trend for organic search traffic?
- What is the aggregated revenue generated from each user based on our independent variables?

Predictive Analysis:

- Which features most contributed to the conversion of visitors? Assuming circumstances in the future are similar, which features would most benefit GStore to focus on? How can GStore use this information to improve?

5. Methodology

We used Jupyter Notebook for preprocessing/cleaning our dataset. Initially, we had 12 variables. After applying JSON on 4 of the 12 variables, we obtained a total of 55 columns for our training set.

```
'''
    INSY 5377-002
    Team 01
    JSON method
'''
def load_gstore_train_df(csv_path = './train.csv', nrows = None):
    JSON_COLUMNS = ['device', 'geoNetwork', 'totals', 'trafficSource']

    df = pd.read_csv(csv_path,
                     converters = {column: json.loads for column in JSON_COLUMNS},
                     dtype = {'fullVisitorId': 'str'},
                     nrows = nrows)

    for column in JSON_COLUMNS:
        column_as_df = json_normalize(df[column])
        column_as_df.columns = [f"{column}.{subcolumn}" for subcolumn in column_as_df.columns]
        df = df.drop(column, axis = 1).merge(column_as_df, right_index = True, left_index = True)
    print(f"Loaded {os.path.basename(csv_path)}. Shape: {df.shape}")
    return df

def load_gstore_test_df(csv_path = './test.csv', nrows = None):
    JSON_COLUMNS = ['device', 'geoNetwork', 'totals', 'trafficSource']

    df = pd.read_csv(csv_path,
                     converters = {column: json.loads for column in JSON_COLUMNS},
                     dtype = {'fullVisitorId': 'str'}, # Important!!
                     nrows = nrows)

    for column in JSON_COLUMNS:
        column_as_df = json_normalize(df[column])
        column_as_df.columns = [f"{column}.{subcolumn}" for subcolumn in column_as_df.columns]
        df = df.drop(column, axis = 1).merge(column_as_df, right_index = True, left_index = True)
    print(f"Loaded {os.path.basename(csv_path)}. Shape: {df.shape}")
    return df
```

```
%%time
'''
    INSY 5377-002
    Team 01
    Run JSON function.
'''

training_df = load_gstore_train_df()
testing_df = load_gstore_test_df()
```

```
Loaded train.csv. Shape: (903653, 55)
Loaded test.csv. Shape: (804684, 53)
Wall time: 2min 15s
```

However, some of these columns did not contribute meaningful information to the analysis (e.g. Null (NaN) values, constants, etc.) and we decided to drop them.

```
'''
    INSY 5377-002
    Team 01
    Listing the constant columns for training dataset.
'''
training_df["totals.transactionRevenue"] = training_df["totals.transactionRevenue"].astype('float')
constant_columns = [col for col in training_df.columns if training_df[col].nunique(dropna=False)==1 ]
constant_columns
```

```
['socialEngagementType',
 'device.browserVersion',
 'device.browserSize',
 'device.operatingSystemVersion',
 'device.mobileDeviceBranding',
 'device.mobileDeviceModel',
 'device.mobileInputSelector',
 'device.mobileDeviceInfo',
 'device.mobileDeviceMarketingName',
 'device.flashVersion',
 'device.language',
 'device.screenColors',
 'device.screenResolution',
 'geoNetwork.cityId',
 'geoNetwork.latitude',
 'geoNetwork.longitude',
 'geoNetwork.networkLocation',
 'totals.visits',
 'trafficSource.adwordsClickInfo.criteriaParameters']
```

```
'''
    INSY 5377-002
    Team 01
    Showing training_df after eliminating constant columns.
'''
training_df = training_df.drop(columns = constant_columns)
training_df
```

As a result, out of the initial 55 columns, we only considered 36 columns for our analysis.

	channelGrouping	date	fullVisitorId	sessionId	visitId	visitNumber	visitStartTime	device.browser
0	Organic Search	20160902	1131660440785968503	1131660440785968503_1472830385	1472830385	1	1472830385	Chrome
1	Organic Search	20160902	377306020877927890	377306020877927890_1472880147	1472880147	1	1472880147	Firefox
2	Organic Search	20160902	3895546263509774583	3895546263509774583_1472865386	1472865386	1	1472865386	Chrome
3	Organic Search	20160902	4763447161404445595	4763447161404445595_1472881213	1472881213	1	1472881213	UC Browser
4	Organic Search	20160902	27294437909732085	27294437909732085_1472822600	1472822600	2	1472822600	Chrome
...	...	...	...	...	...	...	...	...
903648	Social	20170104	5123779100307500332	5123779100307500332_1483554750	1483554750	1	1483554750	Chrome
903649	Social	20170104	7231728964973959842	7231728964973959842_1483543798	1483543798	1	1483543798	Chrome
903650	Social	20170104	5744576632396406899	5744576632396406899_1483526434	1483526434	1	1483526434	Android Webview
903651	Social	20170104	2709355455991750775	2709355455991750775_1483592857	1483592857	1	1483592864	Chrome
903652	Social	20170104	0814900163617805053	0814900163617805053_1483574474	1483574474	1	1483574474	Chrome

903653 rows × 36 columns

5.1 Post-Processing Variables

Independent Variables:	
channelGrouping	totals.bounces
date	totals.newVisits
fullVisitorId	totals.hits
sessionId	totals.pageviews
visitId	trafficSource.campaign
visitNumber	trafficSource.source
visitStartTime	trafficSource.medium
device.browser	trafficSource.keyword
device.operatingSystem	trafficSource.isTrueDirect
device.isMobile	trafficSource.referralPath
device.deviceCategory	trafficSource.adwordsClickInfo.page
geoNetwork.continent	trafficSource.adwordsClickInfo.slot
geoNetwork.subContinent	trafficSource.adwordsClickInfo.gclId
geoNetwork.region	trafficSource.adwordsClickInfo.adNetworkType
geoNetwork.metro	trafficSource.adwordsClickInfo.isVideoAd
geoNetwork.city	trafficSource.adContent
geoNetwork.networkDomain	trafficSource.campaignCode
Dependent Variable:	
totals.transactionRevenue	

After preprocessing/cleaning our dataset, we looked at the correlation of the independent variables and several descriptive analysis/models (Q #1-7), such as demographics, type of device, and traffic source. Finally, before moving to our predictive models, we compared the total revenue for several of the independent variables (Q #8).

For our predictive analysis, we used the following predictive models (Q #9):

- Light Gradient Boosting Machine (LGBM)
- Time Series
- Logistic Regression
- Random Forest

6. Results & Discussion

6.1 Question 1

Which independent variables influence the dependent variable the most?

```
'''
    INSY 5377-002
    Team 01
    Question 1
'''

import matplotlib.pyplot as plt

plt.figure(figsize=(20, 16))
sns.heatmap(training_df.corr(), annot=True)
plt.show()
```

The above code produces a HeatMap that displays the correlation between the dependent and independent variables. The lighter colors indicate a strong relationship between the corresponding pairs of variables.

Figure 1: Displaying Correlation Between Variables

- The heatmap shows there is a very strong correlation between **visitStartTime** and **visitId** (1.00).
- In addition, both also show a strong correlation with **date** (0.88).
- Finally, there is some correlation between **visitNumber** and **totals.transactionRevenue** (0.31).

6.2 Question 2

From where (geographic) are visitors coming? Based on website behaviors, are certain places more important to us than others?

```
'''
    INSY 5377-002
    Team 01
    Question 2
'''
def horizontal_bar_chart(cnt_srs, color):
    trace = go.Bar(
        y=cnt_srs.index[::-1],
        x=cnt_srs.values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace

# Continent
cnt_srs = training_df.groupby('geoNetwork.continent')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace1 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace2 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace3 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')
```

In addition to geoNetwork.continent, the above code was also applied for the following independent variables: geoNetwork.subContinent, geoNetwork.country, geoNetwork.region, geoNetwork.metro, geoNetwork.city, and geoNetwork.networkDomain.

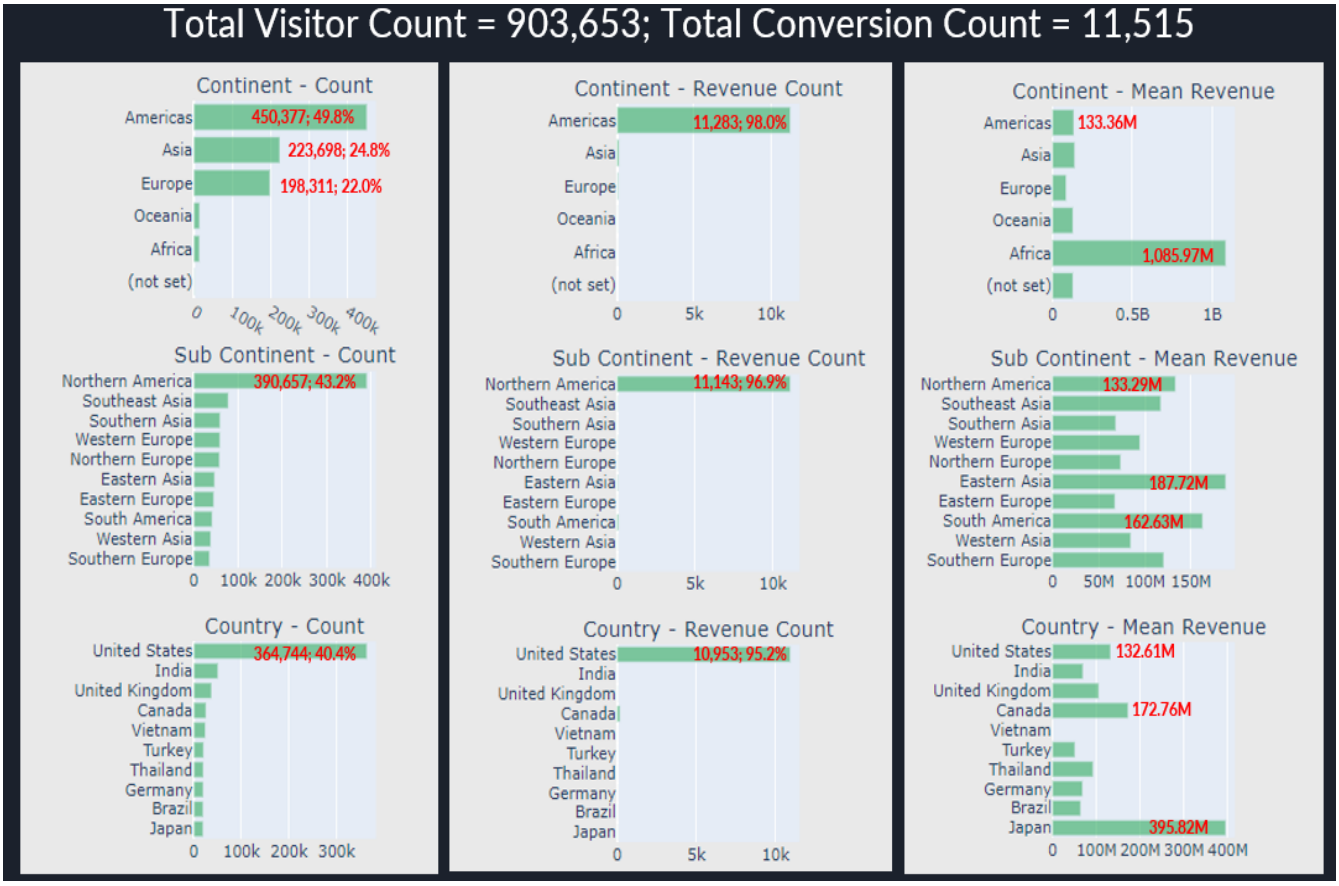

Figure 2.1: High-Level Demographics

Continent - From Figure 2.1 above, it appears that the Americas provide the most visitors (49.8%) for the GStore, followed by Asia (24.8%) and Europe (22.0%). However, it is also clear that the Americas account for the majority of the visitors that actually convert (98.0%). Interestingly though, when you look at the mean revenue on a continent basis, Africa really jumps out. This may indicate that albeit the number of visitors is really small compared to the other continents, those visitors that do convert from the continent of Africa spent more on average than their counterparts in the Americas, Asia, and Europe.

Sub Continent - Drilling down further, it is clear that most visitors are located in Northern America (43.2%) which also accounts for the greatest conversion count (96.9%). However, on a

mean revenue basis, it appears that Eastern Asia and South America outperforms Northern America.

Country - Continuing to drill down, it becomes clear that most visitors call the United States home and account consequently for 95.2% of the total conversion count. Still, Japan and Canada outperform the United States on a mean revenue basis.

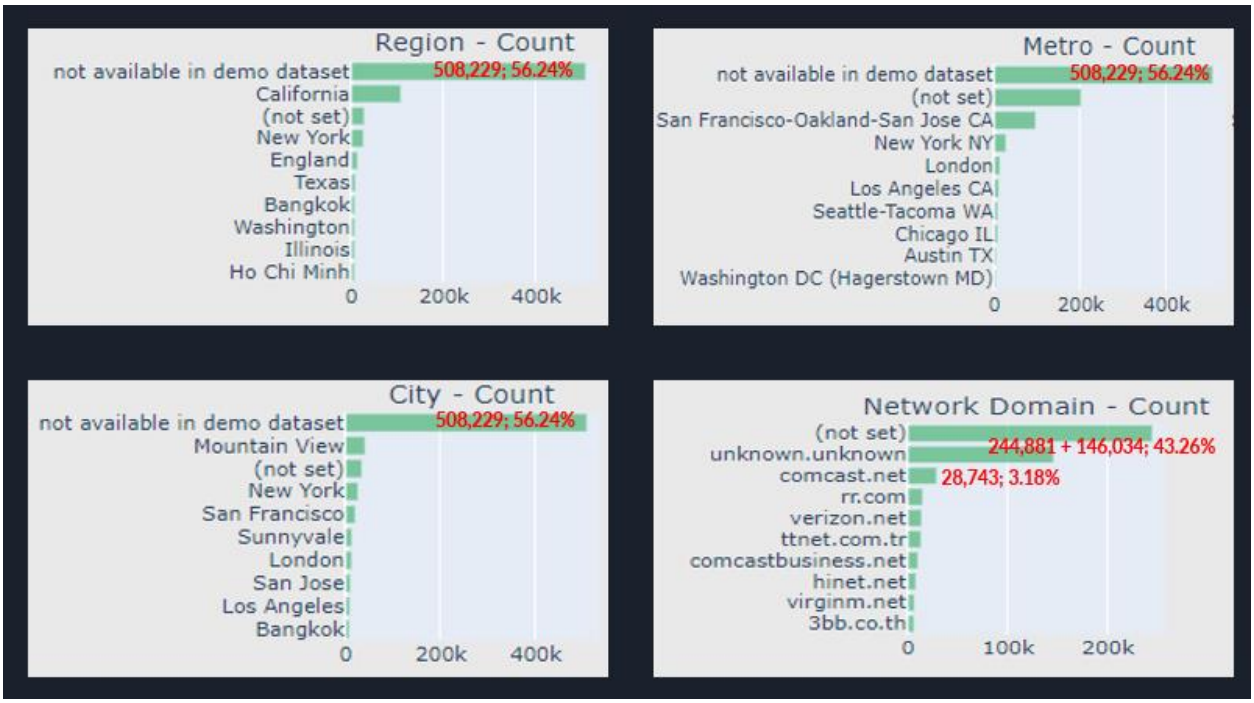

Figure 2.2: Low-Level Demographics

Region/Metro/City/Network Domain - From Figure 2.2 above, the majority of the data is unavailable in the dataset (56.24% for Region/Metro/City) and not set/unknown for Network Domain (43.26%). This might be an indication of visitors using VPNs (Virtual Private Networks) that hide their IP addresses. On a Regional/Metro/City basis, it looks like California brings in the most visitors. While on a Network Domain basis, Comcast.net seems to be the front runner with 3.18% of all visitors. Perhaps it would behoove GStore to consider a partnership with Comcast.

6.3 Question 3

What devices, browser, and operating system do visitors prefer?

```
# Device Browser
cnt_srs = training_df.groupby('device.browser')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace1 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace2 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace3 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')

# Device Category
cnt_srs = training_df.groupby('device.deviceCategory')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace4 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace5 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace6 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')

# Operating system
cnt_srs = training_df.groupby('device.operatingSystem')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace7 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace8 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace9 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')
```

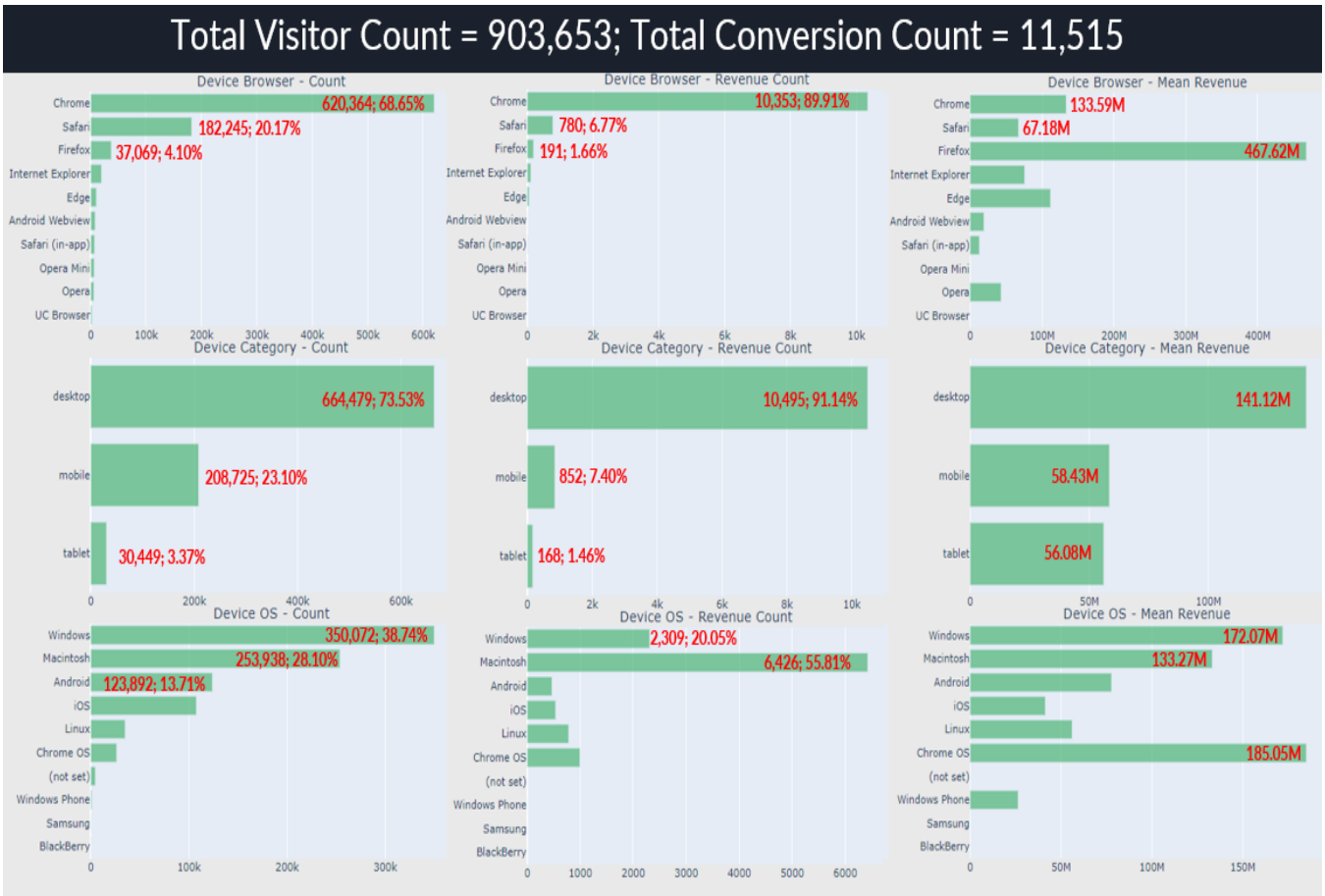

Figure 3: Device Plots

Device Browser - From Figure 3, it appears that most visitors are utilizing Chrome (68.65%), followed by Safari (20.17%) and Firefox (4.10%). In line with the visitor count, Chrome is the device browser with the most converted visitors at 89.91%. However, albeit Firefox only has 1.66% of the total conversion count, it has the highest mean revenue with $467.62M indicating that people that use Firefox are spending on average more at the GStore than those that use any other device browser.

Device Category - Likewise, it looks like most visitors make use of a desktop (73.53%) to visit the store. This translates in it also having the highest conversion count and mean revenue at 91.14% and $141.12M respectively. Unexpectedly, albeit both having the lowest visitor count and conversion count, the average revenue spent by tablet users is almost identical to those that use a mobile device, at $56.08M and $58.43M correspondingly. Hence, it may be beneficial for the GStore to allocate campaign funds towards the tablet segment.

Device Operating System - Similarly, it looks as if most visitors are using either Windows (38.74%), Macintosh (28.10%), or Android (13.71%). Although the biggest percentage of visitors are Windows users, a bigger share of Macintosh users are converting (55.81%). As both Windows and Macintosh users have the highest conversion count, Chrome OS users appear to have the leading mean revenue of the operating system group.

6.4 Question 4

6.4.1 Hits & Pageviews

What are the number of total hits & pageviews?

```
# Hits
cnt_srs = training_df.groupby('totals.hits')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", 'mean']
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace1 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace2 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(60), 'rgba(150, 71, 80, 0.6)')
trace3 = horizontal_bar_chart(cnt_srs["mean"].head(60), 'rgba(150, 71, 80, 0.6)')

# Page views
cnt_srs = training_df.groupby('totals.pageviews')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace4 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace5 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(60), 'rgba(150, 71, 80, 0.6)')
trace6 = horizontal_bar_chart(cnt_srs["mean"].head(60), 'rgba(150, 71, 80, 0.6)')
```

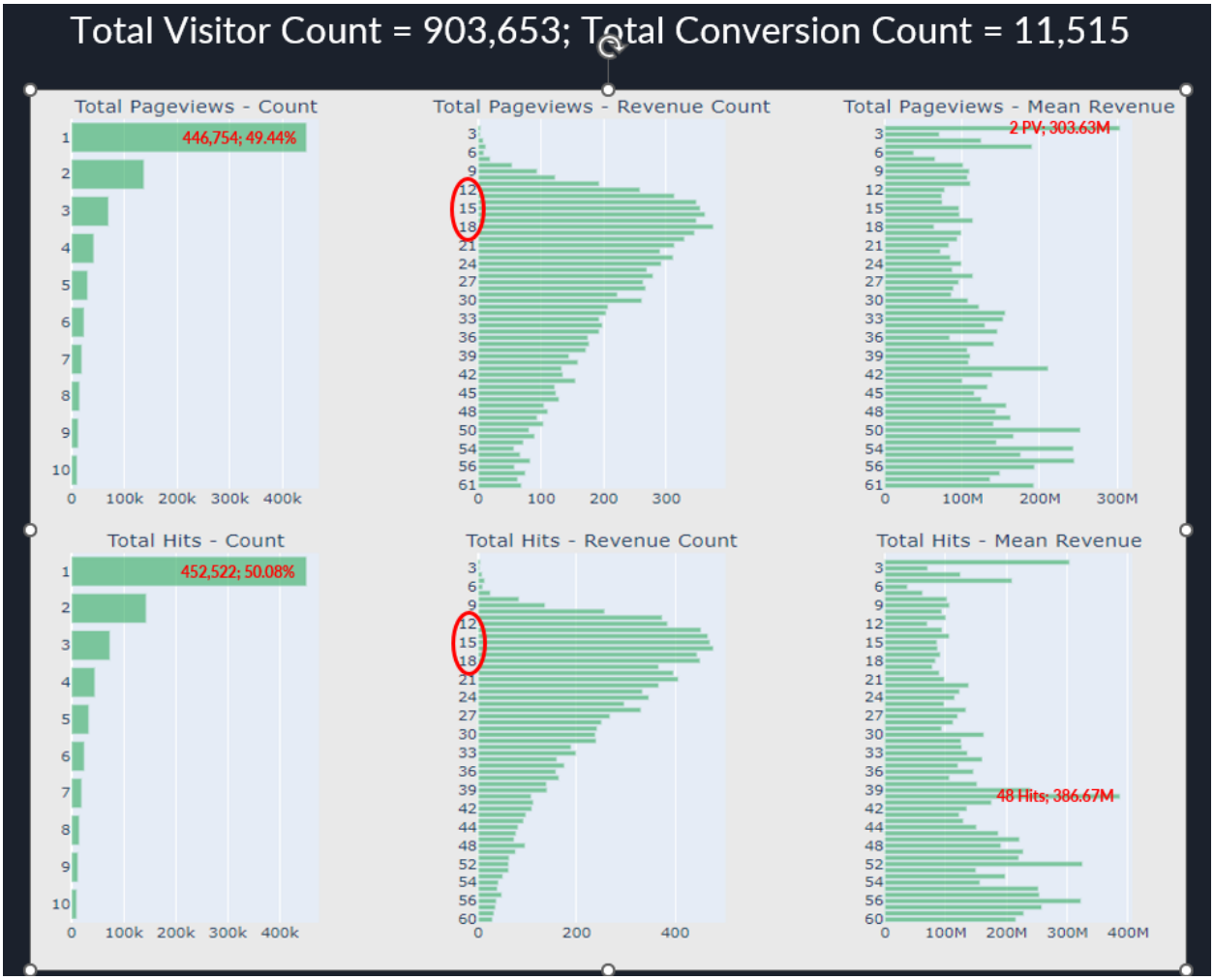

Figure 4.1: Visitor Profile Plots

Pageviews - It seems that a majority of visitors of the GStore only view one page (49.44%) indicating a high bounce rate as the entry page usually doesn't serve as a desired exit page such as the shopping cart/checkout page. This is further confirmed by the biggest conversion count occurring for visitors in the 12-18 range as well as the highest average revenue for a visitor appearing at 2 pageviews ($303.63M) - one entry page and one exit/checkout page.

Hits - Like with pageviews, a large portion of visitors bounce (50.08%) while the biggest conversion count appears in the 12-18 range. However, unlike with pageviews, the highest mean revenue for a visitor happens around 48 hits ($386.67M). This may be an indication that the entry page may be resource intensive, slowing down the loading speed of the page and thereby frustrating potential customers, which explains the high bounce rates. This may suggest for GStore to consider reconfiguring or optimizing their entry page to be resource light and more user-friendly.

6.4.2 Conversion Rate

What is the conversion rate?

```
'''
    INSY 5377-002
    Team 01
    Question 4B
'''
training_df["totals.transactionRevenue"] = training_df["totals.transactionRevenue"].astype('float')
print("Number of unique visitors in training set : ",training_df.fullVisitorId.nunique(),
      " out of rows : ",training_df.shape[0])
print()
group_per_visit = training_df.groupby("fullVisitorId")["totals.transactionRevenue"].sum().reset_index()

plt.figure(figsize=(8,6))
plt.scatter(range(group_per_visit.shape[0]), np.sort(np.log1p(group_per_visit["totals.transactionRevenue"].values)))
plt.title("Distribution of Visits Contributing to transactionRevenue")
plt.xlabel('index', fontsize=12)
plt.ylabel('TransactionRevenue', fontsize=12)
plt.show()
```

```
Number of unique visitors in training set :  714167  out of rows :  903653
```

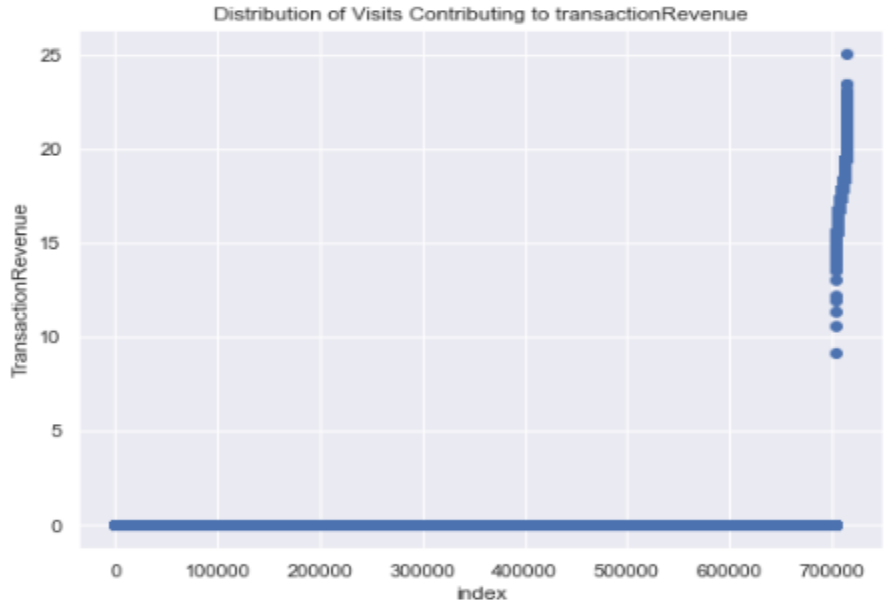

Figure 4.2: Distribution of Visits Contributing to Transaction Revenue

Figure 4.2 shows that only a very small percentage of visitors are responsible for all of the transaction revenue for GStore.

```
'''
    INSY 5377-002
    Team 01
    Question 4C
'''

non_zero_inst = pd.notnull(training_df["totals.transactionRevenue"]).sum()
non_zero_rate = (group_per_visit["totals.transactionRevenue"]>0).sum()
print("Number of non-zero revenue instances in training set: {:,}".format (non_zero_inst),
      " and conversion rate is : {:.4f}".format (non_zero_inst / training_df.shape[0]))
print("Number of non-zero revenue unique customers in training set: {:,}".format (non_zero_rate),
      "and conversion rate is : {:.4f}".format (non_zero_rate / group_per_visit.shape[0]))
```

```
Number of non-zero revenue instances in training set: 11,515  and conversion rate is : 0.0127
Number of non-zero revenue unique customers in training set: 9,996 and conversion rate is : 0.0140
```

6.5 Question 5

What is the likelihood a visitor will see our website just once? What percentage of visitors return?

```
'''
    INSY 5377-002
    Team 01
    Question 5
'''

def horizontal_bar_chart(cnt_srs, color):
    trace = go.Bar(
        y=cnt_srs.index[::-1],
        x=cnt_srs.values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace

# Number of visits
cnt_srs = training_df.groupby('visitNumber')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace1 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace2 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace3 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')

# Creating a subplot
fig = subplots.make_subplots(rows=1, cols=3, vertical_spacing=0.01, horizontal_spacing=0.10,
                          subplot_titles=["Visit Number - Count", "Visit Number - Revenue Count",
                                          "Visit Number - Mean Revenue"])

fig.append_trace(trace1, 1, 1)
fig.append_trace(trace2, 1, 2)
fig.append_trace(trace3, 1, 3)

fig['layout'].update(height=500, width=1000, paper_bgcolor='rgb(233,233,233)', title="Traffic Source Plots")
py.iplot(fig, filename='traffic-source-plots')
```

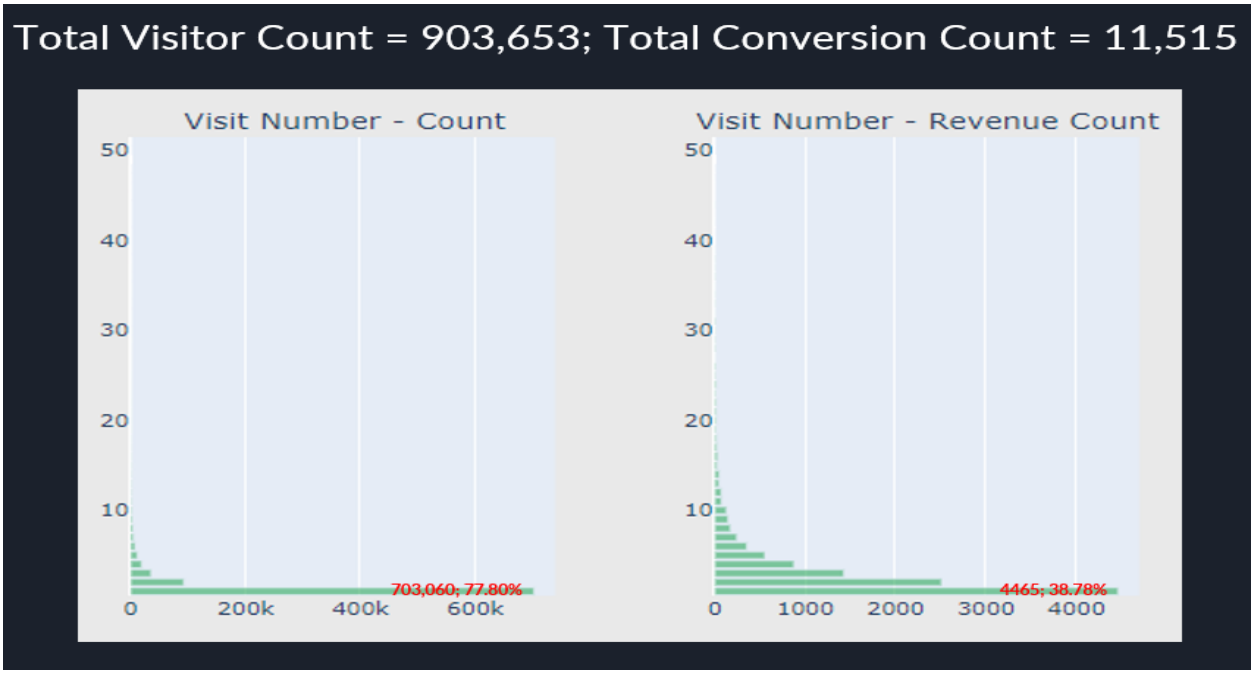

Figure 5.1: Visit/Revenue Count Plots

The above Figure 5.1 shows that the GStore has an active acquisition campaign to attract new customers (growth mode). This is no surprise as the GStore was established in March 2015, and our dataset ranges from Q3, 2016 through Q2, 2017. The marketing program looks to be successful as 38.78% of the conversion count appears to be coming from this segment.

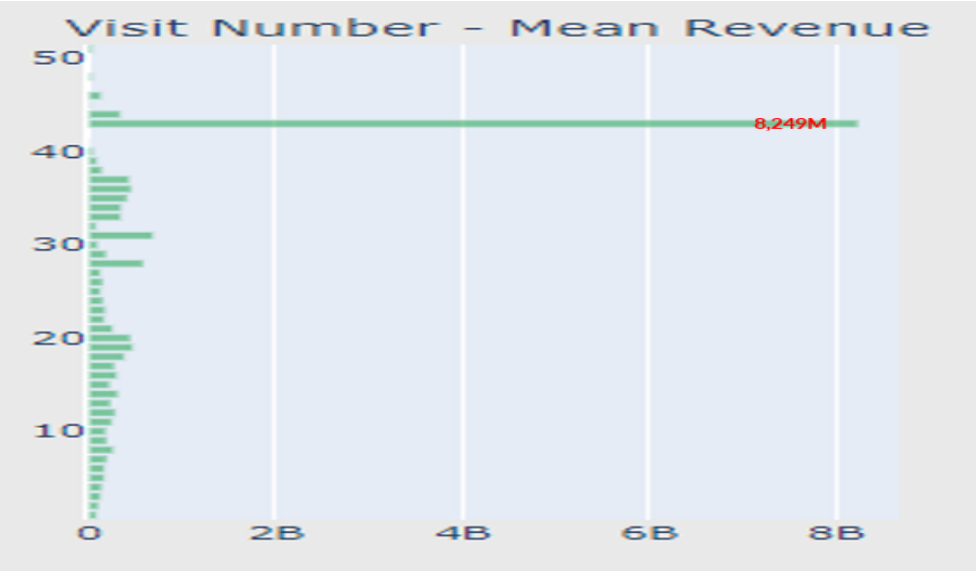

Figure 5.2: Visitor Mean Revenue Plot

However, Figure 5.2 shows that many of the higher average revenue numbers occur after multiple visits to the website, i.e. returning customers. Therefore, the GStore may want to consider re-balancing their acquisition/retention marketing programs to start attracting more of the returning customers who seem to have a higher average order value (AOV). The anomaly of $8,249M at 43 visits might be explained by one or multiple customers visiting the site multiple times to assure themselves before making a large purchase.

6.6 Question 6

Does one traffic source category dominate disproportionally?

```
# Traffic Source - Source
cnt_srs = training_df.groupby('trafficSource.source')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace4 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace5 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace6 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')
```

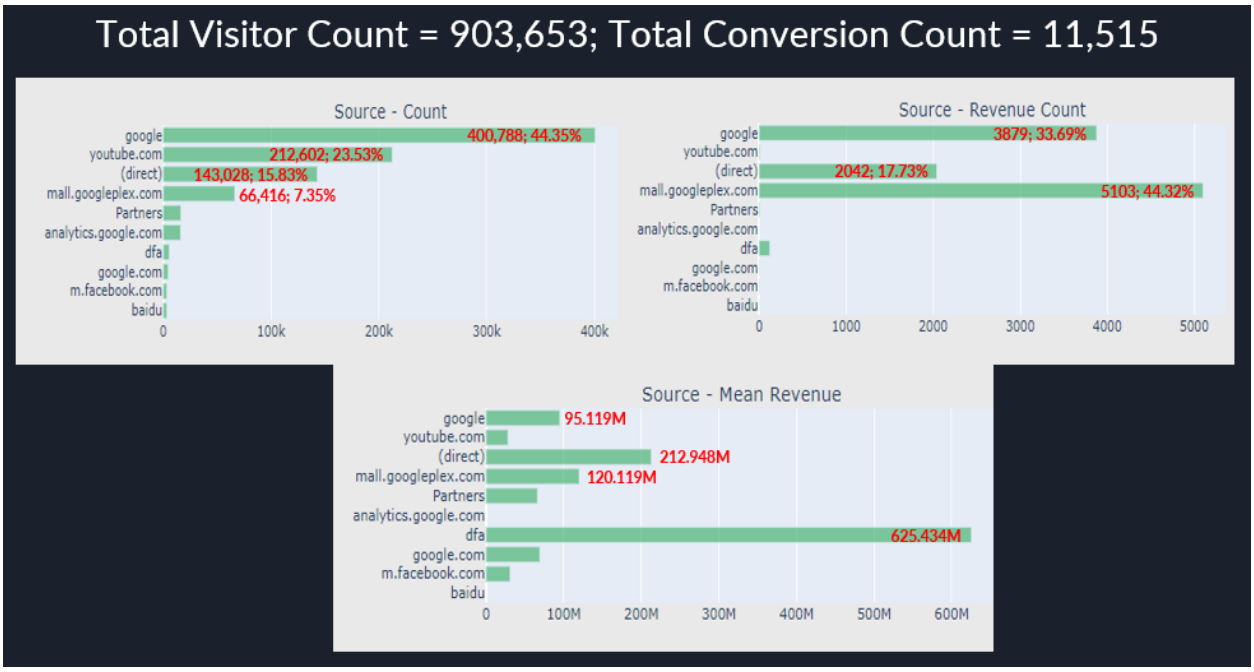

Figure 6: Traffic Source Plots

From the above Figure 6, it appears that a majority of the visitors are directed to the store through Google (google - 44.35% & mall.googleplex.com - 7.35%). This should not be a surprise as the GStore is affiliated to Google. This is confirmed as the revenue count shows a similar pattern (google - 33.69% & mall.googleplex.com - 44.32%). Besides the abundance of normally referred traffic, the output also has a fair amount of directly referred traffic (15.83%). This segment accounts for 17.73% of the conversion count. As for the mean revenue output, it seems that both the Google Marketing Program (DFA - Doubleclick for Advertisers) and the directly referred traffic are leading the pack. Particularly of interest is the directly referred traffic ($212.95M) as it shows the effective offline marketing of the GStore. Directly referred traffic

refers to people who type in their URL directly into a browser – this can be done manually for first time visitors or through a saved bookmark shortcut for repeat customers. Usually, these types of visitors are attracted through billboards, newspaper ads, and TV ads.

6.7 Question 7

What is the trend for Organic Search?

```
# Traffic Source - Medium
cnt_srs = training_df.groupby('trafficSource.medium')['totals.transactionRevenue'].agg(['size', 'count', 'mean'])
cnt_srs.columns = ["count", "count of non-zero revenue", "mean"]
cnt_srs = cnt_srs.sort_values(by="count", ascending=False)
trace7 = horizontal_bar_chart(cnt_srs["count"].head(10), 'rgba(150, 71, 80, 0.6)')
trace8 = horizontal_bar_chart(cnt_srs["count of non-zero revenue"].head(10), 'rgba(150, 71, 80, 0.6)')
trace9 = horizontal_bar_chart(cnt_srs["mean"].head(10), 'rgba(150, 71, 80, 0.6)')
```

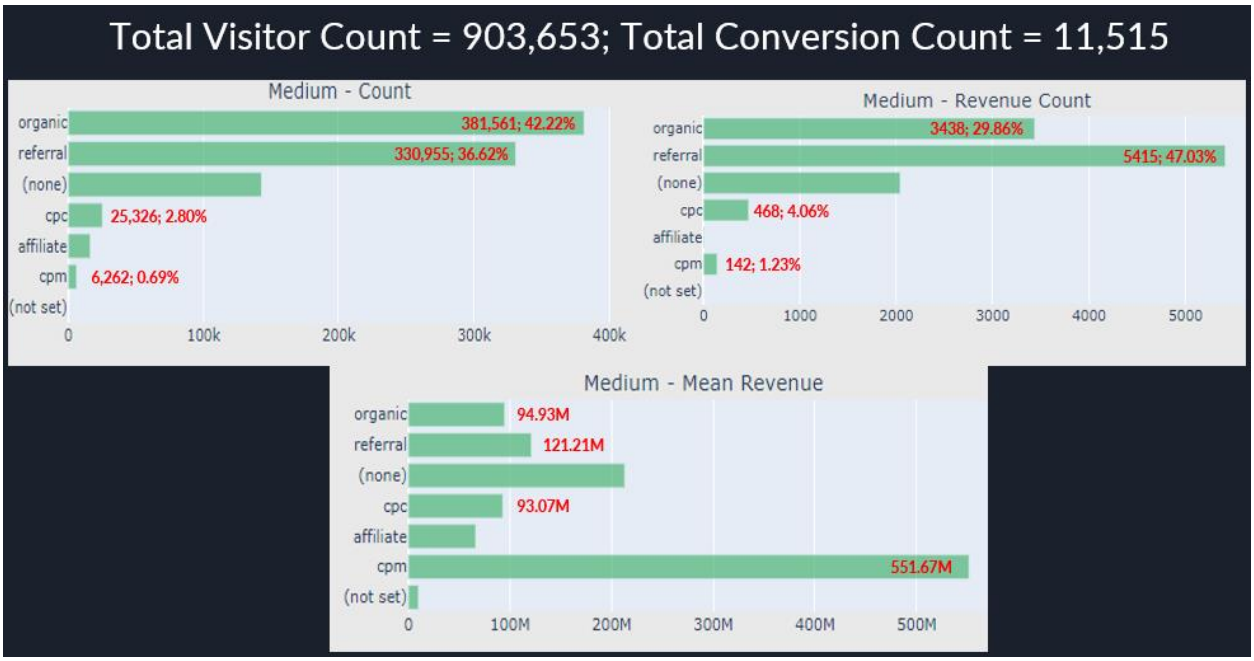

Figure 7: Traffic Medium Charts

Figure 7 shows that a majority of the visitors come to the GStore website organically (42.22%) or through referral (36.62%). From these visitors, referrals account for the largest share of conversion count (47.03%) confirming the strength of the GStore marketing campaigns. However, GStore should focus on improving their entry page to match their marketing messages,

since as indicated before in **Section 6.4.1**, the bounce rate is in excess of 50%. Cost-per-thousand impressions (CPM) appear to bring in those customers with the highest AOV ($551.67M).

6.8 Question 8

What is the aggregated revenue generated from each user based on our independent variables?

```
'''
    INSY 5377-002
    Team 01
    Question 8A
'''
#Country
rev_per_country = training_df.loc[:,['geoNetwork.country','totals.transactionRevenue']]
rev_per_country = rev_per_country.rename(columns={'geoNetwork.country':'Country','totals.transactionRevenue':'Total Revenue'})
rev_per_country = rev_per_country.groupby(['Country']).sum('Total Revenue')
revenue_per_country = rev_per_country.sort_values(by=['Total Revenue'],ascending=False).reset_index()
revenue_per_country.head()
```

The above code was applied for most of the independent variables resulting in the below tables.

	Continent	Total Revenue
0	Americas	1.504672e+12
1	Asia	1.740184e+10
2	Africa	8.687760e+09
3	Europe	6.747030e+09
4	Oceania	1.793230e+09

	SubContinent	Total Revenue
0	Northern America	1.485265e+12
1	South America	1.593731e+10
2	Eastern Asia	1.107522e+10
3	Eastern Africa	5.268700e+09
4	Southeast Asia	3.755990e+09

	Country	Total Revenue
0	United States	1.452441e+12
1	Canada	3.282454e+10
2	Venezuela	1.337490e+10
3	Japan	6.728990e+09
4	Kenya	5.268700e+09

Table 8.1: High-Level Demographic Based Revenue

The above Table 8.1 displays that the biggest part of the revenue based on the GStore demographics is located in the Americas, followed by Asia and Africa. The same conclusion can be drawn when drilling down further to the sub-continent and country levels. As can be seen in Figure 2.1, very few visitors/customers come from the continent of Africa. However, Table 8.1 shows that a fair share of the revenue comes from Africa, more specifically from Kenya.

	City	Total Revenue
0	not available in demo dataset	6.461611e+11
1	New York	2.201435e+11
2	Mountain View	1.261037e+11
3	San Francisco	1.008740e+11
4	Chicago	7.285453e+10

	Network Domain	Total Revenue
0	(not set)	8.482641e+11
1	comcastbusiness.net	1.461464e+11
2	comcast.net	1.231696e+11
3	verizon.net	5.291897e+10
4	unknown.unknown	3.490154e+10

Table 8.2: Low-Level Demographic Based Revenue

As previously discussed in **Question #2** and can also be observed from the Table 8.2 above, the dataset shows limited data when drilling down below country level. As that may be, it appears that most revenue comes from California, Illinois, or New York. In addition, it shows that both customers using Comcast and Verizon bring in the majority of the GStore revenue.

	Browser	Total Revenue
0	Chrome	1.383105e+12
1	Firefox	8.931632e+10
2	Safari	5.239748e+10
3	Internet Explorer	8.255550e+09
4	Edge	6.482970e+09

	Device	Total Revenue
0	desktop	1.480864e+12
1	mobile	4.978581e+10
2	tablet	9.421340e+09

	Operating System	Total Revenue
0	Macintosh	8.563758e+11
1	Windows	3.973127e+11
2	Chrome OS	1.839378e+11
3	Linux	4.391040e+10
4	Android	3.634128e+10

Table 8.3: Device Revenue Table

The above Table 8.3 reveals that most of the GStore revenue comes from customers using a desktop. However, as mentioned in the answer to **Question #3**, it may be beneficial for the GStore to reconsider their current marketing mix such that more of it is directed towards tablets, as its mean revenue is approximate to that of mobile users. As for the operating system, it looks like Macintosh brings in the most revenue. Yet, redirecting some of the marketing funds towards Chrome OS, which has the top average revenue, may potentially push the total revenue of this operating system more towards the top. Regarding the browser, it seems that Chrome brings in the most revenue. But, unexpectedly, Firefox secures the second spot even though Safari has the

second highest visitor count and visitor conversion count. Yet, this can be explained by the average revenue of Firefox which far outperforms that of Safari (see Figure 3).

	Pageviews	Total Revenue
0	22	5.056812e+10
1	14	4.953587e+10
2	26	4.414558e+10
3	13	4.263538e+10
4	40	4.176049e+10

	Number of Visits	Total Revenue
0	1	3.823260e+11
1	2	2.731557e+11
2	3	1.948948e+11
3	4	1.299212e+11
4	5	9.670241e+10

Table 8.4: Pageviews/Visits Revenue Table

Referring to the high bounce rate issue (see Figure 4), the above Table 8.4 suggests that GStore seems to have an ineffective path for converting their visitors. As a remedy, they could investigate ways to convert visitors sooner hereby conserving server overhead and costs. Also, based on the above Table 8.4, the GStore receives most of its revenue from new customers. However, as we observed in Figure 5.2, the average revenue tends to be higher for returning customers. Though the GStore is in a current acquisition mode, it would be beneficial to turn these better mean revenues into higher total revenues by switching some of their focus towards retention instead of acquisition.

	Source	Total Revenue
0	mall.googleplex.com	6.129680e+11
1	(direct)	4.348406e+11
2	google	3.689654e+11
3	dfa	7.692842e+10
4	mail.google.com	2.332791e+10

	Medium	Total Revenue
0	referral	6.563464e+11
1	(none)	4.348406e+11
2	organic	3.263805e+11
3	cpm	7.833746e+10
4	cpc	4.355890e+10

Table 8.5: Source Revenue Table

From the above Table 8.5, it is clear that most of the visitors bringing in revenue come from a source associated with Google (4 out of 5). As previously mentioned in **Question #6**, it was observed that the GStore has an effective offline marketing program (direct). However, this

should not come as a surprise since direct visitors make up 15.83% of all visitors, accounting for 17.73% of visitor conversion count, and have the 2nd largest average revenue amongst sources. Table 8.5 above also portrays the effectiveness of the GStore marketing campaign as the largest portion of total revenue comes through referral. This is supported by Figure 7, which shows that 47.03% of visitor conversion count is associated with referral. Interestingly, although Figure 7 indicates that only 0.69% of all visitors and 1.23% of visitor conversion count are from CPM, the above Table 8.5 shows its significance in the total revenue of the GStore.

6.9 Question 9

Which features most contributed to the conversion of visitors? Assuming circumstances in the future are similar, which features would most benefit GStore to focus on? How can GStore use this information to improve?

For our research, we built 3 predictive models in Python.

- Initially, we utilized the Light Gradient Boosting Machine (LGBM), as it is applicable towards very large datasets.
- Next, our attempts with a Logistic Regression resulted in an overfitted model, due to the complex nature of the dataset. To reduce overfitting, we tried oversampling and undersampling, which did not have the desired results. Next, we performed scaling with Standard Scaler, which finally gave us the desired results with more realistic accuracy scores.
- In addition, we ran a Random Forest Classifier predictive model to further improve upon the results obtained from the previous model. Even after fine tuning with Grid Search and

Stratified K-Fold Cross Validation, the accuracy obtained from the Random Forest model was unsatisfactory.

- Finally, although getting positive results using Logistic Regression, we decided to utilize a Time Series analysis to understand and analyze the patterns behind the online traffic volume.

6.9.1 Light Gradient Boosting Machine (LGBM)

We employed a Light Gradient boosting Machine (LGBM) to conduct further analysis into feature importance. Given our dataset's complexity and ambitious scale, LGBM is especially appropriate for our situation. In addition, LGBM is notably "light" in terms of resource consumption.

```
# This will run the Light Gradient Boosting Machine (LGBM).
def run_lgb(train_X, train_y, valid_X, valid_y, test_X):
    params = {
        "objective" : "regression",
        "metric" : "rmse",
        "num_leaves" : 30,
        "min_child_samples" : 100,
        "learning_rate" : 0.1,
        "bagging_fraction" : 0.7,
        "feature_fraction" : 0.5,
        "bagging_frequency" : 5,
        "bagging_seed" : 2018,
        "verbosity" : -1
    }
    '''
    Regression means it's learning.

    RMSE means Root-Mean Squared Error. We would like to keep this at a minimum.

    num_leaves represents the number of leaves we want each iteration of the decision tree to grow out towards.

    The bagging_fraction represents the percentage of the sampling of the training set's rows for each iteration
    of a decision tree.

    The feature_fraction represents the percentage of sampling from the features (or columns).

    The bagging frequency means the rate at which data is randomly "bagged", or used to create subsets
    of the training sets. These subsets are used to train the decision trees.

    Bagging seed is a randomizer that determines the bagging behavior. Bagging means to combine predictions for trees
    generated in machine learning.
    '''

    lgbm_train = lgb.Dataset(train_X, label = train_y)
    lgbm_valid = lgb.Dataset(valid_X, label = valid_y)
    lgbm_model = lgb.train(params, lgbm_train, 1000, valid_sets = [lgbm_valid], early_stopping_rounds = 100,
                                                                                verbose_eval = 100)

    '''
    Here is where the model predicts using the testing and validation datasets. It will only use the best iteration, or
    in other words, where the RMSE (Root-Mean Squared Error) is smallest. Because it's a metric measuring error, we
    would want this to be smallest!
    '''
    test_y_hat = lgbm_model.predict(test_X, num_iteration= lgbm_model.best_iteration)
    valid_y_hat = lgbm_model.predict(valid_X, num_iteration= lgbm_model.best_iteration)
    return test_y_hat, lgbm_model, valid_y_hat

# Training the model.
pred_test, lgbm_model, pred_val = run_lgb(learning_X, learning_y, valid_X, valid_y, test_X)
```

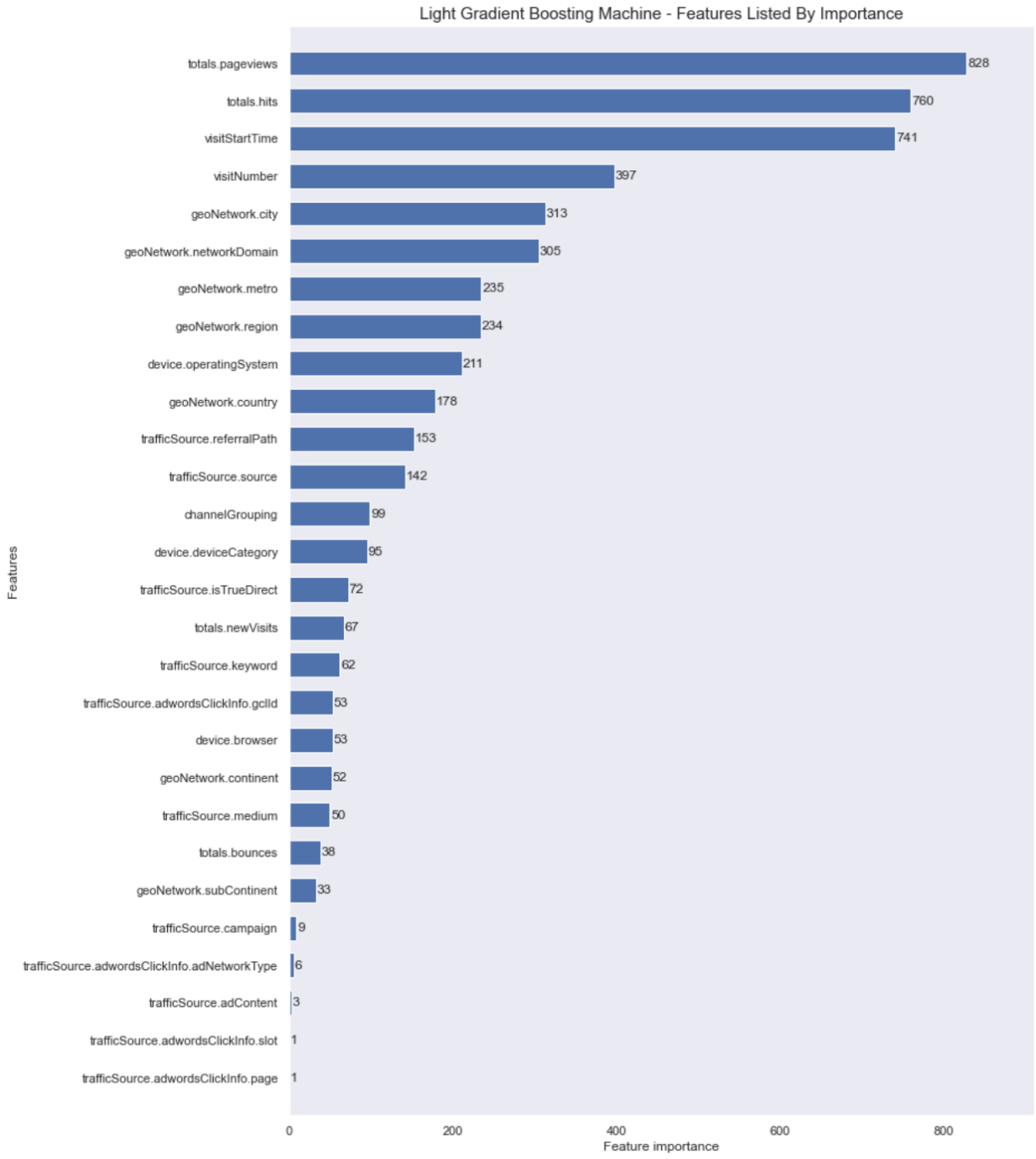

Figure 9.1: LGBM Feature Importance Chart

Among the features above in Figure 9.1, pageviews are quite high. We can infer this is natural for the GStore. After all, its dataset involves a shopping website where visitors are inherently browsing through multiple pages on the GStore site. Greater exposure to pages on the site can potentially lead to purchases while still effectively pushing visitors down the digital marketing

funnel. In addition, some visitors may be hitting the back or forward buttons on their browser. For this reason, any analysis should be based on visits or visitors instead of pageviews.

6.9.2 Time Series

The following Time Series displays the daily number of visits from Q3 2016 through Q2 2017.

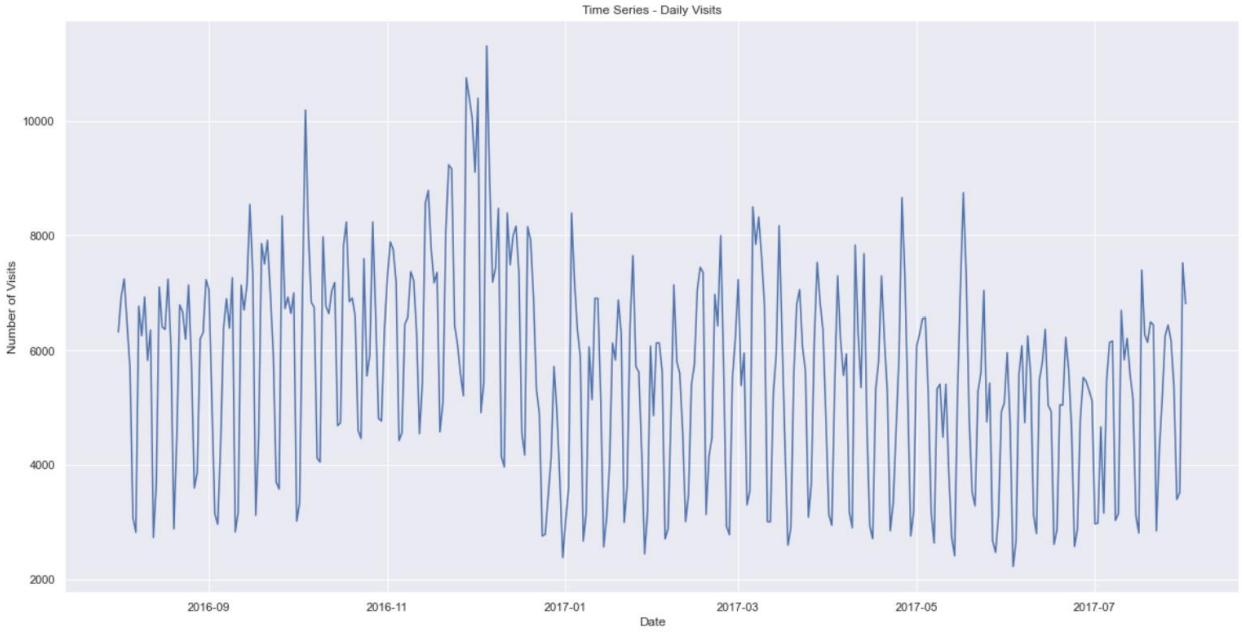

Figure 9.2: Time Series of Daily Visits

Numerous spikes in the number of visits indicate:

- Campaigns for GStore's website (i.e. Black Friday, Holiday sales).
- Google's unveiling of their new line of smartphones, Pixel 1 was announced and released during October 2016.

It can be observed that overall visits typically are lower outside of the last quarter of the year. To improve conversion prospects, GStore can take measures to increase overall build-up towards the end of the year.

- Looking at this Time Series, we observed that we could utilize June and July as a baseline for our Logistic Regression, seen later in **Section 6.9.3**.

6.9.3 Logistic Regression

Our initial attempts with Logistic Regression encountered problems of overfitting.

```
In-sample Accuracy Score:  98.51206392148228
Out-of-sample Accuracy Score:  98.55583046471601
```

For that reason, we investigated different ways to adjust our dataset. First, we contracted our dataset to make it less prone to seasonality and significant events. By observing our Time Series analysis in **Section 6.9.2**, the months of June and July of 2017 were selected, as the GStore witnessed a lower overall number of visitors during these months. Using this time-based data sample, we looked into various methods to mitigate for overfitting and class imbalances. Since the number of visitors who convert consist of a small portion of our dataset, we tried oversampling. This resulted in a more equal distribution of converters and non-converters.

```
logistic_regression_model = LogisticRegression(max_iter = 1000, class_weight = w)

smt = SMOTE(random_state = 0)
x_train_res, y_train_res = smt.fit_resample(x_train_df, y_train_df)
```

```
Prior to OverSampling, counts of label '1': 946
Prior to OverSampling, counts of label '0': 62632

After OverSampling, counts of label '1': 62632
After OverSampling, counts of label '0': 62632
```

```
y_train_hat = logistic_regression_model.predict(x_train_res)
y_test_hat = logistic_regression_model.predict(x_test_res)

# Displays the Accuracy scores for in-sample (training) and out-of-sample (testing) data
print("In-sample Accuracy Score: ", accuracy_score(y_train_res, y_train_hat, normalize = True) * 100)
print("Out-of-sample Accuracy Score: ", accuracy_score(y_test_res, y_test_hat, normalize = True) * 100)
```

```
In-sample Accuracy Score:  50.0
Out-of-sample Accuracy Score:  50.0
```

The resulting scores of 50% accuracy for both in-sample and out-of-sample are unacceptable. We re-examined our dataset and believe this is due to the inherent complexity of the data.

Because of this result, we looked into the optimal resampling rates for our dataset based on the best Receiver-Operating Characteristic (ROC) and Area Under Curve (AUC) score.

```
from collections import Counter
from imblearn.over_sampling import SMOTE
import pandas as pd
import numpy as np
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from sklearn.svm import SVC

from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from numpy import mean

x_logreg = pd.concat([x_train_df.copy(), x_test_df.copy()], sort = False)
y_logreg = np.concatenate([y_train_df.copy(), y_test_df.copy()])

# Parameters for oversampling and undersampling.
oversampling_params = [0.3, 0.4, 0.5]
undersampling_params = [0.7, 0.6, 0.5]

for o in oversampling_params:

    for u in undersampling_params:

        model = SVC()
        over = SMOTE(sampling_strategy = o)

        under = RandomUnderSampler(sampling_strategy = u)
        steps = [("over", over), ("under", under), ("model", model)]
        pipeline = Pipeline(steps=steps)

        scores = cross_val_score(pipeline, x_logreg, y_logreg, scoring = "roc_auc", cv = 5, n_jobs=-1)
        score = mean(scores)

        print("SMOTE oversampling rate:%.1f, Random undersampling rate:%.1f , Mean ROC AUC: %.3f" % (o, u, score))
```

```
SMOTE oversampling rate:0.3, Mean ROC AUC: 0.579
SMOTE oversampling rate:0.3, Mean ROC AUC: 0.594
SMOTE oversampling rate:0.3, Mean ROC AUC: 0.588
SMOTE oversampling rate:0.4, Mean ROC AUC: 0.563
SMOTE oversampling rate:0.4, Mean ROC AUC: 0.615
SMOTE oversampling rate:0.4, Mean ROC AUC: 0.572
SMOTE oversampling rate:0.5, Mean ROC AUC: 0.592
SMOTE oversampling rate:0.5, Mean ROC AUC: 0.599
SMOTE oversampling rate:0.5, Mean ROC AUC: 0.583
```

We evaluated various metrics that would be appropriate in oversampling our data. Using the mean scores of the ROC and AUC, we found the optimal resampling rates and mean score were:

- 0.4 for SMOTE oversampling.
- 0.615 for mean ROC AUC score.

For a reasonable ROC AUC score, it would be ideal to be relatively high (i.e. close to 0.9). From our observation, we find this did not provide sufficient insight.

As we mentioned before, the number of visitors who convert consist of a small portion of our dataset. Since oversampling did not work well, we used the **StandardScaler()** to standardize our data.

```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import precision_score, recall_score, precision_recall_curve, f1_score, fbeta_score, make_scorer, classi


std_scale = StandardScaler()
x_train_scaled = std_scale.fit_transform(x_train_df)
x_test_scaled = std_scale.transform(x_test_df)

logistic_regression_model = LogisticRegression(max_iter = 1000, class_weight = w)

%time logistic_regression_model.fit(x_train_scaled, y_train_df)
y_test_hat = logistic_regression_model.predict(x_test_scaled)
y_train_hat = logistic_regression_model.predict(x_train_scaled)


print('Accuracy = ', logistic_regression_model.score(x_test_scaled, y_test_df).round(2),
          'Precision = ', precision_score(y_test_df, y_test_hat).round(2),
          'Recall = ', recall_score(y_test_df, y_test_hat).round(2),
          'F1_score = ', f1_score(y_test_df, y_test_hat).round(2)
         )

# Displays the accuracy scores for the June (training) and the July (testing) datasets.

print("In-Sample Accuracy Score: ", accuracy_score(y_train_df, y_train_hat, normalize = True) * 100)

print("Out-Of-Sample Accuracy Score: ", accuracy_score(y_test_df, y_test_hat, normalize = True) * 100)
```

```
Wall time: 661 ms
Accuracy =  0.94 Precision =  0.19 Recall =  0.98 F1_score =  0.32
In-Sample Accuracy Score:  94.00893390795558
Out-Of-Sample Accuracy Score:  93.97321428571429
```

The resulting accuracy scores around 94% showed the Logistic Regression model using Standard Scaler to be a better fit for our analysis.

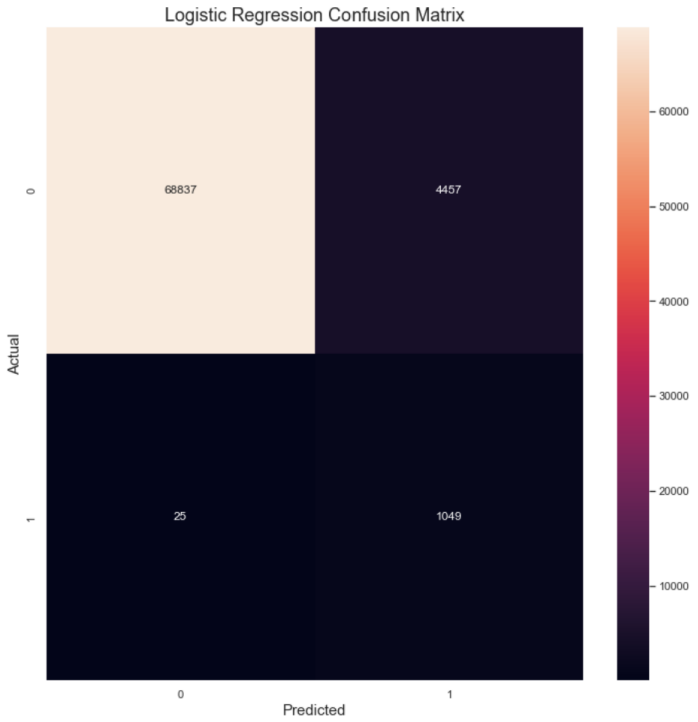

Figure 9.3: Logistic Regression Confusion Matrix

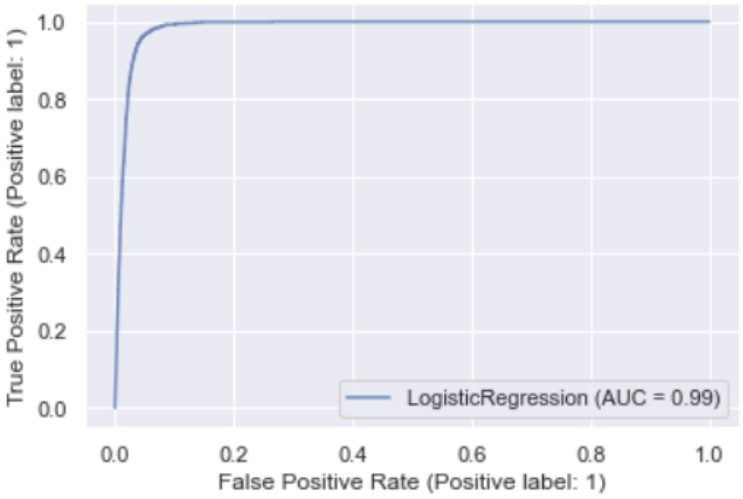

Figure 9.4: Logistic Regression AUC Graph

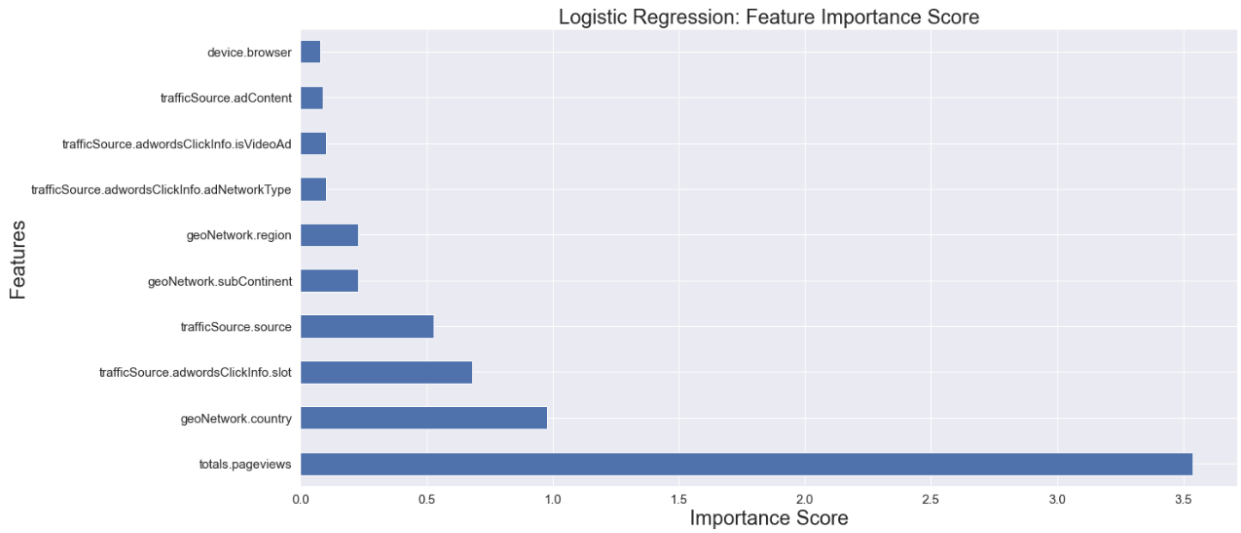

Figure 9.5: Logistic Regression Feature Importance

The above Figure 9.5 illustrates which features most contributed towards the conversion of visitors for the months of June and July. Though there are similarities to Figure 9.1 in **Section 6.9.1** in that pageviews are indicated as the highest, some of the geonetwork demographics are among the highest. Perhaps since June and July are the months during which people vacation for many countries, visitors are traveling abroad and viewing the GStore on their personal devices. Potentially, it could improve GStore's presence by looking into penetration marketing opportunities in places where there is room for growth to capitalize on. If considered, GStore should take further action to improve their reach and acquisition programs during these months, where turnout is generally lower.

6.9.4 Random Forest Classifier

Random Forest Classifier models are powerful predictors, and we thought this would be suited to analyze our dataset. Our attempts resulted in an in-sample accuracy score of 98.76% and an out-of-sample accuracy score of 98.74%. These indicate that our Random Forest predictive model showed to be overfitting.

```
'''
    INSY 5377-002
    Team 01
    Question 9
'''
RandomForest_model = RandomForestClassifier() # By default, n_estimators = 100; Still testing for 100 trees.

# Performing Grid Search with Cross Validation to find the optimal value for maximum depth.
param_grid = {'max_depth': [4, 5, 6, 7], 'min_samples_leaf': [500, 1000, 1500], 'max_features': np.arange(5, 10)}

# Stratified 5-fold cross validation.
cv = StratifiedKFold(n_splits = 5, random_state = 0, shuffle=True)
grid = GridSearchCV(RandomForest_model, param_grid, cv = cv,
                    return_train_score = True, scoring = 'accuracy')

%time grid.fit(x_train, y_train)

# Reports the optimal value for max_depth and the average scores for the stratified 5-fold cross validation.
print("Best Parameter: {}".format(grid.best_params_))
print("Best Cross Validation Score: {}".format(grid.best_score_))
```

```
Wall time: 2h 50min 33s
Best Parameter: {'max_depth': 7, 'max_features': 9, 'min_samples_leaf': 500}
Best Cross Validation Score: 0.9875268157150486
```

Even after fine tuning our model with Stratified K-Fold Cross Validation and Grid Search, our Random Forest Classifier still appeared to be overfitting (Cross Validation Score of 98.75%).

Using the demographics of visitors as features, our model was attempting to predict a binary dependent variable indicating whether they would convert or not:

- "0" indicates the visitor did not convert.
- "1" indicates the visitor did convert.

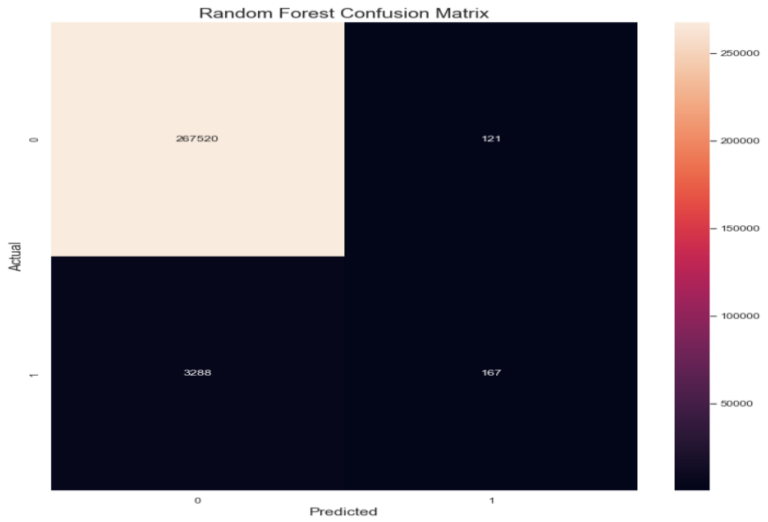

Figure 9.6: Random Forest Confusion Matrix

Our dataset contains a large class imbalance, where only a slim minority of visitors would convert. Therefore, the resulting confusion matrix in Figure 9.6 indicates that the Random Forest Classifier is ineffective in the prediction of visitor conversion prospects.

7. Conclusion

Our **descriptive analysis** of GStore's online metrics and customer behavioral patterns reveal a number of business insights we would like to recommend. The following suggestions outlined in the below paragraphs serve to improve GStore's online business prospects.

GStore mostly comes about their customers organically. However, the conversion count for referrals demonstrates the effectiveness of GStore's marketing initiatives (Figure 7) Additionally, given that the majority of the website traffic comes from sources connected to Google, it is clear that brand image is crucial in drawing visitors to the site (Figure 6). Direct traffic has the second highest average income ($212.95M) among all traffic sources, proving that Google also has an effective offline marketing strategy. As currently direct traffic is ranked third by visitor count, it would make sense for them to increase their campaign contributions towards this segment especially in areas of geographic interest as indicated by Figure 2.1. As for normally referred traffic, GStore should prioritize optimizing their website not only for Google Chrome and Safari users, but also for Firefox users as its mean revenue is the highest amongst these competing browsers. This is affirmed by Table 8.3 as Firefox is currently bringing in more revenue than Safari. Additionally, Google should look into site optimization for mobile users. The ubiquitous ability of mobile devices to access the web can increase GStore's accessibility for all users. Moreover, GStore may want to shift some of their focus towards acquiring tablet users, as their average revenue was similar to that of mobile users. Finally, Table 8.5 indicates that cost-per-

thousand impressions (CPM) surpasses cost-per-click (CPC) in total revenue albeit it only bringing in ¼ of CPC's visitors. Perhaps the GStore doesn't encourage impulsive buys and therefore would benefit from shifting some campaign funds towards CPM which promotes impressions that convert into sales afterwards.

As indicated by Figure 1, there is some correlation between **visitNumber** and **totals.transactionRevenue**. Nevertheless, the first step in converting these visitors is to make sure that the campaign message matches the entry page message. This might be an issue as GStore experiences an approximate bounce rate of 50 percent (Figure 4.1). Perhaps, they could re-evaluate their entry page to bring this number down and push visitors further down the digital marketing funnel. The current conversion rate of 1.27% obtained from the output of **Question 4** appears to be strongly influenced by new customers (Figure 5.1). The GStore was relatively new at the time of our dataset, and hence the ratio of new versus returning customers is not unusual. However, if you look at the average revenue per time visited, the number is higher for return visitors. Hence, the GStore may benefit from shifting their marketing mix more towards retention to increase their overall number of customers.

For our **predictive analysis**, Light Gradient Boosting Machine (LGBM) and Logistic Regression with Standard Scaler provided the best fit. However, it is our group's recommendation to make use of LGBM on larger datasets while utilizing Logistic Regression for more concentrated series (1 – 2 months). The Logistic Regression model was computed for out-of-seasons months and/or non-events to establish a baseline. It would be fair to assume that hosting additional campaigns throughout the year to maintain the sales momentum outside the Holiday season and special promotional periods would extend GStore's year-around reach.

GStore's direct affiliation with Google naturally attracts visitors. Utilizing this to the business' advantage while continuing targeted campaigns and improvements to the site can expand their awareness and improve conversion prospects.

8. References

Google Merchandise Store. [Logo of Google Merchandise Store] [Online image]. (n.d.). Retrieved July 14, 2022, from https://shop.googlemerchandisestore.com/

LightGBM. (n.d.). *LightGBM*. https://lightgbm.readthedocs.io/en/latest/Python-Intro.html

Masarat, S. (2018). *Tutorial (Preprocessing, Processing, Evaluation).* Kaggle. https://www.kaggle.com/code/smasar/tutorial-preprocessing-processing-evaluation

RStudio. (2018, November 23). *Google Analytics Customer Revenue Prediction*. Kaggle. https://www.kaggle.com/competitions/ga-customer-revenue-prediction/overview

SRK. (2018). *Simple Exploration+Baseline - GA Customer Revenue*. Kaggle. https://www.kaggle.com/code/sudalairajkumar/simple-exploration-baseline-ga-customer-revenue

Tan, J. (2020, November 11). *How to deal with imbalanced data in Python.* Towards Data Science. https://towardsdatascience.com/how-to-deal-with-imbalanced-data-in-python-f9b71aba53eb