

Subject Code : BUAN 6356.503 – Business Analytics with R – Project Report

A Project Report on

**GOOGLE ANALYTICS CUSTOMER REVENUE
PREDICTION**
(A Kaggle competition)
Predict how much GStore Customers will spend

Authored By:

Vikas Srikanth - VXS180022@utdallas.edu

TABLE OF CONTENTS

PART A – GROUP/JOINT REPORT

1.	Acknowledgement	3
2.	Introduction	4
3.	Brief Exploratory Data Analysis (EDA)	8
4.	Algorithm	24
5.	Results	28
6.	Recommendations	33
7.	References	34

PART B – INDIVIDUAL REPORT **35**

1.	Abstract	36
2.	Experience of Working on the Project	36
3.	Successes	36
4.	Challenges	37
5.	Recommendations	37

PART C – R CODE **38**

ACKNOWLEDGEMENT

It has been a great opportunity to gain a lot of experience in real time projects, followed by the knowledge of how to actually design and analyse them. For that we would like to thank our professor, **Dr. Sourav Chatterjee**, who made it possible. We would like to acknowledge him for his efforts in providing us with useful information and for making the path clear.

INTRODUCTION

Background

The 80/20 rule has proven true for many businesses—only a small percentage of customers produce most of the revenue. As such, marketing teams are challenged to make appropriate investments in promotional strategies. The 80/20 rule, also known as the [Pareto Principle](#), is attributed to the Italian economist, Vilfredo Pareto. In one of his papers, Pareto noted that about 80% of the land in Italy belonged to approximately 20% of the country's total population. In essence, the Pareto Principle infers that there's an 80-to-20 relationship between effects and their causes. The **80/20 rule** is a **principle** that holds true in many areas and is: Your 80% most significant results come from 20% of your actions. The **rule** is often used in companies to see the 20% of people that make 80% of the difference; this helps determine layoffs if and when they need to happen.

Objective

RStudio, the developer of free and open tools for R and enterprise-ready products for teams to scale and share work, has partnered with Google Cloud and Kaggle to demonstrate the business impact that thorough data analysis can have. Here, we were challenged to analyse a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict revenue per customer. Hopefully, the outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of GA data. We are tasked with analysing the GStore data so that we can predict the revenue it can generate from a customer. We are predicting the natural log of the sum of all transactions per user. For every user in the test set, the target is:

$$Y_{\text{user}} = \sum \text{transaction}_{\text{user}}$$

$$\text{target}_{\text{user}} = \ln(y_{\text{user}} + 1)$$

Data Exploration

Data is taken from <https://www.kaggle.com> and contains three datasets which are as follows:

- 1) Training dataset : It contains 903653 observations of 12 variables.
- 2) Test dataset : It contains 804684 observations of 12 variables.
- 3) Submission dataset : It contains all the unique customer ids.

The data in the CSV file is in JSON format. JSON, or JavaScript Object Notation, is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application, as an alternative to XML. There are two primary parts of JSON data which are key and value. Together known as the key/value pair.

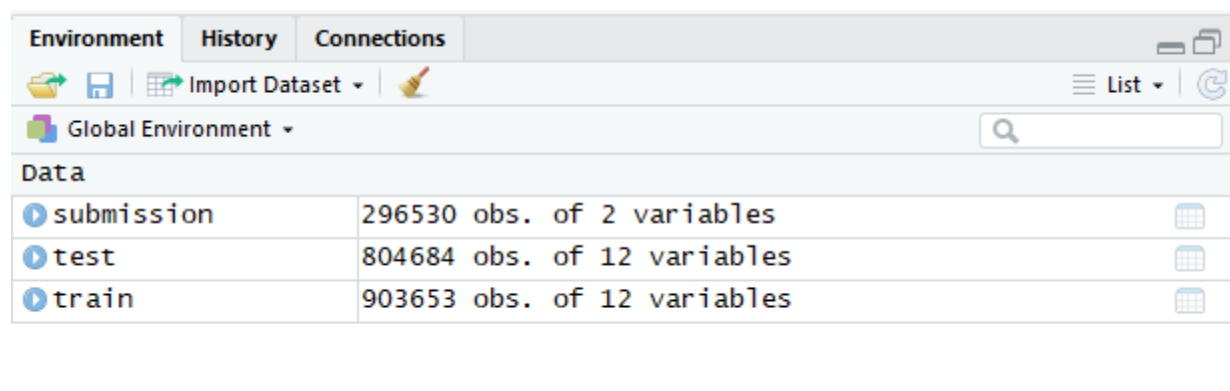
Key: It is always a string enclosed in quotation marks.

Value: It can be a string, number, Boolean expression, array or object.

First, we converted the JSON format file to data frames by declaring a function to convert it. This was done so that each **Key** could be converted as a separate column. After converting JSON format it was found that 12 variables were converted to 55 variables for training dataset and 53 variables for test dataset.

From there the missing values of every variable were discovered and variables which had more than 50% missing values were omitted as those variables would not be helpful during the prediction stage.

Following is the table below which shows the initial unchanged dataset when imported in R.



Global Environment	
Data	
submission	296530 obs. of 2 variables
test	804684 obs. of 12 variables
train	903653 obs. of 12 variables

Fig a. Shows the number of observation in each dataset

The submission file has two columns—one with unique customer ID (*fullVisitorId*) and the other for predicted log revenue (*PredictedLogRevenue*).

Definitions We Should Know

In order to utilize the wealth of metrics that GStore provides in the data set, we need to know exactly what the metrics mean.

Common Google Analytics Metrics & Definitions

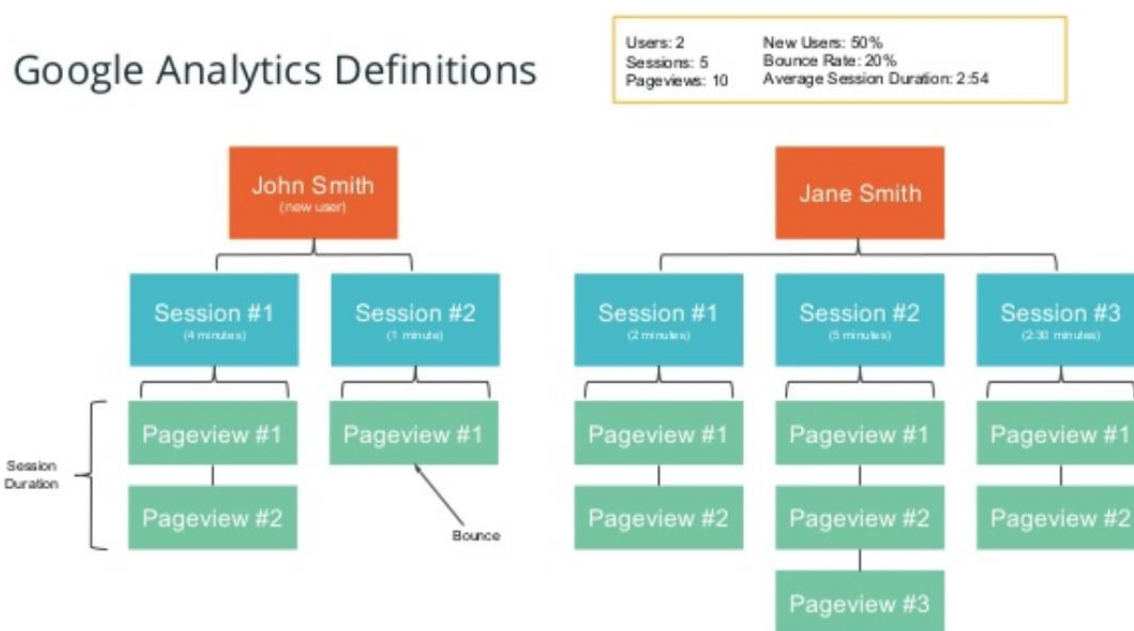
METRIC	DEFINITION	EXAMPLE
Users	The number of people that have visited the site.	Lets you know how many unique people come to the site compared to total sessions.
New Users	Percentage of visitors that have not been to the site before. Useful for trying to reach new audiences.	Helpful if you want to focus on reaching new audiences, or evaluating returning visitors.
Session	A single visit to a website, starting when visitor enters the site and ending when visitor leaves the site.	Total visits to the site. The most commonly used all-around metric.
Pageviews	The count of all pages viewed. May be multiple pageviews within a single session.	Helpful to understand what pages are being viewed after the initial landing page.
Unique Pageviews	The number of unique pages viewed. Does not include multiple views of the same page.	Helpful to know how much visibility content is getting on a per session basis.

Above are some of the must-know, common metrics in Google Analytics. Sessions are probably the most useful metric when using Google Analytics and give you arguably the most insight. Sessions and pageviews are directly related; a single session can include many page views. Within page views, unique page views indicate if users are visiting multiple individual pages or not.

Common Google Analytics Metrics & Definitions (cont'd)

METRIC	DEFINITION	EXAMPLE
Pageviews per Session	The average number of pages viewed per session.	Helpful for ad-based sites to improve engagement, showing more ads.
Bounce Rate	The percentage of people that visit a single page before leaving. Useful for evaluating content performance.	High bounce rates can be bad for sites based on ad-revenue, or totally fine for local businesses.
Avg Session Duration	The average length of time that visitors spend on the site during their session.	Helpful to understand how long visitors typically spend on the site overall.
Average Time on Page	The average length of time a visitor spends on a particular page.	Most helpful to evaluate page content on an individual basis.

These metrics are arguably less valuable but still worth noting. Page views per session and bounce rate are noteworthy for ad-based sites where one is trying to get a lot of page views to drive ad impressions and ad revenue. Average session duration indicates the average amount of time the user spends on the entire site and average time on page indicates the length of time the user spends on a specific page.



This diagram represents the real-time flow of some of the previously mentioned definitions. John Smith, a new user, and Jane Smith, a returning user, represent two visitors to the site. John had 2 different sessions whereas Jane had 3 different sessions. Within that, there are varying numbers of page views associated with each session.

Data Fields

- **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.
- **sessionId** - A unique identifier for this visit to the store.
- **socialEngagementType** - Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- **totals** - This section contains aggregate values across the session.
- **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).

Algorithm

We searched for algorithms that can be used to predict transaction revenue and found out that Random Forest, Gradient Boosting Machines (GBM), Linear Discriminant Analysis (LDA) and Deep Learning algorithms could be used.

We decided to use the Random Forest algorithm for our prediction. It is a supervised classification algorithm. From its name we can see that it creates a forest and makes it random. There is direct relationship between the number of trees in the forest and the result it can get, i.e., the more trees there are in forest, the more accurate the result. One thing to keep in mind is that creating a forest is not the same as constructing the decision with information gain or gain index approach. The difference between the Random Forest algorithm and the decision tree algorithm is that in Random Forest, the processes of finding the root node and splitting the feature nodes will run randomly.

The advantage of Random forest is that it can be used for both classification and regression tasks. Overfitting is one critical problem that may make the results worse, but for Random Forest algorithm, if there are enough trees in the forest, the classifier won't overfit the model. Another advantage is that the classifier of Random Forest can handle missing values and the Random Forest classifier can be modelled for categorical values.

Result

As mentioned earlier, submission file has 2 columns – One with unique customer ID and other with log of revenue predicted for each customer. With our random forest model, we predict the log of revenue for each of the customer and calculate the RMSE (root mean square error). We also check for the variable importance in terms of their contribution to the model. After predicting the revenue, we save the results as a CSV file and submit on the Kaggle competition page.

BRIEF EXPLORATORY DATA ANALYSIS (EDA)

1. Missing Data

Quite a few of the variables that are used in the training data set contain missing values. We plotted the percentage of missing data for each variable below.

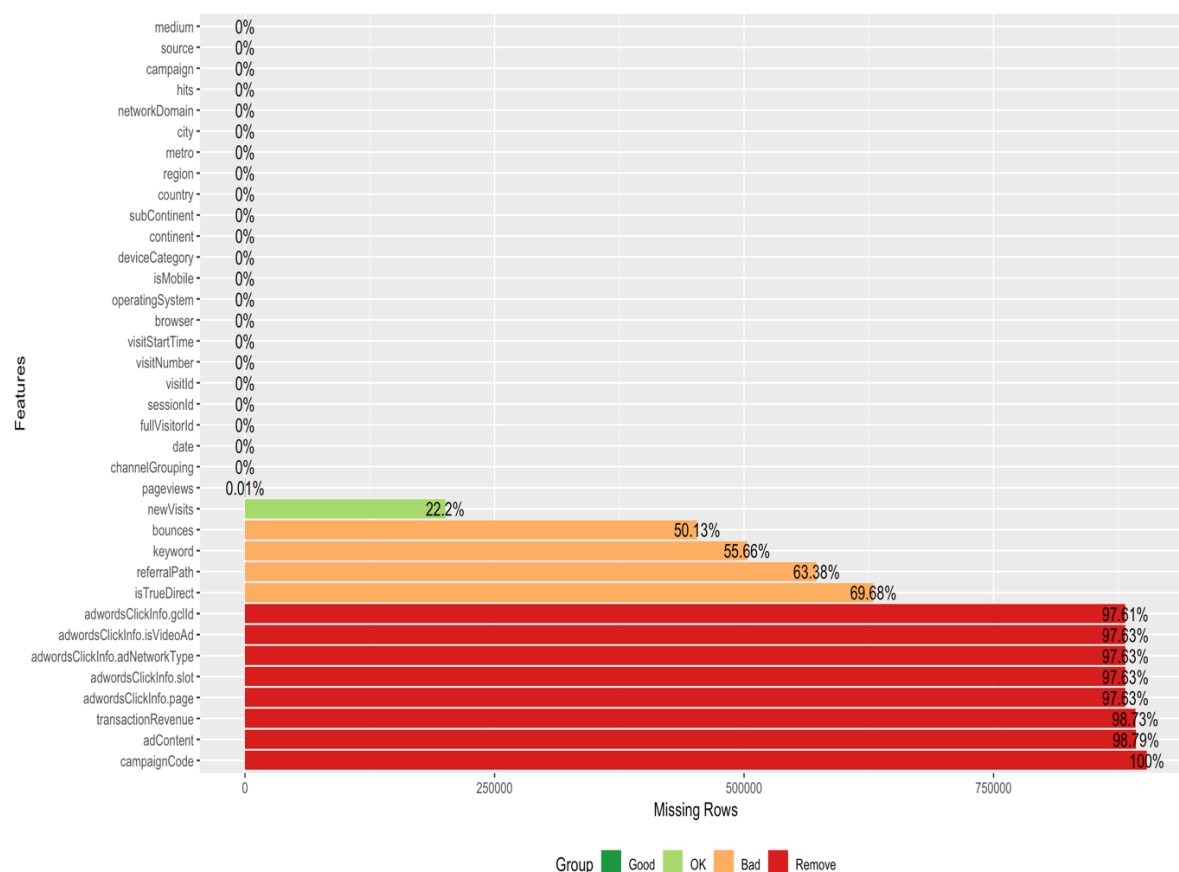


Fig. 1: Graph showing variables and the percentage of missing values they contain

It can be seen that most of the missing data seems related to the AdWords Campaigns. Furthermore, from the missing data in *transactionRevenue* we can assume that transactions did not take place in those instances. Excluding *transactionRevenue*, we can ignore the data in red from the set as it is not useful to conduct analyses with.

2. The Response Variable

From the graphs for the *transactionRevenue* variable, we can see that the data is skewed very sharply to the right. Not only that, but from the graph of the frequency distribution of the natural log of the response variable, we can note that the revenue has a lot of larger values.

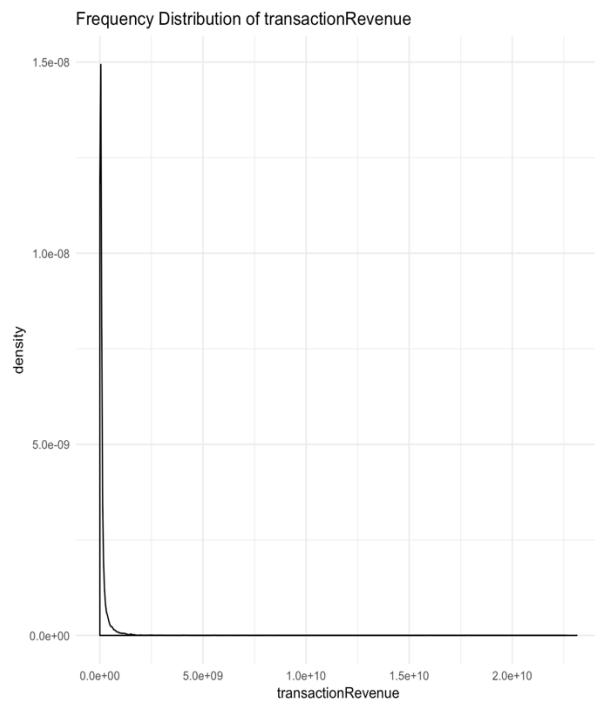


Fig. 2a. Graph showing the frequency distribution of transaction revenue

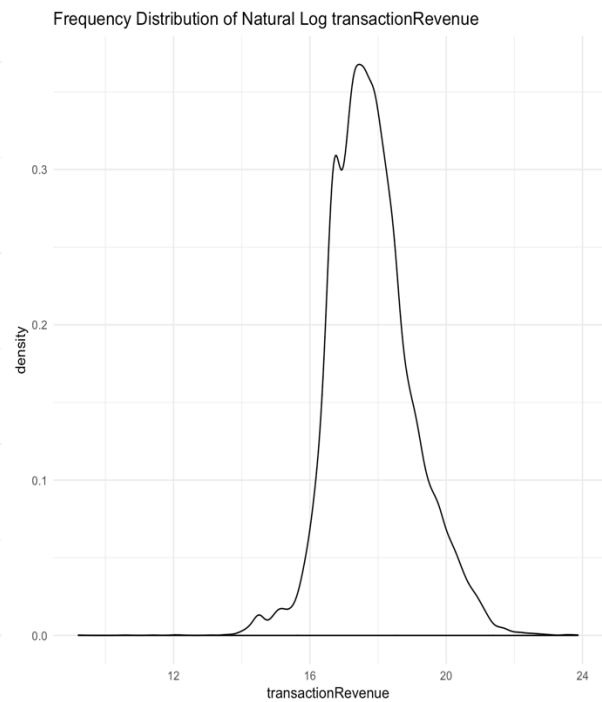


Fig. 2b.: Graph showing frequency distribution of natural log of transaction revenue.

3. Transaction Revenue Plotted Against Time

3.1 Transaction Revenue of the Training Set and Test Set

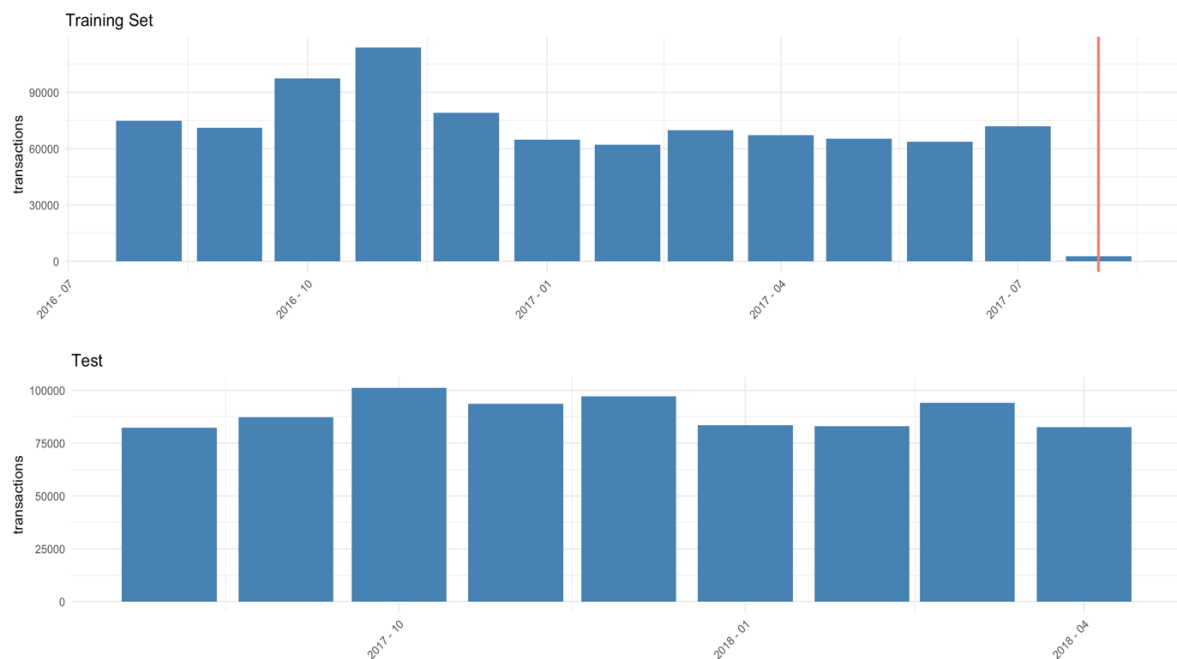


Fig. 3.1.: Graph showing transaction revenues of both the training set (top) and test set (bottom)

Above is a graph for both the training set and the test set to show how *transactionRevenue* varies over time. Since this data by itself is not of much help in prediction, we plot the sessions over time and the transaction revenue over time to compare them.

3.2 Time Series of Transaction Revenue and Sessions Over Time

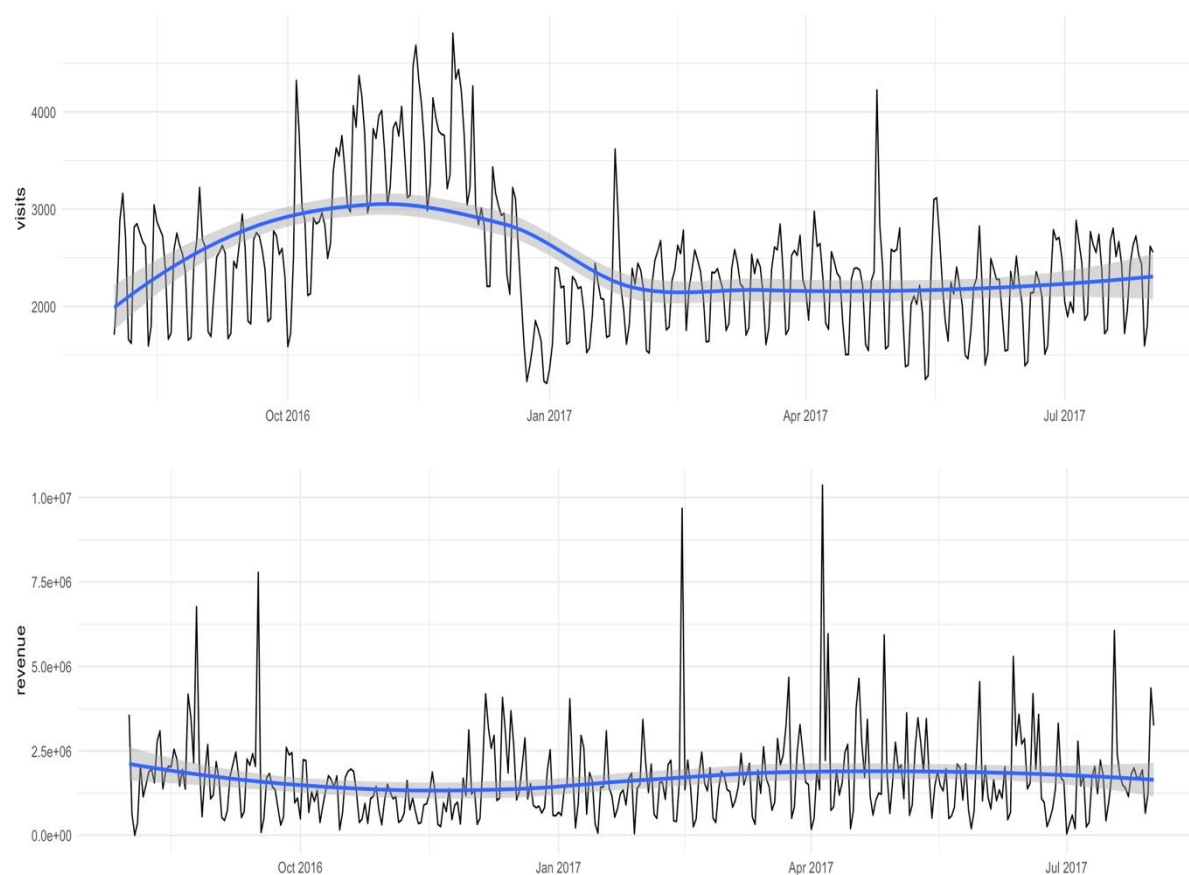


Fig. 3.2: Graph of visits over time (top) and graph of revenue over time (bottom) for the training set

From the time series we note that the number of visits does not result in more transactions. In fact, we can see that around November and December—when the number of visits reaches its peak—the transaction revenue is not high. The highest revenue that we can see seems to be in April of 2017.

The patterns do not seem to be very clear in how months affect revenue, but we can see that there are some that exist regardless. For example, the ratio of revenue to the number of session is low for November and December, however it is higher for April. This is somewhat of a surprise since November and December are months of holiday shopping and sales, however the transaction revenue does not seem to fit into what our expectations are.

4. Transaction Revenue and Frequency of Browsers Used

One of the other factors that is analysed is the frequency and transaction revenue of the browsers used.

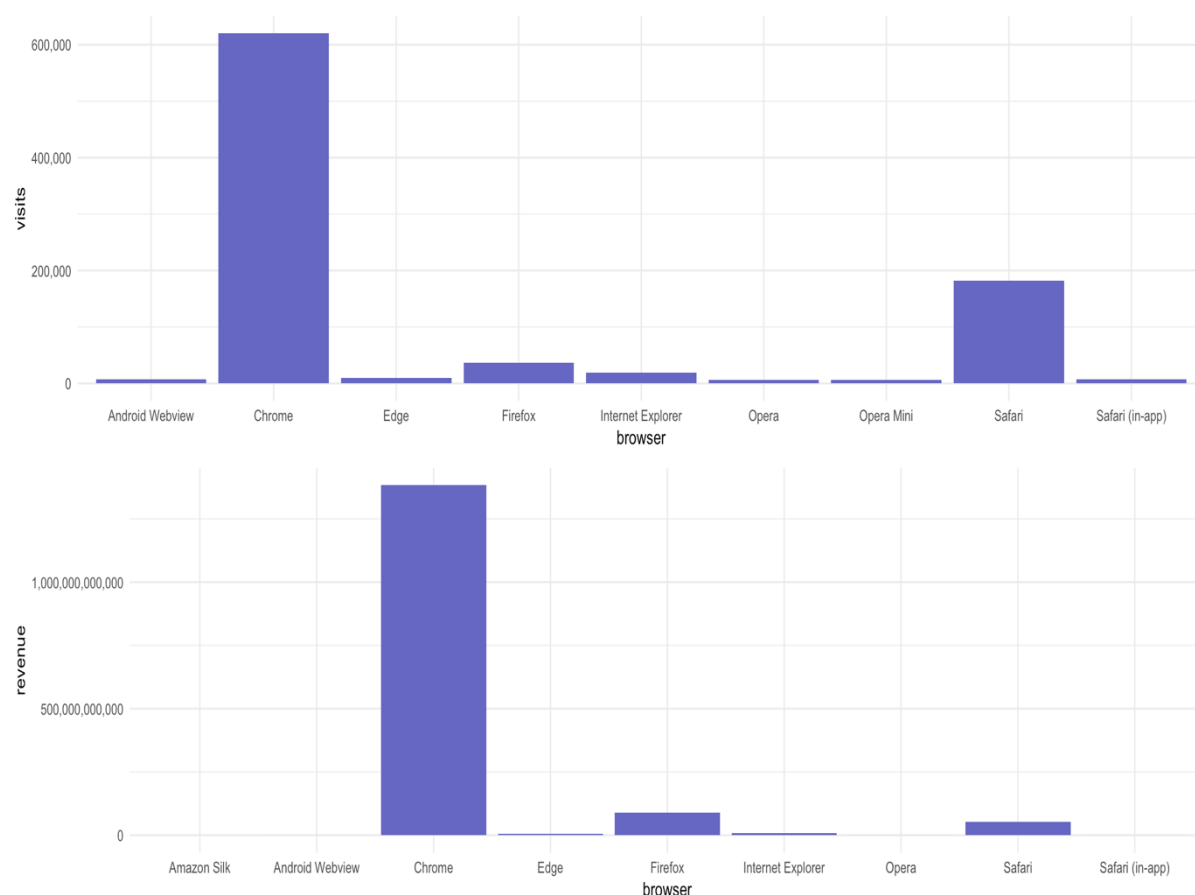


Fig. 4.: Graphs of visits over browser used (top) and revenue against browser used (bottom) for the training set

From the graph above we can see that the browsers which are used most often to access the site are Chrome, Firefox, and Safari. By far the most sessions and revenues come from the Chrome browser—which is to be expected as Chrome is one of the largest browsers used. Firefox has a healthy balance between the number of session and the revenue, while Safari has many sessions but very little revenue when compared.

5. Transaction Revenue and Frequency of Channel Groupings

Channel grouping needs to be defined before it can be analysed. Channels are the different modes through which customers come to the website while groupings are based on the different sources of traffic.

Illustrated in the graph we can see that Organic Search and Social both lead to relatively the highest amount of views while also resulting in the least amount of revenue. On the other hand, Referral and Display have a lower amount of sessions while resulting in a comparatively large amount of revenue.

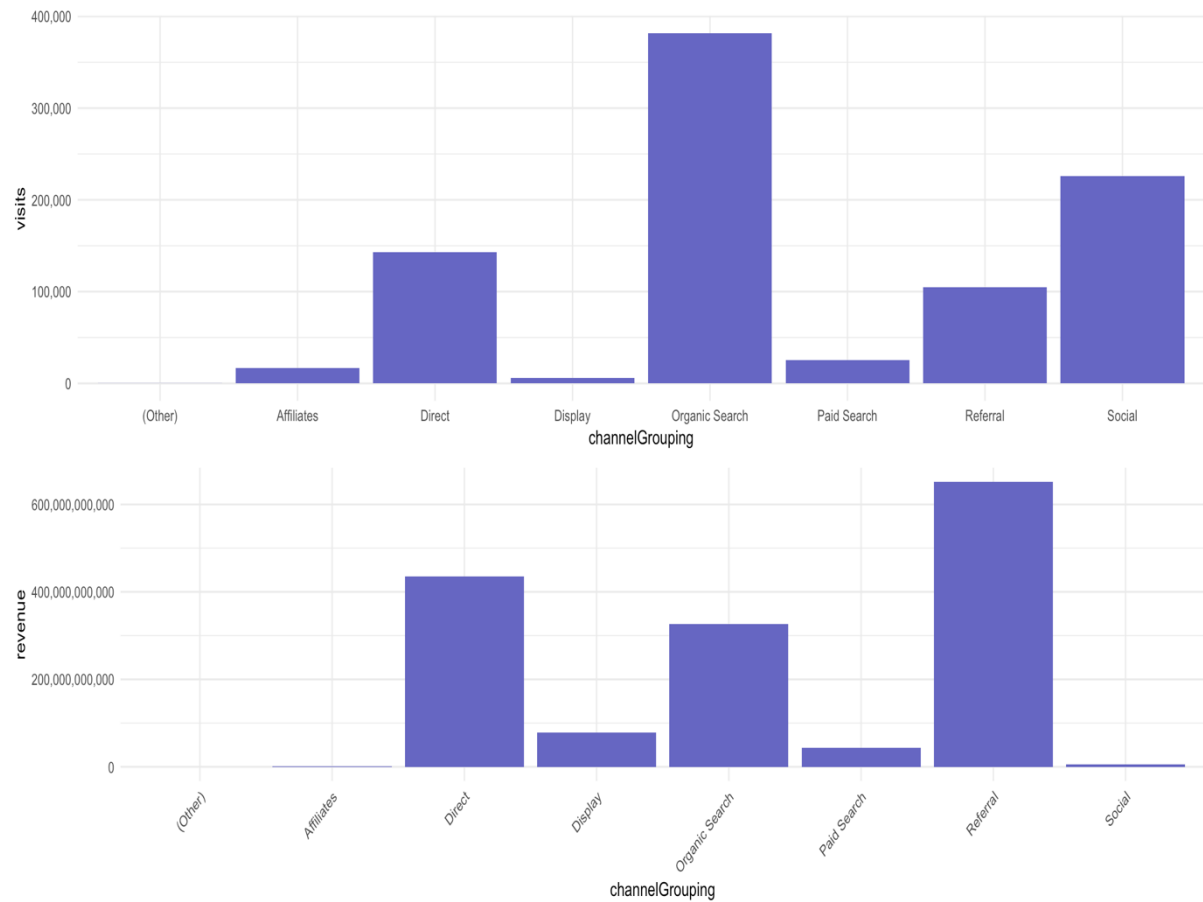


Fig. 5.: Graphs of visits against channelGrouping (top) and revenue against channelGrouping (bottom) for the training set

6. Transaction Revenue and Frequency for Operating Systems

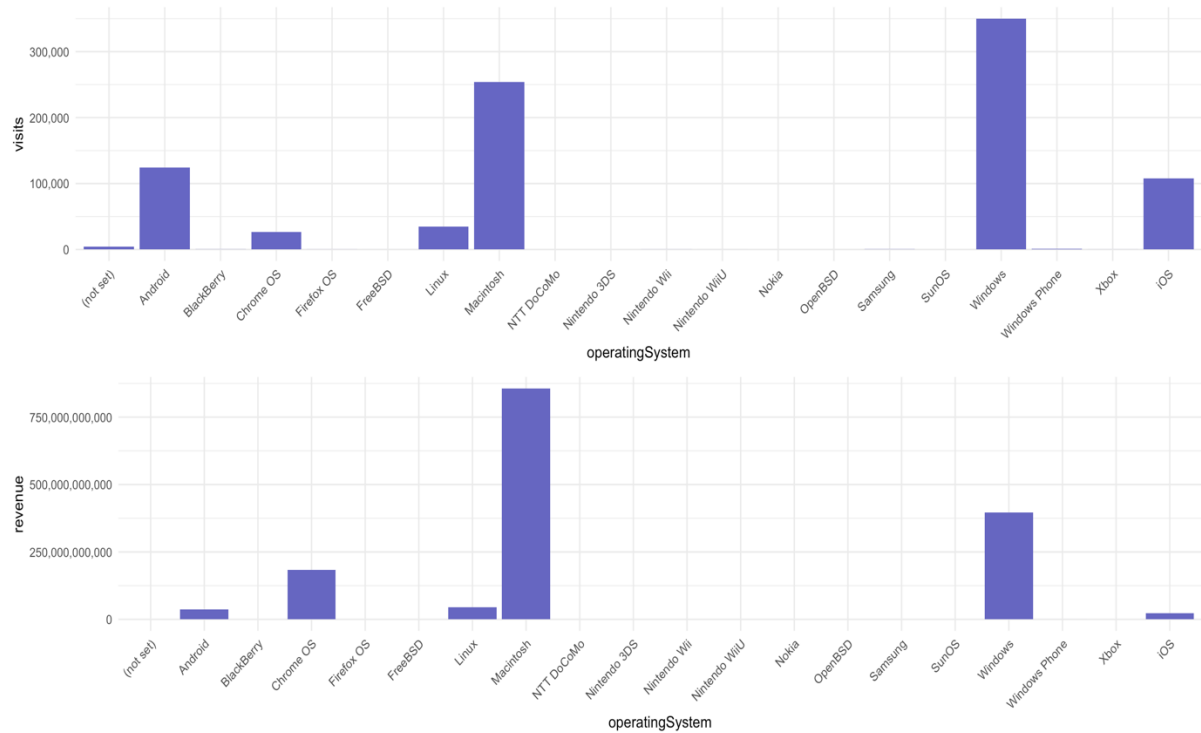


Fig. 6.: Graph of visits against operatingSystem (top) and revenue against operatingSystem (bottom) for the training set

In order to understand the trends of the data, the different Operating Systems and how they interacted with frequency and transaction revenue were also considered.

It can be noted that (not set) has a larger number of sessions than the Windows Phone, however, since it does not have any revenue and is rather meaningless, it is ignored. Interestingly, we can see that though Windows accounted for the largest amount of sessions, Macintosh was the one that resulted in more transactions by a larger margin.

7. Transaction Revenue and Frequency of Device Category

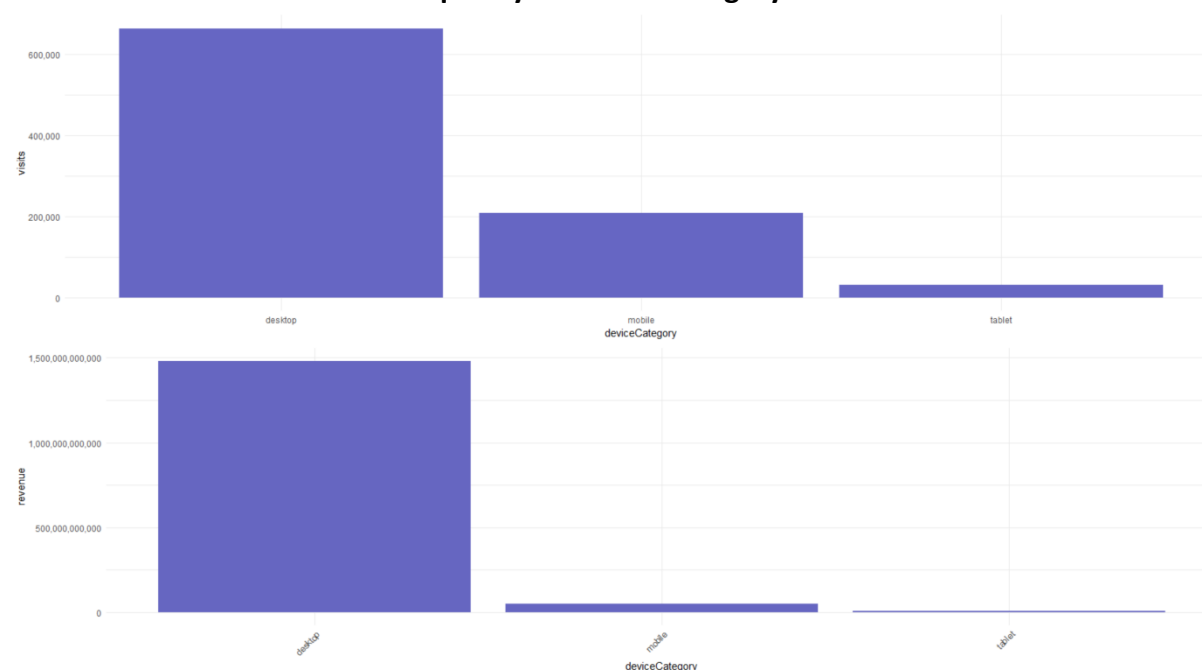


Fig. 7.: Graph of visits against deviceCategory (top) and revenue against deviceCategory (bottom) for training set

There are three device categories: desktop, mobile and tablet. Out of these desktop has both the highest number of visits and the revenue. It is interesting to note that mobile has high number of visits but less revenue. Tablet has both less number of visits and revenue.

8. Transaction Revenue and Frequency for Different Countries

Let us now consider if the number of visits is proportionate to revenue across different countries.

United States has the highest number of visits and revenue. It is interesting to find that countries like major countries like India and United Kingdom have higher number of visits but very less transaction revenue. It can be seen that Canada is second in terms of revenue.

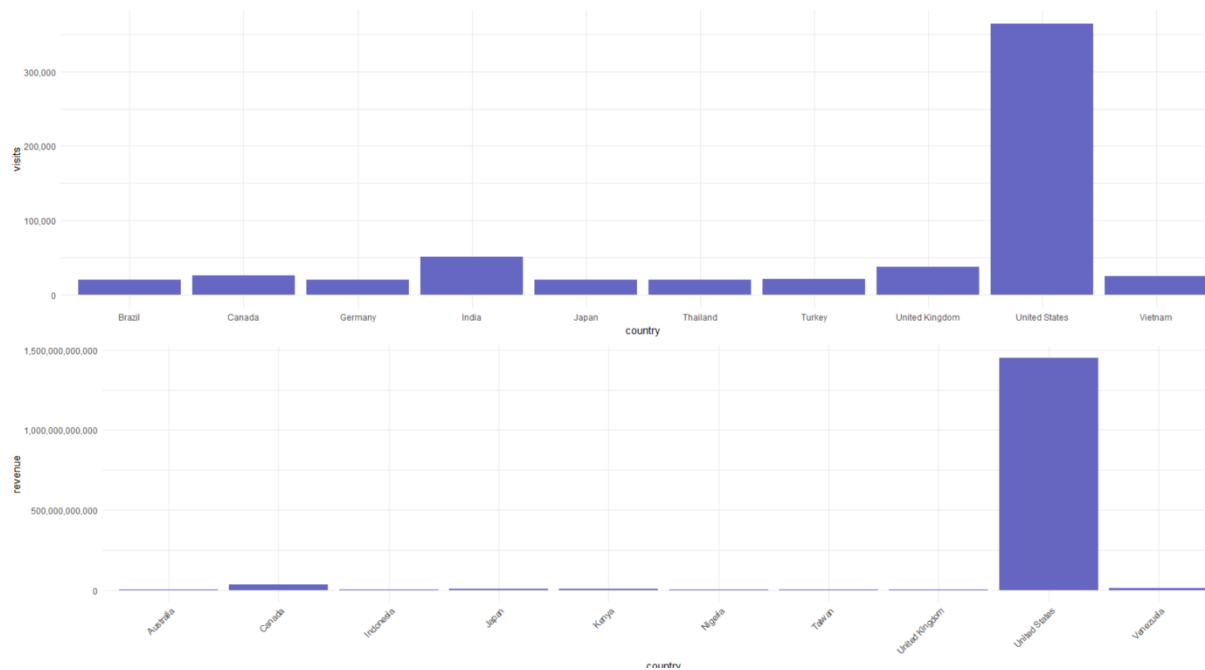


Fig. 8.: Graph of visits against countries (top) and revenue against countries (bottom) for training set

9. Transaction Revenue and Frequency for Different Continents

A similar trend is seen for continents as well. America has both the highest number of visits and revenue. It is interesting to note that though Asia and Europe has significant number of visits, they barely produce any revenue.

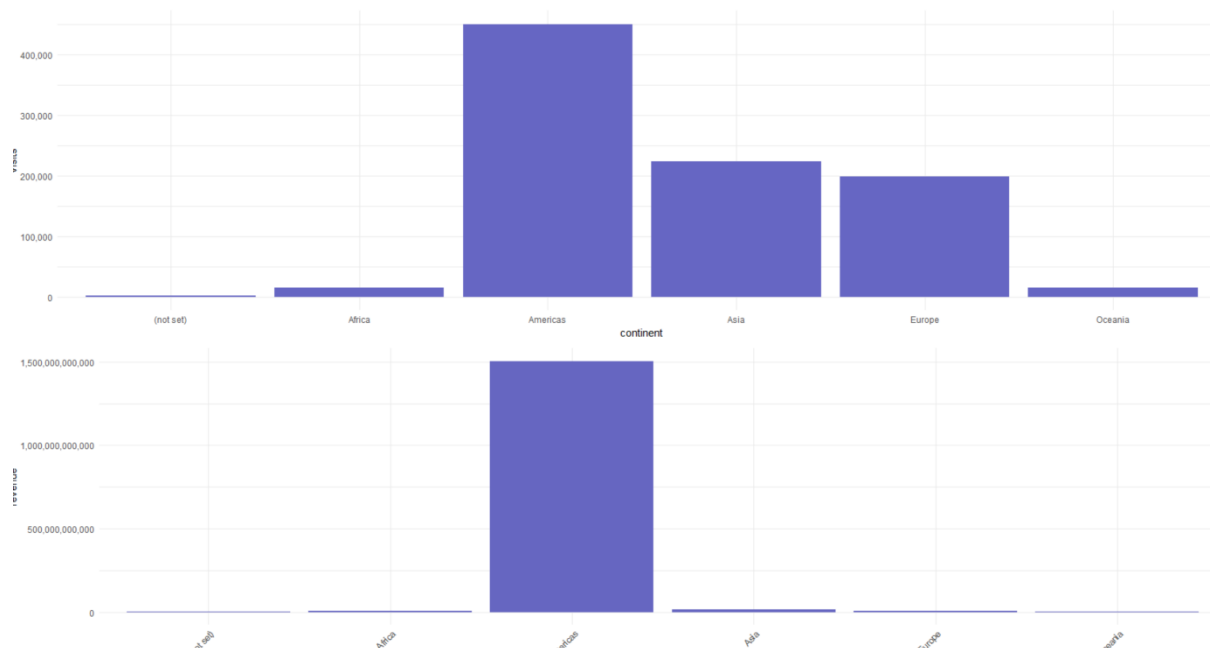


Fig. 9.: Graph of visits against continents (top) and revenue against continents (bottom) for training set

10. Transaction Revenue and Frequency for Different Subcontinents

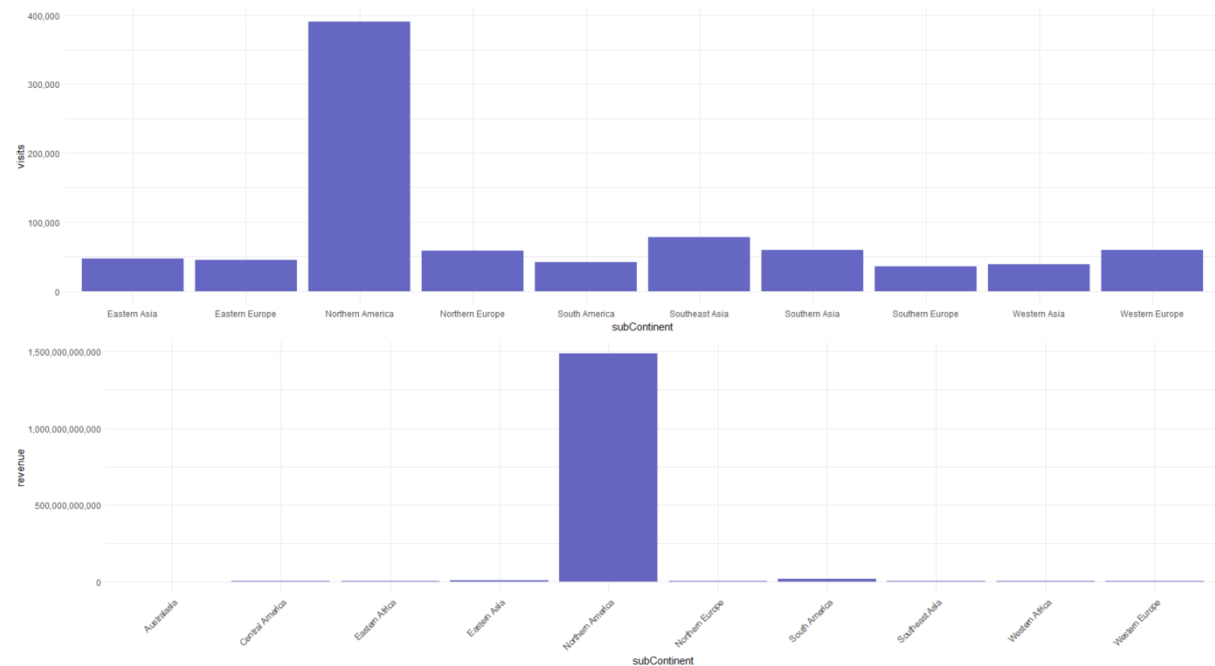


Fig. 10.: Graph of visits against subcontinents (top) and revenue against subcontinents (bottom) for training set

Looking at subcontinent perspective, we note the same trend. Transaction revenue and number of visits is highest for Northern America. In fact every region has significant number of visits, but they barely produce any revenue.

11. Transaction Revenue and Frequency for Different Regions

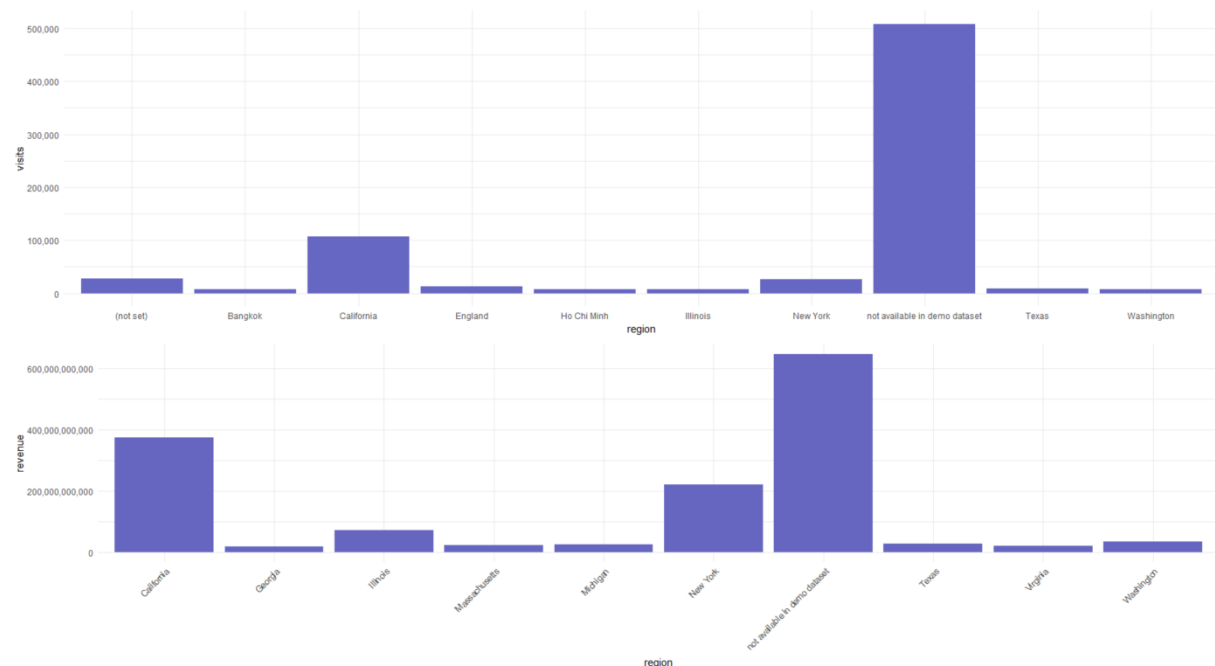


Fig. 11: Graph of visits against regions (top) and revenue against regions (bottom) for training set

From the trend we recognise that much of the information is not available in demo dataset. However we can see that New York and California are the two regions with higher number of visits as well as transaction revenue.

12. Transaction Revenue and Frequency for Different Campaigns

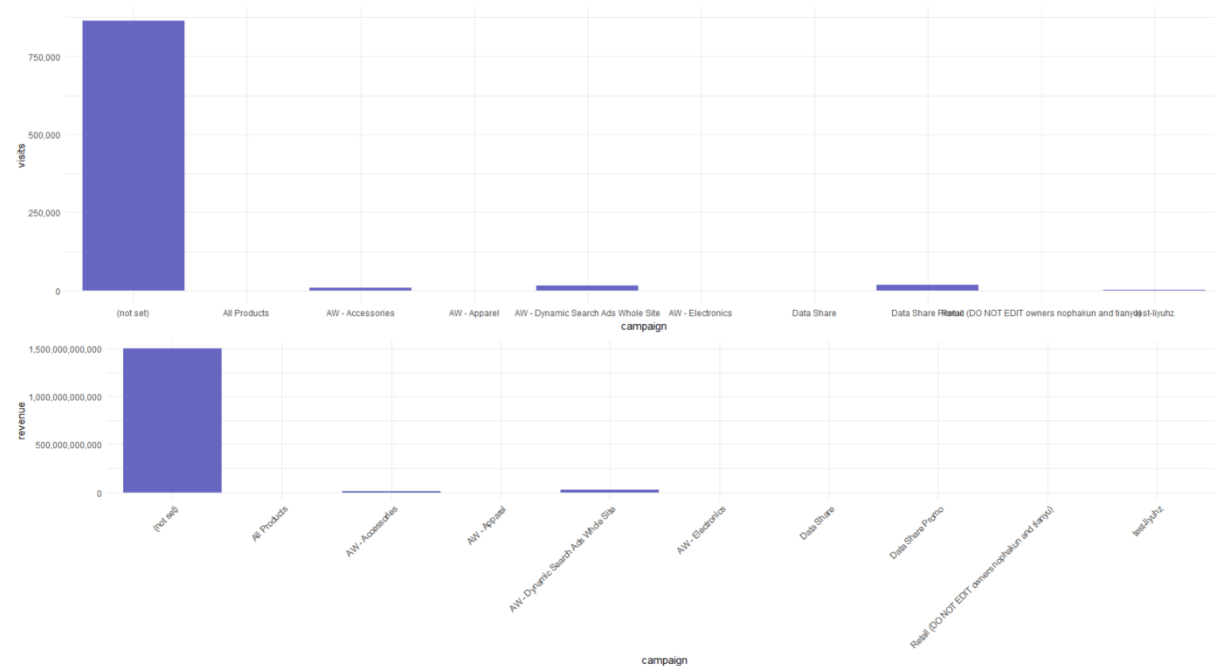


Fig. 12: Graphs of visits against campaign (top) and revenue against campaign (bottom) for training set

It can be seen that not much information is provided about campaign type. However, it is interesting to note that AW-Accessories and AW- Dynamic search Ads Whole Site have higher visits and also some revenue, whereas Data Share Promo has high number of visits but no revenue at all.

13. Log of Transaction Revenues v/s Log of Page Views

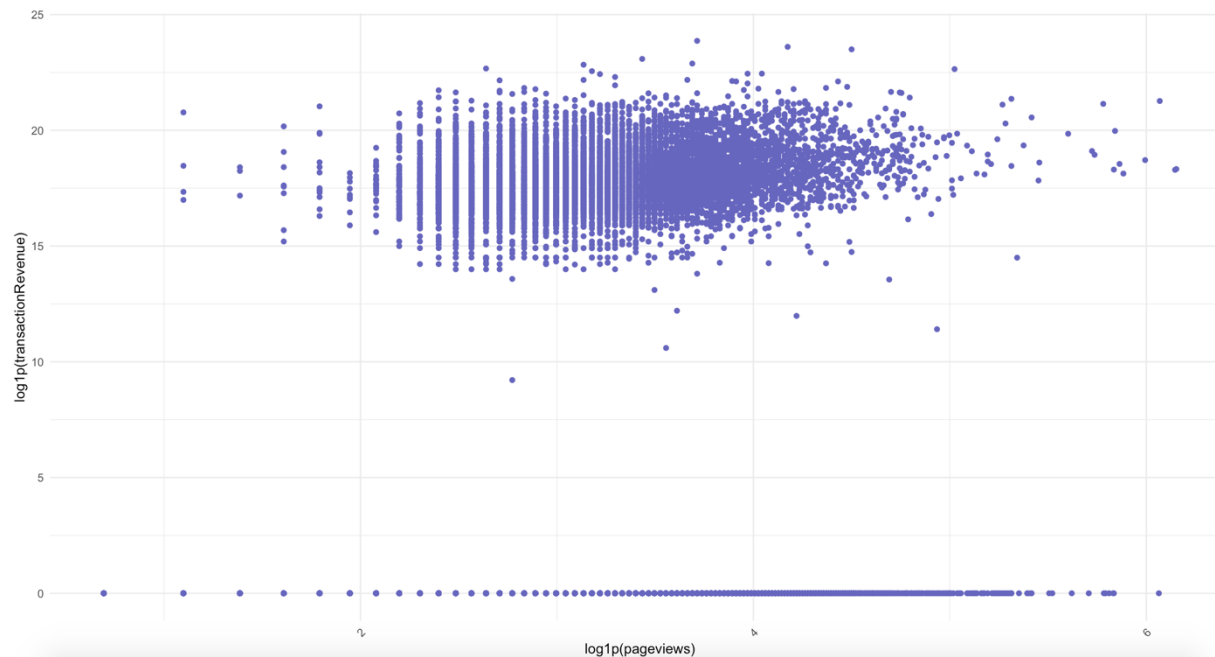


Fig. 13: Graph of log of transaction revenue against log of pageviews of training set

Page view shows how many times the page is visited. From the log graph, it is observed that transaction revenue increases with increase in the number of page views. It is evident that more page views imply greater popularity for the product, which will ultimately result in higher transaction revenue.

14. Log of Transaction Revenue v/s Log of Hits

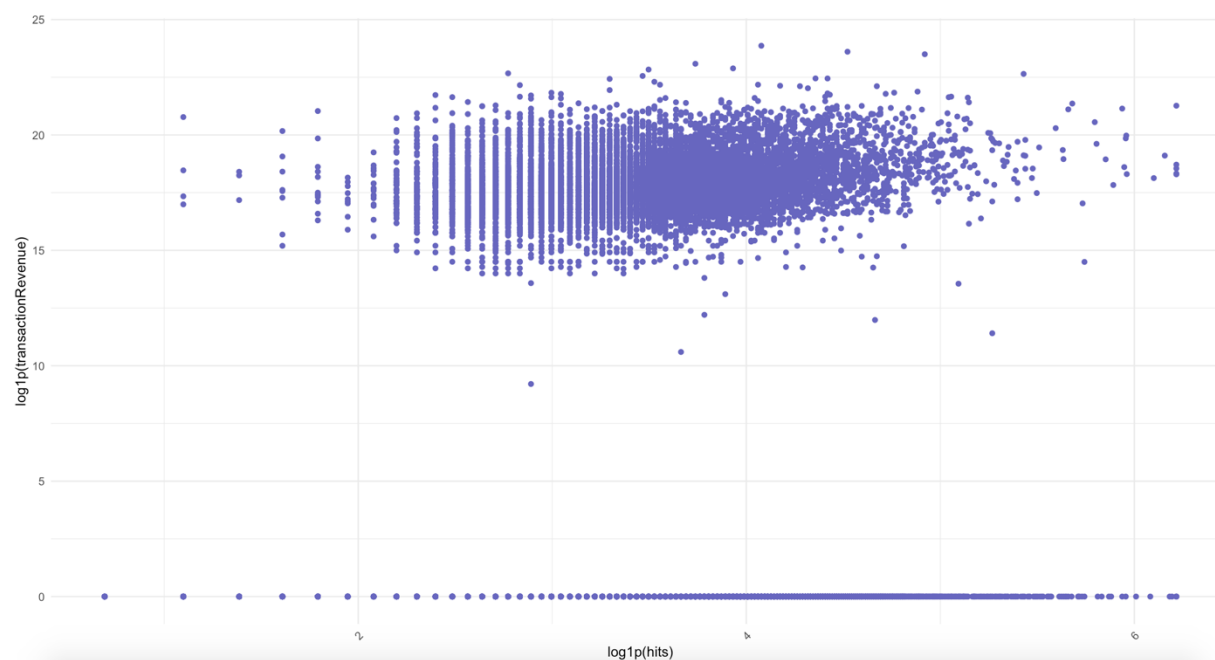


Fig. 14: Graph of log of transaction revenue against log of hits for training set

The log graph for hits is almost similar to the log graph of page views.

A hit is generated when a file is requested and served on your website. From the graph it is seen that transaction revenue is high for higher number of hits. It is evident that more page views imply greater popularity for the product, which will ultimately result in higher transaction revenue.

15. Data Wise Aggregation

We have plotted the trends of Daily Visits, Daily new Visits, Daily Hits and Daily Bounces against time.

Visits: These are series of hits from any particular IP address

Hits: A single file request in the access log of a Web serve. While a hit is a meaningful measure of how much traffic a server handles, it can be a misleading indicator of how many pages are being viewed.

Bounces: A bounce is calculated specifically as a session that triggers only a single request to the Analytics server, such as when a user opens a single page on the site and then exits without triggering any other requests to the Analytics server during that session.

Checking The Trends For Training Data

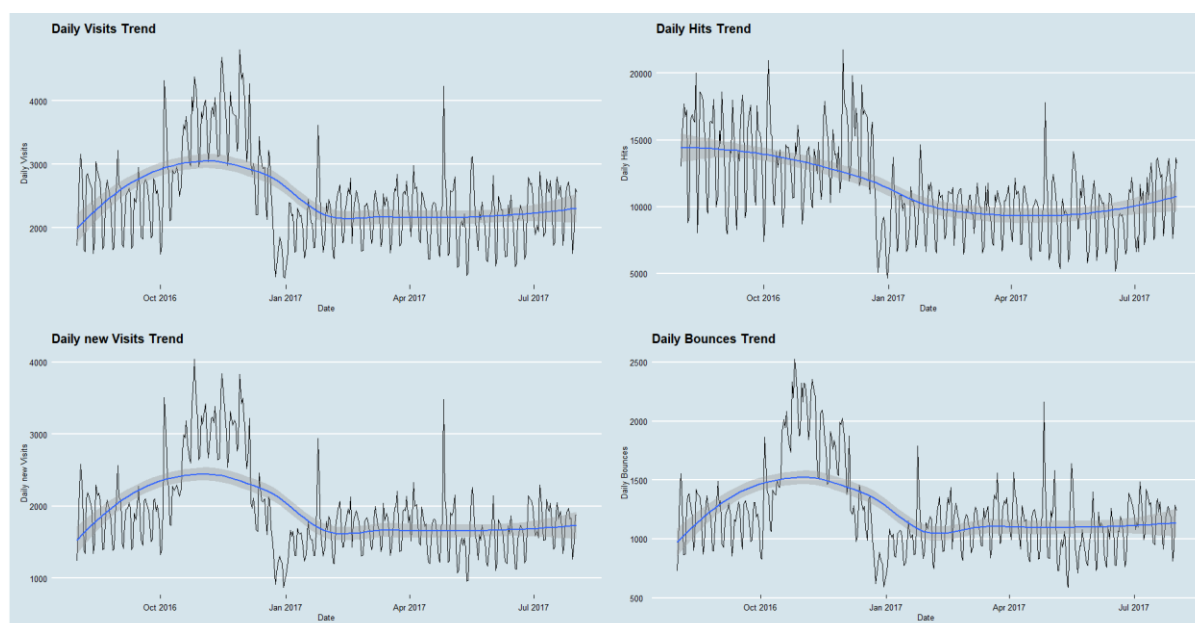


Fig. 15.1: graphs of daily trends for visits, hits, new visits, and bounces (from left to right top) for training set

Checking The Trends For Test Data

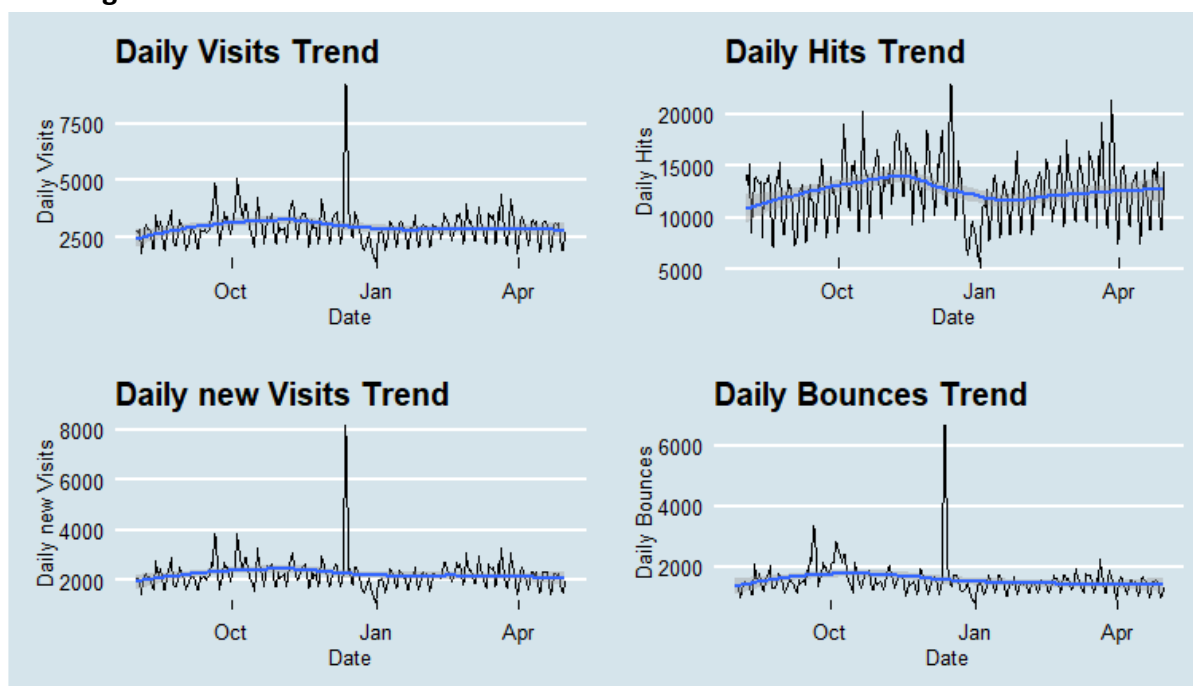


Fig. 15.2: graphs of daily trends for visits, hits, new visits, and bounces (from left to right top) for test set

a. Daily Visits (and new Visits) Trend

There is an interesting comparison between Daily Visits and Daily new Visits Trend. It is seen that the number of visits is higher in the holiday season (i.e during November and December). For other months the trend follows a regular pattern.

b. Daily Hits Trend

The trend shows that the number of hits is high in the second half of year (i.e July to December) than in the first half.

c. Daily Bounces Trend

Daily Bounces Trend is similar to Daily Visits trend. The number of Bounces is high during the holiday season of November and December.

Plain bounce might not be entirely useful, hence let's couple it with another useful GA metric - **Bounce Rate**

Bounce rate = Bounces/Visits

Bounce rate is single-page sessions divided by all sessions, or the percentage of all sessions on your site in which users viewed only a single page and triggered only a single request to the Analytics server.

d. Daily Bounce Rate Trend for Training Data

From the bounce rate trend given below, it is seen that the bounce rate is highest in November and lowest in June. The bounce rate trend is almost similar to the bounce trend.

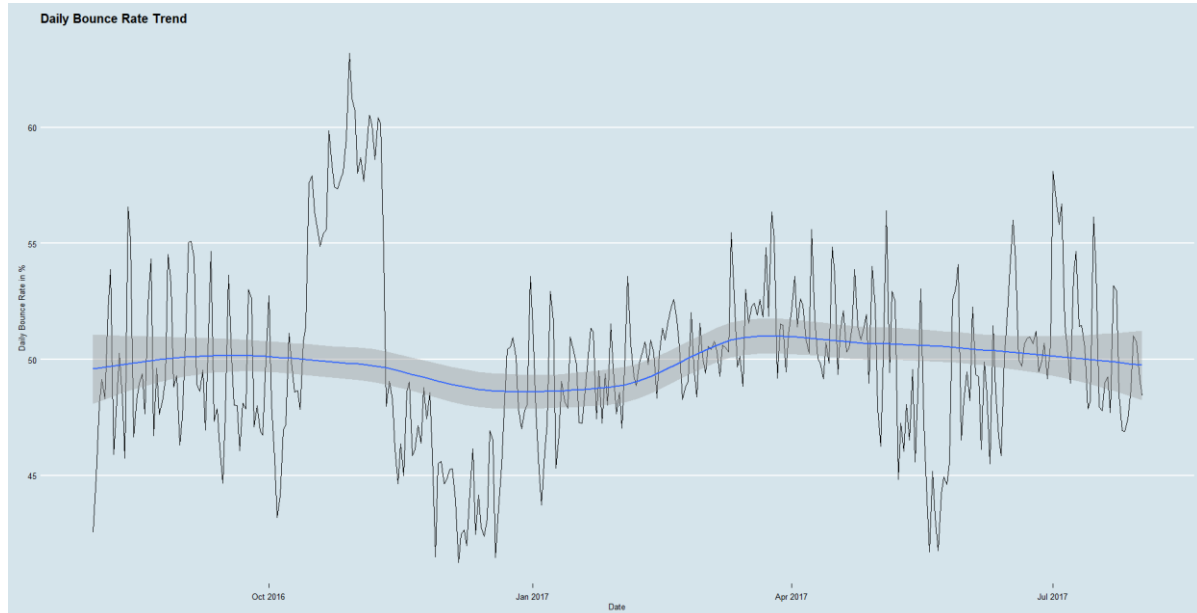


Fig. 15.3: graphs of daily bounce rate trend for training set

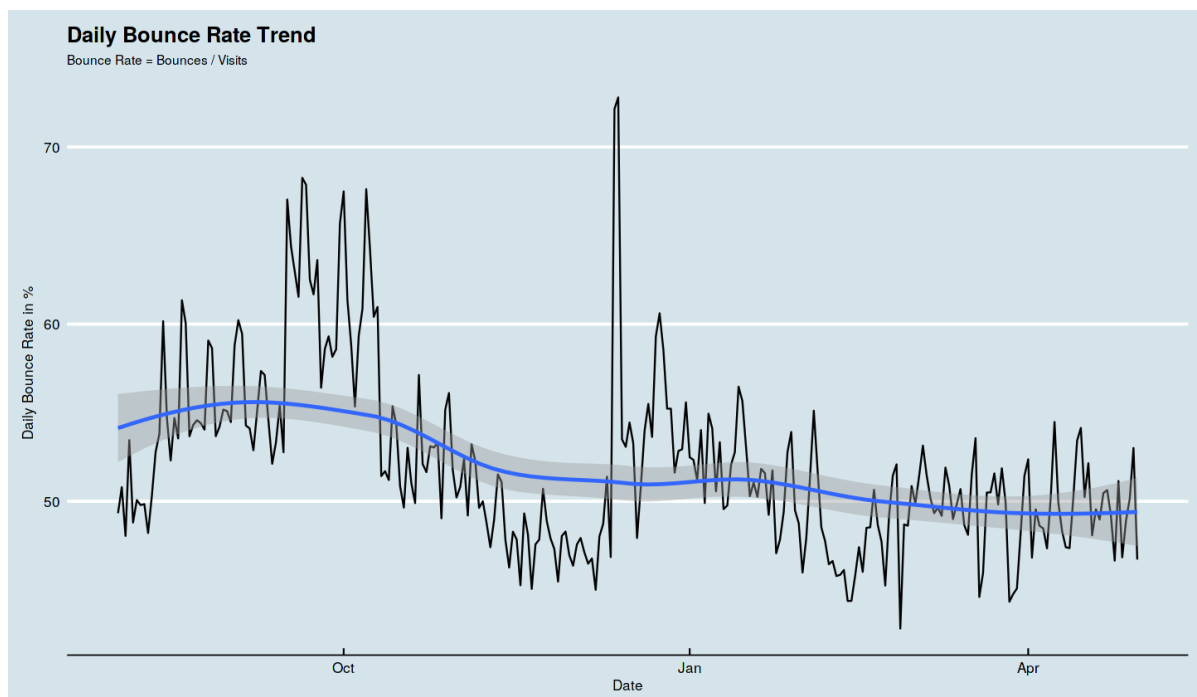
e. Daily Bounce Rate Trend for Test Data

Fig. 15.4: graphs of daily bounce rate trend for test set

The daily bounce rate of test data is similar to the training data. The bounce rate is higher during October compared to other months.

16. Transaction Revenue and Frequency for *isMobile*

We also plotted the transaction revenue and frequency for the *isMobile* variable.

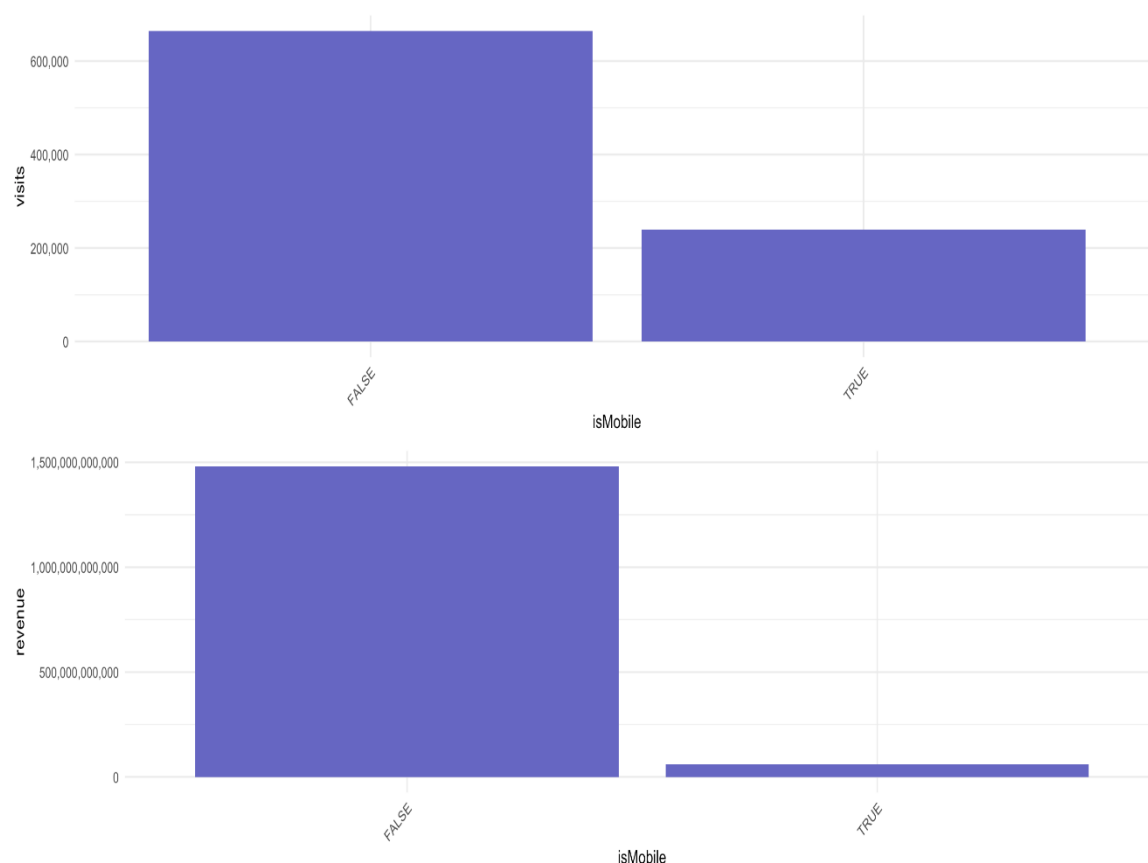


Fig. 16: graphs of visits against ismobile (top) and revenue against ismobile (bottom) for training set

Summary of Exploratory Data Analysis

1. The variables present in *tr_totals* play significant role in the analysis. We have used two important parameters namely Transaction revenue and Visits as metrics to analyse their relationship with other variables.
2. The variables derived from *tr_device* are quite useful in the EDA. For example browser, operating system and mobile device provides knowledge on which device parameters has higher transaction revenue and visits.
3. The variables that are derived from *tr_geoNetwork* have higher significance in the analysis. It is seen that America produces highest transaction revenue when compared to other countries.

4. The variables in traffic source are not much useful in the analysis as most of these variables have lot of missing values. For example, data related to *adwords* click doesn't provide any useful information.
5. From the various graphs analysed above, we find the following predictors to be most important in developing an algorithm:

operatingSystem, channelGrouping, isMobile and region.

Feature Engineering

Feature engineering is one process of using domain knowledge of the data to create features that makes any machine learning algorithm work. By doing this, it increases the predictive power of the algorithm, resulting in improved model accuracy on unseen data.

In our code, weeks, day of week, type of operating system of training and testing data is converted to a format which the machine learning algorithm understands.

One Hot Encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to machine learning algorithms to do a better job in prediction. In our code, we have created dummy variables that allows us to perform one hot encoding on the data frame we have created.

ALGORITHM

With the data cleaned and analysed, we have leads to go about from here. We have to choose an algorithm that builds a classification and prediction model to predict revenue for each customer in the test/validation set. We chose to build a random forest classifying and predicting model to forecast the transaction revenue.

Comparing packages for which had more advantages than the regular *randomforest*, we found that random forest in H2O package gave better accuracy for our prediction.

Firstly, normal random forest package in R splits based on the Gini criterion and whereas H2O trees are split based on reduction in Squared Error (even for classification). Moreover, H2O also uses histograms for splitting and can handle splitting on categorical variables without dummy encoding.

Secondly, a normal random forest package builds really deep trees, resulting in pure leaf nodes which means the predicted outcome or an observation is either going to be 0 or 1 in each tree. This can lead to a bad AUC score which causes poor prediction in outcome. On the other hand, the trees in H2O's random forest aren't quite as deep and therefore aren't as pure, allowing for

predictions that have some more granularity to them and in turn gives us a better AUC score and, hence, better prediction.

ALGORITHM USED: RANDOM FOREST FROM H2O PACKAGE

H2O is an open source machine learning platform where companies can build models on large data sets (no sampling needed) and achieve accurate predictions. It is incredibly fast, scalable and easy to implement at any level.

One of the algorithms which it supports is Random Forest, which has been used in this project for prediction. Instead of choosing normal random forest package we chose to use `h2o.randomForest()` package since there are differences in the algorithms and their default hyperparameters.

Random Forest is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees. The individual trees here are referred as decision trees which are one of the most popular learning methods commonly used for data exploration. Each decision tree in the forest considers a random subset of features when forming questions and only has access to a random set of the training data points. This increases diversity in the forest leading to more robust overall predictions and the name 'random forest.'

Each decision tree is constructed using the following algorithm:

1. Let the number of training cases be N , and the number of variables in the classifier be M .
2. Consider number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M .
3. Choose a training set for this tree by choosing n times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose m variables on which you base the decision at that node. Calculate the best split based on these m variables in the training set.
5. Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction (This is usually the case when we are predicting a continuous value of a particular variable). But in other case when we need to classify and the targets are a discrete class label, in that case, the random forest will take

a majority vote for the predicted class (It has been clearly depicted in the diagram given below on this page). It is used in many real-world applications such as predicting patients for high risks, prediction of parts failures in manufacturing, predicting loan defaulters etc.

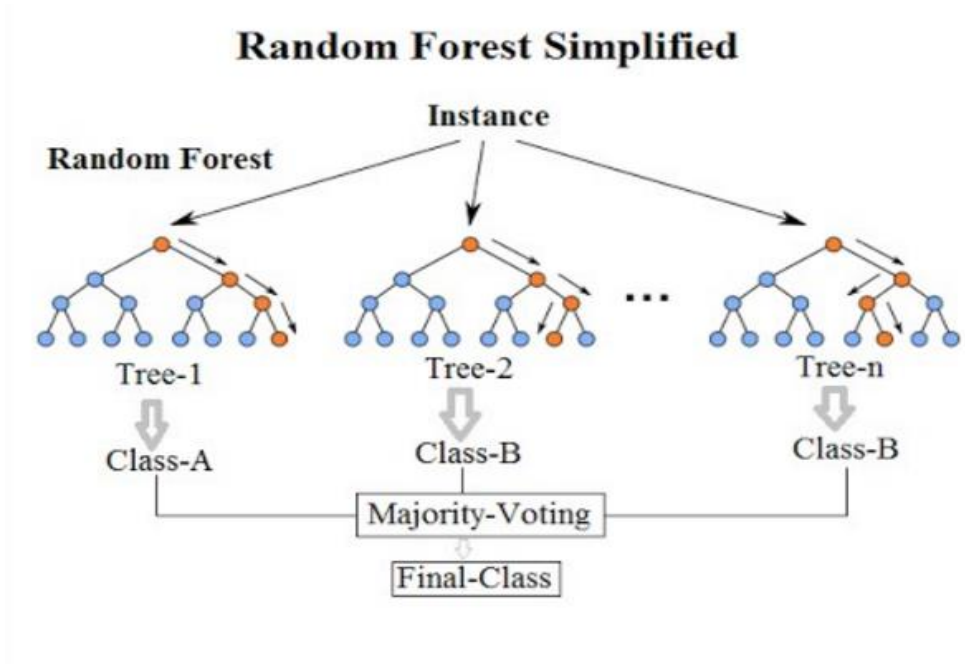


Fig.: Random Forest Simplified Diagram

Random Forest Algorithm Flow Chart

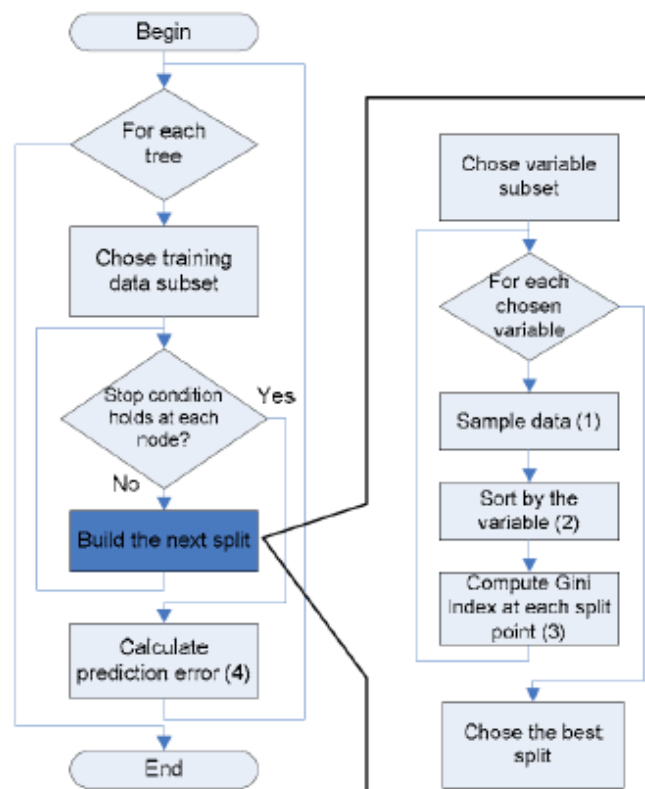


Fig: Random Forest Algorithm Flowchart

The reason why we preferred Random Forests over Gradient Boosted Decision Trees is that firstly Random Forests are much easier to tune as compared to Gradient boosted machines since it only requires one hyperparameter to set which is the number of features randomly selected at each node but on the other hand, Gradient boosted machines require several hyperparameters that include the number of trees, the depth (or number of leaves), and the shrinkage (or learning rate). So since we are dealing with large datasets it would have been much complex to tune GBMs as compared to Random Forests. Secondly, as compared to Gradient Boosted Machines, Random Forests are much harder to overfit. Although Random Forests do overfit but are more robust to overfitting as compared to GBMs and require less tuning to avoid overfitting.

Strengths of Random Forest

- It is highly accurate and is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier.
- It has a fast runtime.
- It can handle large number of input variables without variable deletion.
- It can run efficiently on large databases.
- It gives an estimate of which variables are important in the classification.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data is missing.
- Forests which are generated can be used for future use on other data.
- Cluster identification can be used to generate tree-based clusters through sample proximity
- To give information about the relation between the variables and the classification, the prototypes are computed.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- It's capabilities can be extended to unlabelled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

Weaknesses of Random Forest

- Random forests sometimes tend to overfit for some datasets with noisy classification/regression tasks.
- When used for regression they cannot predict beyond the range in the training data.
- For data including categorical variables with different number of levels, random forests are biased in favour of those attributes with more levels. Therefore, the variable importance scores from random forest are not reliable for this type of data.
- It is basically used when you are just looking for high performance with less need for interpretation.
- With random forest, we can train a model with a relative small number of samples and get pretty good results. It will, however, quickly reach a point where more samples will not improve the accuracy.

RESULTS

Description About the Model Used

H2O.Randomforest() – Builds a Random Forest Model on an H2OFrame. It builds trees faster than the R's random forest function. Trees are independent and can be built parallelly.

Summary(model) – gives the summary of the model with all the basic details

Regression Model – DRF

Distributed Random Forest (DRF) is a powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree.

Each of these trees is a weak learner built on a subset of rows and columns. More trees will reduce the variance. Both classification and regression take the average prediction over all of their trees to make a final prediction, whether predicting for a class or numeric value.

Model Summary gives information about the following –

1. Number of trees
2. Number of internal trees
3. Model size in bytes
4. Min depth

5. Max depth
6. Mean depth
7. Min leaves
8. Mean leaves

By default, the number of trees selected will be 50. Hence we see number of internal trees = 50. Min and Max depth is 30 (by default). Mean leaves gives the average number of leaves put together from all 50 trees.

Scoring History

RMSE—Root mean square Error—is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

Our model started RMSE score with 2.23 and started reducing with the increase in number of trees.

Variable Importance

Variable importance is determined by calculating the relative influence of each variable: whether that variable was selected during splitting in the tree building process and how much the squared error (over all trees) improved as a result.

Relative importance quantifies the contributions of a specific predictor against those made by other individual predictors in predicting the response variable. The number that you might be thinking of that needs to be between 1 and 100 is the scaled importance. Lastly, the reason you are not getting a confusion matrix in your output is that you have a regression model rather than a classification model.

Confusion matrices are only produced for classification models.

```
Model Details:
=====
H2ORegressionModel: drf
Model Key: DRF_model_R_1537875163242_1
Model Summary:
 number_of_trees number_of_internal_trees model_size_in_bytes min_depth
1           50           50       7368591       30
 max_depth mean_depth min_leaves max_leaves mean_leaves
1       30 30.00000   11399   11974 11683.94000
```

H2ORegressionMetrics: drf

** Reported on training data. **

** Metrics reported on Out-Of-Bag training samples **

MSE: 3.309478

RMSE: 1.819197

MAE: 0.3122479

RMSLE: 0.3545644

Mean Residual Deviance : 3.309478

Scoring History:

	timestamp	duration	number_of_trees	training_rmse
1	2018-09-25 11:36:08	0.044 sec	0	
2	2018-09-25 11:36:31	23.758 sec	1	2.23010
3	2018-09-25 11:36:55	47.044 sec	2	2.17017
4	2018-09-25 11:37:19	1 min 11.849 sec	3	2.13121
5	2018-09-25 11:37:41	1 min 33.241 sec	4	2.09892

training_mae training_deviance

1		
2	0.31932	4.97333
3	0.31489	4.70962
4	0.31452	4.54205
5	0.31509	4.40545

	timestamp	duration	number_of_trees	training_rmse
46	2018-09-25 11:52:50	16 min 42.700 sec	45	1.82215
47	2018-09-25 11:53:12	17 min 4.668 sec	46	1.82115
48	2018-09-25 11:53:33	17 min 25.635 sec	47	1.82081
49	2018-09-25 11:53:55	17 min 46.913 sec	48	1.81979
50	2018-09-25 11:54:15	18 min 6.915 sec	49	1.81948
51	2018-09-25 11:54:38	18 min 30.547 sec	50	1.81920

training_mae training_deviance

46	0.31236	3.32024
47	0.31228	3.31657
48	0.31227	3.31535
49	0.31219	3.31162
50	0.31223	3.31052
51	0.31225	3.30948

```

Variable Importances: (Extract with `h2o.varimp`)
=====

Variable Importances:
      variable relative_importance scaled_importance
1      pageviews  34945116.000000      1.000000
2         hits   27174210.000000      0.777625
3    visitNumber  13931599.000000      0.398671
4   trainReferral  2563985.000000      0.073372
5      trainMacintosh  1832805.625000      0.052448
percentage
1 0.350281
2 0.272388
3 0.139647
4 0.025701
5 0.018372
---

```

Fig: Model details, scoring history, and variable importance of the variables in the dataset (model summary)

`h2o.varimp_plot(model)` gives us a bar plot for the important variables identified by the model.

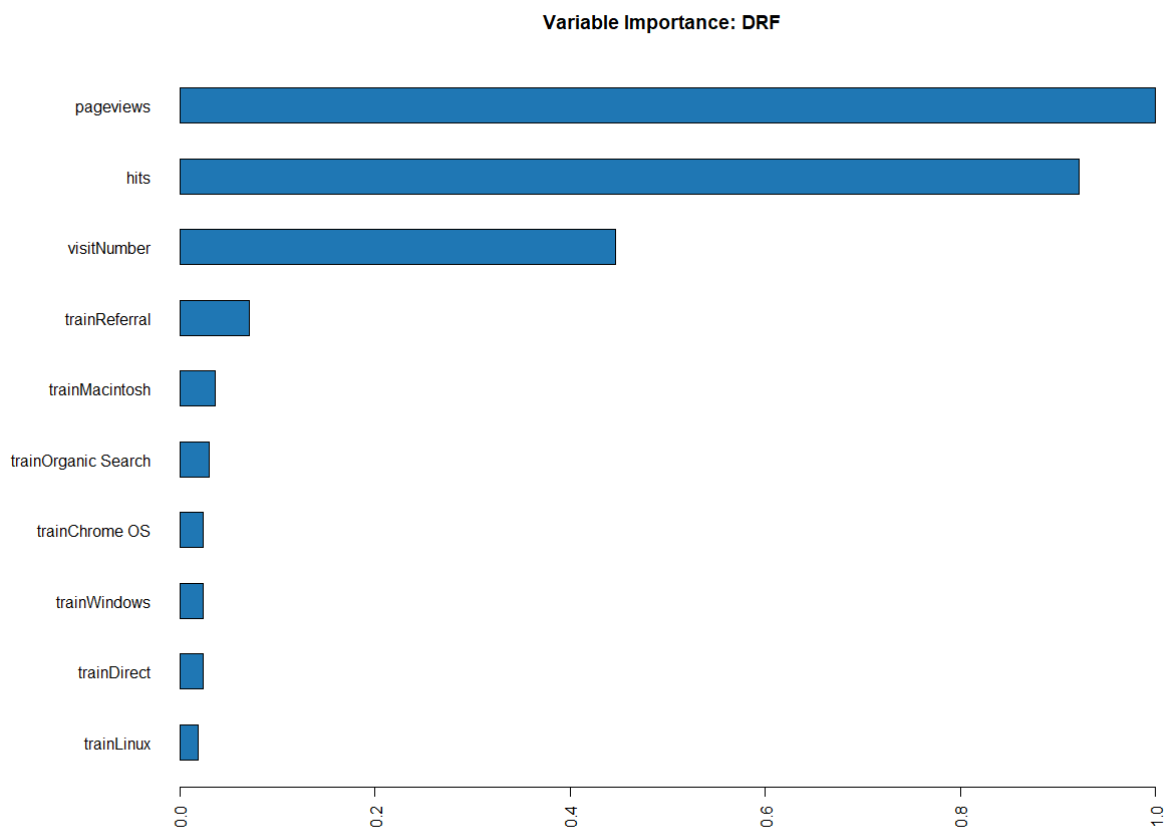


Fig: The variables in the data set ranked by level of importance (most important at the top)

For nonlinear, non-monotonic response functions, variable importance measures are often the only commonly available quantitative measure of the machine-learned relationships between independent variables and the dependent variable in a model. Variable importance measures rarely give insight into even the average direction that a variable affects a response function. They simply state the magnitude of a variable's relationship with the response as compared to other variables used in the model.

Pageviews contributes 35.03% in our model to predict the revenue. Where as *hits*, *visitNumber* and *trainReferral* together contribute to about 43.77%. These 4 variables put together contribute around 78.8% in predicting. The remaining variables contribute the outstanding 21.2% in prediction.

We submitted the submission.csv file in the competition with an RMSE of around 1.8 which was a pretty decent score. This gives us another opportunity to work on finetuning the algorithm.

We couldn't produce confusion matrix or ROC plot for our model against the testing set as this was a real time competition data. We will get to see the accuracy only with time as we had to predict the transaction revenue from Dec 1st to Jan 31st. For now, RMSE is the only parameter which can be used to measure accuracy.

	A	B	C
1		fullVisitorId	PredictedLogRevenue
2	1	259678714014	0.20799605
3	2	49363351866189	0.023409615
4	3	53049821714864	0.000702418
5	4	59488412965267	0.023409615
6	5	85840370633780	0.000727109
7	6	91131414287111	0.000702418
8	7	117255350596610	3.183040864
9	8	118334805178127	0.000702418
10	9	130646294093000	0.000727109
11	10	150005271820273	0.001956753
12	11	166374699289385	0.001956753
13	12	174453501096099	0.000702418
14	13	18122977590134	0.34655519
15	14	20731284570628	0.000702418
16	15	232022622082281	0.000702418
17	16	271086753662651	0.000702418
18	17	282648818935742	0.000727109
19	18	3038793126460	0.010067221
20	19	313524203455157	0.000727109
21	20	324924635296742	0.010939558
22	21	338548677636278	0.000727109
23	22	354865008116989	0.000702418
24	23	359974620542953	0.009401048
25	24	384434116640351	3.830080731
26	25	385653946068037	0.001316923
27	26	397214032106948	0.000702418
28	27	414003317636552	0.375947195
29	28	421850492821864	0.000702418

Fig. shows the screenshot of the submission file (first 29 customers with their ID and corresponding predicted log of transaction revenue).

Above table shows a sample of how the submission file data looks like. Column C has the log of predicted revenue for each of the unique customer (customer ID) who were in the test data file.

Recommendations

1. In case we use, h2o random forest, increasing the number of trees to a large number (does tuning by over building) and reducing the number of variables will result in a lower error.
2. Using XGBoost or deep learning algorithms to build models to predict transaction revenue would also be helpful.

REFERENCES

Web Sources

- "Google Analytics Customer Revenue Prediction | Kaggle". 2018. *Kaggle.Com*. <https://www.kaggle.com/c/ga-customer-revenue-prediction>.
- "Distributed Random Forest (DRF) — H2O 3.22.0.2 Documentation". 2018. *Docs.H2o.Ai*. <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html>.
- "How to make any plot in ggplot2?". 2018. *R-statistics.co*. <http://r-statistics.co/ggplot2-Tutorial-With-R.html>
- Cuda, Alyssa. 2018. "Everything You Need To Know About Google Analytics I WP Engine Blog". *WP Engine*. <https://wpengine.com/blog/the-ins-and-outs-of-google-analytics/>.
- Synced. "How Random Forest Algorithm Works in Machine Learning." October 24, 2017. *Medium.com*. <https://medium.com/@Synced/how-random-forest-algorithm-works-in-machine-learning-3c0fe15b6674>.
- Danny Butvinik. "----- What Is Random Forest Algorithm? *** First, Random Forest Algorithm Is a Supervised Classification Algorithm. We Can Se by Expertlytics." 2018. *Pintaram.com*. https://www.pintaram.com/u/datascience.bible/1712817124181073749_6887876413.
- Analytics Vidhya Content. "Use H2O and Data.table to Build Models on Large Data Sets in R." 2016. *Analytics Vidhya.com*. <https://www.analyticsvidhya.com/blog/2016/05/h2o-data-table-build-models-large-data-sets/>.
- Koehrsen, William. "Random Forest Simple Explanation – William Koehrsen – Medium." December 27, 2017. *Medium.com*. <https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d>.
- For JSON datatype - <https://developers.squarespace.com/what-is-json>
- 80/20 rule : https://en.wikipedia.org/wiki/Pareto_principle
- <https://8fit.com/fitness/the-8020-rule-of-fitness/>
- Random forest :<https://syncedreview.com/2017/10/24/how-random-forest-algorithm-works-in-machine-learning/>

Journals

- Ho, Tin. (1995). "Random decision forests". *Document Analysis and Recognition, International Conference on*. 1. 278 - 282 vol.1. 10.1109/ICDAR.1995.598994.
- Breiman, Leo. *Machine Learning*, 45, no. 1 (2001): 5-32. doi:10.1023/a:1010933404324.
- Abou-Warda, Horeya, Nahla A. Belal, Yasser El-Sonbaty, and Sherif Darwish. "A Random Forest Model for Mental Disorders Diagnostic Systems." *Advances in Intelligent Systems and Computing Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2016*, 2016, 670-80. doi:10.1007/978-3-319-48308-5_64.

Subject Code : BUAN 6356.503 – Business Analytics with R

PART - B

Learning Report on

GOOGLE ANALYTICS CUSTOMER REVENUE PREDICTION

(A Kaggle competition)

Predict how much GStore Customers will spend

An Individual project report,

By-

Vikas Srikanth

VXS180022@utdallas.edu

- **ABSTRACT:**

In this project (competition), we're challenged to analyse a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict future revenue per customer between December 1st 2018 to January 31st 2019.

The outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of GA data.

Our task was to build an algorithm that predicts the natural log of the sum of all transactions per customer.

Submissions are scored on the root mean squared error.

RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2},$$

where \hat{y} is the natural log of the predicted revenue for a customer and y is the natural log of the actual summed revenue value plus one.

For each fullVisitorId in the test set, we must predict the **natural log** of their total revenue in PredictedLogRevenue.

- **EXPERIENCE OF WORKING ON THE PROJECT:**

Personally, I considered doing this project as the dataset was imbalanced and it dealt with very large number of observations. This gave us a chance to write good stories through exploratory data analysis. This project was a great learning opportunity for me as I got end to end hands on experience in performing data cleaning, EDA, building a model and predicting the future revenue.

The dataset initially had 12 columns and some were in JSON format with imbalance. I had to spend good amount of time on cleaning the data (including JSON to data frame conversion). When I carefully looked at data, I observed that there were 892138 entries with zero transaction revenue and 11515 entries with transaction revenue greater than 0. I have presented good plots to explain relationship between the transaction revenue with all the other independent predictors in the group report.

Next, I had to choose an algorithm which gave me better accuracy. I chose Random Forest algorithm from h2o.ai package as it did a pretty good job of predicting the transaction revenue. I have explained the model interpretation in the group report.

- **SUCCESES:**

There were many positive outcomes from this project-

1. As I mentioned earlier, this project gave me an opportunity to apply lessons learnt in classroom on real time data.
2. Cleaned an imbalanced and different format data set to a useable data frame for prediction.
3. Studied about different algorithms that could be used. Implemented the project using Random Forest algorithm in h2o package.
4. Achieved a good and competitive RMSE score and submitted the predicted revenue in an ongoing competition on Kaggle.

(Reference: <https://www.kaggle.com/vxs180022>)

- **CHALLENGES:**

There were many challenges faced while working on this project –

1. Cleaning Data and EDA – with an imbalanced dataset like what we had, it was very difficult to produce predictions that not only looks good on the metrics but also looks like real data.
2. Kaggle Competition updated the data set.
3. Dimensionality reduction and considering the best contributors.
4. Reducing the RMSE score by choosing the right predictor variables.
5. Model Selection.

- **RECOMMENDATIONS :**

- I recommend that you use all the data to train a model that generalizes better. Another approach would be to eliminate from the training set the users who never bought, assuming that they will continue to behave in the same way, since we will be predicting the purchases of the same users in a future period of time
- A (maximum likelihood estimates for a) customers purchase in a period can be written as expected number of visits in that period * expected transaction value per visit. However, we have to work with the condition that the known customer returns at least once, so we have to work with the conditional expectation of $\max(1, \text{number of visits})$. I have not seen any kernel correcting for this either.
- First run a classifier to distinguish zero from non-zero revenue. Then run a regression on only the cases where the revenue is non-zero.

R CODE

I have included some of the packages to get desired results and get good graphs to highlight the relationships between the variables. Please note that I have not included **install.packages** in my code (please include it while running the code).

```
# LOAD THE LIBRARIES

library(data.table)
library(jsonlite)
library(readr)
library(dplyr)
library(tidyverse)
library(magrittr)
library(lubridate)
library(purrr)
library(ggplot2)
library(gridExtra)
library(countrycode)
library(highcharter)
library(zoo)
library(TTR)
library(xts)
library(digest)
library(ggExtra)
library(grid)
library(mime)
library(Biobase)
library(caret)
library(scales)
library(tibble)
library(timeDate)
library(ggthemes)
library(DataExplorer)
library(cowplot)
library(rlist)

# LOAD THE DATA SETS #

train <- read_csv("train.csv")
test <- read_csv("test.csv")
submission <- read_csv("sample_submission_v2.csv")

# MAKE A FUNCTION TO CONVERT JSON TO DF #

jsontodf <- function(col){
  list.stack(lapply(col, function(j){
    as.list(unlist(fromJSON(j)))) , fill=TRUE)
})

tr_device <- jsontodf(train$device)
tr_geoNetwork <- jsontodf(train$geoNetwork)
tr_totals <- jsontodf(train$totals)
tr_trafficSource <- jsontodf(train$trafficSource)

# FLATTENING THE JSON
```

```

flatten_json <- . %>%
  str_c(., collapse = ",") %>%
  str_c("[", ., "]") %>%
  fromJSON(flatten = T)

parse <- . %>%
  bind_cols(flatten_json(.device)) %>%
  bind_cols(flatten_json(.geoNetwork)) %>%
  bind_cols(flatten_json(.trafficSource)) %>%
  bind_cols(flatten_json(.totals)) %>%
  select(-device, -geoNetwork, -trafficSource, -totals)

test <- parse(test)
te_device <- test[,c("browser",
"browserVersion", "browserSize", "operatingSystem", "operatingSystemVersion",
"mobileDeviceModel", "mobileInputSelector", "mobileDeviceInfo",
"mobileDeviceMarketingName", "flashVersion", "language", "screenColors", "screenResolution", "deviceCategory")]
te_geoNetwork <-
test[,c("continent", "subContinent", "country", "region", "metro", "city", "cityId", "networkDomain", "latitude",
"longitude", "networkLocation")]
te_totals <- test[,c("visits", "hits", "pageviews", "bounces", "newVisits")]
te_trafficSource <-
test[,c("campaign", "source", "medium", "keyword", "adwordsClickInfo.criteriaParameters", "isTrueDirect",
"referralPath", "adwordsClickInfo.page", "adwordsClickInfo.slot", "adwordsClickInfo.gclid",
"adwordsClickInfo.adNetworkType", "adwordsClickInfo.isVideoAd", "adContent")]

#Check to see if the training and test sets have the same column names
setequal(names(tr_device), names(te_device))
setequal(names(tr_geoNetwork), names(te_geoNetwork))
setequal(names(tr_totals), names(te_totals))
setequal(names(tr_trafficSource), names(te_trafficSource))

#As expected, all are equal except for the totals - which includes the
target, transactionRevenue
#Clearly this should only appear in the training set
names(tr_totals)
names(tr_device)
names(tr_geoNetwork)
names(tr_trafficSource)
names(te_totals)

#Combine to make the full training and test sets
train <- train %>%
  cbind(tr_device, tr_geoNetwork, tr_totals, tr_trafficSource) %>%
  select(-device, -geoNetwork, -totals, -trafficSource)

test <- test %>%
  cbind(te_device, te_geoNetwork, te_totals, te_trafficSource) %>%
  select(-device, -geoNetwork, -totals, -trafficSource)

```

```

glimpse(train)
glimpse(test)

#Variable Typecasting
train$visits <- as.numeric(train$visits)
train$hits <- as.numeric(train$hits)
train$pageviews <- as.numeric(train$pageviews)
train$bounces <- as.numeric(train$bounces)
train$newVisits <- as.numeric(train$newVisits)
train$transactionRevenue <- as.numeric(train$transactionRevenue)

test$visits <- as.numeric(test$visits)
test$hits <- as.numeric(test$hits)
test$pageviews <- as.numeric(test$pageviews)
test$bounces <- as.numeric(test$bounces)
test$newVisits <- as.numeric(test$newVisits)

train$date <- as.Date(strptime(train$date, format = "%Y%m%d"))
test$date <- as.Date(strptime(test$date, format = "%Y%m%d"))

head(train,2)
#class(train)
str(train)

# GETTING RID OF COLUMNS WITH ONLY ONE UNIQUE VALUE SINCE THEY WON'T BE
USEFUL #

uniqvals <- sapply(train, n_distinct)
print(uniqvals)
todel <- names(uniqvals[uniqvals == 1])
train %<>% select(-one_of(todel))
test %<>% select(-one_of(todel))

# PLOTTING THE PERCENTAGE OF MISSING VALUES IN THE TRAINING SET #

plot_missing(train)

# PLOTTING THE TARGET VARIABLE (TRANSACTION REVENUE) FREQUENCY DISTRIBUTION
#

train %>%
  ggplot() + geom_density(aes(as.numeric(transactionRevenue))) +
  labs(x = "transactionRevenue",
       title = "Frequency Distribution of transactionRevenue") +
  theme_minimal() -> p1

train %>%
  ggplot() + geom_density(aes(log(as.numeric(transactionRevenue)))) +
  labs(x = "transactionRevenue",
       title = "Frequency Distribution of Natural Log transactionRevenue")
+
  theme_minimal() -> p2

plot_grid(p1,p2)

# PLOTTING THE TRANSACTION REVENUE FOR TRAINING AND TEST SET

t1 <- train %>% mutate(date = ymd(date),

```



```

      year_month = make_date(year(date), month(date))) %>%
group_by(year_month) %>% count() %>%
ggplot(aes(x = year_month, y = n)) +
geom_bar(stat="identity", fill="steelblue") +
labs(x = "", y = "transactions", title = "Training Set") +
theme_minimal() +
scale_x_date(labels = date_format("%Y - %m"))+
theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
geom_vline(aes(xintercept = max(year_month), colour = "red"), size = 1) +
theme(legend.position="none")

t2 <- test %>% mutate(date = ymd(date),
                      year_month = make_date(year(date), month(date))) %>%
group_by(year_month) %>% count() %>%
ggplot(aes(x = year_month, y = n)) +
geom_bar(stat="identity", fill="steelblue") +
labs(x = "", y = "transactions", title = "Test") +
theme_minimal() +
scale_x_date(labels = date_format("%Y - %m"))+
theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t1,t2)

# WE SHOULD ASSUME THAT NA IN TRANSACTION REVENUE MEANS THAT TRANSACTION
# WAS NOT MADE AND SO #
# WE SHOULD CONVERT THE NA TO 0 #

tr <- train$transactionRevenue
summary(tr)
tr[is.na(tr)] <- 0
summary(tr)

# VISITS TO SITE VS TRANSACTION REVENUE OVER TIME #

t3 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(date) %>%
  summarize(visits = n()) %>%
  ungroup() %>%
  ggplot(aes(x = date, y = visits)) +
  geom_line() +
  geom_smooth() +
  labs(x = "") +
  theme_minimal()

t4 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(date) %>%
  summarize(revenue = mean(value)) %>%
  ungroup() %>%
  ggplot(aes(x = date, y = revenue)) +
  geom_line() +
  geom_smooth() +
  labs(x = "") +
  theme_minimal()

grid.arrange(t3,t4)

# NOW TO PLOT TRANSACTION REVENUE AND FREQUENCY OF USE OF DIFFERENT
# BROWSERS #

```

```

t5 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(browser) %>%
  summarize(visits = n()) %>%
  top_n(9, wt=visits) %>%
  ggplot(aes(x = browser, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t6<- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(browser) %>%
  summarise(revenue = sum(value)) %>%
  top_n(9, wt=revenue) %>%
  ggplot(aes(x = browser, y = revenue)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

grid.arrange(t5,t6)

# TO GET TRANSACTION REVENUE AND FREQUENCY OF DIFFERENT CHANNEL GROUPINGS #

t7 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(channelGrouping) %>%
  summarize(visits = n()) %>%
  ggplot(aes(x = channelGrouping, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t8 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(channelGrouping) %>%
  summarise(revenue = sum(value)) %>%
  ggplot(aes(x = channelGrouping, y = revenue)) +
  geom_bar(fill="#6666c2", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t7,t8)

# TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR OPERATING SYSTEMS #

t9 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(operatingSystem) %>%
  summarize(visits = n()) %>%
  ggplot(aes(x = operatingSystem, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

```

```

t10 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(operatingSystem) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = operatingSystem, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t9, t10)

```

TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT DEVICES

```

t11 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(deviceCategory) %>%
  summarize(visits = n()) %>%
  top_n(10, wt=visits) %>%
  ggplot(aes(x = deviceCategory, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t12 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(deviceCategory) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = deviceCategory, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t11, t12)

```

TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT COUNTRIES

```

t13 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(country) %>%
  summarize(visits = n()) %>%
  top_n(10, wt=visits) %>%
  ggplot(aes(x = country, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t14 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(country) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = country, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +

```

```

theme_minimal() +
scale_y_continuous(labels = comma) +
theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t13,t14)

# TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT CONTINENT #

t15 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(continent) %>%
  summarize(visits = n()) %>%
  top_n(10,wt=visits) %>%
  ggplot(aes(x = continent, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t16 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(continent) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10,wt=revenue) %>%
  ggplot(aes(x = continent, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t15,t16)

# TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT SUBCONTINENT #

t17 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(subContinent) %>%
  summarize(visits = n()) %>%
  top_n(10,wt=visits) %>%
  ggplot(aes(x = subContinent, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t18 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(subContinent) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10,wt=revenue) %>%
  ggplot(aes(x = subContinent, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t17,t18)

```

```
# TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT REGION #
```

```
t19 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(region) %>%
  summarize(visits = n()) %>%
  top_n(10, wt=visits) %>%
  ggplot(aes(x = region, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t20 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(region) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = region, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t19, t20)
```

```
# TO PLOT TRANSACTION REVENUE AND FREQUENCY FOR DIFFERENT CAMPAIGN #
```

```
t21 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(campaign) %>%
  summarize(visits = n()) %>%
  top_n(10, wt=visits) %>%
  ggplot(aes(x = campaign, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme_minimal()

t22 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(campaign) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = campaign, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t21, t22)
```

```
#Date Wise Aggregation
```

```
date_wise <- train %>%
  group_by(date) %>%
  summarise(daily_visits = sum(visits, na.rm = TRUE),
            daily_hits = sum(hits, na.rm = TRUE),
```

```

        daily_pageviews = sum(pageviews, na.rm = TRUE),
        daily_bounces = sum(bounces, na.rm = TRUE),
        daily_newVisits = sum(newVisits, na.rm = TRUE),
        daily_transactionRevenue = sum(transactionRevenue, na.rm =TRUE)
    )

ggplot(date_wise,aes(date,daily_visits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Visits Trend",
        x = "Date",
        y = "Daily Visits") +
  geom_smooth() -> p1

ggplot(date_wise,aes(date,daily_hits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Hits Trend",
        x = "Date",
        y = "Daily Hits") +
  geom_smooth() -> p2

ggplot(date_wise,aes(date,daily_newVisits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily new Visits Trend",
        x = "Date",
        y = "Daily new Visits") +
  geom_smooth() -> p3

ggplot(date_wise,aes(date,daily_bounces)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Bounces Trend",
        x = "Date",
        y = "Daily Bounces") +
  geom_smooth() -> p4

plot_grid(p1,p2,p3,p4, ncol = 2)

date_wise$daily_bouncerate <- (date_wise$daily_bounces /
date_wise$daily_visits) * 100

ggplot(date_wise,aes(date,daily_bouncerate)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Bounce Rate Trend",
        subtitle = "Bounce Rate = Bounces / Visits ",
        x = "Date",
        y = "Daily Bounce Rate in %") +
  geom_smooth()

ggplot(date_wise,aes(date,daily_transactionRevenue)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Transaction Revenue Trend",
        #subtitle = "Bounce Rate = Bounces / Visits ",
        x = "Date",
        y = "Daily Transaction Revenue") +
  geom_smooth() -> p1

ggplot(date_wise,aes(date,log(daily_transactionRevenue))) + geom_line() +
  theme_economist() +
  labs(title = "Daily Log Transaction Revenue Trend",
        #subtitle = "Bounce Rate = Bounces / Visits ",
        x = "Date",
        y = "Daily Log Transaction Revenue") +

```

```

geom_smooth() -> p2

plot_grid(p1,p2)

#Now, Let's replicate the same for Test Data
date_wise_test <- test %>%
  group_by(date) %>%
  summarise(daily_visits = sum(visits, na.rm = TRUE),
            daily_hits = sum(hits, na.rm = TRUE),
            daily_pageviews = sum(pageviews, na.rm = TRUE),
            daily_bounces = sum(bounces, na.rm = TRUE),
            daily_newVisits = sum(newVisits, na.rm = TRUE)
  )
ggplot(date_wise_test,aes(date,daily_visits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Visits Trend",
       x = "Date",
       y = "Daily Visits") +
  geom_smooth() -> p1
ggplot(date_wise_test,aes(date,daily_hits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Hits Trend",
       x = "Date",
       y = "Daily Hits") +
  geom_smooth() -> p2
ggplot(date_wise_test,aes(date,daily_newVisits)) + geom_line() +
  theme_economist() +
  labs(title = "Daily new Visits Trend",
       x = "Date",
       y = "Daily new Visits") +
  geom_smooth() -> p3
ggplot(date_wise_test,aes(date,daily_bounces)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Bounces Trend",
       x = "Date",
       y = "Daily Bounces") +
  geom_smooth() -> p4
plot_grid(p1,p2,p3,p4, ncol = 2)

date_wise_test$daily_bouncerate <- (date_wise_test$daily_bounces /
date_wise_test$daily_visits) * 100
ggplot(date_wise_test,aes(date,daily_bouncerate)) + geom_line() +
  theme_economist() +
  labs(title = "Daily Bounce Rate Trend",
       subtitle = "Bounce Rate = Bounces / Visits ",
       x = "Date",
       y = "Daily Bounce Rate in %") +
  geom_smooth()

# TO PLOT DIFFERENT TRANSACTION REVENUE AND FREQUENCY FOR ISMOBILE #

t11 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(isMobile) %>%
  summarize(visits = n()) %>%
  ggplot(aes(x = isMobile, y = visits)) +
  geom_bar(fill="#6666c3", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

```

```

t12 <- train %>%
  bind_cols(as_tibble(tr)) %>%
  group_by(isMobile) %>%
  summarise(revenue = sum(value)) %>%
  top_n(10, wt=revenue) %>%
  ggplot(aes(x = isMobile, y = revenue)) +
  geom_bar(fill="#6666c1", stat="identity") +
  theme_minimal() +
  scale_y_continuous(labels = comma) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

grid.arrange(t11, t12)

# _____ # _____ #

#Let's move to Log as we have to predict Log of Revenue and some naive
feature engineering

train$transactionRevenue[is.na(train$transactionRevenue)] <- 0
train$log_revenue <- loglp(train$transactionRevenue)

#feature engineering
train$dayofweek <- weekdays(train$date)
train$month <- months.Date(train$date)
train$operatingSystem <- ifelse(train$operatingSystem %in%
c("Android", "Macintosh", "Linux",
"Windows", "iOS", "Chrome OS"),
train$operatingSystem,
"Others")

train$operatingSystem <- ifelse(train$operatingSystem %in%
c("Android", "Macintosh", "Linux",
"Windows", "iOS", "Chrome OS"),
train$operatingSystem,
"Others")

test$dayofweek <- weekdays(test$date)
test$month <- months.Date(test$date)

test$operatingSystem <- ifelse(test$operatingSystem %in%
c("Android", "Macintosh", "Linux",
"Windows", "iOS", "Chrome OS"),
test$operatingSystem,
"Others")

#Create Dummy variables -

install.packages("dummies")
library(dummies)

train <- cbind(train, dummy(train$channelGrouping), dummy(train$isMobile),
dummy(train$region),
dummy(train$operatingSystem))
#train$isMobile <- as.numeric(train$isMobile)

```



```

test$dayofweek <- weekdays(test$date)
test$month <- months.Date(test$date)

test$operatingSystem <- ifelse(test$operatingSystem %in%
c("Android", "Macintosh", "Linux",
"Windows", "iOS", "Chrome OS"),
test$operatingSystem,
"Others")

test <- cbind(test, dummy(test$channelGrouping), dummy(test$isMobile),
dummy(test$region),
dummy(test$operatingSystem))

library(h2o)
#h2o.init(nthreads=-1,max_mem_size='10G')
h2o.init(nthreads = -1, max_mem_size = '10g', ip = 'localhost', port =
54321)
#h2o.init()

x <- as.h2o(train)

features <- colnames(train)[!(colnames(train) %in% c("date",
"visitStartTime",
"visitEndTime",
"fullVisitorId",

"sessionId",
"visitId",
"flashVersion", "browserSize", "transactionRevenue",
#"revenue_yes",
# "channelGrouping",
"log_revenue"))]

model <- h2o.randomForest(x=features,
y="log_revenue",
ntrees = 50,
max_depth = 30,
nbins_cats = 100,
training_frame=x)

summary(model)

# To plot features by important
h2o.varimp_plot(model)

# To create the submission file

submission <- data.frame(fullVisitorId=test$fullVisitorId,
PredictedLogRevenue=predictions)
sample_sub <- read_csv("sample_submission.csv")
names(submission) <- names(sample_sub)

submission <- submission %>% group_by(fullVisitorId) %>%
summarise(PredictedLogRevenue = sum(PredictedLogRevenue))

```

```
nrow(submission) == nrow(sample_sub)
write.csv(submission, "ga_pred.csv", row.names=T) #Writing it into a new CSV
file
class(submission)

table(submission$fullVisitorId==sample_sub$fullVisitorId)
```

```
# EXTRA INFORMATION #
```

```
# I have tried to show the frequency distribution of continuous and
discrete variables that were part of the data frame (it's not included in
the main code)
```

```
options(repr.plot.width=12)
```

```
plot_density(train, title = "Frequency Distribution Analysis")
```

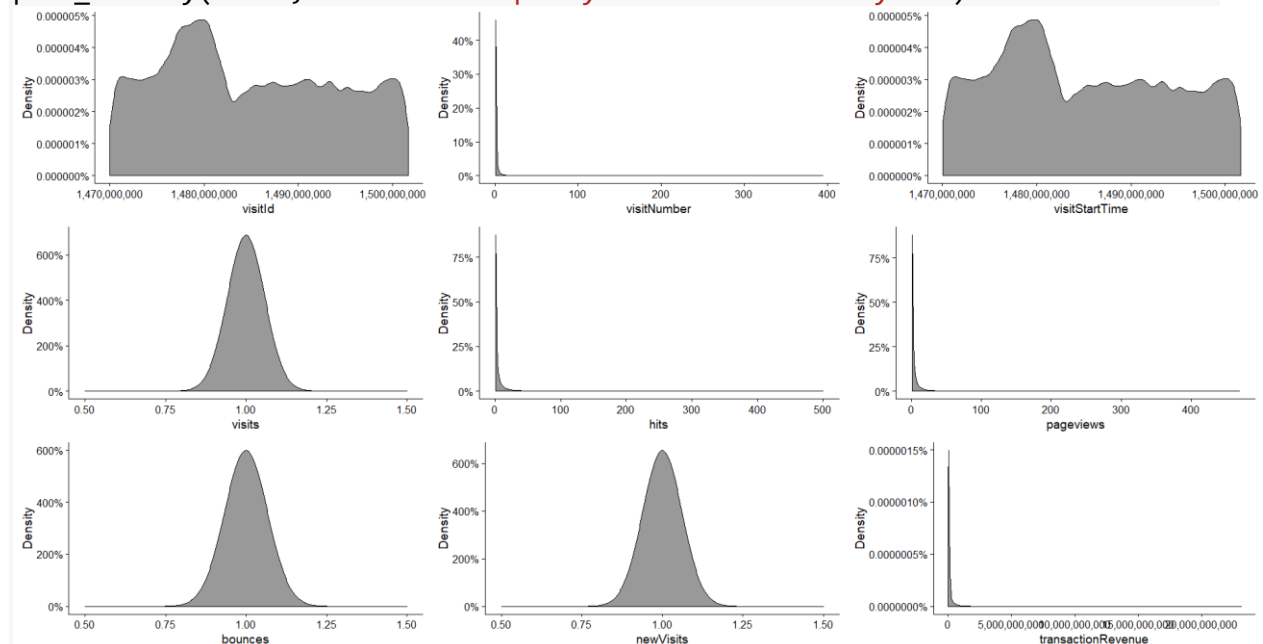


Fig. shows the frequency distribution analysis of visited, visitNumber, visitStartTime, visits, hits, pageViews, bounces, newVisits and transaction revenue

Target Variable Frequency Distribution

```
In [11]:
```

```
library(cowplot)
```

```
train %>%
ggplot() + geom_density(aes(as.numeric(transactionRevenue))) +
labs(x = "transactionRevenue",
title = "Frequency Distribution of transactionRevenue") +
theme_minimal() -> p1
```

```
train %>%
ggplot() + geom_density(aes(log(as.numeric(transactionRevenue)))) +
labs(x = "transactionRevenue",
title = "Frequency Distribution of Natural Log transactionRevenue") +
theme_minimal() -> p2
```

```
plot_grid(p1,p2)
```

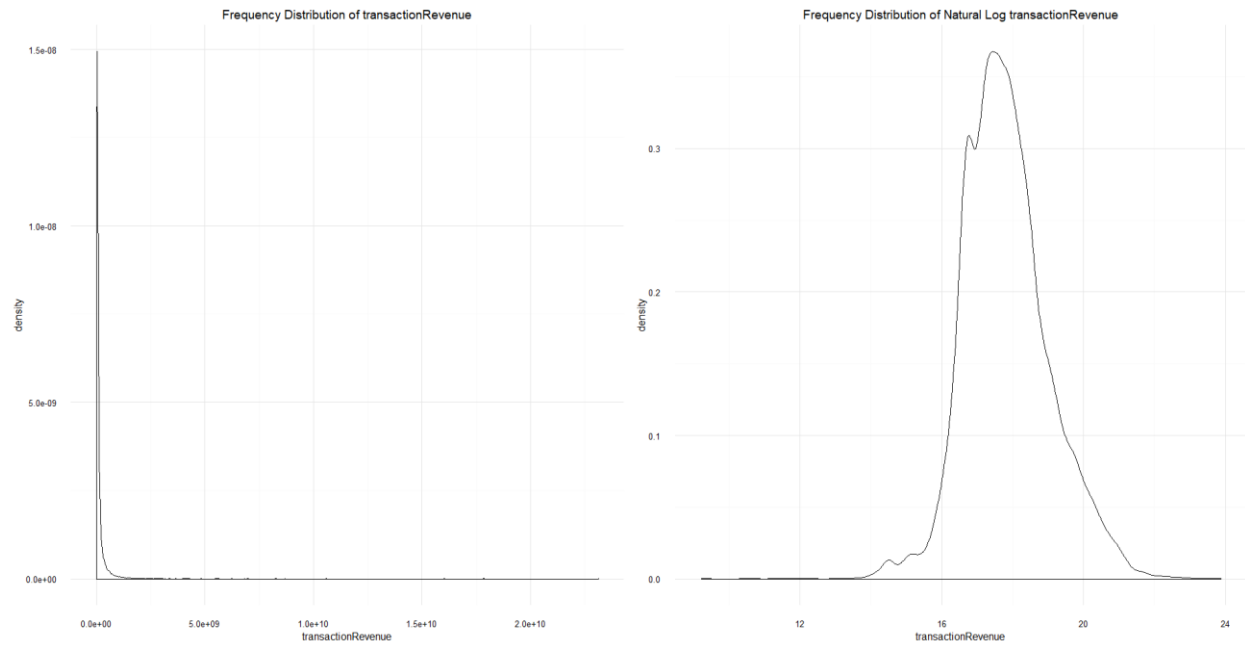


Fig – displays the frequency distribution of continuous variables (analysis)–Variables: transaction revenue and natural_log(transaction revenue)

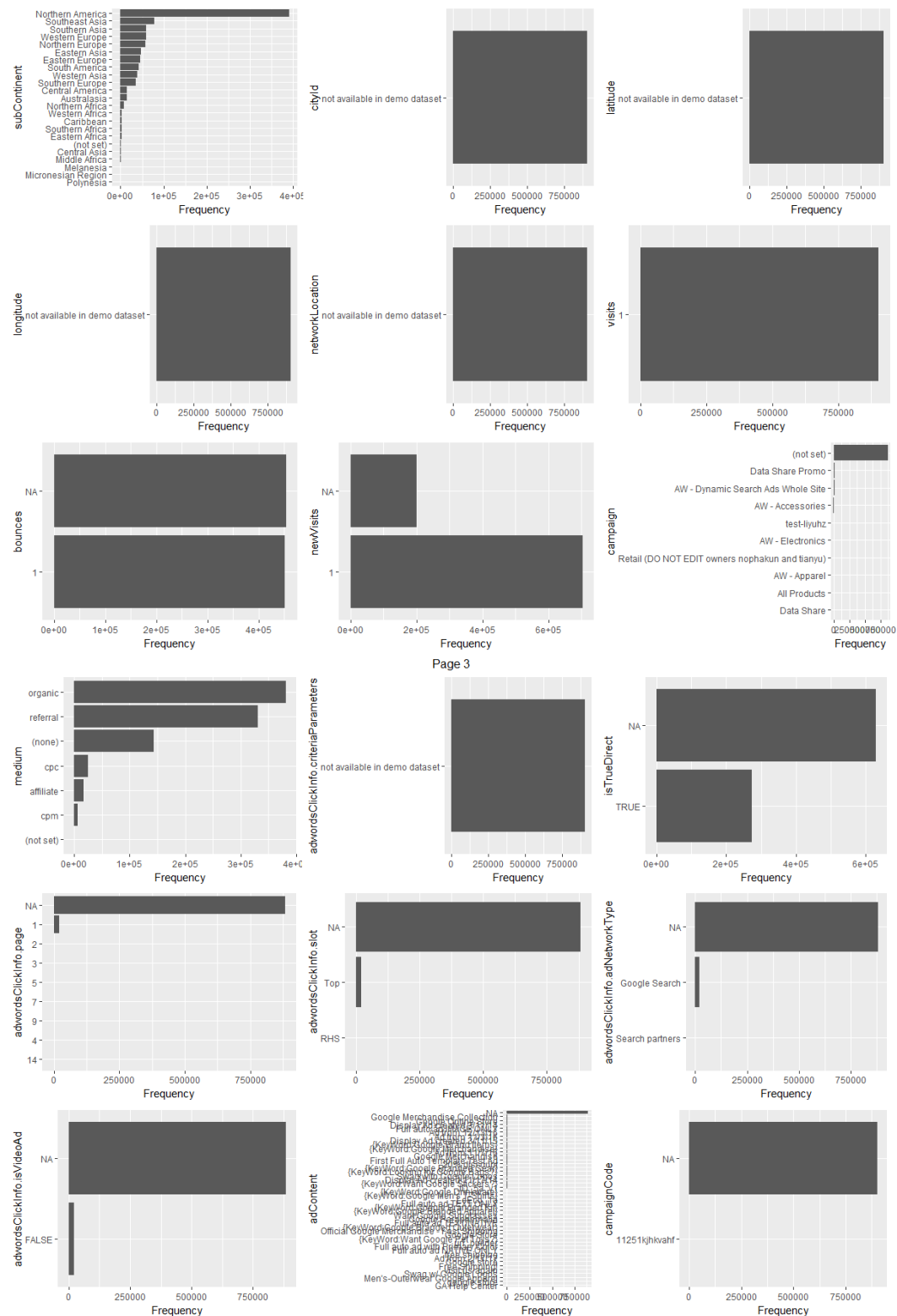
FREQUENCY DISTRIBUTION – DISCRETE VARIABLES (ANALYSIS)

#Discrete Variable Analysis

```
options(repr.plot.width=12)  
plot_bar(train)
```

Organic Search -





Page 4

Figures shows the frequency distribution analysis plots for all the discrete variables in the data.