Springboard DSC Capstone Project 2

Google Analytics Customer Revenue Prediction

Neha Jain Surana

Nov 2018

Introduction

Client

Data and its acquisition

Data Wrangling

- Parse Json columns
- Delete non useful columns
- Fill up missing values
- Check validity of geo related data

Data Exploration

- Analyze the target variable
- Analyze the visitor Ids
- Analyze the date and visitStartTime
- Analyze visitNumber - visit number of a visitor
- Analyze channelGrouping
- Analyze device fields
- Analyze geo fields
- Analyze new visits
- Analyze hits
- Statistical Inference

Data Modeling

- Theory and Approach followed
- Overfitting and Underfitting
- Model evaluation criteria
- Data and code organization
- Data preparation
- First (Baseline) Model
  - Predictions at human level
  - First baseline model using Decision Tree
  - Light GBM
- Feature Engineering
- Modeling using new Features

Model Comparisons

Recommendations for the client

Conclusions and Future Analysis

References

Introduction

The primary purpose of any business is to make profits for its owners and stakeholders. Also, the 80/20 rule has proven true for many businesses–only a small percentage of customers produce most of the revenue.

For an e-store like Google merchandise store, it is nothing different. As such, marketing teams are challenged to make appropriate investments in promotional strategies. But for that, they need as accurate a prediction as possible, of the revenue per customer.

Although, the main prediction is not for future, i.e. we predict revenues for the visits already done. But there might be a second phase of the project where we predict for future dates.

Client

Here the client is the company Google and its specific e-store called Google Merchandise Store (GStore). We hope to provide highly accurate revenue predictions per customer visiting the GStore. The outcome should be more actionable operational changes and a better use of marketing budgets. Also, it would be serve as an evidence for those companies who choose to use data analysis on top of GA data, because the current data comes from the same.

Data and its acquisition

The data is provided by the client in the form of csv file. There is some additional test data that can be used to evaluate and report the algorithm success. The main data file has around 900 thousand rows corresponding to each cusomter visit to the GStore. There are multiple columns which contain JSON blobs of varying depth. In one of those JSON columns, totals, the sub-column transactionRevenue contains the revenue information we are trying to predict.

Loading the data will not be very simple because the json blobs need to be parsed. I am using the read_csv function provided by the python-pandas library to load the data into datafrme. Json converters are used to parse the Json fields. The result is a table like structure that can be used for processing.

Some of the 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).
- **hits** - This row and nested fields are populated for any and all types of hits. Provides a record of all page visits.
- **customDimensions** - This section contains any user-level or session-level custom dimensions that are set for a session. This is a repeated field and has an entry for each dimension that is set.
- **totals** - This set of columns mostly includes high-level aggregate data.

Data Wrangling

Metadata about number of unique and null values for each column, helps to decide what clean up and preprocessing of the data is needed.

Parse Json columns

Columns ('device', 'geoNetwork', 'totals', 'trafficSource') were found to be json blobs. Parsed these columns and added them back to the train/test data. Train data has 55 columns and 903653 rows. The data types are: bool(1), int64(4), object(50)

Delete non useful columns

Columns with 1 unique value and zero null value are not useful in any way and we delete them.

	data type	unique values	null values
socialEngagementType	object	1	0
device.browserSize	object	1	0
device.browserVersion	object	1	0
device.flashVersion	object	1	0
device.language	object	1	0
device.mobileDeviceBranding	object	1	0
device.mobileDeviceInfo	object	1	0
device.mobileDeviceMarketingName	object	1	0
device.mobileDeviceModel	object	1	0
device.mobileInputSelector	object	1	0
device.operatingSystemVersion	object	1	0
device.screenColors	object	1	0
device.screenResolution	object	1	0
geoNetwork.cityId	object	1	0
geoNetwork.latitude	object	1	0
geoNetwork.longitude	object	1	0
geoNetwork.networkLocation	object	1	0
totals.visits	object	1	0
trafficSource.adwordsClickInfo.criteriaParameters	object	1	0

Fill up missing values

The columns having only one unique value other than nulls, can be treated as boolean. The null can be treated as True or False depending on the other value present. But from the perspective of fitting the model it doesn't matter.

```
Column                                  : Old values            : New value
s
totals.bounces                          : ['1' nan]             : [1 0]
totals.newVisits                        : ['1' nan]             : [1 0]
trafficSource.adwordsClickInfo.isVideoAd : [nan False]          : [1 0]
trafficSource.campaignCode              : [nan '11251kjhkvahf'] : [1 0]
trafficSource.isTrueDirect              : [nan True]            : [0 1]
```

The columns having multiple values and nulls, have to filled up for missing values.

```
Column                                      : Unique values  : Null values

totals.pageviews                            : 213            : 100
(Very few null values. null values might indicate no page view. so fill up m
issing by '0's.)

totals.transactionRevenue                   : 5332           : 892138
(Huge number of null values. Null revenue means '0', so replace by it.)

trafficSource.adwordsClickInfo.page         : 8              : 882193
(Huge number of null values. So might not be important variable. It's an in
t, so convert
it to int data type. Filling up null with '0's.)

trafficSource.adContent                     : 44             : 892707
trafficSource.adwordsClickInfo.adNetworkType : 2             : 882193
trafficSource.adwordsClickInfo.slot         : 2              : 882193
trafficSource.keyword                       : 3659           : 502929
trafficSource.referralPath                  : 1475           : 572712
(Huge number of null values. So might not be important variable. It's a stri
ng value. Filling
up nulls by 'Unknown')

trafficSource.adwordsClickInfo.gclId        : 17774          : 882092
(It's alpha numeric strings. Filling up nulls by 'Unknown')
```

Check validity of geo related data and clean up if needed

- Find out unique geo combinations.
- There are values like '(not set)' and 'not available in demo dataset'. We can't do much about them.
- Divided the 6 geo columns into 2 parts. The unique combinations of country, sub-continent, continent seems to be fine. There are 222 such rows. Eyeballing them also looks good.
- The other unique combinations for city, metro, region seems to have some invalid data. Same city is associated with different regions. This can happen when same city name occurs in two places. So we further valide such combinations.
- We found that many of such combinations don't have valid match with country. We delete 3765 such rows. For e.g.

geoNetwork.city	geoNetwork.metro	geoNetwork.region	geoNetwork.country	geoNetwork.subContinent	geoNetwork.continent	counts
Abu Dhabi	(not set)	Abu Dhabi	United States	Northern America	Americas	1
Amsterdam	(not set)	North Holland	United States	Northern America	Americas	7
Amsterdam	(not set)	North Holland	Ireland	Northern Europe	Europe	2
Amsterdam	(not set)	North Holland	United Kingdom	Northern Europe	Europe	2
Amsterdam	(not set)	North Holland	Norway	Northern Europe	Europe	1
Amsterdam	(not set)	North Holland	Germany	Western Europe	Europe	1
Ankara	(not set)	Ankara	Ireland	Northern Europe	Europe	1
Ann Arbor	Detroit MI	Michigan	Ireland	Northern Europe	Europe	2
Ann Arbor	Detroit MI	Michigan	France	Western Europe	Europe	1
Ashburn	Washington DC (Hagerstown MD)	Virginia	China	Eastern Asia	Asia	2
Atlanta	Atlanta GA	Georgia	Puerto Rico	Caribbean	Americas	2

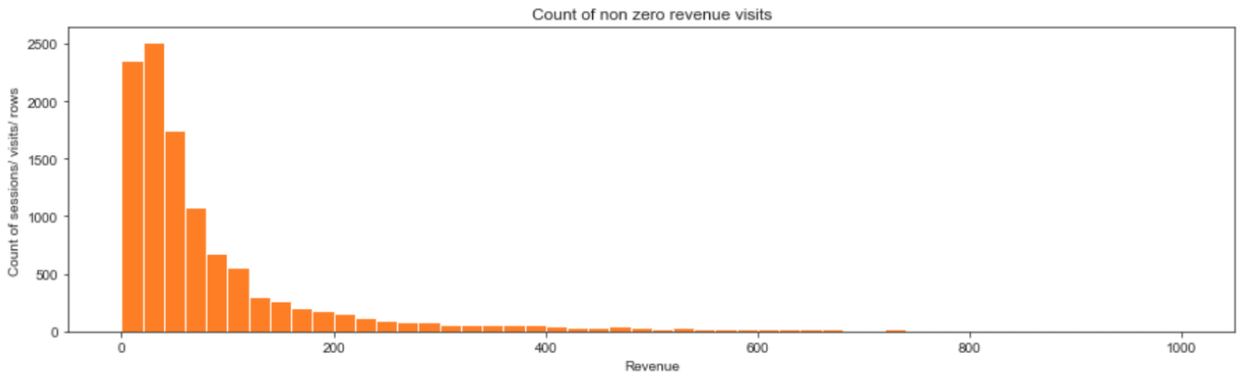
Data Exploration

Let us explore all the fields and their relation to the target variable.

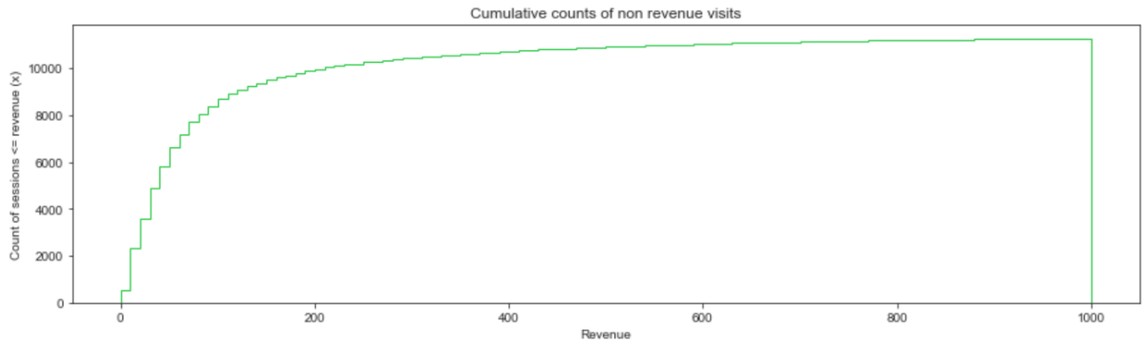
Analyze the target variable

The given transaction revenues data is actually the original revenue multiplied by a million. So actual revenues are millionth of the given numbers. The reason behind is to keep the precision of the numbes upto certain length, so that the values are not rounded while being transformed in multiple systems.

The percent of revenue generating sessions is only 1.27%. And that means most of the visits don't generate any revenue.



This gives a rough picture that there are large number of small valued transactions and very few large valued.



This picture better shows that most of the non zero transactions have revenue of less than 200. In fact we see below that 90% of all revenues are less than 279. The median revenue is around $50.

Few things noticed by comparing revenue generating visits vs no-revenue visits. So these all could serve as good features as they have differences in values for revenue vs no-revenue visits.

- Bounces are all zeros for revenue generating visits
- Campaign code are all ones
- Maximum visitNumber is 315 vs 395
- Minimum pageview is 2 vs 0
- Maximum pageviews is 469 vs 429
- Maximum adwordclickinfo page is 1 vs 14

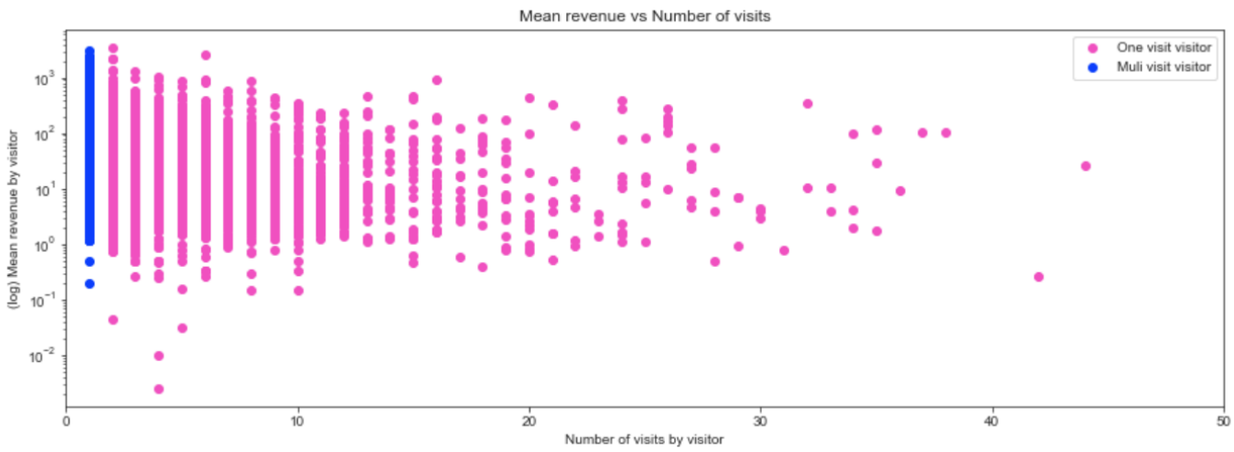
Analyze the visitorIds

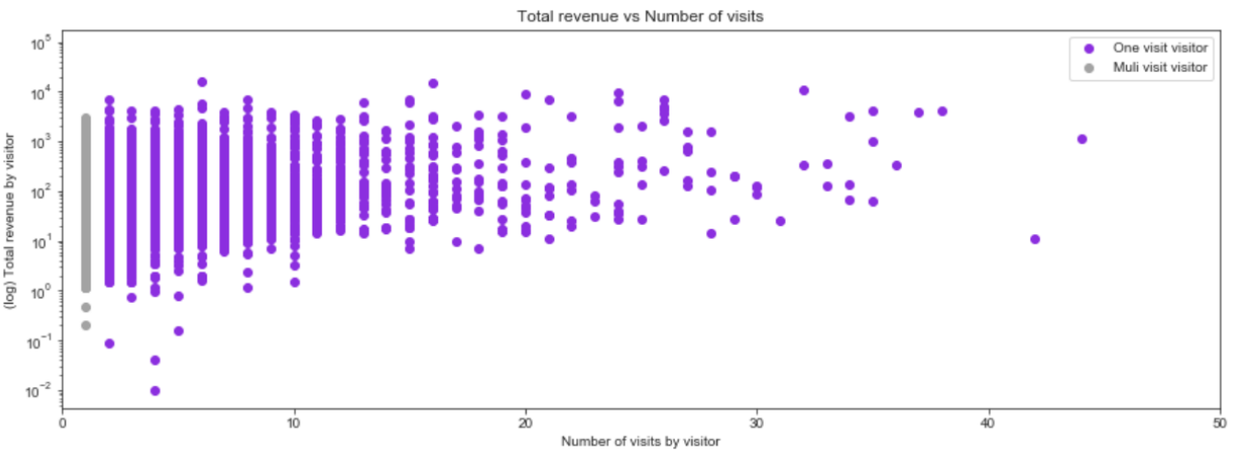
There are 714167 unique visitor ids for the total of 903653 visits. This tells that there will be huge number of visitors who visit the store only once.

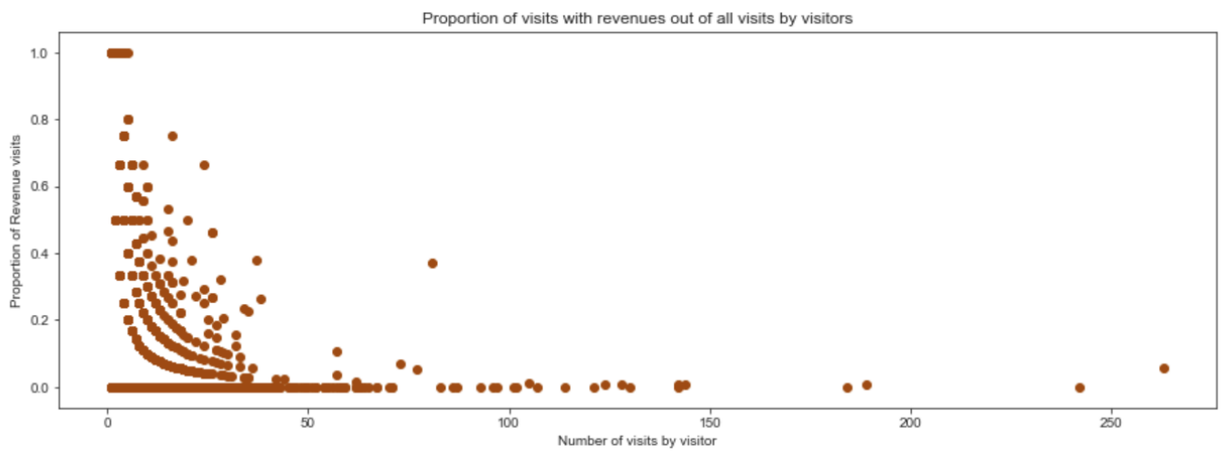
Percentage of unique users for:

- revenue visits 88.25992150021806
- non revenue visits 80.88129190712083
- all visits 80.25376491296694

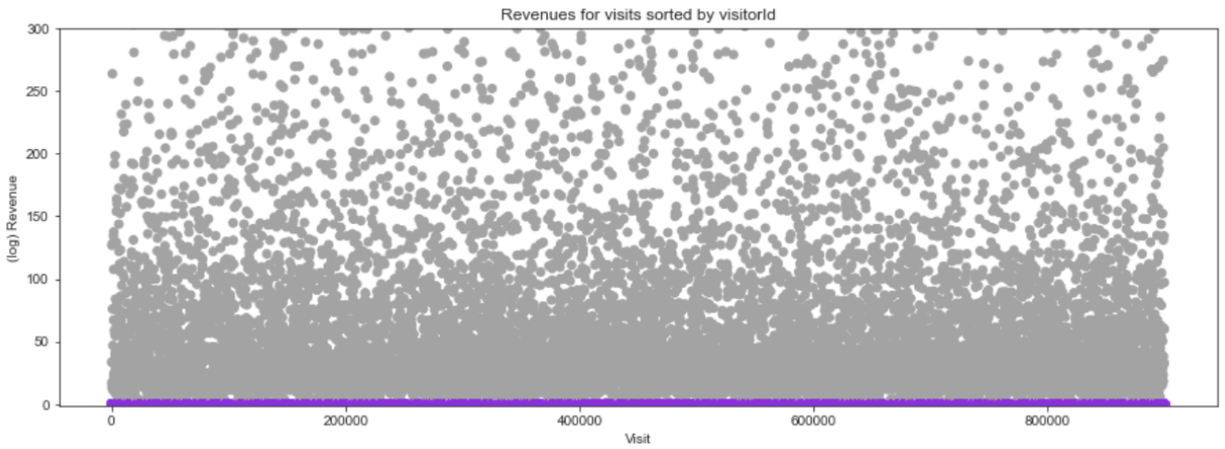
Percent for unique visitors is slightly more for revenue visits as compared to non revenue visits which is little weird because we would expect more visits by some visitor when they buy something.







Above all analysis means that when the number of visits by a visitor increases it necessarily doesn't mean that more revenue will be generated



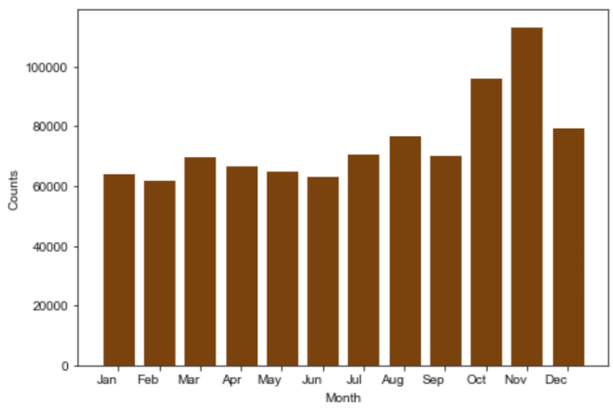
Understanding about visitorIds and revenues

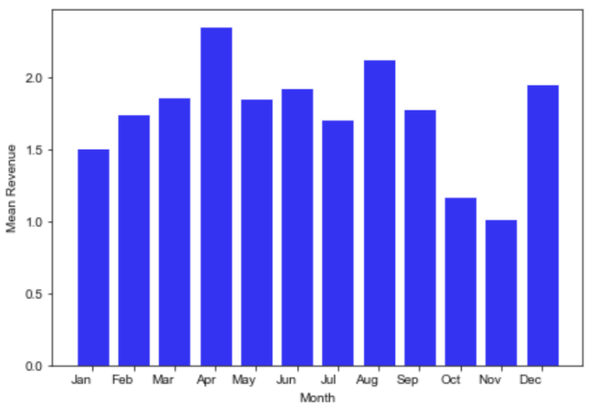
- Revenues are evenly spread for all the visits
- Proportion of revenue visits decreases as the number of visits by the visitor increases. That is, if a visitor visits the site multiple times, it doesn't mean he/she is buying more and more.
- Total revenue increases slights if the number of visits increase
- Some visitors have generated revenue multiple times in multiple visits. Maximum is 30 revenue transactions by a visitor.

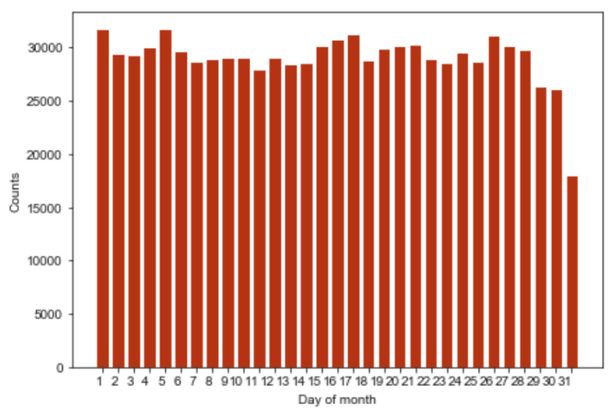
Analyze the date and visitStartTime

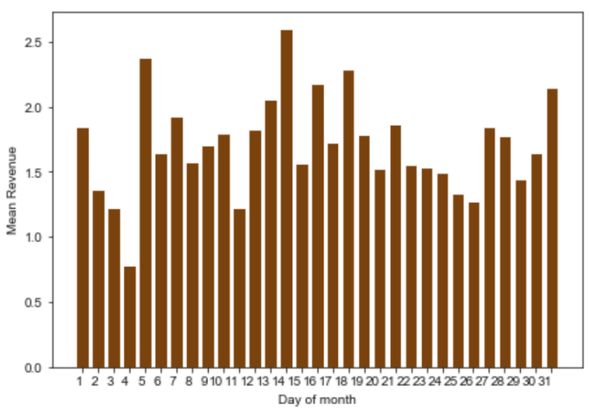
We parse the Date and VisitStartTime into following components and visualize them with respect to the revenue. Also, show below are some interesting charts.

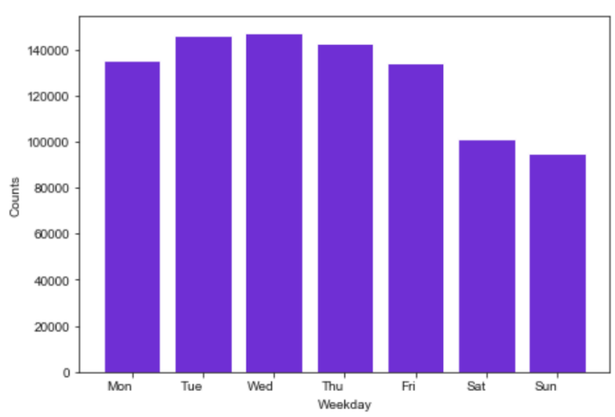
- Year
- Month
- Day (of the month)
- Day (of the year, Jan 1 is day 1)
- Weekday
- Hour
- Local hour (local hour based on timezone)

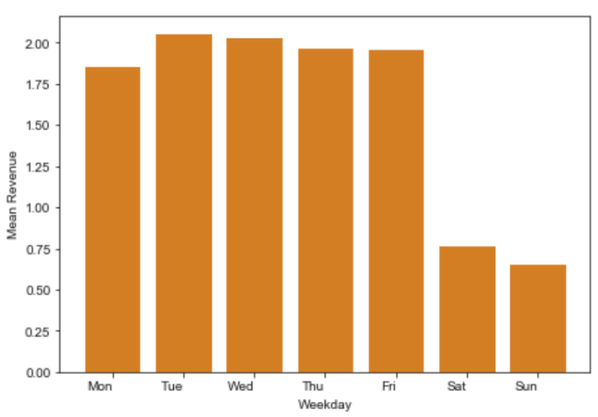


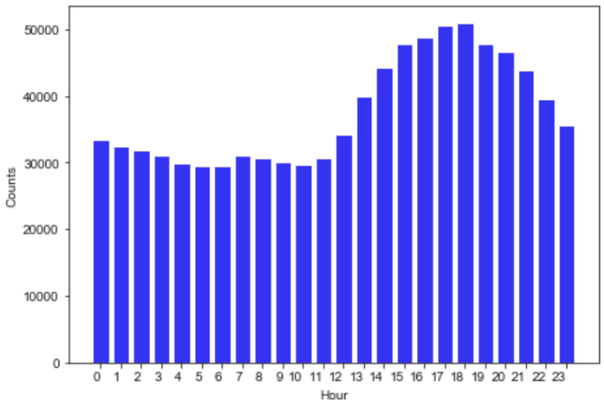


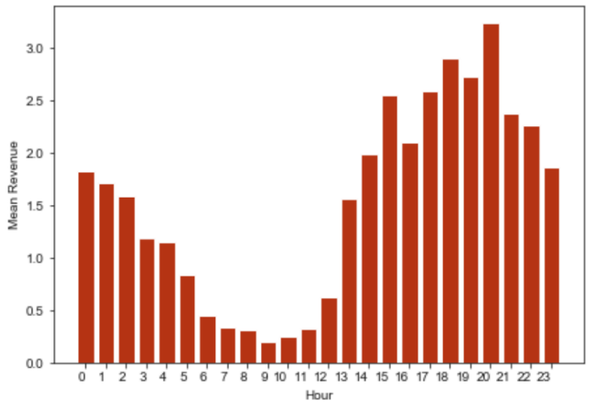


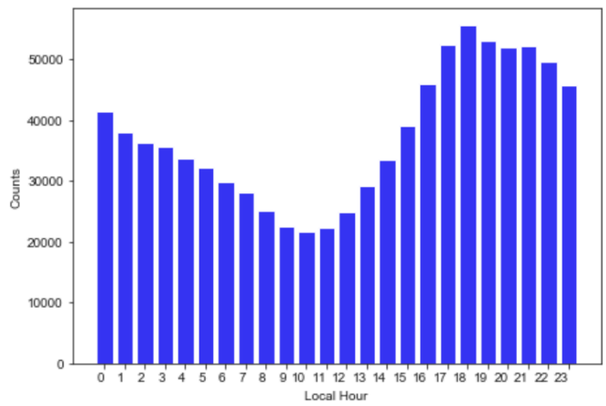


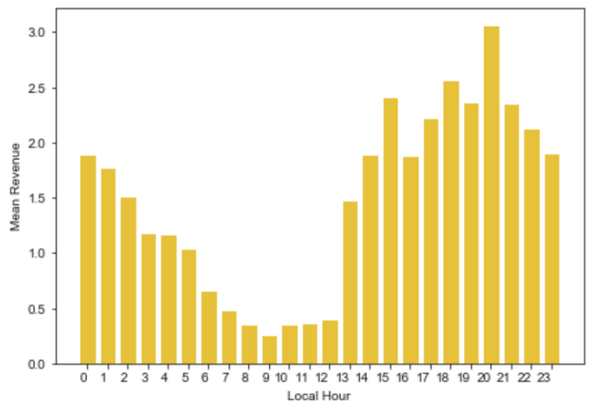


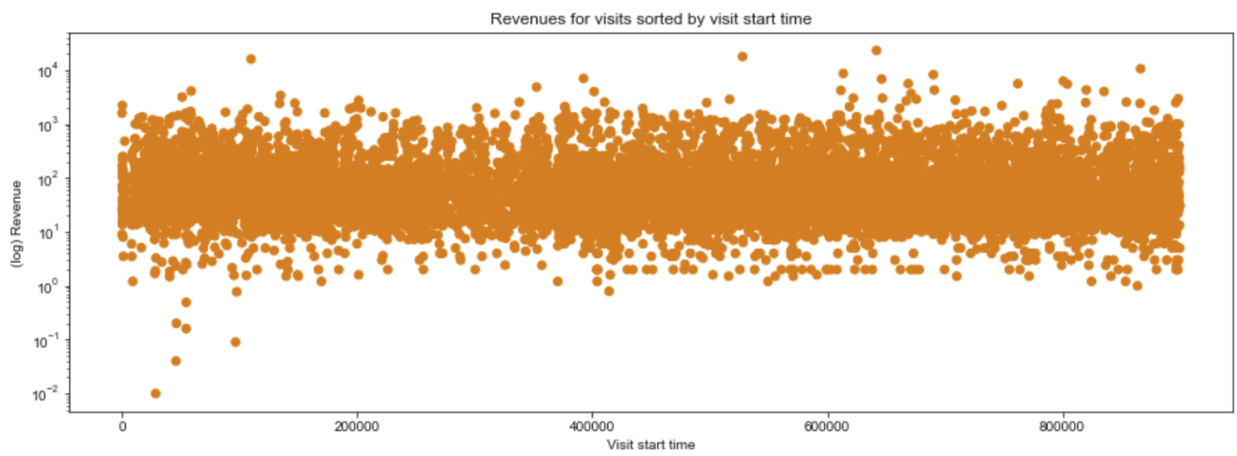


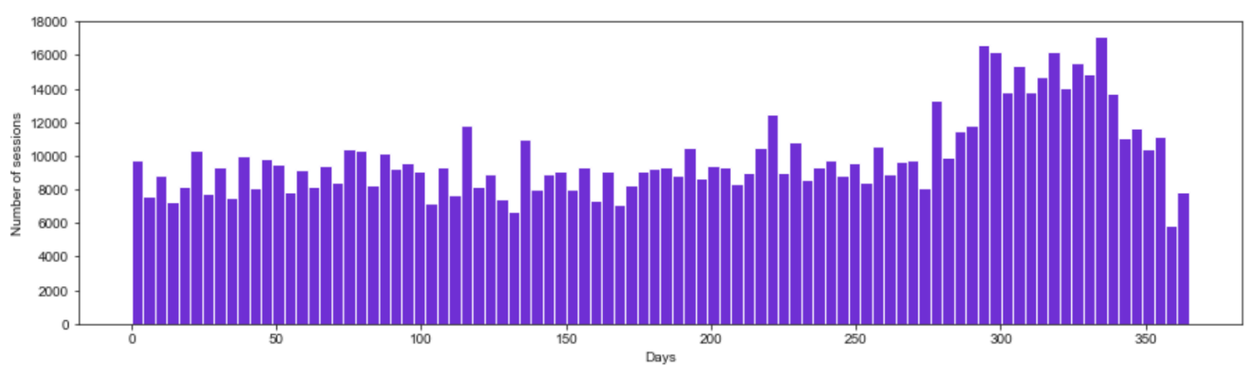

Counts
Hour


Mean Revenue
Hour


Counts
Local Hour


Mean Revenue
Local Hour


Revenues for visits sorted by visit start time
(log) Revenue
Visit start time


Number of sessions
Days

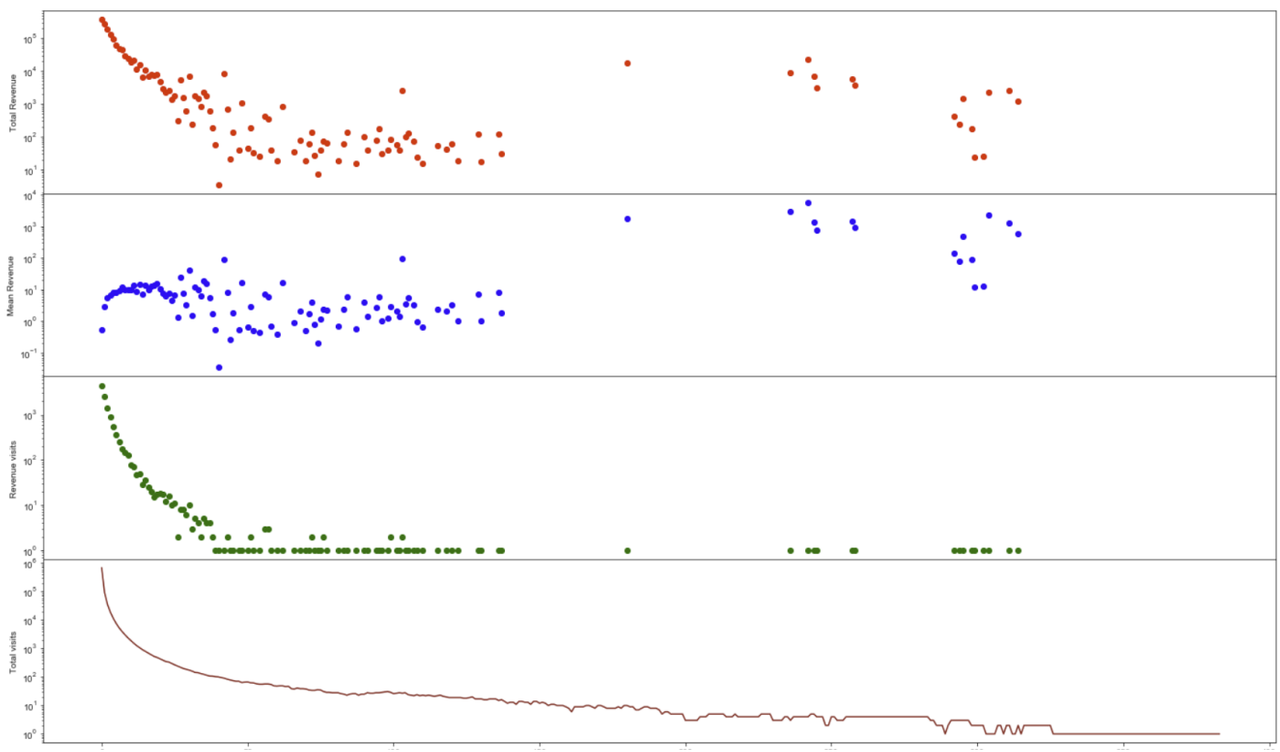
Understanding about date and visitStartTime

- The visit counts increase in the festival months. But the mean revenue fluctuates in those months.
- Both the visit counts and mean revenues drop on weekends. May be people are busy doing other stuff then buying things online.
- Assuming large portion of visitors are from North America, the 24 hour pattern shows mean revenue increase in day time, maximizes around 8pm and then drops.
- We dont see any pattern for day of the month mean revenue. And the visit counts are pretty uniform. It means all days of the month are same as far as visiting the GStore is considered.
- We see weekly patterns for visits and therefore mean and total revenues. And although the visits surge around 50-120 days the mean revenues surge around 120-150 days.

Analyze visitNumber - visit number of a visitor

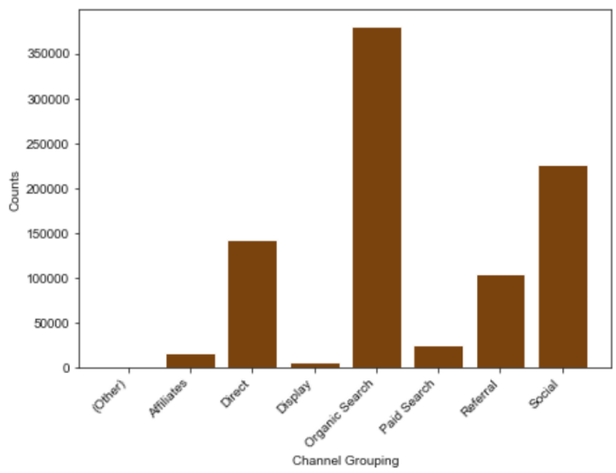
Understanding about visitNumber

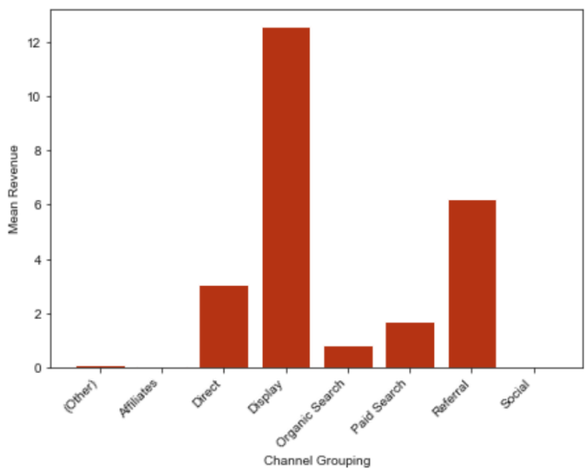
- Most of the visitors visit less than 5 times. For e.g. number of visit first or second time is around 1 million and it drops very fast. Very few visitors visit for more than 50th time or more.
- Same is the case for visits that produce revenue.
- It is little strange that there are very few revenue visits such that it is 150-250th visit for that visitor.
- Mean revenues are uniformly distributed from 0.1 to 10 for visit numbers 1 to 150. And it is significantly higher for greater visit numbers. It indicates that oftern more number of visits are needed before high valued purchase.



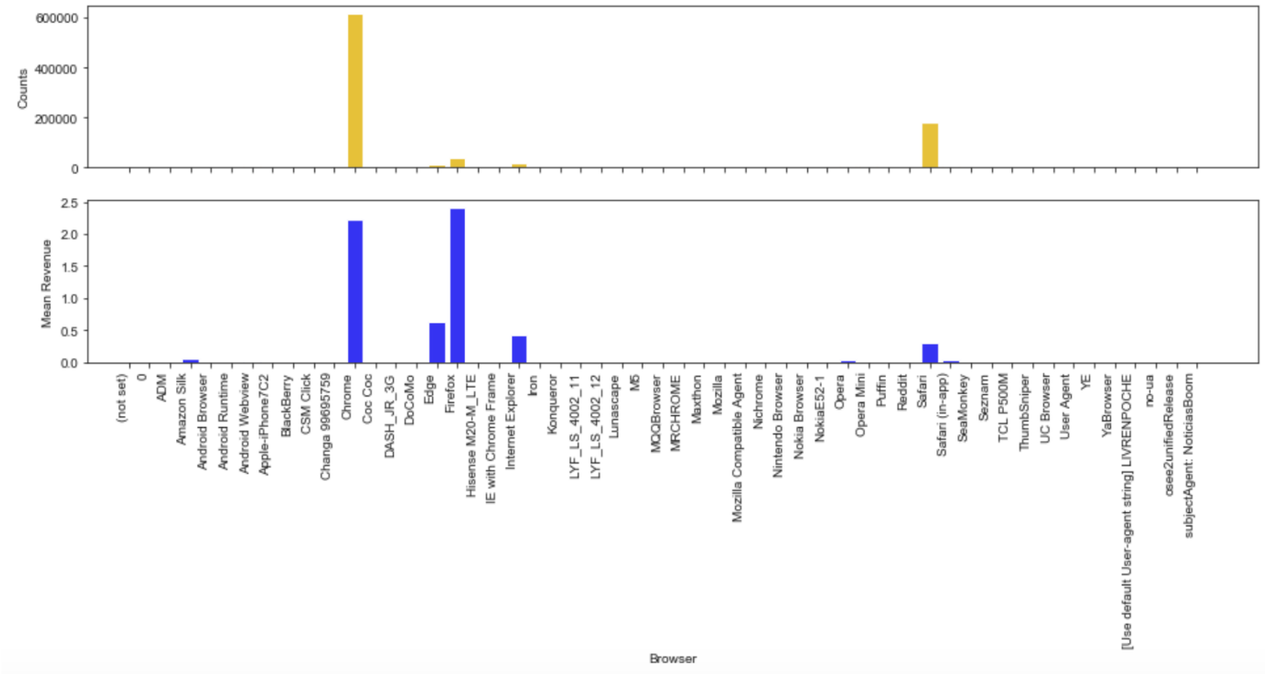
Analyze channelGrouping

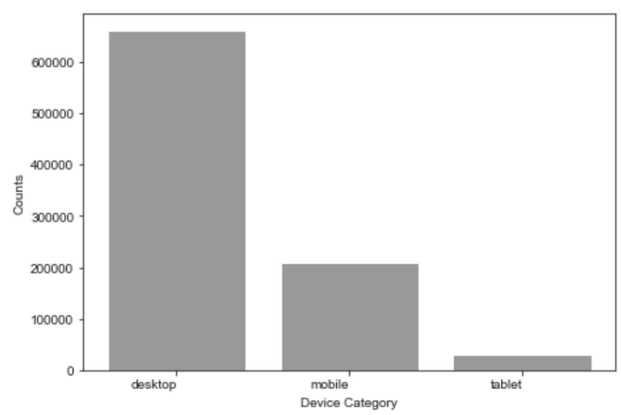
The counts and mean revenues for different channels are almost inversely proportional. That is, the mean revenue is high if a channel grouping is less frequent. For e.g. 'Display' channel.

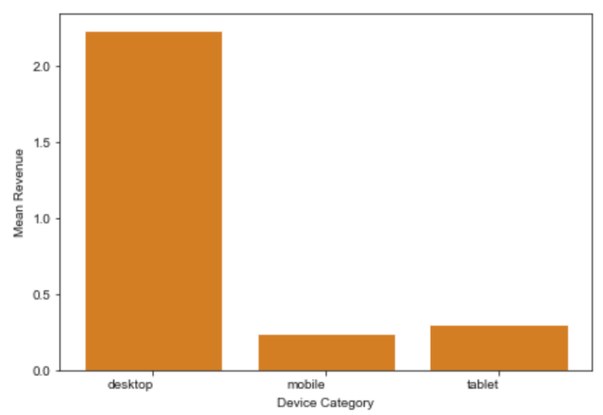


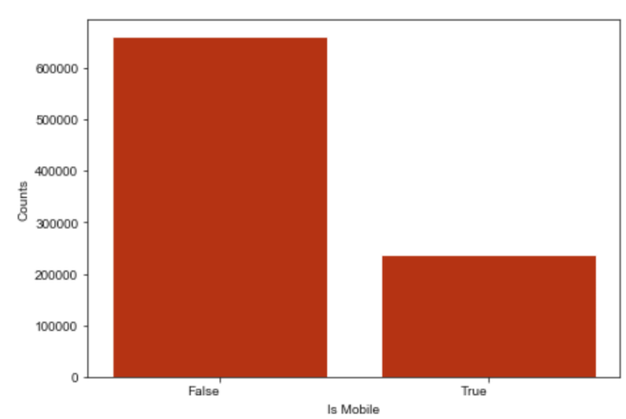


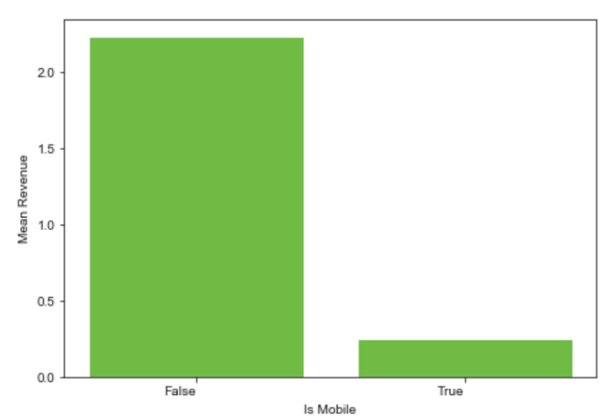
Analyze device fields

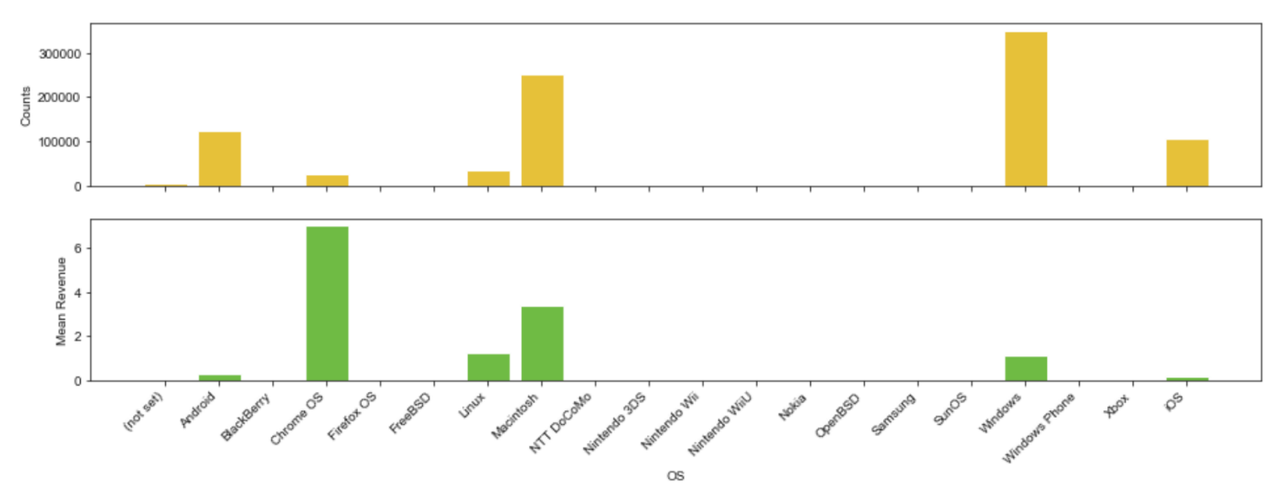




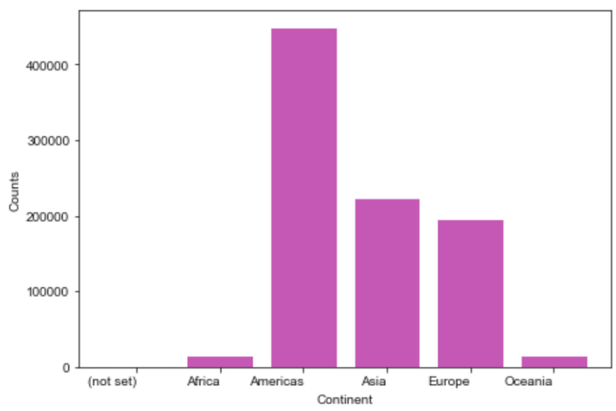


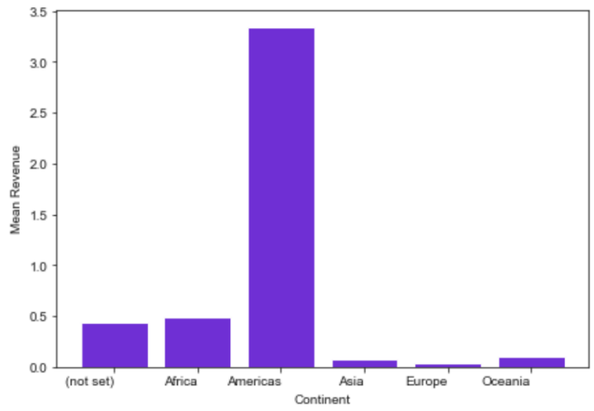


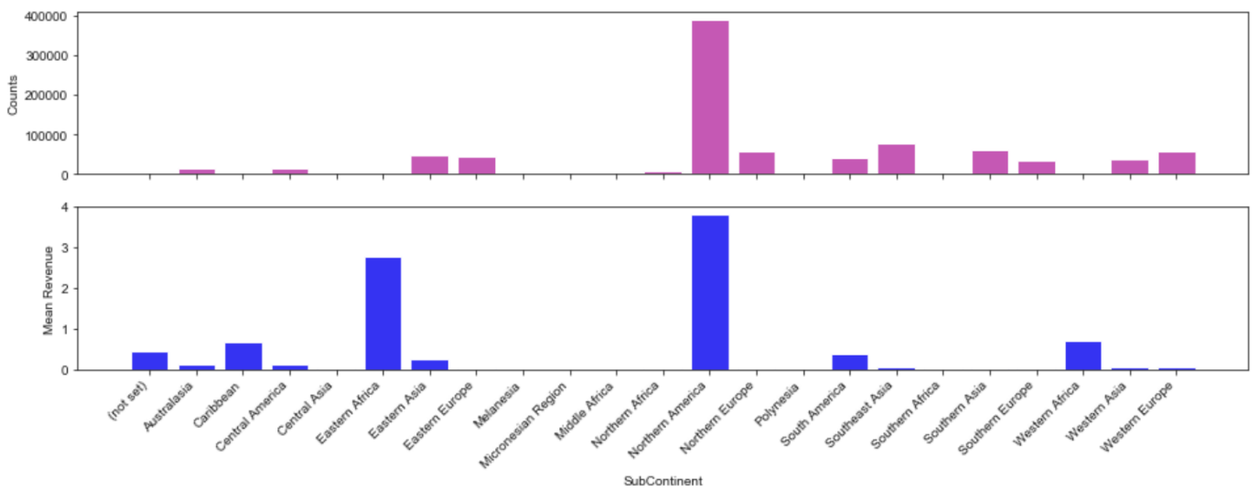




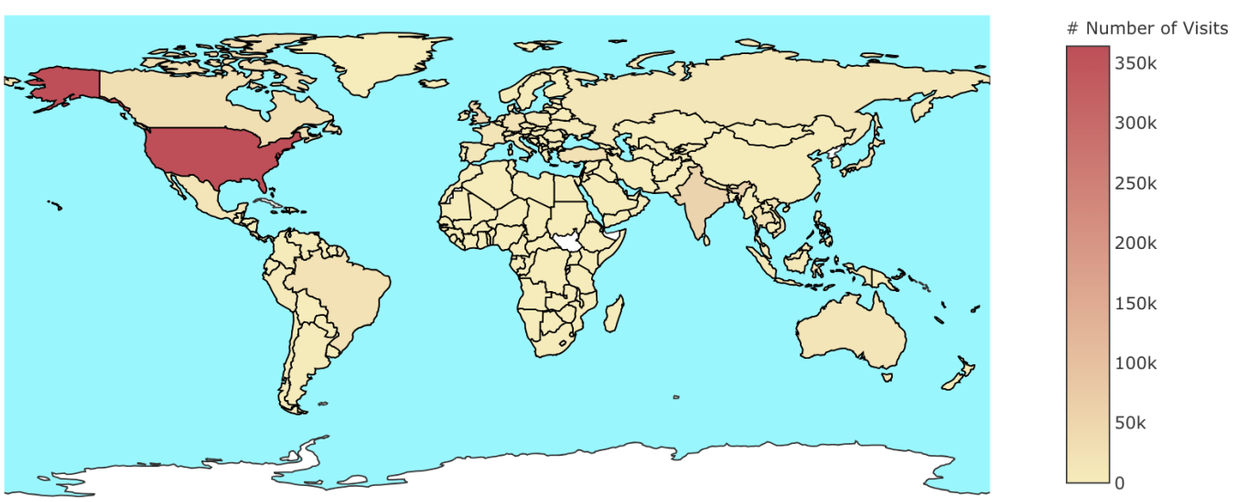
Analyze Geo fields



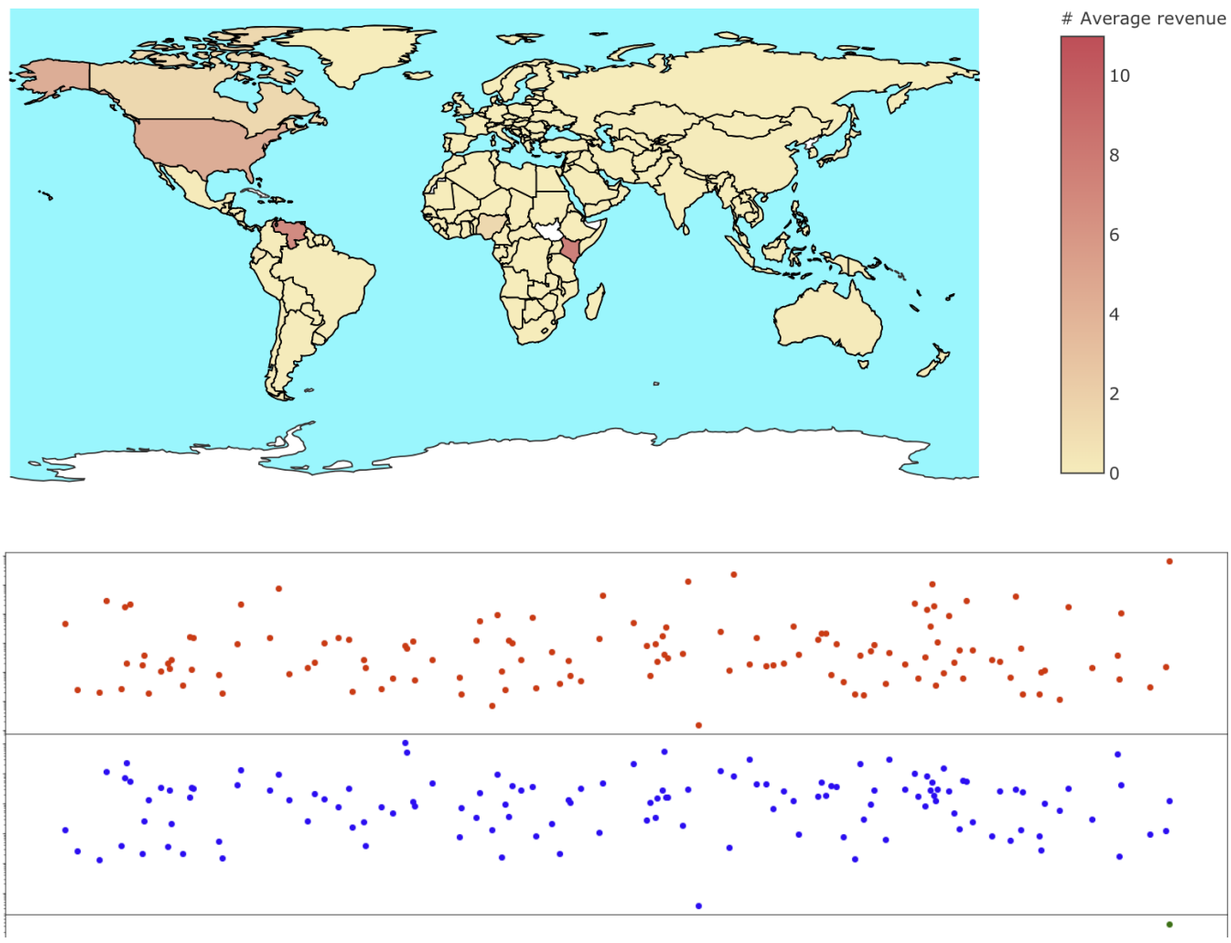




Visit by countries

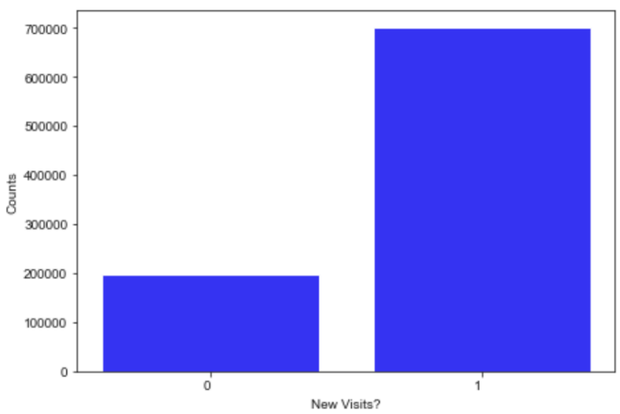


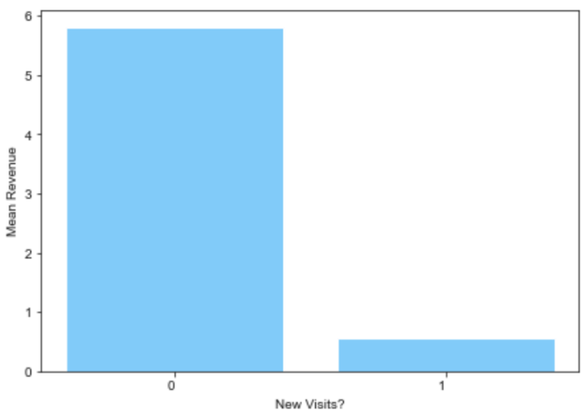
Average revenue by countries



Analyze new visits

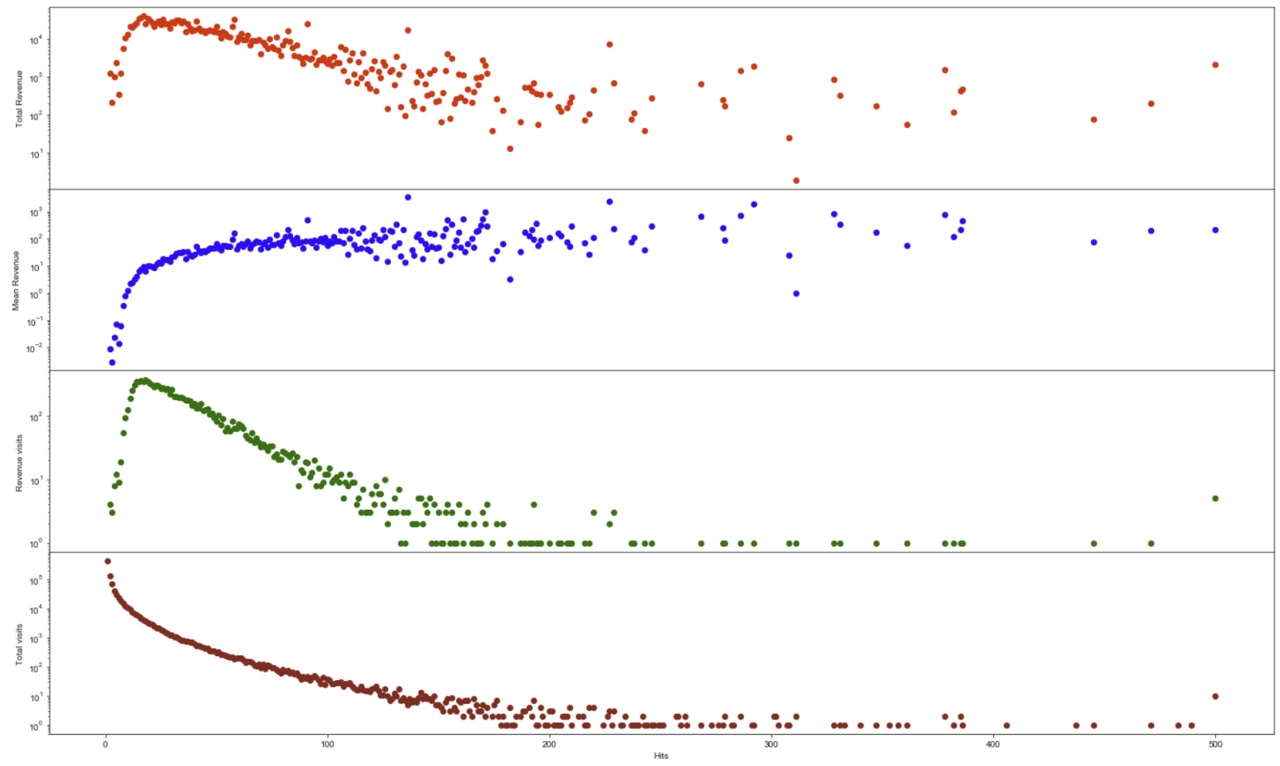
More than 70% visits are new visits or first time visits. But as expected the mean revenue from old visitors is high as compared to new visitors.

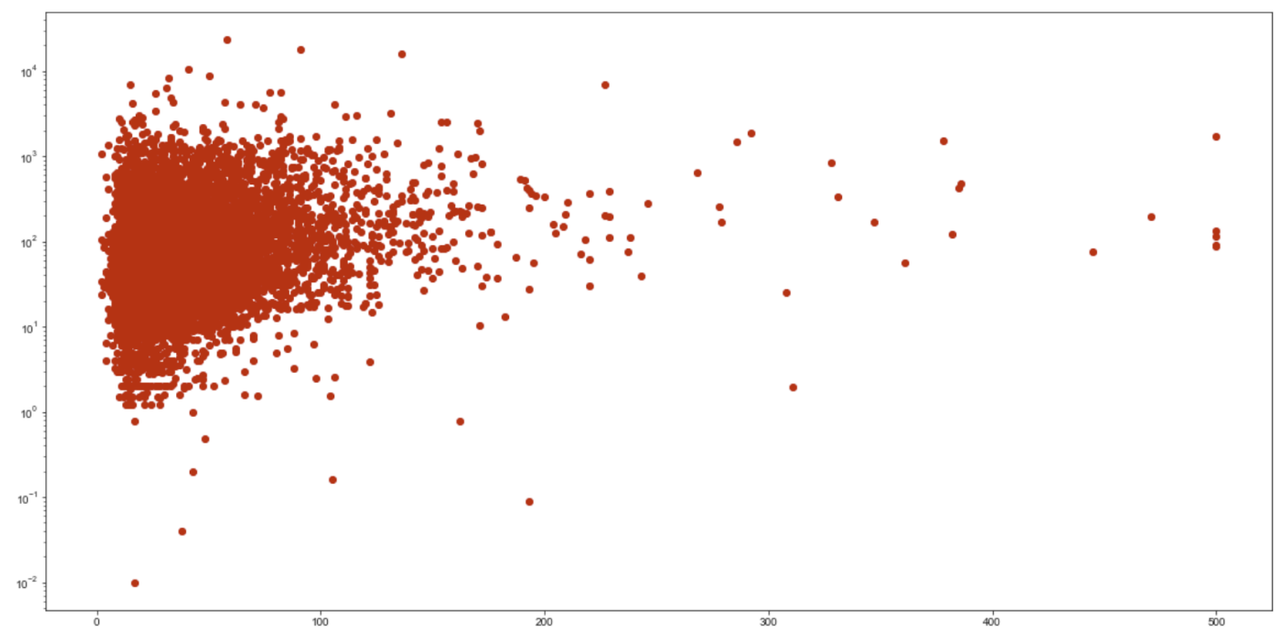




Analyze hits

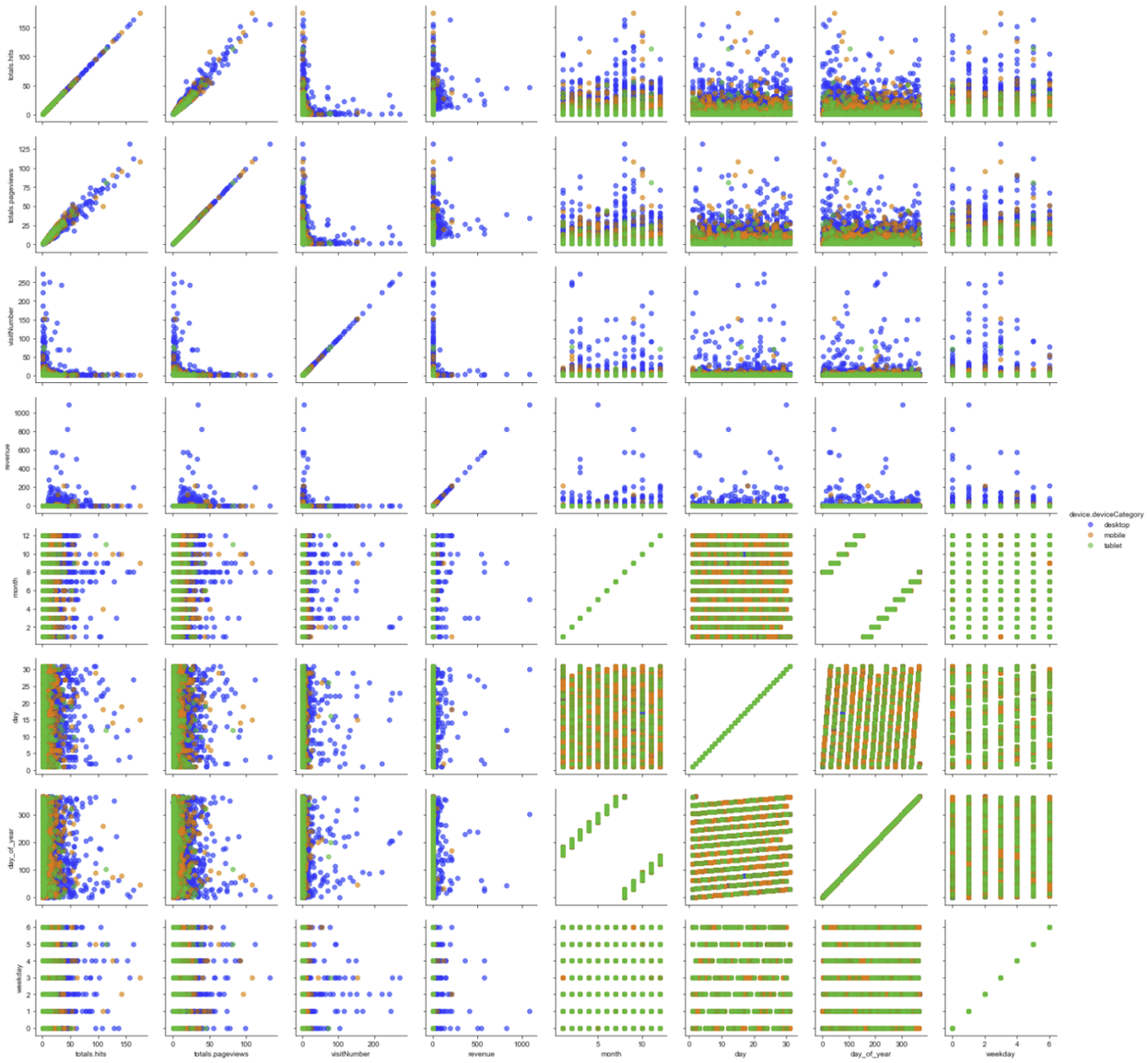
Scatter plot indicates that hits below 100 get huge number of transactions as compared to other hit values. Page views has similar distribution. And it shows that hits and page views are linearly related.





Statistical Inference

Out of all the given variables, we find out what relations, if any, exist between different pairs. Specially the relation with the target variable. We only take a randome sample for the plots.



Hypothesis Testing

By looking at all these plots, one thing is clear that there is linear relationship between the hits and the page views. Logically also it makes sense. The more number of pages are browsed it will produce more hits/downloads. Now let us see if the correlation is statistically signficant. If the correlation is high we can try omitting one of the variable in our feature matrix to train the machine learning models. Such features produce multi-collinearity issues while model fitting.

Formally, the correlation coefficient, r, tells us about the strength and direction of the linear relationship between x and y. However, the reliability of the linear model also depends on how many observed data points are in the sample. We need to look at both the value of the correlation coefficient r and the sample size n, together. The hypothesis test lets us decide whether the value of the population correlation coefficient ρ is "close to zero" or "significantly different from zero". We decide this based on the sample correlation coefficient r and the sample size n.

Null Hypothesis: H0: $\rho = 0$

Alternate Hypothesis: Ha: $\rho \neq 0$

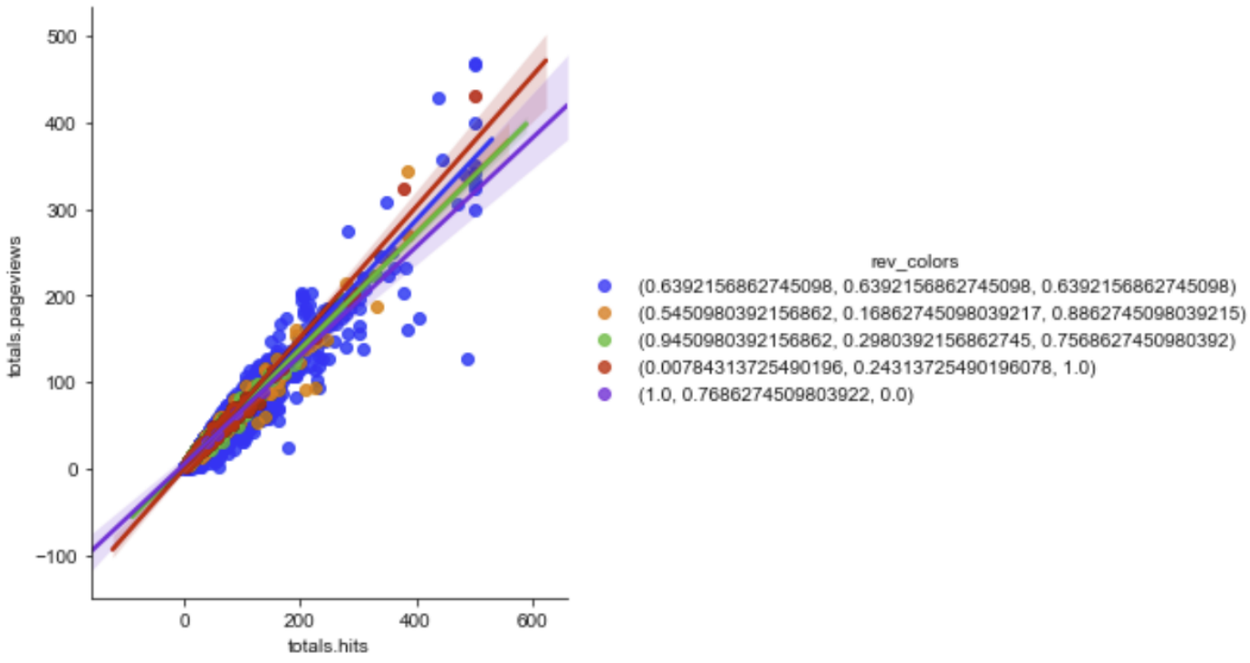
OR

Null Hypothesis H0: The population correlation coefficient IS NOT significantly different from zero. There IS NOT a significant linear relationship(correlation) between x and y in the population.

Alternate Hypothesis Ha: The population correlation coefficient IS significantly DIFFERENT FROM zero. There IS A SIGNIFICANT LINEAR RELATIONSHIP (correlation) between x and y in the population.

at significance level (α = 0.01)

We will accept or reject the Null hypothesis based on the p values. We will apply this to all the feature pairs and find the highest correlated features. And we can compare that with our previous results too.



Conclusion

The above p value is zero and we reject the null hypothesis that the population correlation is zero. The variables hits and pageViews have very high statistically significant correlation of 0.98 at significance level of 1%.

Modeling

Theory and Approach followed

Our goal is to develop a model that describes the relationship between the features in the data and the target class (product category). So far we have got good understanding of the data and we can apply the machine learning algorithms and find out the best model. Technically speaking, this is Supervised Regression ML problem. Here we have the training data with the actual labels. We need to model that and use the model to predict labels for new data.

We will start with a quick and dirty baseline model. And improve upon our model in interative fashion. Following are some ideas to keep in mind.

Overfitting and Underfitting

When the model fits the training data too perfectly but not the test data, it is called overfitting. Such a model will not be useful for predictions. While at the other extreme end our model might fit the training data very poorly. Consequently, it can't fit test data any better and will give inaccurate predictions. Generally its a trade off between overfitting and underfitting. This is also called **Bias-Variance trade-off**. When the model fits too well to the training data (a subset of population) it has low Bias, that is, it mimics the data very well. But it might fail miserably in mimicking the test data (another subset of population). Or it has variance between different subsets of data. As we try to make this variance low, it will increase the Bias.

In our case, the number of data points is very large (900 thousand) as compared to the number of features. This will help in avoiding overfitting. We are going to always compare train and test accuracy to see how well the model fits. We should also design train-test split of the data wisely to regulate the bias-variance trade off.

Model Evaluation Criteria

Evaluating our model closely relates with how the model is going to be used by the client.

Root Mean Squared Error (RMSE): This is the evaluation criteria used by the client (Google). So this is our main criterion to compare and improve the models. It is defined as

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

where y hat 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.

Data and code organization

- **Data files generated and saved instead of data pipelines in the runtime**

  Although data pipelines are generally used to transform train and test data sets. In this project the data is huge and we don't want to run all the steps of datapipeline while development. Moreover there is not much of data transformation. Mainly we do the Standard Scaling and save the resulting data as a file in the /data folder. This file is used for training of most the models. Similiarly the cleaned data files are also saved into the /data folder.

- **Python functions written to apply the DRY principle**

  Since lot of things are repetitive like fitting the data, evaluating the results etc. we have written Python functions that are used multiple times.

Data preparation before modeling

- **Cardinality reduction**

  The data has a large number of categorical variables and the cardinality of those variables is high. That is the number of unique values for some of the categories is huge. For e.g. network domain has 27996 unique values. Ultimately categorical variable get one hot encoded with one columns for each unique

value. And this will drastically increase the size of data for modeling. Because of resource(memory and compute) constraints, we start with merging the less frequent values into a grouped dummy value. Specifically, all the values ocurring less than 1% time for a particular category are grouped into a single value dummy_column_name. Some examples:

```
device.operatingSystem
number of unique values 20
dummy_device_operatingSystem
reduced number of unique values 7
remaining of unique values ['Windows' 'Macintosh' 'Linux' 'Android' 'iOS' 'Chrome OS'
 'dummy_device_operatingSystem']

geoNetwork.city
number of unique values 649
dummy_geoNetwork_city
reduced number of unique values 9
remaining of unique values ['dummy_geoNetwork_city' 'not available in demo dataset' '(not set)'
 'London' 'Mountain View' 'New York' 'San Francisco' 'Sunnyvale'
 'San Jose']

geoNetwork.country
number of unique values 222
dummy_geoNetwork_country
reduced number of unique values 22
remaining of unique values ['Turkey' 'Australia' 'Spain' 'Indonesia' 'United Kingdom' 'Italy'
 'dummy_geoNetwork_country' 'Netherlands' 'India' 'France' 'Brazil'
 'Poland' 'Germany' 'Canada' 'Thailand' 'Taiwan' 'Russia' 'Philippines'
 'Mexico' 'United States' 'Japan' 'Vietnam']
```

- **Select Basic Features**
  - We keep all other features except for the following
  - ID variables are not useful, as they are unique to each mostly. Also, they don't make any business sense.
  - Year variable can't be used because the data does not spread over many years.
  - Continent data can be removed because we have finer granular Sub-continent.
  - IsMobile will be a part of categorical variable deviceCategory. It is redundant and can be dropped.
  - Date is used to derive month, day etc. date variables and can be dropped.
  - Also, for initial set of models we are not using date features, to keep it simple
- **Encode categorical variables**

  We use pd.get_dummies() because it gives us the new column names nicely like "geoNetwork.country_Spain" etc.
- **Train Test split based on time**

  Since the data is based on time and we are predicting later dates based on earlier dates, we should divide our train/test based on time too. We took threshold of date "2017-06-01" and based our train/test as before/after this date.

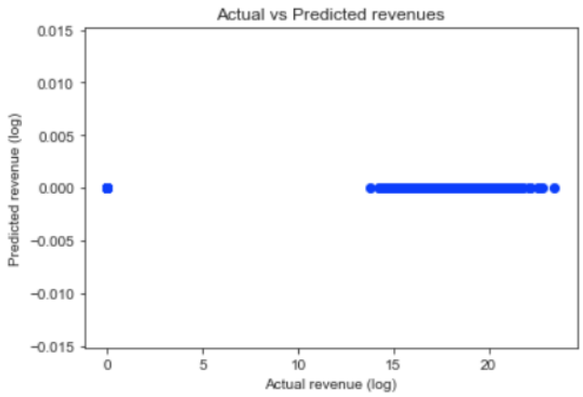
First (Baseline) Models

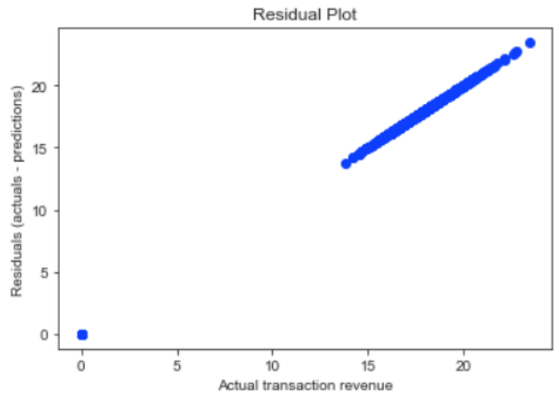
Predictions at human level

The whole point of doing machine learning is to do predictions that are better than that we humans can do, given the huge amount of data. So before doing any kind of machine learning, let us do a very simple human level prediction. That will give us the worst case to start with. We call it dumb because we expect ML models to do better than that.

We create a dumb model, that predicts all revenues per visitor to be zero. The reason for 0s is that we have large number of visits with 0 revenues. So at first thought, we can just expect future visits to be 0 revenue. However we can also predict all revenues to be the mean of exisiting revenues as well. Evaluating that model gives the RMSE of **2.290**.

Let us visualize how it looks. It shows how bad our model is. The residual plot shows a clear linear pattern too.

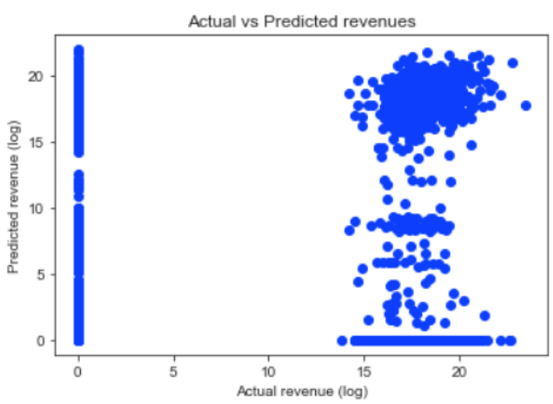


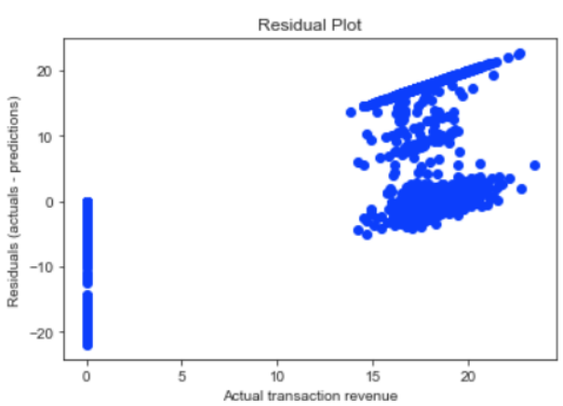


First baseline model using Decision Tree

- Decision Trees are very fast to implement.
- They tend to overfit but good to try out for our first quick and dirty model.
- They are intuitive and belong to non-parametric models.
- In simple words, a tree is grown by successively splitting the data set into 2 branches based on a threshold value for a given feature. There can be a stopping criteria like the depth of a tree etc. If no criteria, the tree can grow big with single leaves at deepest levels. That would ofcourse be overfitting.
- The default parameters are good to try for our first model.

The RMSE of **2.298** is no better than our previous RMSE. But the charts below show that this is a better model.



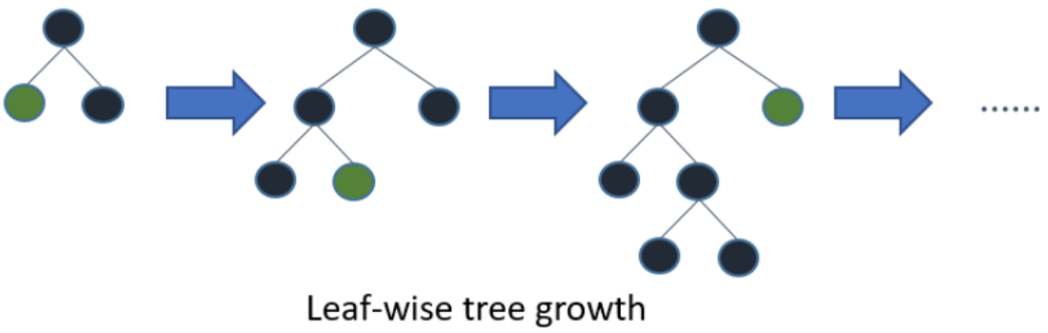


Light GBM

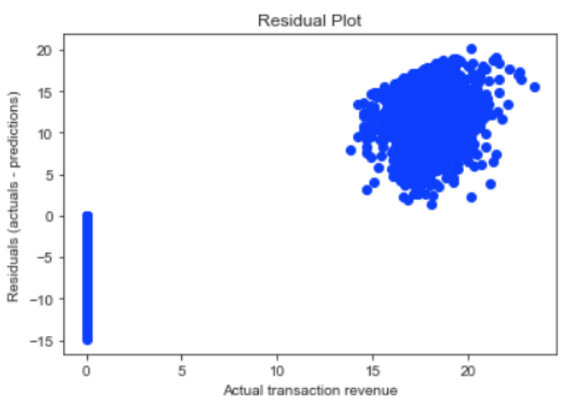
- We tried Random Forest and XGB but with only a sample of data set. Both of them are memory intensive and not feasible with the given amount of resources. We will not try these models further.
- Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm.
- It splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.
- It gives high accuracy with the risk of overfitting.
- Most important thing for us is that it is super fast, hence the name Light. Therefore, we are able to use the whole dataset to fit unlike XGB.

Before is a diagrammatic representation by the makers of the Light GBM to explain the difference clearly.

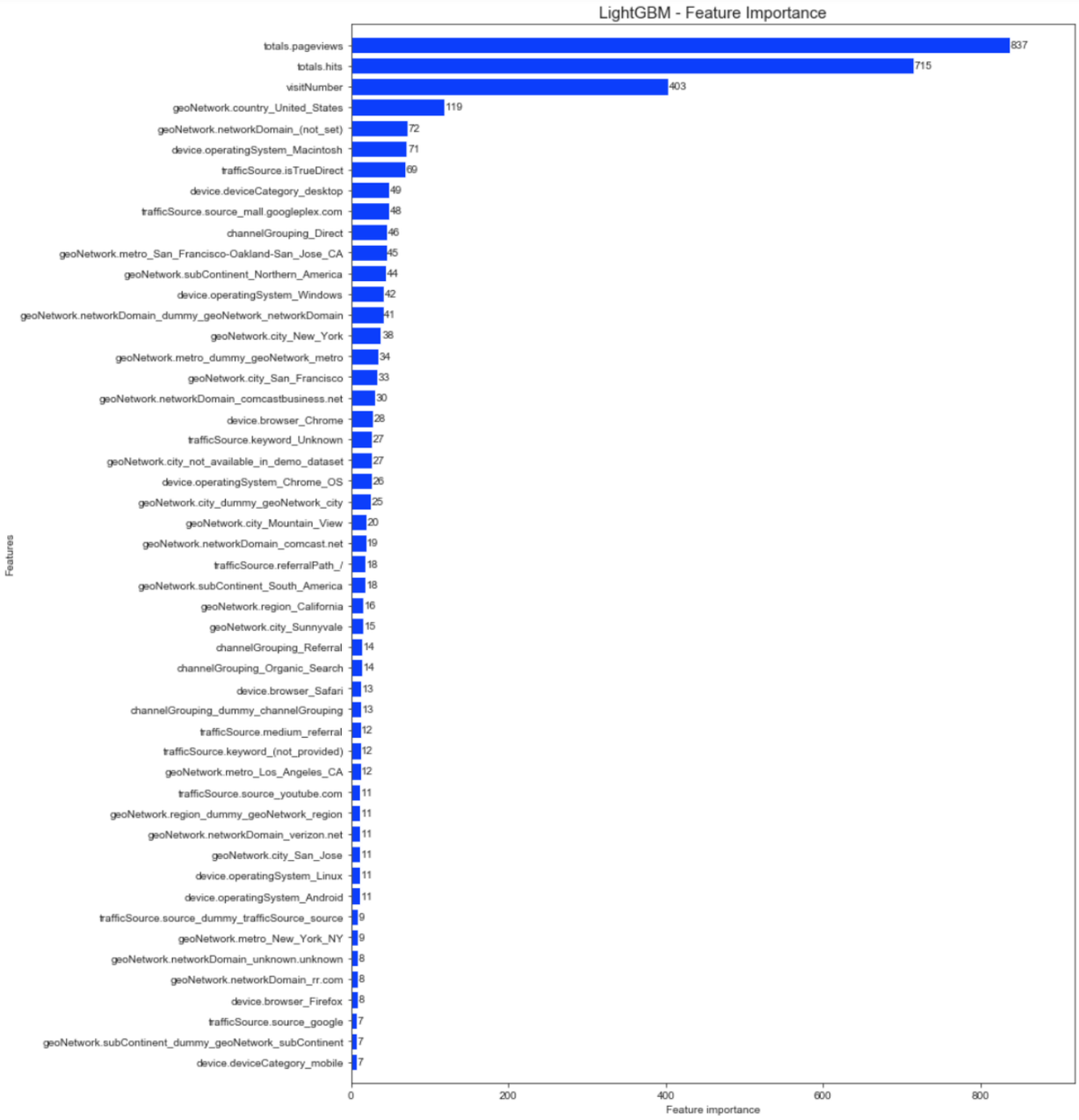
Level-wise tree growth in XGBOOST.



The RMSE of **1.756** is better than our previous RMSEs. The charts below show that this is indeed better model.



Feature importance as per the model. This is useful if want to make model more interpretable and reduce less useful featutes.


LightGBM - Feature Importance
totals.pageviews 837
totals.hits 715
visitNumber 403
geoNetwork.country_United_States 119
geoNetwork.networkDomain_(not_set) 72
device.operatingSystem_Macintosh 71
trafficSource.isTrueDirect 69
device.deviceCategory_desktop 49
trafficSource.source_mall.googleplex.com 48
channelGrouping_Direct 46
geoNetwork.metro_San_Francisco-Oakland-San_Jose_CA 45
geoNetwork.subContinent_Northern_America 44
device.operatingSystem_Windows 42
geoNetwork.networkDomain_dummy_geoNetwork_networkDomain 41
geoNetwork.city_New_York 38
geoNetwork.metro_dummy_geoNetwork_metro 34
geoNetwork.city_San_Francisco 33
geoNetwork.networkDomain_comcastbusiness.net 30
device.browser_Chrome 28
trafficSource.keyword_Unknown 27
geoNetwork.city_not_available_in_demo_dataset 27
device.operatingSystem_Chrome_OS 26
geoNetwork.city_dummy_geoNetwork_city 25
geoNetwork.city_Mountain_View 20
geoNetwork.networkDomain_comcast.net 19
trafficSource.referralPath_/ 18
geoNetwork.subContinent_South_America 18
geoNetwork.region_California 16
geoNetwork.city_Sunnyvale 15
channelGrouping_Referral 14
channelGrouping_Organic_Search 14
device.browser_Safari 13
channelGrouping_dummy_channelGrouping 13
trafficSource.medium_referral 12
trafficSource.keyword_(not_provided) 12
geoNetwork.metro_Los_Angeles_CA 12
trafficSource.source_youtube.com 11
geoNetwork.region_dummy_geoNetwork_region 11
geoNetwork.networkDomain_verizon.net 11
geoNetwork.city_San_Jose 11
device.operatingSystem_Linux 11
device.operatingSystem_Android 11
trafficSource.source_dummy_trafficSource_source 9
geoNetwork.metro_New_York_NY 9
geoNetwork.networkDomain_unknown.unknown 8
geoNetwork.networkDomain_rr.com 8
device.browser_Firefox 8
trafficSource.source_google 7
geoNetwork.subContinent_dummy_geoNetwork_subContinent 7
device.deviceCategory_mobile 7
Features
0
200
400
600
800
Feature importance

Feature Engineering

Based on the EDA done till now and gaining some insights and intuition about the data, we can now derive some new features. After testing the machine learning models, we can keep them if useful. We have already added some features at the time of EDA.

Previously added new features

- **hour** - Extracted this feature from visitStartTime. The login hour might serve some business meaning.
- **local_hour** - This is hour but calculated keeping in mind the location of the visitor. This is more meaningful than just UTC hour.
- **day** - We saw some patterns within the month. This feature can capture that pattern.
- **day_of_year** - Again based on EDA there are variations in the data for all the days of the year.
- **weekday** - Captures the weekly patterns of the data.
- **month** - Captures the monthly pattern of the data.

New features

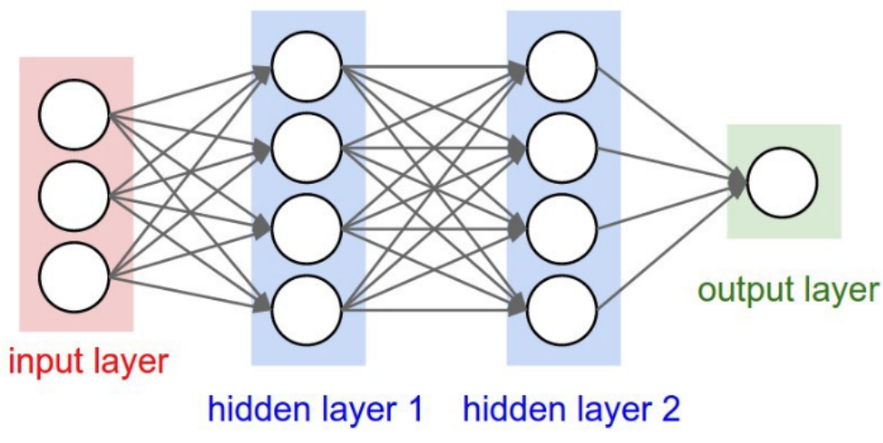
- **quarter** - The whole year can be divided into quarters. This feature might capture holiday season or financial year start etc.
- **week_day_end** - As seen from EDA there are more logins or weekends but less sales. It might indicate that people have leisure time and they browse and not necessarily interested in buying.
- **holidays** - At the end of the year there is first rise in the number of the logins and then some days later there is rise in revenue transactions. We can define that period as holiday period.
- **month_period** - Whole month can be divided into start, mid and end. Start might indicate the arrival of salaries etc. Month end might be related to more expenditure.
- **active_hours** - As seen during the EDA that morning hours are generally low on revenue transactions. This makes sense because people are generally busy getting ready for the work day ahead.

Modeling using new Features

Even after adding the new features the models didn't improve much. We didn't try XGB and Random Forest as we don't have enough resources to use full dataset for them. Neither can we try hyperparameter tuning on them. The results for Decision Tree and LightGBM remains the same.

We also tried Artificial Neural Nets. In case of huge data sets, ANNs generally work good. An ANN consists of artificial neurons called nodes that accept inputs, processes it to generate the output. A neural network generally consists of 3 layers:

- Input Layer: This layer is responsible to accepts large volumes of inputs in various formats like text, numbers, image pixels, audio, etc.
- Hidden Layers: This layer is responsible to perform data manipulation, mathematical calculation, feature extraction, etc.
- Output Layer: This layer generates the desired output based on the information it receives from the previous layers.



But they need longer run times. Training with new features took around 3mins. Also, NNs need lot of paramter tuning, which is not feasible for this project. So for future trials we will only use Light GBM. Here is our ANN model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 100)               14900
_________________________________________________________________
dense_2 (Dense)              (None, 65)                6565
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 66
=================================================================
Total params: 21,531
Trainable params: 21,531
Non-trainable params: 0
```

The results are worse than that of Light GBM. The test rmse is **1.85**. And here is the glimpse of training:

```
Train on 762857 samples, validate on 137031 samples
Epoch 1/10
762857/762857 [==============================] - 22s 29us/step - loss: 3.1421 - val_loss: 3.2976
Epoch 2/10
762857/762857 [==============================] - 22s 29us/step - loss: 2.9006 - val_loss: 3.3016
Epoch 3/10
762857/762857 [==============================] - 22s 28us/step - loss: 2.8718 - val_loss: 3.1547
Epoch 4/10
762857/762857 [==============================] - 22s 28us/step - loss: 2.8475 - val_loss: 3.1759
Epoch 5/10
762857/762857 [==============================] - 22s 29us/step - loss: 2.8316 - val_loss: 3.2209
Epoch 6/10
762857/762857 [==============================] - 22s 28us/step - loss: 2.8146 - val_loss: 3.1894
Epoch 7/10
762857/762857 [==============================] - 22s 28us/step - loss: 2.7982 - val_loss: 3.1525
Epoch 8/10
762857/762857 [==============================] - 22s 29us/step - loss: 2.7845 - val_loss: 3.1851
Epoch 9/10
762857/762857 [==============================] - 22s 29us/step - loss: 2.7781 - val_loss: 3.2073
Epoch 10/10
762857/762857 [==============================] - 22s 28us/step - loss: 2.7705 - val_loss: 3.2833
CPU times: user 5min 42s, sys: 1min 20s, total: 7min 3s
Wall time: 3min 38s
```

Model Comparisons

Light GBM performed the best, both in terms of accuracy and lower RMSE and the training speed. There is lot of scope for further feature engineering and parameter tuning. Hopefully that will give better results.

Model	RMSE (Metric)	Details
Dumb Model	2.29	All predictions as 0s
Decision Tree	2.298	Baseline Decision Tree
Random Forest	1.869	Random Forest with only 10K training rows
XGBoost	1.964	XGBoost Regressor with only 10K training rows. Params are {learning_rate=0.1, n_estimators=1000, max_depth=5, min_child_weight=1, gamma=0, subsample=0.8, colsample_bytree=0.8, objective='reg:linear', nthread=4, scale_pos_weight=1, seed=19}
Light GBM	1.756	Light GBM with basic features
Decision Tree	2.351	Decision Tree with new features
Light GBM	1.761	Light GBM with new features
Neural Nets	1.84	Neural Network with new features with 2 hidden layers of 100 and 65 neurons

Recommendations for the client

We lots of EDA and data modeling, there are many insights gained. Here are our recommendations for the client GStore.

- The model can be used to predict the revenues per visitor. Ultimately, this will help in targeting the advertisement towards indidvidual visitor based on their potential expenditure.
- Holiday season as seen from our charts, can be exploited. We see more visits during that time period but not much increase in revenue. The advertisement data can be used to better adverstise in the holiday season.
- Local active hours show that the morning hours are least active and we could target particular segment of customers like old or non working people. This might also help in balancing the traffic to the GStore throughout the day.
- Video Ads seem to be wasted. Might spent the advertising budget elsewhere.
- Traffic source shows that "Google search" bring lot of revenue.
- There are few customers who are loyal and visited the GStore many times as compared to average visitor. Some of them generated greater revenues. There might be a chance that they are looking for something specific and not being able to find it and hence visiting the website often to check out. These visitors can be studied more. Also, they can be rewarded for being loyal.

Conclusions and Future analysis

We analyzed the data and predicted the revenues for the test data for visits for a time period. The problem was defined as such on Kaggle. However the problem can be modifies a bit and can serve even greater business puposes. We can actually predict revenues for future dates, for which we have no data collected whatsoever. We intend to carry that out as the second phase of the project. It will involve Time Series modeling and the model will be based on per visitor. The goal can be summarized as: Predict for the current visitors, if they will generate any revenue for future month (or some period) and if yes predict the amount too.

The current problem was hard because of the huge amount of data (1 million rows approx.). Due to the lack of available resources we could not try out XGB or NeuralNets on the full data (full cardinality categorical features) and with cross validation. Given more resources we can try those models.

We can try many more new features like:

- Cross features like device_browser, browser_networkDomain, hits_to_pageviews_ratio
- We can try assigning ranks to unique values of categorical variables. The ranks can be based on mean revenue for that value of category or count of visits. For e.g. the "Northen America" region has the highest mean revenue. And so it can be given rank 1st and so on

References

https://www.kaggle.com/c/ga-customer-revenue-prediction (https://www.kaggle.com/c/ga-customer-revenue-prediction)

https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/ (https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/)

https://scikit-learn.org (https://scikit-learn.org)

https://medium.com (https://medium.com)