Customer Revenue Prediction

Domain Background

Any company wants to increase the revenue by minimizing the investment. Only a small proportion of the customers produce most of the revenue. Knowing this proportion is essential for marketing teams to make appropriate investments in the promotional strategies. The forecast can be based on past sales data, industry wide comparisons and economic trends. Predicting revenue not only enable how to manage the workforce, cash flow and resources but also helps the company when looking to acquire investment capital. Revenue forecast allows you to spot potential issues while there's still time to avoid or migrate them. According to research from the Aberdeen Group, companies are more likely to grow revenue by 10% year-over-year when they have an accurate revenue forecast system.

Problem Statement

The main objective of the project is to apply Machine Learning techniques to accurately predict revenue per customer. The tasks involved are the following:

1. Process the train and test datasets
2. Normalize the data
3. Train the model that can predict the revenue for the customer
4. Predict the customer revenue for the test dataset

Metrics

Given a historic sales and associated revenue information, we want to be able to predict the revenue of a customer in future time period. The accuracy is determined by the Root Mean Squared Error(RSME). Lower the value more accurate is the prediction.

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

where $\widehat{y}$ is the natural log of the predicted revenue and y is the natural log of the actual summed revenue for a customer.

Analysis

Data Exploration

The dataset has approximately 1 million store visits for customers and the revenue generated on those visits. The visits range from August 2016 to August 2017 with 903653 visits in the train and 804684 visits in the test dataset. The dataset have the following 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.
- **sessionId** - The sessionId of the user visited the Store.
- **device** - The specifications for the device used to access the Store.
- **geoNetwork** - This section contains information about the geography of the user.
- **socialEngagementType** - Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- **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).

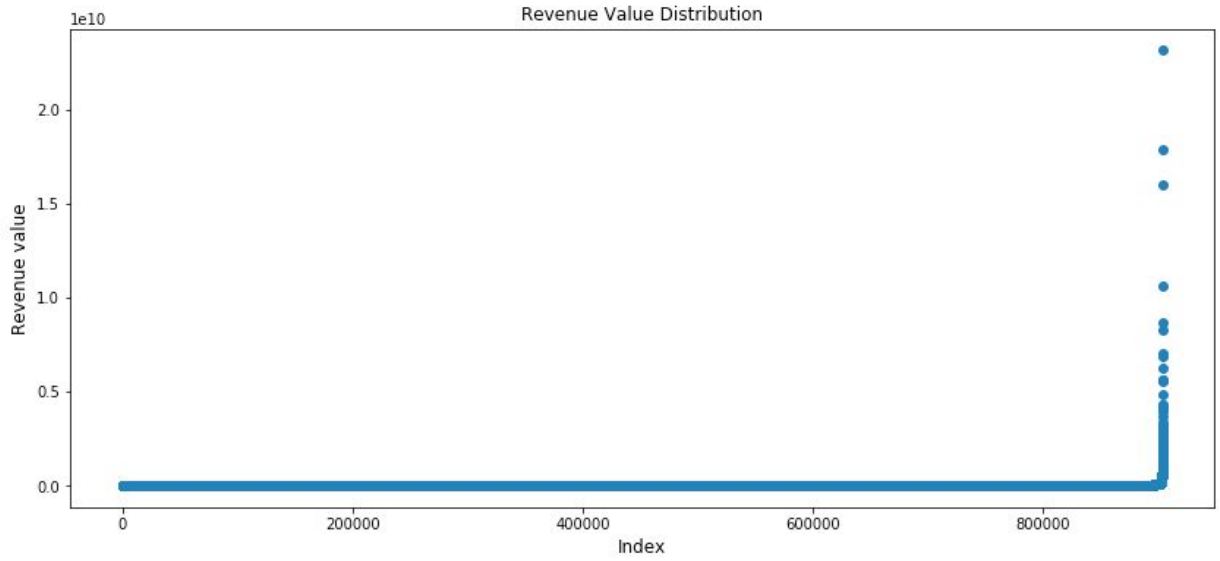
As it can be in any dataset some of the above fields maybe in ineligible or not useful, which means they need to be discarded during the preprocessing.

Exploratory Visualization

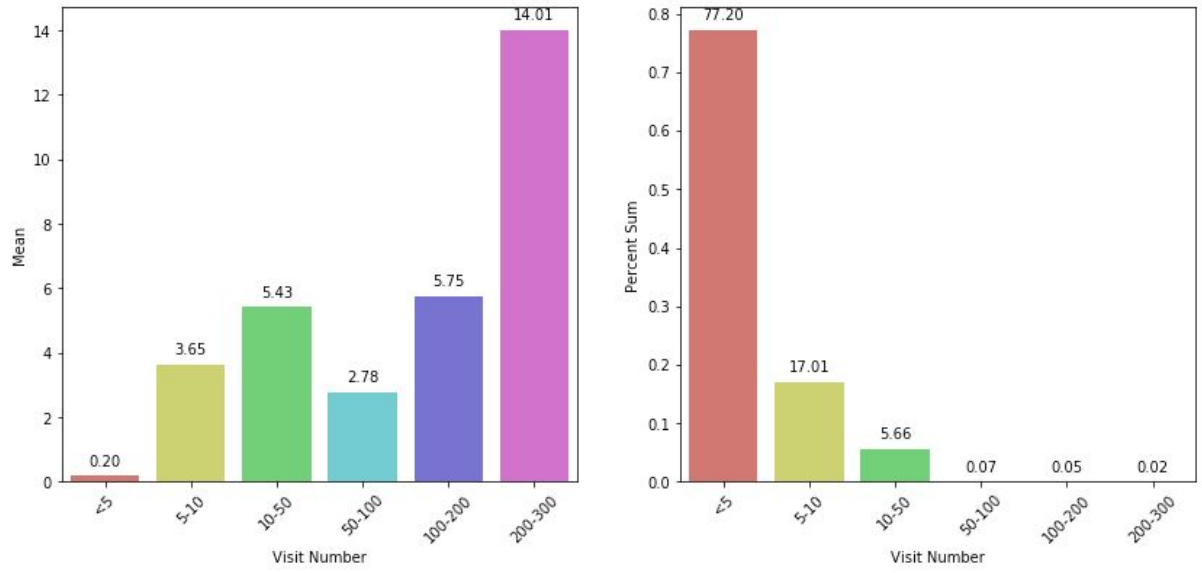
From the revenue distribution it confirms the first two lines of the competition overview.

Since we are predicting the natural log of the sum of all transactions of the user, let us sum up the transaction revenue at user level and take a log and then do a scatter plot.

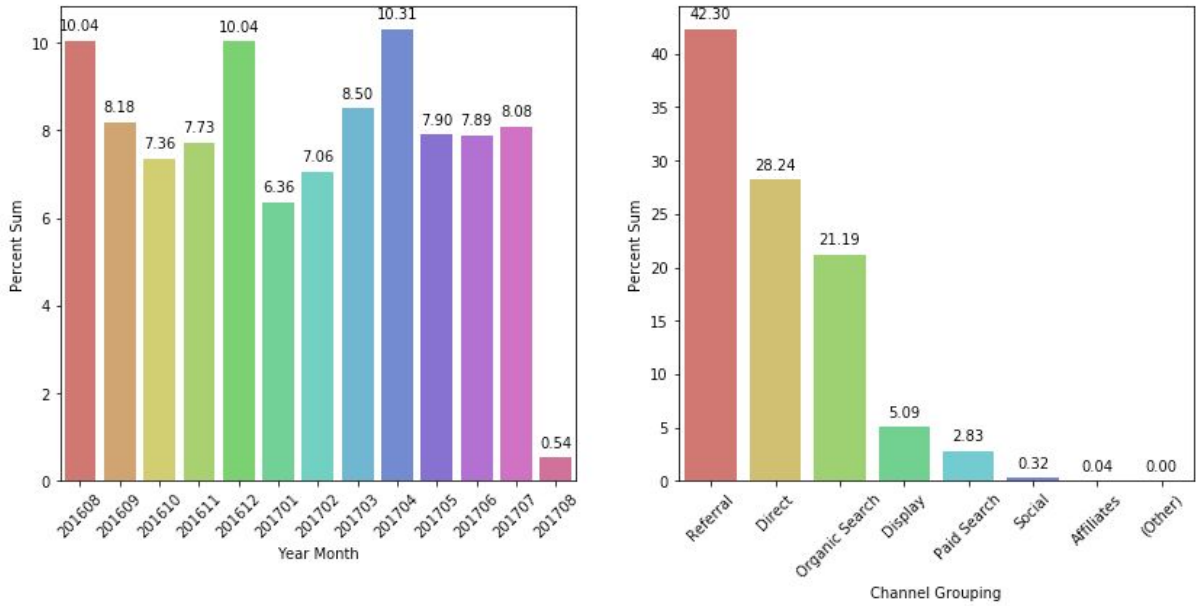
The 80/20 rule has proven true for many businesses–only a small percentage of customers produce most of the revenue. As such, marketing teams are challenged to make appropriate investments in promotional strategies



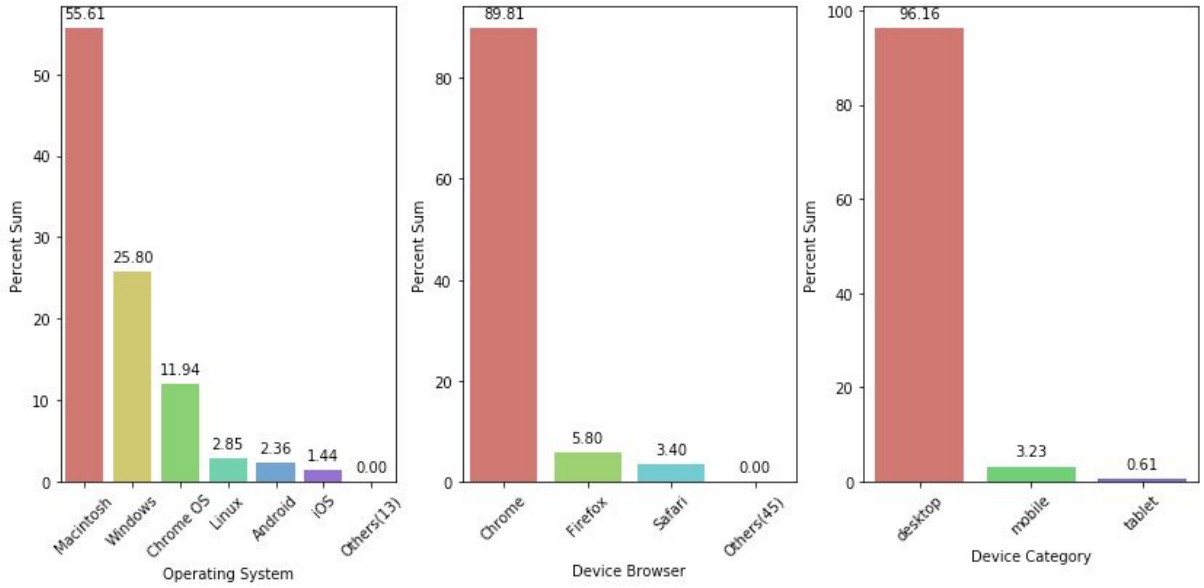
The visitNumber feature provides interesting insights. 77.2% of the revenues come from the customers who have visited the store less than 5 times, 17.01% comes from customers who have visited between 5 and 10 times whereas only 0.02% comes from those who visited more than 200 times. However the mean of the revenue is highest for the greater than 200 visit customers. Although fewer customers visits the store greater than 200 times they are loyal customers from who the company drives the most per capita revenue and these customers could be potentially be a good target. The below figure illustrates the relationship between visit number and revenue.



Also analysing the revenue by month, it shows that month of April has the highest revenue aside from December(which is expected). So during this month company can offer additional promotional strategies so as to increase the revenue. Also the referrals is also fetching more income. We can combine this along with the monthly revenue to target the potential customers. The below figure illustrates the above analysis



On further analysis, we can find that Macintosh operating system customers generates 55.61%, Chrome web browser customers generates 89.81% and desktop customers generates 96.16% of the revenue. We can target users on these features to maximize the revenue. The below figure illustrates the analysis.



Algorithms and Techniques

Out of the 903653 visits to the store only 9495 are revenue generating visits. Post the data cleaning and validation steps, I am considering the following features in the model.

1. channelGrouping
2. device.browser
3. device.deviceCategory
4. device.operatingSystem
5. geoNetwork.country
6. trafficSource.medium
7. trafficSource.source
8. totals.hits
9. totals.newVisits
10. totals.pageviews

11. visitNumber

The model is a Long Short-Term Memory Recurrent Neural Network, capable of learning long-term dependencies and are explicitly designed to avoid the long-term dependency problem. The following parameters can be tuned to optimize the classifier:

1. Training Parameter
    a. Training length(number of epochs)
    b. Batch size (how many visits to look at once during a single training step)
    c. Solver type (what algorithm to use for learning)
    d. Learning rate (how fast to learn)
2. Neural network Architecture
    a. Number of layers
    b. Number of units in the layer

During the training, both the training and validation sets are loaded into the RAM. After that batches are selected and loaded to GPU memory for processing. The batches are passed through the LSTM networks and are fully connected using a linear network and using ReLU activation function to obtain the output. I will be calculating the validation loss every epoch to see how the model is performing.

Benchmark

To create an initial benchmark for the regressor, I used LightGBM model and achieved and RSME of 1.8654. Later I'll discuss the slightly improved model that will predict the revenue better. This model is implemented in 2_Baseline_LGBM_Model notebook.

Methodology

Data Preprocessing

The preprocessing is done in 1_Exploratory_Data_Analysis notebook and consists of the following:

1. Flattening the JSON objects in the raw input datasets

2. Default some features to appropriate values
3. Removing all the columns that have constant or have more than 90% null values
4. Normalizing the data using LabelEncoder and MinMaxScaler
5. Splitting the data into training and validation sets
6. Save the training and validation sets to csv files

Implementation

During the implementation process the model was trained on the preprocessed training data This was done in 3_LSTM_Model notebook and can be further divided into the following steps:

1. Load both the training and the validation sets into memory
2. Implement helper functions:
    a. _get_data_loader: creates a data loader from training/validation data
    b. model: generates the model and forward function to train the data
    c. train: runs through the specified epochs and batch size and calculates the validation loss
    d. predict: given an input, it predicts the output of the input feature
3. Define the network architecture and training parameters
4. Define loss function and optimizer
5. Train the network, logging the validation/training loss and validation accuracy
6. Save and freeze the network

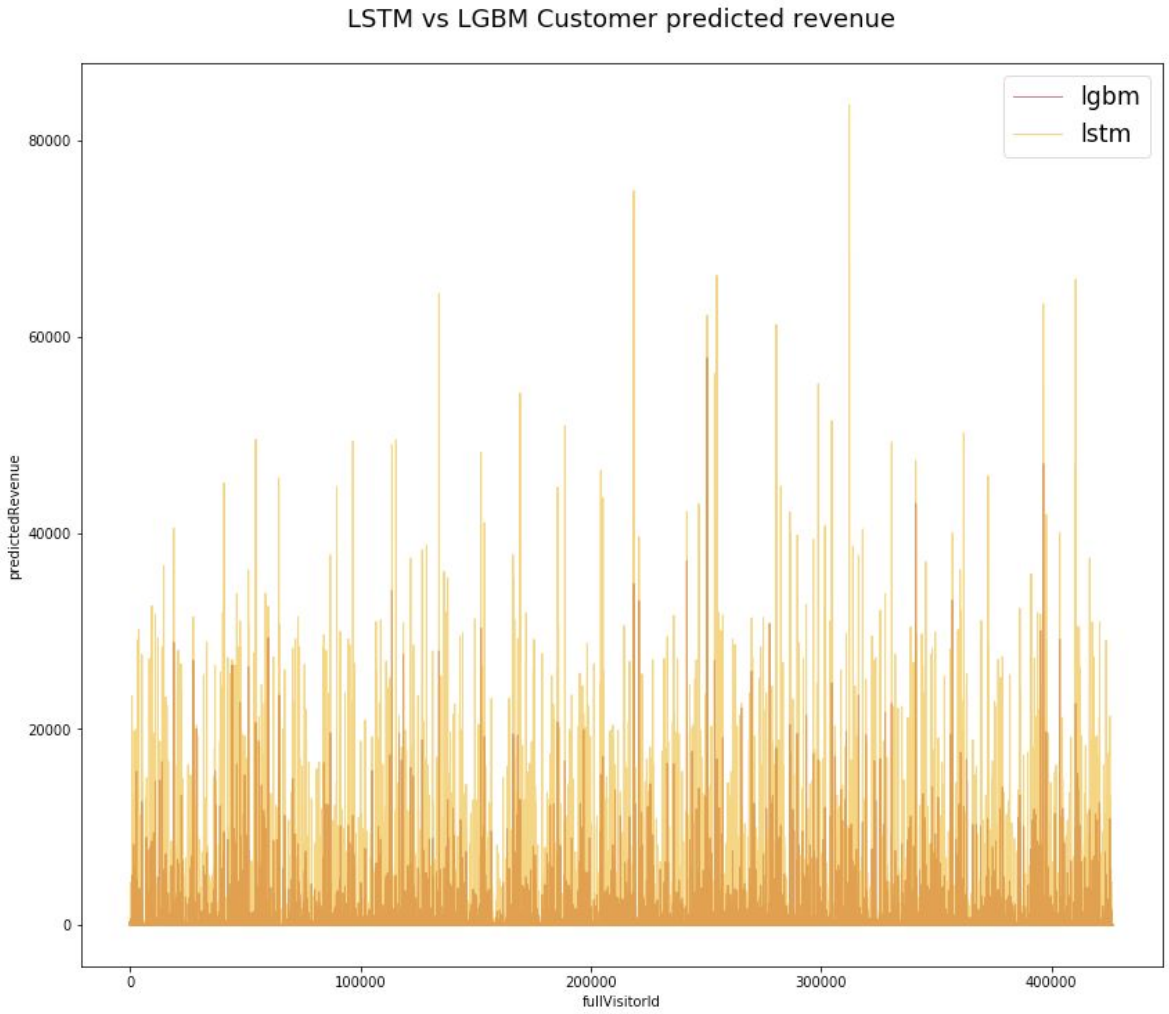
Refinement

As mentioned in the benchmark section, the LightGBM model achieved a RSME of 1.8654. Even after adjusting the parameters(e.g. Learning rate, number of layers, hidden units etc) the LSTM model has only achieved similar RSME of 1.8613 when compared to the benchmark model.

Results

During development, a validation set was used to evaluate the model. The models were compared in 4_Model_Comparision notebook.

The final architecture and hyperparameter were chosen because they performed the best among the tried combinations. To verify the robustness of the final model, the inferred results on the test data set is compared with the benchmark model results. LSTM model predicted revenue for more customers than the LightBGM model did. The above observations can be visualized in the below:



Justification

Even though both models have almost the same RSME, the LSTM model has predicted more revenue for customers than LBGM predicted for 426712 customers. So out of 617242 customers LSTM model predicted better or same for 69.13% of the customers.

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and related datasets were found
2. The data was downloaded and preprocessed
3. A benchmark was created for the regressor
4. The regressor was trained using the data multiple times until a good set of parameters were found
5. The Keras LSTM Model was created and trained
6. The results of both the models were compared

I found step 3 and 4 the most difficult, as I had to familiarize myself with the files of the LightGBM and Keras LSTM, both of which were the technologies that I was not familiar with before the project.

As far as most interesting aspects of the project, I am very glad that I found the competition in kaggle. I am sure I will be working on more competitions in future. I am also happy to use Tensorflow, Keras libraries.