Capstone Project

Sandeep Deshpande

Machine Learning Nano Degree

November 6, 2018

Google Analytics Customer Revenue Prediction

Definition

Project Overview

Google Analytics Customer Revenue Prediction is a Kaggle competition to predict how much GStore customers will spend on buying Google branded merchandise. Lot of marketing dollars are spent on retaining customers, and on understanding customers spending habits. The information gleaned from understanding customer spending habits will help the marketing teams with marketing budget allocation. Revenue prediction also helps with retention of customers by providing sales and promotions.

This competition is sponsored by R Studio and Google Cloud and developers are challenged to demonstrate the impact by using machine learning.

In my research I have noticed that many people have worked on similar projects, one interesting project was Revenue Forecasting for Enterprise Products (Bansal, n.d.); another one is on Municipal Revenue Prediction by Ensembles of Neural Network and Support Vector Machine (PETR HÁJEK, n.d.) and one on predicting stock prices, Equity forecast: Predicting long term stock price movement using machine learning (Milosevic, n.d.). I also came across a paper which used CNN for Sales forecast, Sales Forecast in E-commerce using Convolutional Neural Network (Wang, n.d.)

Problem Statement & Metrics

The criterion for this competition is to predict the natural log of the sum of all transactions per user. For every user in the test data, the target is to calculate

$$y_{user} = \sum_{i=1}^{n} transaction_{user_i}$$

$$target_{user} = \ln(y_{user} + 1)$$

The goal of the project is to calculate root mean squared error. Since this is a regression problem, I will run various algorithms and find lowest value for RMSE.

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

Analysis

Data Exploration

The data for the competition is provided by Google Merchandise store (GStore) on Kaggle. The train.csv file has 903653 and 12 features. For the capstone project I am using partial data from train.csv file i.e., the last 3 months of data starting from May 1, 2017 till the end August 1, 2017, it is 203,317 rows and 12 features. Each row in the dataset corresponds to one visit to the store and each user is uniquely identified by fullVisitorId field. The table below shows the sample data.

	channelGrouping	date	device	fullVisitorId	geoNetwork	sessionId	socialEngagementType	totals	trafficSource	visitId	visitNumber	visitStartTime
0	Organic Search	20160902	{'browser': 'Chrome', 'browserVersion': 'not a...	1131660440785968503	{'continent': 'Asia', 'subContinent': 'Western...	1131660440785968503_1472830385	Not Socially Engaged	{'visits': '1', 'hits': '1', 'pageviews': '1',...	{'campaign': '(not set)', 'source': 'google', ...	1472830385	1	1472830385
1	Organic Search	20160902	{'browser': 'Firefox', 'browserVersion': 'not ...	3773060208779278890	{'continent': 'Oceania', 'subContinent': 'Aust...	3773060208779278890_1472880147	Not Socially Engaged	{'visits': '1', 'hits': '1', 'pageviews': '1',...	{'campaign': '(not set)', 'source': 'google', ...	1472880147	1	1472880147
2	Organic Search	20160902	{'browser': 'Chrome', 'browserVersion': 'not a...	3895546263509774583	{'continent': 'Europe', 'subContinent': 'South...	3895546263509774583_1472865386	Not Socially Engaged	{'visits': '1', 'hits': '1', 'pageviews': '1',...	{'campaign': '(not set)', 'source': 'google', ...	1472865386	1	1472865386

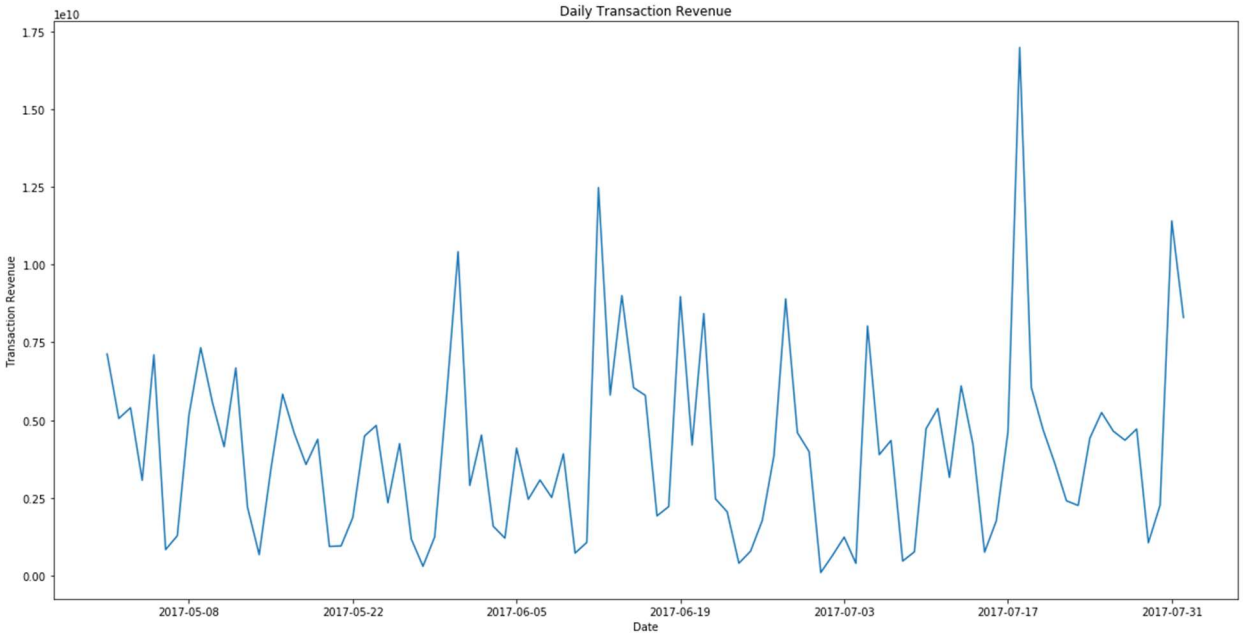
Data Fields

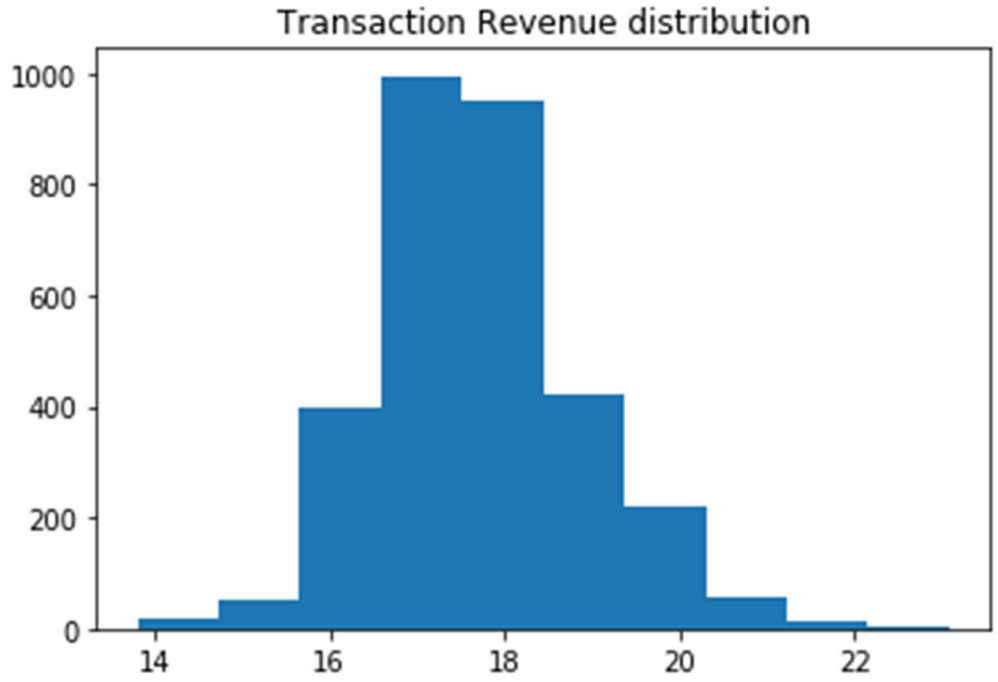
- *channelGrouping* – (categorical) The channel via which the user came to the Store.
- *date* – (date) The date on which the user visited the Store.
- *device* – (JSON) The specifications for the device used to access the Store.

- *fullVisitorId*- (numeric need to be treated as string) A unique identifier for each user of the Google Merchandise Store.
- *geoNetwork* – (JSON) This section contains information about the geography of the user.
- *sessionId* – (numeric need to be treated as string) A unique identifier for this visit to the store.
- *socialEngagementType* - (categorical) Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- *totals* – (JSON) This section contains aggregate values across the session.
- *trafficSource* – (JSON) This section contains information about the Traffic Source from which the session originated.
- *visitId* - (numeric need to be treated as string) 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* – (Numeric) The session number for this user. If this is the first session, then this is set to 1.
- *visitStartTime* – (Numeric) The timestamp (expressed as POSIX time).

There are four features (device, geoNetwork, totals and trafficSource) that have JSON blobs in them, after extracting the JSON blobs we have 55 features in the data set. One of the extracted features is totals_ transactionRevenue that has revenue information and that is what I need to predict.

On further investigation of totals_ transactionRevenue feature in the full data set I discovered that 98.73% of rows are null. Since I am using partial data, I analyzed the partial dataset and found that the subset data also has 98.46% null values. The distribution of revenue data between the full data set and partial data set is not significantly different. To better visualize the revenue data, we have excluded rows with zero value in transactionRevenue, and plotted log of daily transaction revenue and log of transaction revenue distribution below. The distribution chart shows a normal distribution.





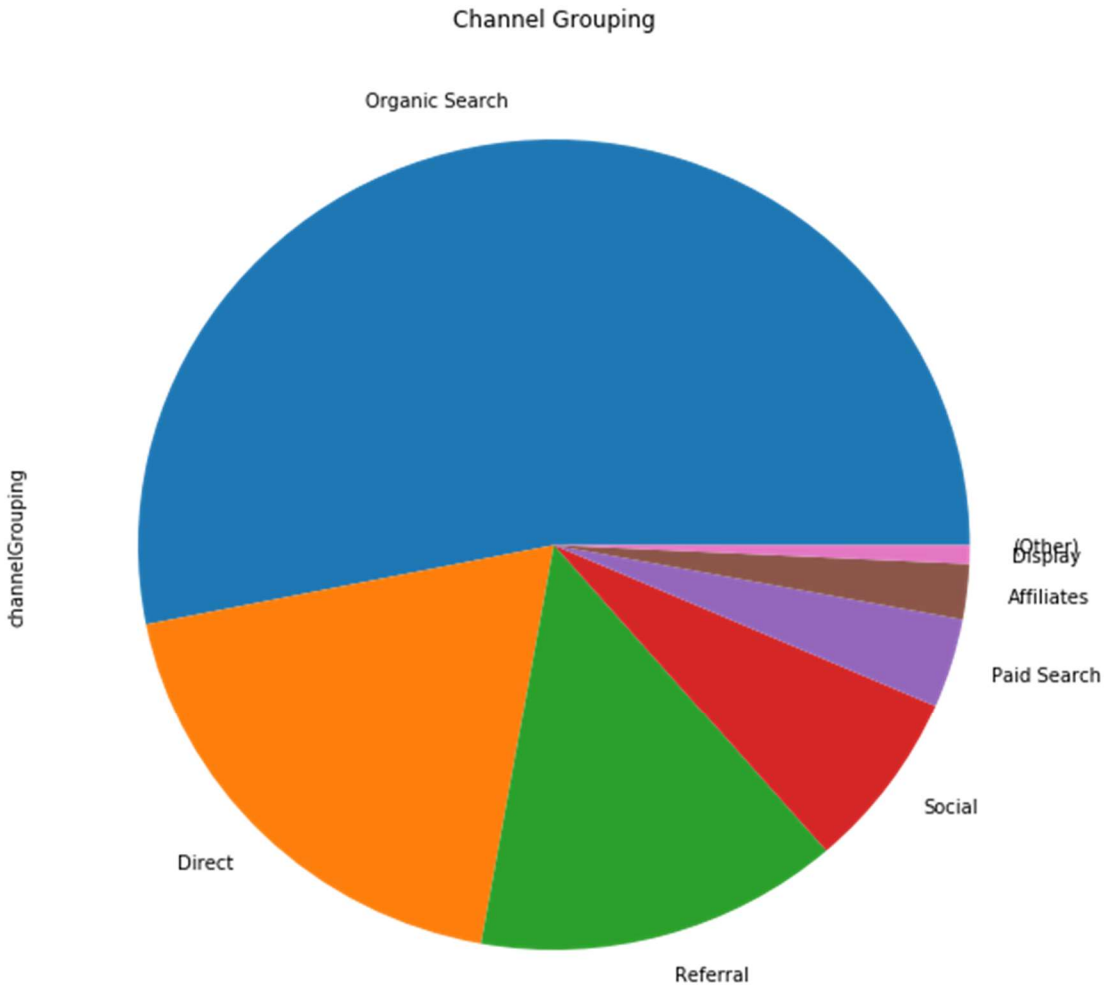
The table below shows descriptive statics of the data that I will use to predict totals_transactionRevenue. The data below shows that most of the rows have zero in totals_transactionRevenue field.

	DATE	VISITID	VISITNUMBER	VISITSTARTTIME	TOTALS_TRANSACTIONREVENUE
COUNT	203317.000000	203317.000000	203317.000000	203317.000000	203317.000000

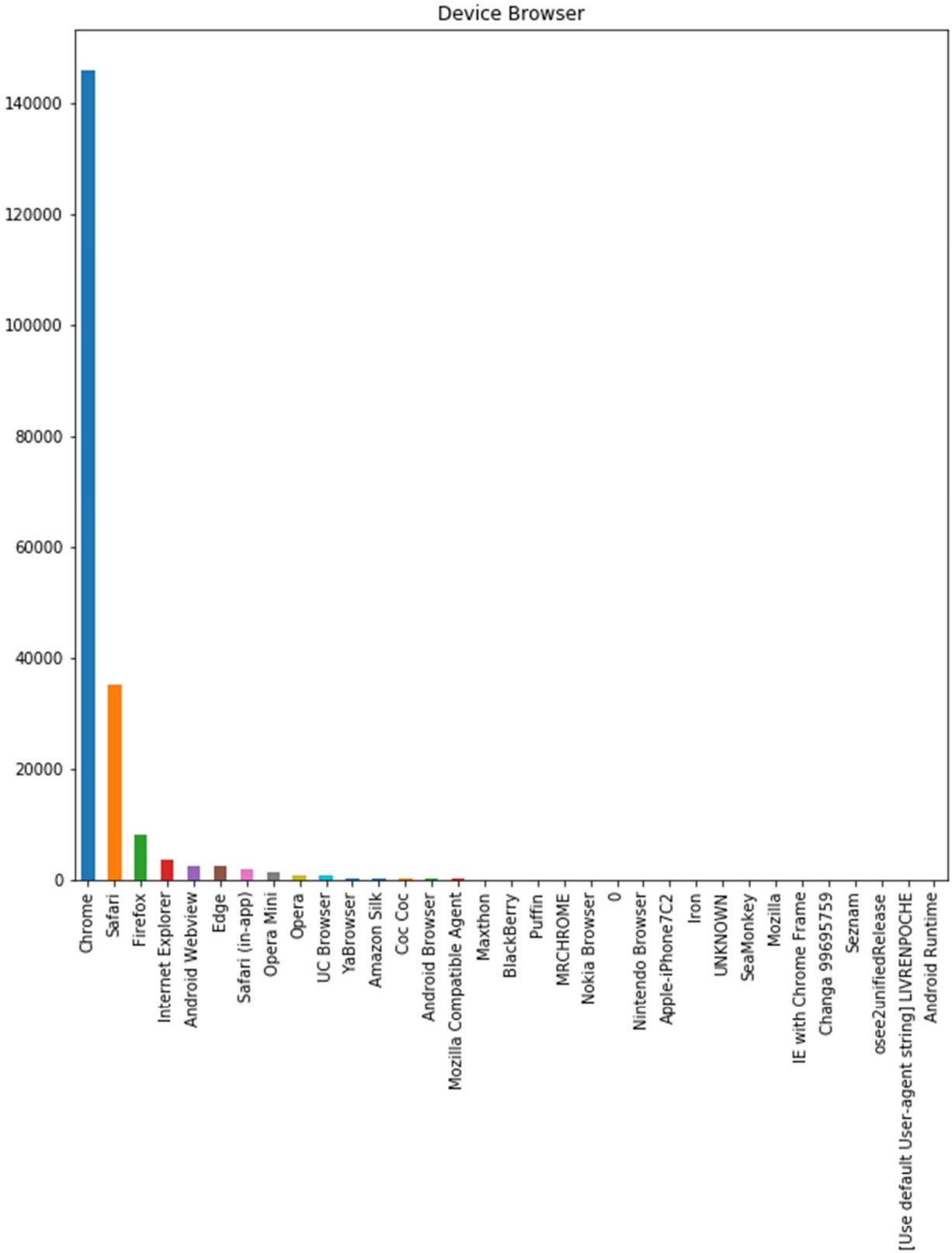
MEAN	20170621.358032	1497731978.155648	2.248248	1497731979.554523	1849747.881387
STD	85.156229	2347706.884221	8.850502	2347706.786266	46098374.370036
MIN	20170501.000000	1493620320.000000	1.000000	1493622000.000000	0.000000
25%	20170524.000000	1495654230.000000	1.000000	1495654230.000000	0.000000
50%	20170618.000000	1497840895.000000	1.000000	1497840895.000000	0.000000
75%	20170711.000000	1499791184.000000	1.000000	1499791184.000000	0.000000
MAX	20170801.000000	1501657193.000000	395.000000	1501657193.000000	10589140000.000000

Exploratory Visualization

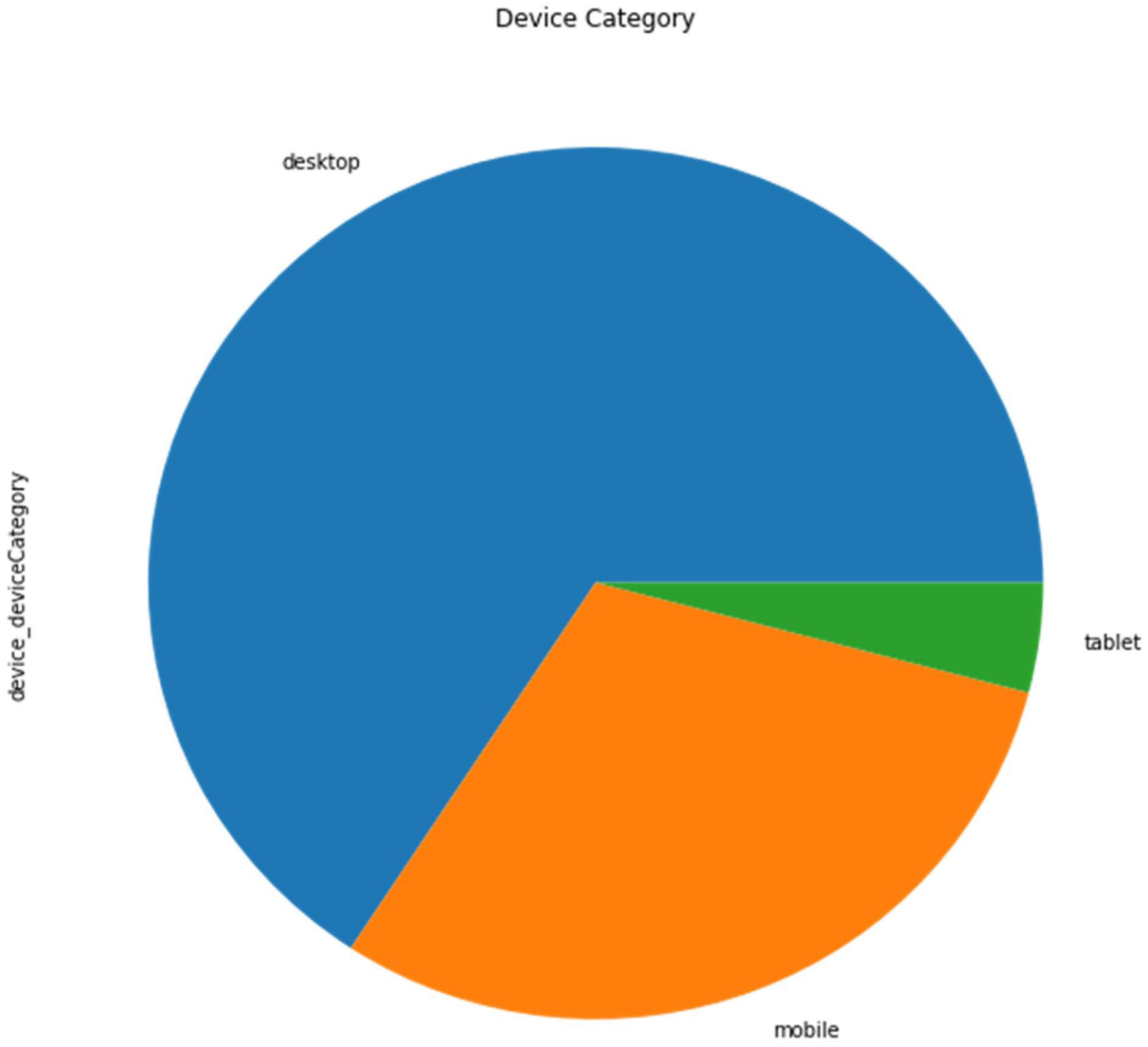
I started by visualizing Channel Grouping feature of the data. The below graph helps to see the source of traffic to the store. More than 50% of the traffic originates as Organic Search followed by Direct channel.



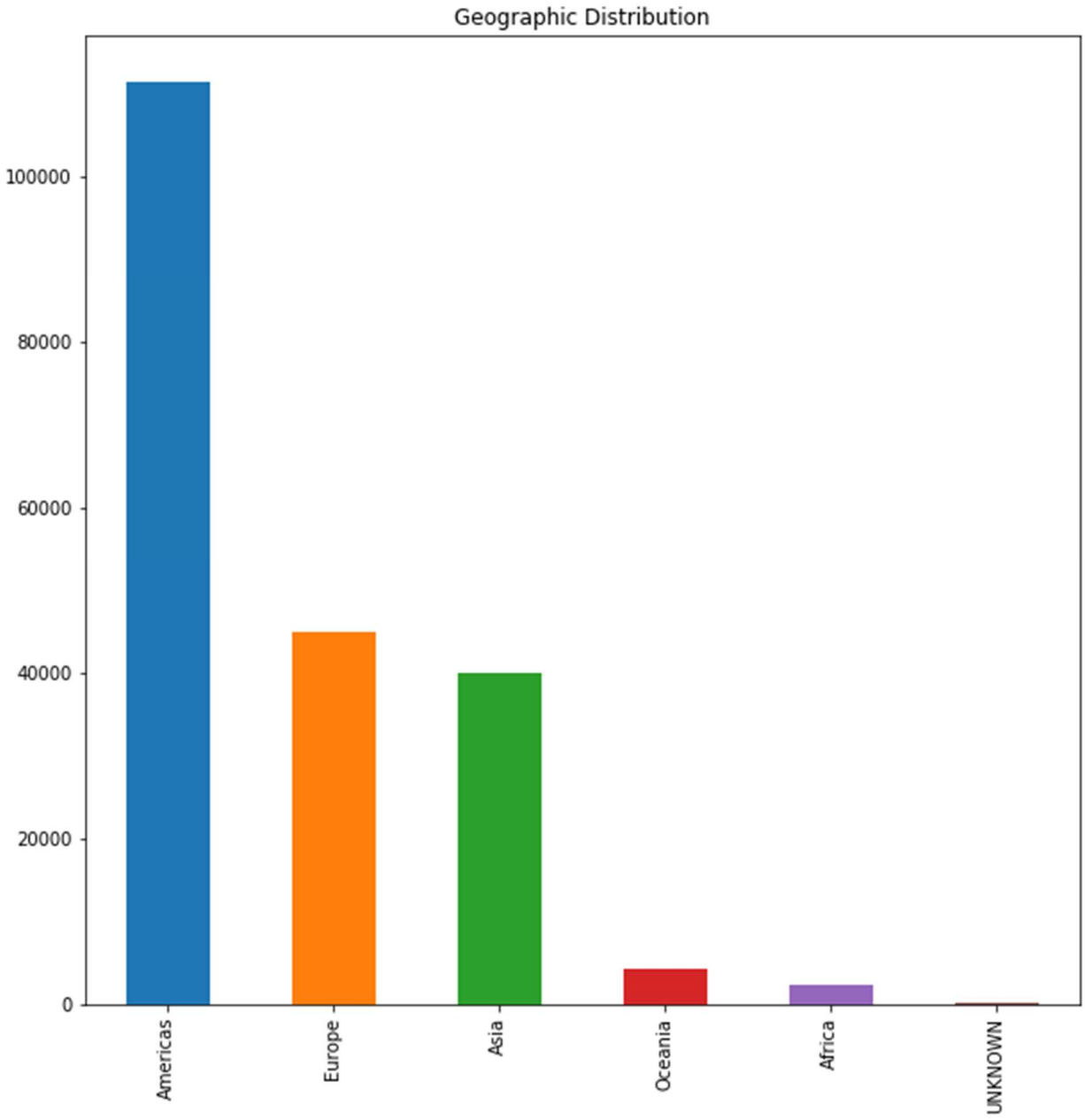
After looking at Channel Grouping, I visualized the traffic from device browsers. Chrome, Safari and Firefox are top three browsers from where traffic is originating. Traffic from Internet Explorer and Edge combined is close to the traffic from Firefox.


Device Browser
140000
120000
100000
80000
60000
40000
20000
0
Chrome
Safari
Firefox
Internet Explorer
Android Webview
Edge
Safari (in-app)
Opera Mini
Opera
UC Browser
YaBrowser
Amazon Silk
Coc Coc
Android Browser
Mozilla Compatible Agent
Maxthon
BlackBerry
Puffin
MRCHROME
Nokia Browser
0
Nintendo Browser
Apple-iPhone7C2
Iron
UNKNOWN
SeaMonkey
Mozilla
IE with Chrome Frame
Changa 99695759
Seznam
osee2unifiedRelease
[Use default User-agent string] LIVRENPOCHE
Android Runtime

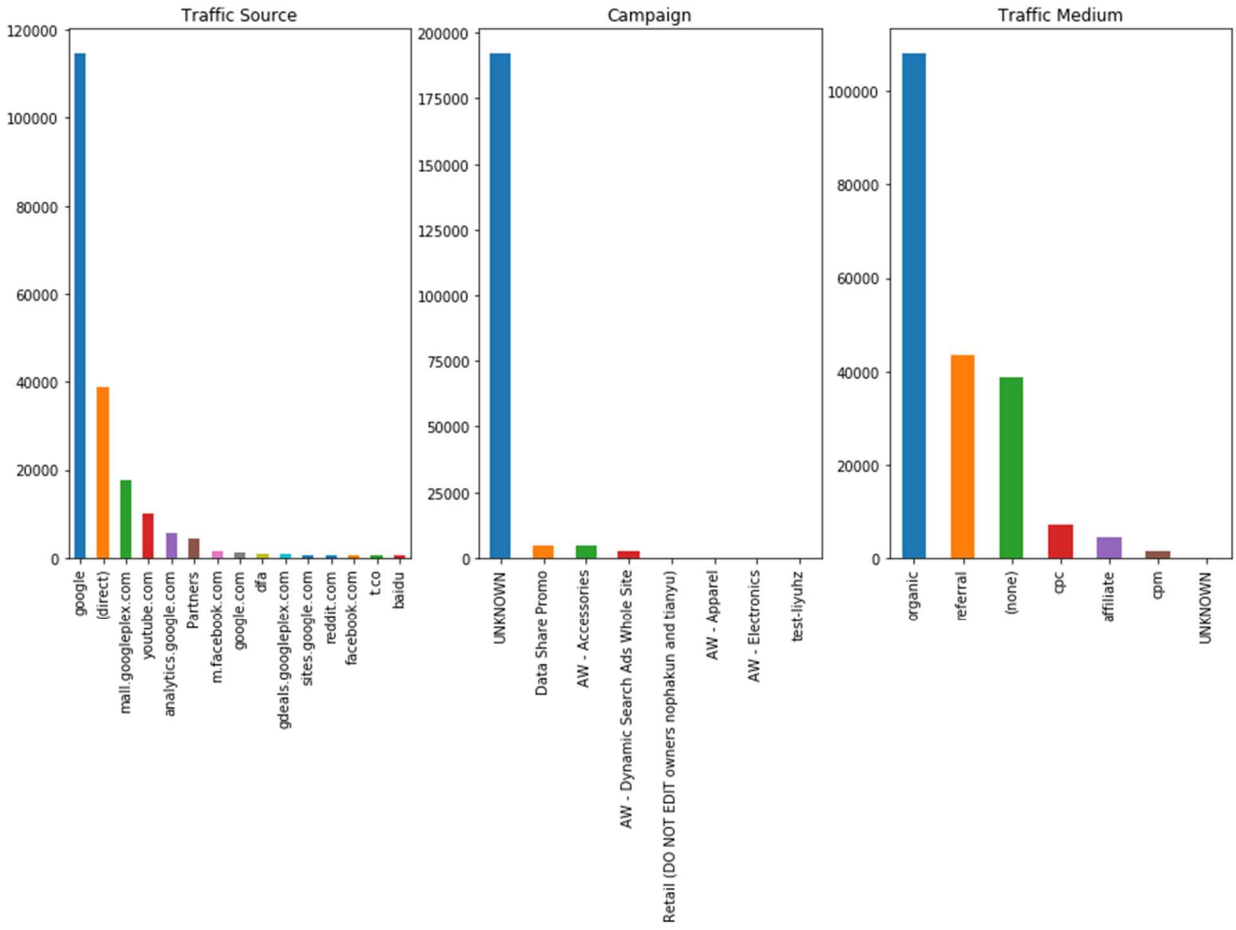
In the next graph I explore the traffic from Device Category. As you can see from the figure below most of the traffic is coming from Desktop, followed by Mobile and Tablet.



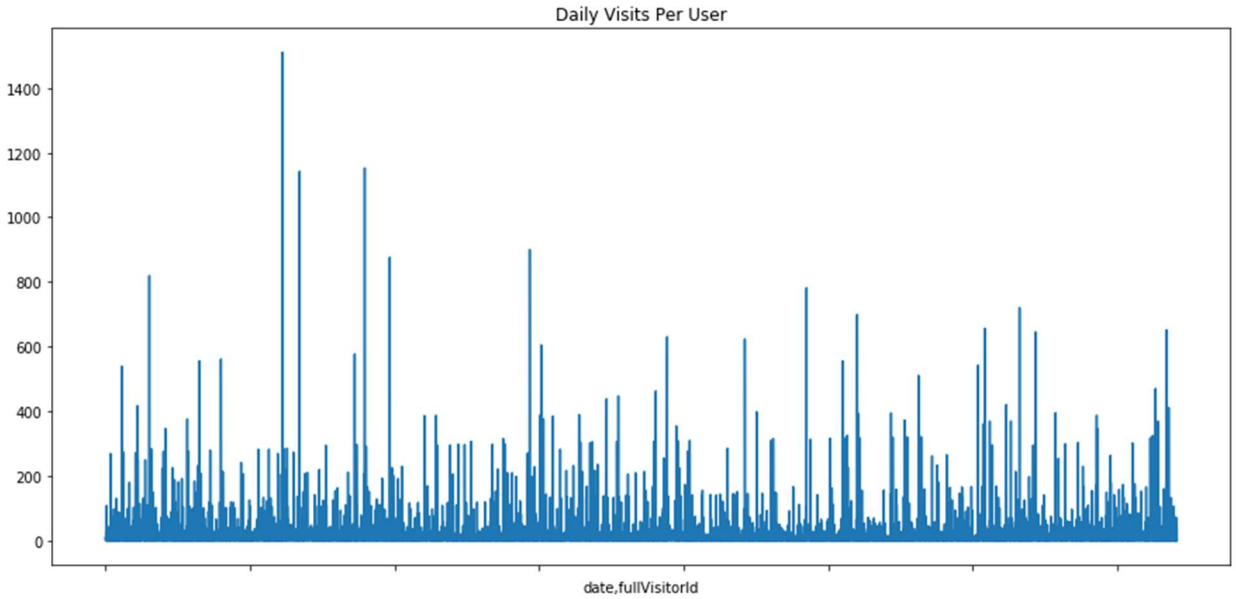
In the next graph I look at the traffic patterns from geographic perspective. The figure below shows the traffic patterns originating from different continents.

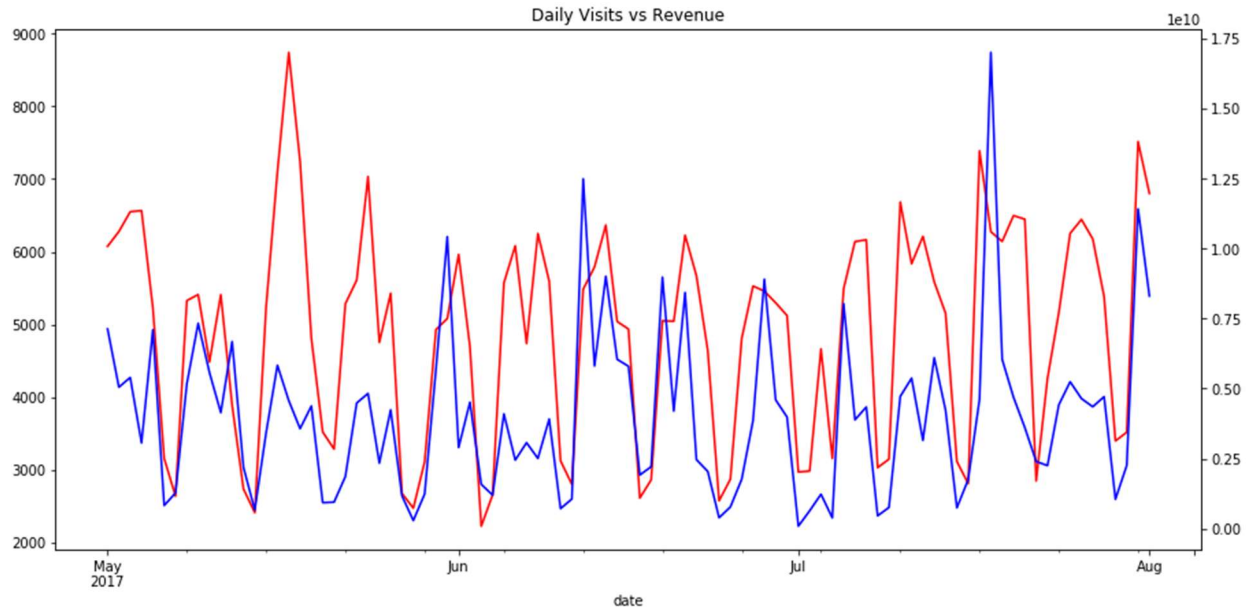


In the next set of graphs, I look at trafficSource related features like Source, Campaign and Medium. These graphs reveal the source of traffic, marketing campaign that is driving the traffic and medium from where the traffic is originating.



Finally, in last two set of graphs I visualize the daily visits per user and then corelate the daily visits with revenue earned. All these visualizations help us see how the data has impact on the revenue of the site.





Algorithms and Techniques

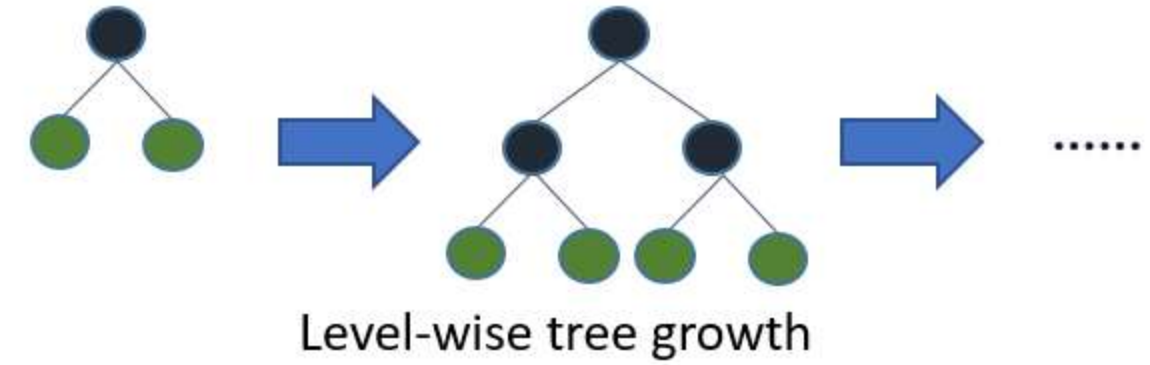
I have used the following algorithms to predict and compare the transactionRevenues and RMSE. This is a supervised learning regression problem as we predict revenues for each user who visits the web site.

Random forest is a type of supervised machine learning algorithm that is based on ensemble learning. The random forest algorithm combines multiple algorithms of the same type i.e. multiple decision *trees*, resulting in a *"forest of trees"*. Random forest is very stable and works well even if some data is changed or is missing. However, it is a complex algorithm and take up more computational resources and time to train compared to other comparable algorithms.

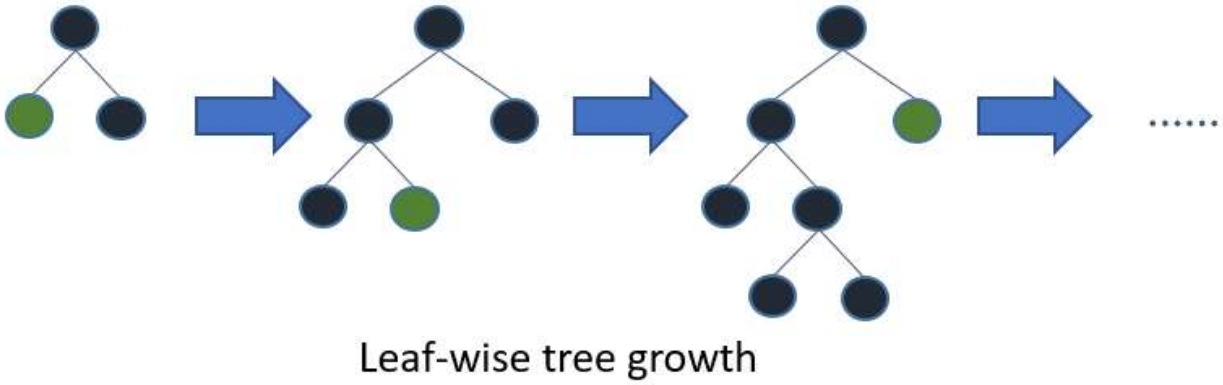
XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the gradient boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems accurately and quickly.

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. So, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy compared to other boosting algorithms. Light GBM is very fast, hence the word 'Light'.

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



Level-wise tree growth in XGBOOST.



Leaf wise tree growth in Light GBM.

Leaf wise splits lead to increase in complexity and may lead to overfitting and it can be overcome by specifying another parameter max-depth which specifies the depth to which splitting will occur.Light GBM uses less memory, has a faster training speed, higher efficiency, and is accurate. It can be used on large data sets.

Linear regression is a useful and widely used statistical learning method used for supervised learning. Linear regression is used to predict a quantitative response (Y) from the predictor variable (X). Linear Regression is used assuming that there's a linear relationship between X and Y.

AdaBoost, or "Adaptive Boosting", is a type of ensemble service algorithm that can be used in conjunction with many other types of learning algorithms to improve their performance. AdaBoost is sensitive to noisy data and outliers.

Support vector machines (SVMs) are a set of supervised learning methods used for regression, classification and outlier detection. SVM is effective in high dimensional spaces, it uses subset of training points in the decision function, it also memory efficient.

Benchmark

For my initial benchmark, I have used Random Forest Regressor algorithm. My goal in this project is to get lowest possible value of root mean squared error (RMSE) as stated by Kaggle Competition. The initial bench mark value was 1.7667. To optimize the performance, I ran GridSearchCV and I had a slight improvement in RMSE value 1.7625.

Next, I ran various algorithms to compare their performance with benchmark including ensemble and tree-based models. The summary of the results is given below.

Algorithms	**RMSE**
RandomForestRegressor (Initial)	1.7667
RandomForestRegressor (GridSearchCV)	1.7625
LinearRegression	1496720742.7189
XgBoost	1.7293
LightGBM	1.7194
AdaBoostRegression	2.3122
SVM Regressor	2.2041
LightGBM (GridSearchCV)	1.7240

Two things that standout from the results above are that Linear Regression for some reason gives very large value, so we can ignore it for this project. The second thing we can observe is that LightGBM Regressor algorithm gives lowest RMSE value. I tried fine tuning the parameters of LightGBM using GridSearchCV and there was no improvement in RMSE value.

Methodology

Data Processing & Cleaning

In this section I will prepare data for processing, first by deleting those features that have null values in more than 50% of their rows. Some of the features have values set to 'not available in demo dataset', I have deleted these columns as well. I have also deleted 'socialEngagementType' feature as it has only one value, which is not very helpful for predicting revenue. The 'trafficSource_source' categorical feature has domains and subdomains from where traffic originated. I have merged the subdomain and replaced with the main domain of the site e.g (mail.google.com, finance.google.com where replaced by google). Finally, I deleted those features that do not add any value to our regression such as 'visitId', 'fullVisitorId' etc. The cleaned data had 22 features in them.

	channelGrouping	visitNumber	device_browser	device_deviceCategory	device_isMobile	device_operatingSystem	geoNetwork_city	geoNetwork_continent	geoNetwork_country	geoNetwork_subContinent	...
4822	Direct	1	Chrome	mobile	True	Android	UNKNOWN	Asia	Turkey	Western Asia	...
4823	Organic Search	1	Chrome	desktop	False	Macintosh	Paris	Europe	France	Western Europe	...
4824	Direct	1	Chrome	desktop	False	Chrome OS	Amsterdam	Europe	Netherlands	Western Europe	...
4825	Organic Search	1	Chrome	desktop	False	Windows	UNKNOWN	Asia	China	Eastern Asia	...
4826	Organic Search	1	Safari	tablet	True	iOS	UNKNOWN	Americas	United States	Northern America	...

5 rows × 22 columns

I normalized the numerical features by calling MinMaxScaler from Skit Learn library. For the categorical features I did One-hot encoding by calling pandas.get_dummies() functions. After one-hot encoding the features, I had 668 columns in my data frame. Finally, before splitting data for training and testing purpose I had to cleanup the column names that had "[","]", and "<" and replace these characters with "_". The three characters caused errors while executing XgBoost Algorithm.

As stated in my proposal I split the data 80% for training and 20% for testing randomly by calling train_test_split function from skit learn. The size of training sample was 162,653 and the size of testing sample was 40,664.

Implementation

I start with loading the dataset that was acquired from Kaggle, there are four fields with JSON blob that have to be flattened to extract all the features. Once the data is loaded, I performe exploratory analysis of the data to see the distribution of revenue feature. I then proceeded to clean the data, as part of data cleanup, I replaced null values in transactionRevenue field with zero values. I also deleted columns that have null values in majority fields. I also deleted a set of columns that did not have any meaningful data and fixed some of the values in the fields to prepare for running the model.

Before starting the work on model evaluation, I split the data into training 80% and testing 20%. This being a supervised learning problem I have trained some tree-based models and some ensemble models.

I have split the implementation in two stages. In the first stage I run the RandomForestRegressor to get the initial benchark, then I fine tune it using GridSearchCV. As the stated goal for this competition is to get lowest RMSE, I get a slight improvement in the RMSE values.

In the second stage I run various models like Linear Regression, XgBoost Regression, AdaBoost Regression, Support Vector Machine and Light GBM Regression. I choose Light GBM for further tuning as it had the lowest RSME value. I tuned LGBM model's hyperparameters using GridSearchCV. I focused on tuning Learning Rate, Bagging Fraction, and Number of Leaves as I wanted to improve accuracy while maintaining speed.

Refinement

I performed tuning of hyperparameters of Light GBM using GridSearchCV from SciKit Learn and the values are shown below in a table.

Parameter	Objective	Tested Values	Best Values
learning_rate	Better Accuracy	[0.05,0.1, 1]	0.1
num_leaves	Better Accuracy	[100, 200, 300]	200
bagging_fraction	Faster Speed	[0.3, 0.7]	0.3

The results of running the initial unoptimized light gbm model where better than the results of optimized model. There was a slight increase in RMSE value, the initial RMSE value being 1.7194 and optimized models RMSE value being 1.7240. There is a need to further tune the model, as I continue to work towards the competition, I will work on tuning other hyperparameters like max_bin, bagging_freq to improve the results.

Results

Model Evaluation and Validation

The competitions submissions are scored on lowest RMSE Value. Our goal for this project was to find a model that would help in achieving the lowest RMSE value. I started by applying Random Forest out of the box and then subsequently applied many of the supervised machine learning algorithms mentioned above. The metrics and fine tuning of algorithm is calculated and performed using SicKit learn library. The Light GBM solution that I selected performed satisfactorily compared to the baseline algorithm. There is a 2.68% improvement in RMSE Value. Below is the code snippet and result from LGBM model that had lowest RMSE Value.

```
In [42]: import lightgbm as lgbm

d_train = lgbm.Dataset(X_train, label=y_train)
d_test = lgbm.Dataset(X_test, label=y_test, reference=d_train)

params = {}
params['learning_rate'] = 0.1
params['bagging_frequency'] = 10
params['objective'] = 'regression'
params['metric'] = 'rmse'
params['bagging_fraction'] = 0.7
params['num_leaves'] = 50
params['feature_fraction'] = 0.50
params['max_depth'] = 10
params['min_child_samples'] = 100
params['verbosity'] = 1

clf = lgbm.train(params, d_train, 1000, d_test, verbose_eval=100, early_stopping_rounds=200)

Training until validation scores don't improve for 200 rounds.
[100]   valid_0's rmse: 1.72233
[200]   valid_0's rmse: 1.72024
[300]   valid_0's rmse: 1.72208
[400]   valid_0's rmse: 1.72605
Early stopping, best iteration is:
[212]   valid_0's rmse: 1.7194
```

Justification

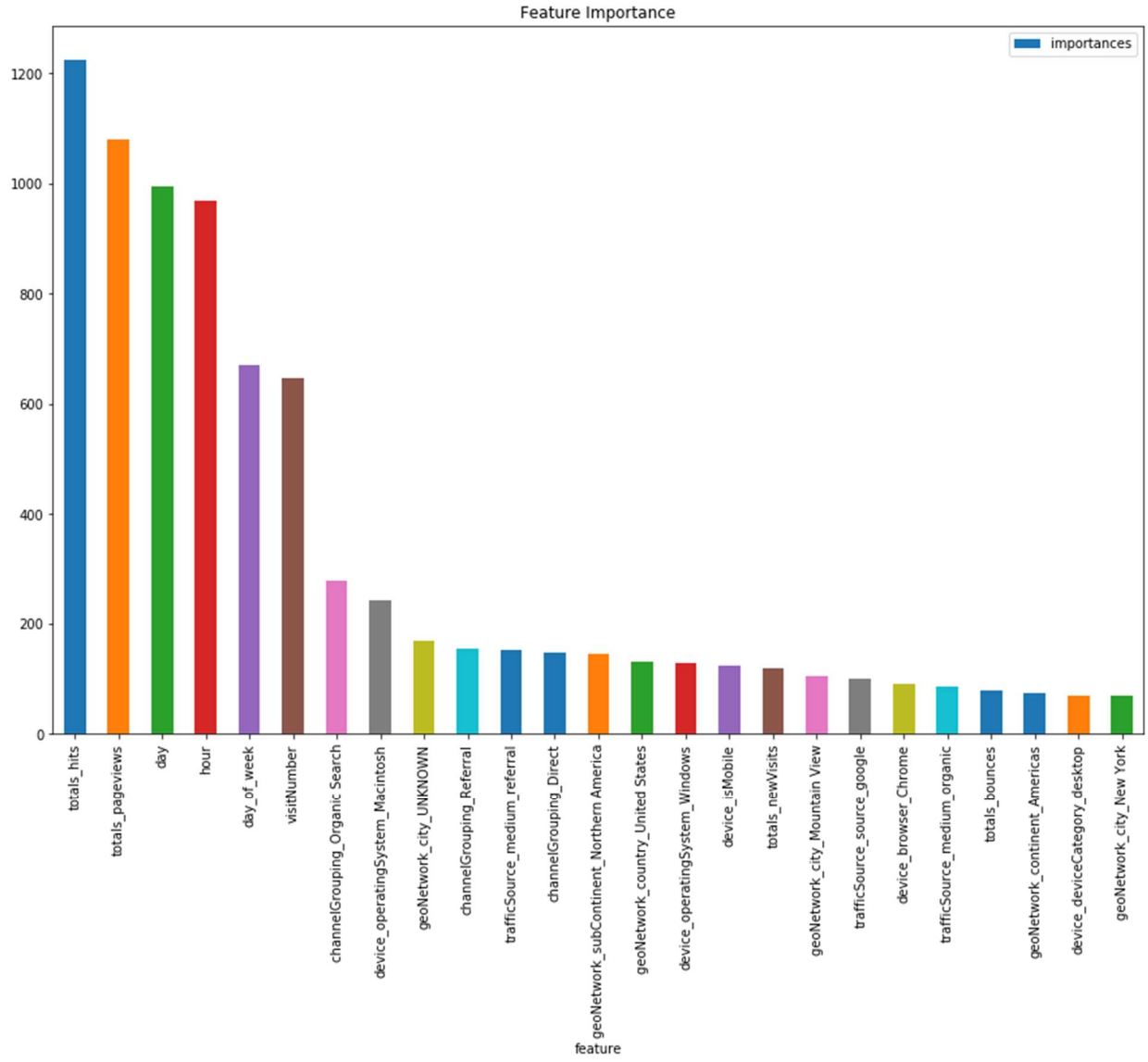
Of all the models I used LightGBM performed the best both in terms of speed and accuracy compared to all other algorithms. Fine tuning LightGBM model did not improve the RSME Values significantly. There is room for improvement. As I continue to work on the competition, I will work on further fine tuning the hyperparameters.

	Benchmark Model (Random Forest)	Untuned Model (Light GBM)	Tuned Model (Light GBM)
RSME	1.7667	1.7194	1.7240

Conclusion

Free-Form Visualization

The feature importance is visualized in the figure below.



The importance of features like totals_hit, totals_pageviews, day, hour, day_of_week, visitNumber, channelGrouping and device_operatingSystem is much higher that rest of the features.

Reflection

The most important aspect of this project was to understand and clean the data. The data available from Kaggle had JSON blobs, I had to extract these JSONs to understand the data. Many of the columns had NaN values or a string stating data was ‘not set’ or ‘not available in demo dataset’. Once I removed and cleaned the data, data visualization became easy.

Categorical columns had large set of values that needed to be encoded. One-hot encoding resulted in creating 668 entries. This ended up slowing down some of the algorithms significantly. The worst case

was SVM which took 2 days to complete. Given the time constraint I was unable to run a deep neural network model.

Improvements

There are couple of improvements that I would like to make. On the data front I want to explore a different encoding mechanism like LGBM Categorical encoding, in this project I have used one-hot encoding using SciKit learn library that resulted in over 668 entries. If I can reduce the number of features that would significantly improve the speed of running different algorithms. Another improvement that I want to work on is fine-tuning the LGBM algorithm to improve the RSME Values. There are additional parameters that I could change to improve the accuracy but I should also be cautious about not reducing speed and overfitting. Finally, I want run a deep neural network model to train and predict the transactionRevenues.

References

https://www.kaggle.com/c/ga-customer-revenue-prediction
https://en.wikipedia.org/wiki/Marketing_mix
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html
https://www.analyticsvidhya.com/blog/2015/05/data-visualization-python/
https://scentellegher.github.io/programming/2017/07/15/pandas-groupby-multiple-columns-plot.html
https://matplotlib.org/examples/api/two_scales.html
https://lightgbm.readthedocs.io/en/latest/Features.html
https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/
https://xgboost.ai/about
https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/
https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06
https://en.wikipedia.org/wiki/AdaBoost
https://scikit-learn.org/stable/index.html
https://scikit-learn.org/stable/modules/svm.html#svm-regression