PINGILI LOHITH

MACHINE LEARNING NANODEGREE

CAPSTONE PROJECT

DEC 25TH, 2018

Definition

Project Overview:

The Pareto principle (also known as the 80/20 rule) states that, for many events, roughly 80% of the effects come from 20% of the causes. The 80/20 rule has proven true for many businesses– only a small percentage of customers produce most of the revenue. It's a challenge in such businesses to invest in promotional strategies that target the small fraction of customers who drive the revenue.

Related research:

https://hbr.org/2017/02/ai-is-going-to-change-the-8020-rule

https://www.swrve.com/images/uploads/whitepapers/swrve-monetization-report-2016.pdf

Problem Statement:

Predict revenue per customer by analyzing a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset.

As the problem statement requires the prediction of revenue per customer, the solution is a regression model. Various regression algorithms like Linear Regression, Decision Tress, bagging, XGBoost should be applied to find the best performing algorithm.

Note: This project is from the a Kaggle competition. The initial data was changed as there was a data leakage. As I already started working on the data and my proposal was approved I used the initial data for my capstone project. The original data from Kaggle had train and test sets with the 'y' variable for test set held back by Kaggle for evaluation. Because of the change in competition, I used only train data as the sole data source. The data used for my capstone project can be found here.

Solution Statement:

As the problem statement requires the prediction of revenue per customer, the solution is a regression model. Various regression algorithms like Linear Regression, SVM, Decision Trees, Adaboost, XGBoost should be applied to find the best performing algorithm.

Evaluation Metrics:

RMSE (Root Mean Squared error) is the evaluation metric used for model selection. RMSE is shows how close the predicted values are to the actual values. RMSE is calculated by the formula,

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

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. Natural log is taken as predicted variable is income and there's a wide distribution. Also '1' is added before natural log transformation as most of the predicted variables have a zero value.

R2 score is used to check the statistical robustness of the model. R2 score (Co-efficient of determination) gives a measure of the variance of the target variable that can be explained by the input variables. R2 score is calculated by the formula,

$$\sqrt{\frac{1}{N} * \sum_{i=1}^{N}(y_i \; - \; y_i')^2}$$

So, the metrics used are R2 score and RMSE. RMSE is chosen over MAE because RMSE penalizes difference in predicted value and actual value more than MAE. For this data set, where the range of 'y' variable is really high, and the data is highly imbalanced, RMSE is a better choice in evaluating the accuracy of the model. The chosen metrics will help in evaluating the robustness and accuracy of the model.

ANALYSIS

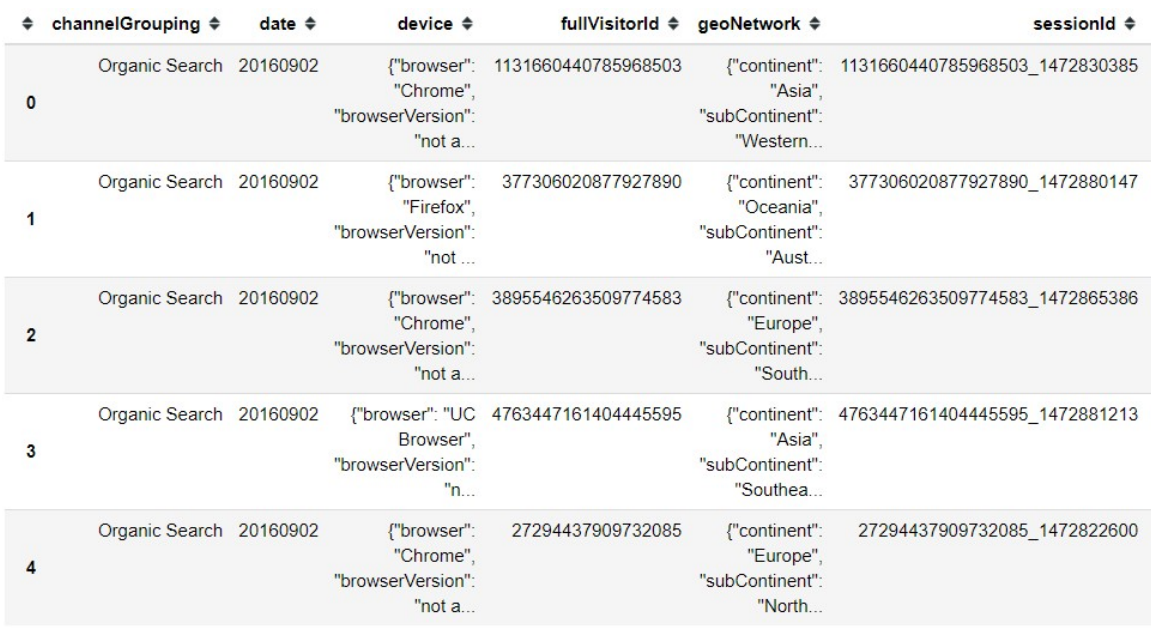
Data Exploration: Train dataset has 903653 rows with 12 columns. There are multiple sub columns for 4 of the attributes. The description of the attributes:

- 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.

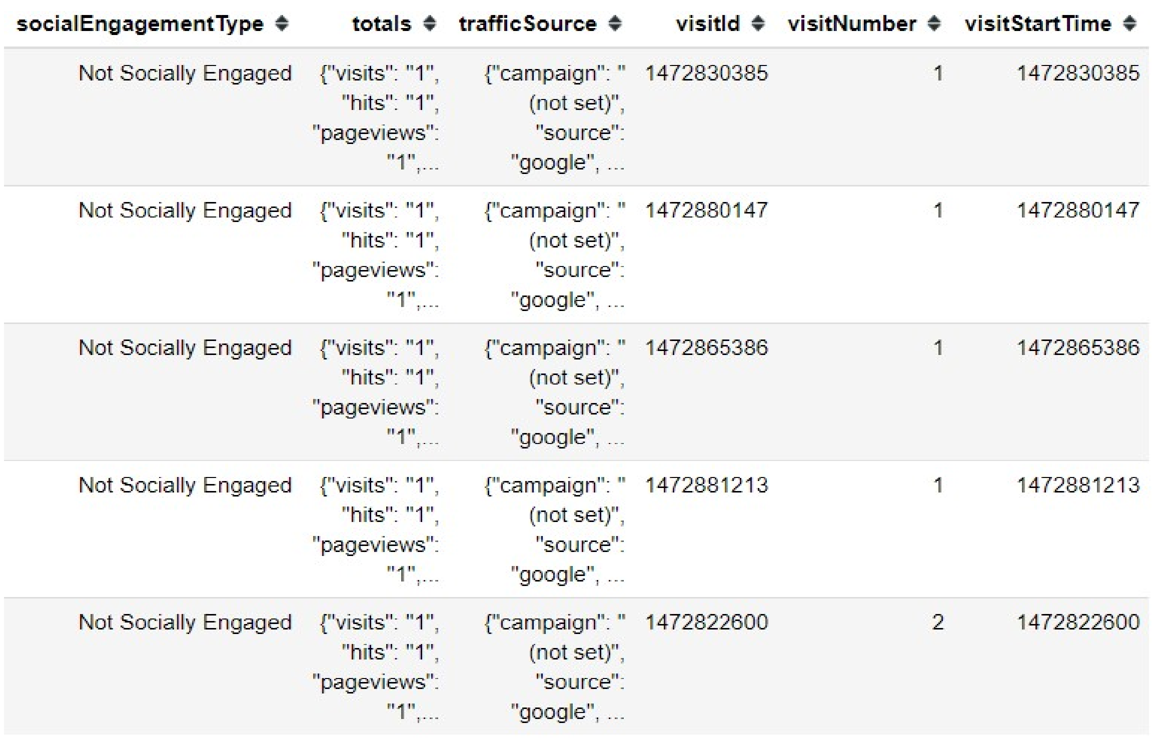
- visitId - An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.
- visitNumber - The session number for this user. If this is the first session, then this is set to 1.
- visitStartTime - The timestamp (expressed as POSIX time).

  ‘Transaction revenue’ sub-column from ‘Totals’ attribute summed per userID is the ‘y’ variable.

Sample of the data:



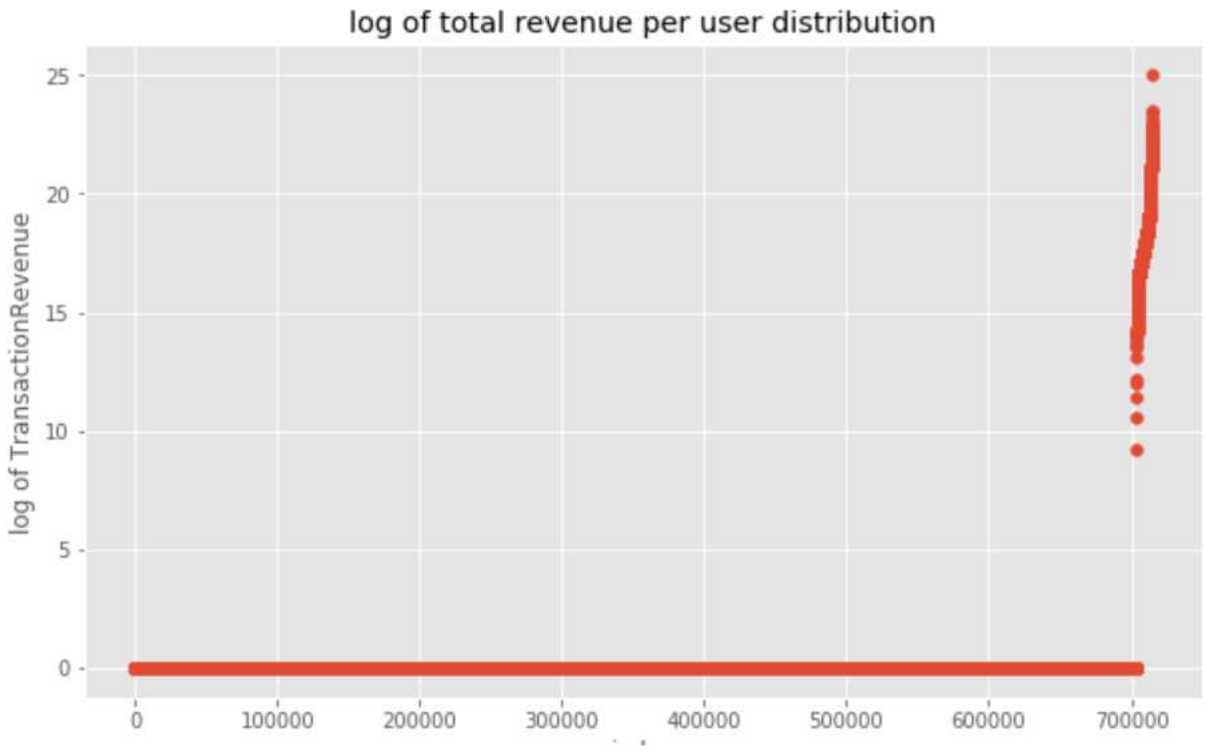
♦	channelGrouping ♦	date ♦	device ♦	fullVisitorId ♦	geoNetwork ♦	sessionId ♦
0	Organic Search	20160902	{"browser": "Chrome", "browserVersion": "not a...	1131660440785968503	{"continent": "Asia", "subContinent": "Western...	1131660440785968503_1472830385
1	Organic Search	20160902	{"browser": "Firefox", "browserVersion": "not ...	377306020877927890	{"continent": "Oceania", "subContinent": "Aust...	377306020877927890_1472880147
2	Organic Search	20160902	{"browser": "Chrome", "browserVersion": "not a...	3895546263509774583	{"continent": "Europe", "subContinent": "South...	3895546263509774583_1472865386
3	Organic Search	20160902	{"browser": "UC Browser", "browserVersion": "n...	4763447161404445595	{"continent": "Asia", "subContinent": "Southea...	4763447161404445595_1472881213
4	Organic Search	20160902	{"browser": "Chrome", "browserVersion": "not a...	27294437909732085	{"continent": "Europe", "subContinent": "North...	27294437909732085_1472822600



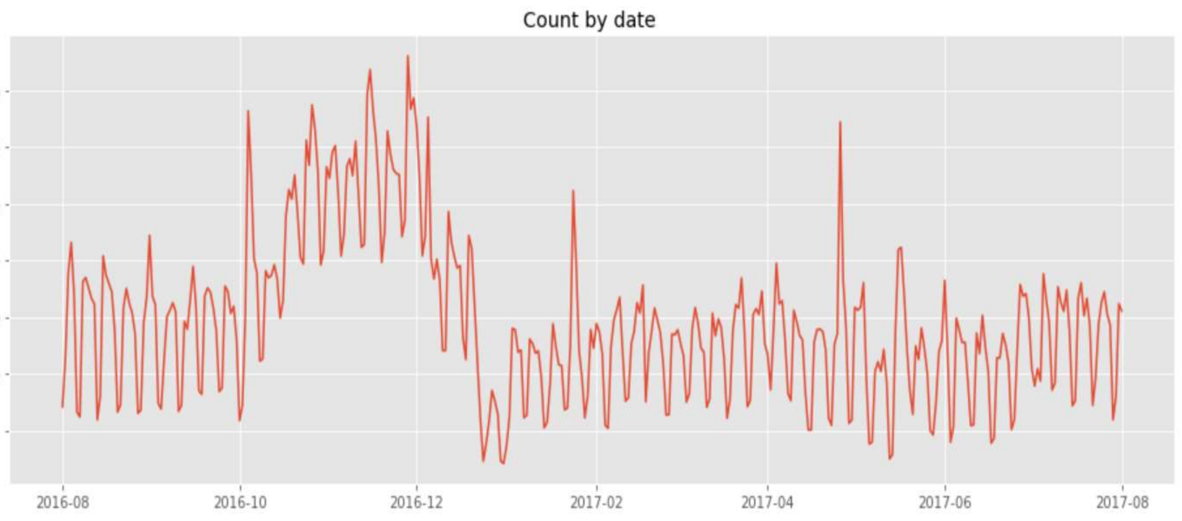
socialEngagementType ♦	totals ♦	trafficSource ♦	visitId ♦	visitNumber ♦	visitStartTime ♦
Not Socially Engaged	{"visits": "1", "hits": "1", "pageviews": "1",...	{"campaign": "(not set)", "source": "google", ...	1472830385	1	1472830385
Not Socially Engaged	{"visits": "1", "hits": "1", "pageviews": "1",...	{"campaign": "(not set)", "source": "google", ...	1472880147	1	1472880147
Not Socially Engaged	{"visits": "1", "hits": "1", "pageviews": "1",...	{"campaign": "(not set)", "source": "google", ...	1472865386	1	1472865386
Not Socially Engaged	{"visits": "1", "hits": "1", "pageviews": "1",...	{"campaign": "(not set)", "source": "google", ...	1472881213	1	1472881213
Not Socially Engaged	{"visits": "1", "hits": "1", "pageviews": "1",...	{"campaign": "(not set)", "source": "google", ...	1472822600	2	1472822600

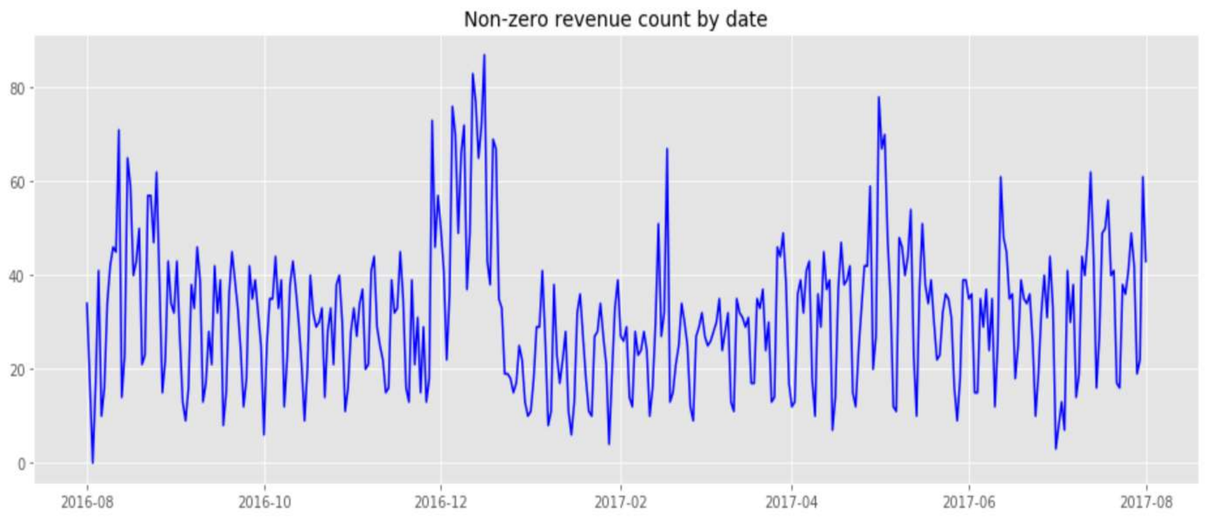
Characteristics of the data:

Exploratory Visualization:

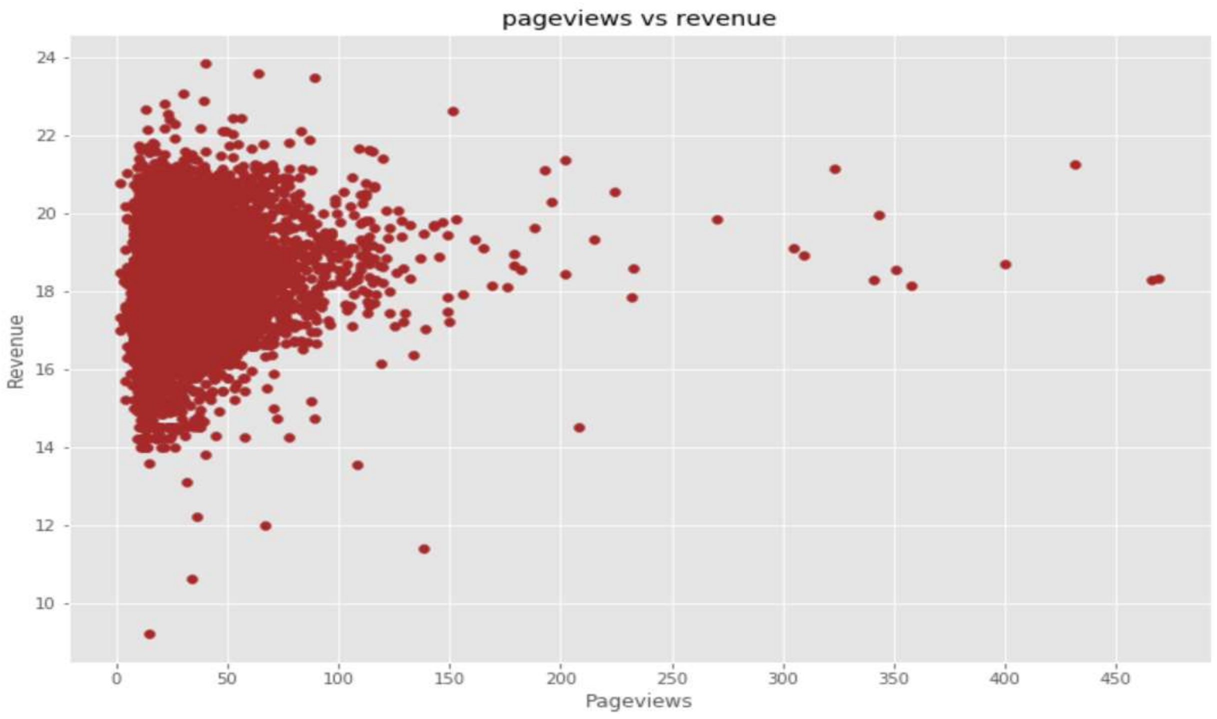


- Number of instances with non-zero revenue in data set is 11515 out of 903653 instances and the ratio is 0.0127427231470487

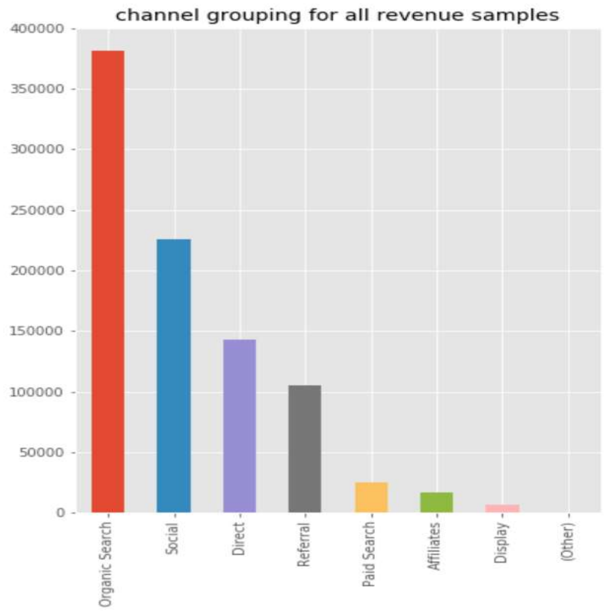


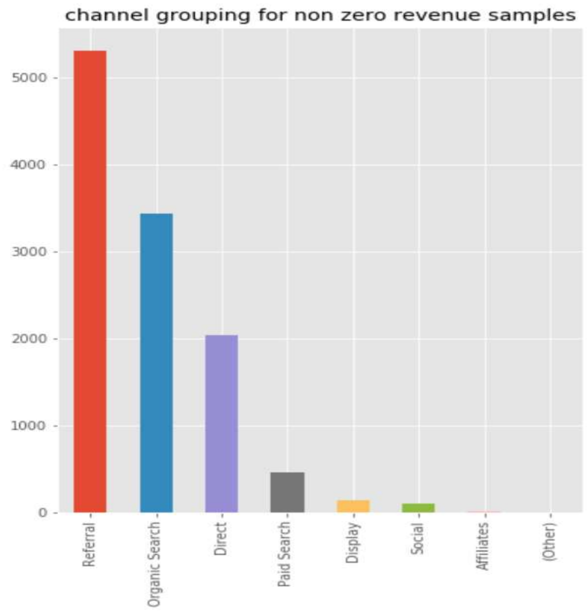


The plots show that there is correlation between number of users and revenue for that day. During october'16, though there is an increase in number of users there is no increase in revenue.

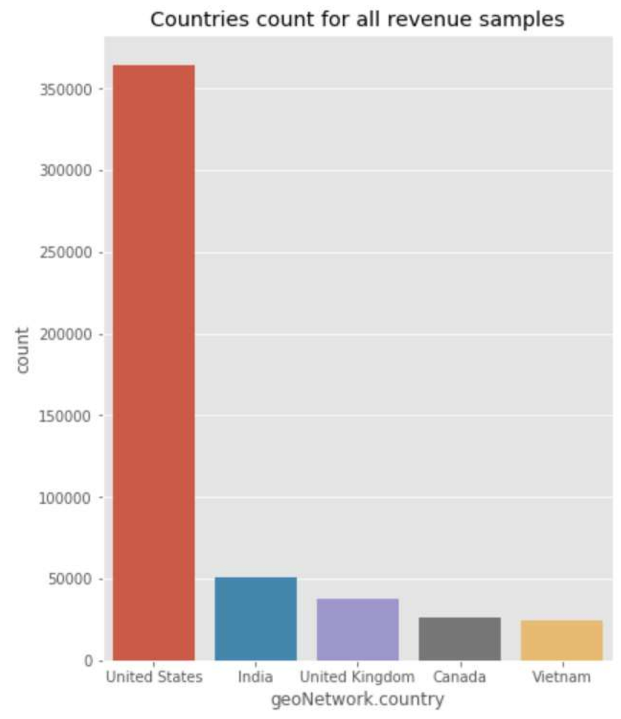


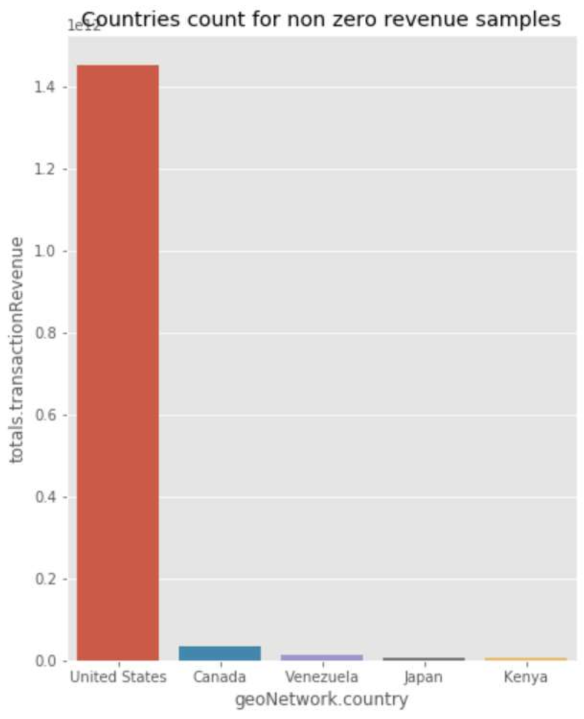
Most of the revenue generating users has page views less than 100. This shows that increase in number of page views doesn't translate to increase in revenue.





1. Organic search, social and Direct are the most common types of channels.
2. Referral is the channel type through which most non-zero transactions occur.





Most of the users are from USA, India and UK. Most of the revenue is generated from users in USA.

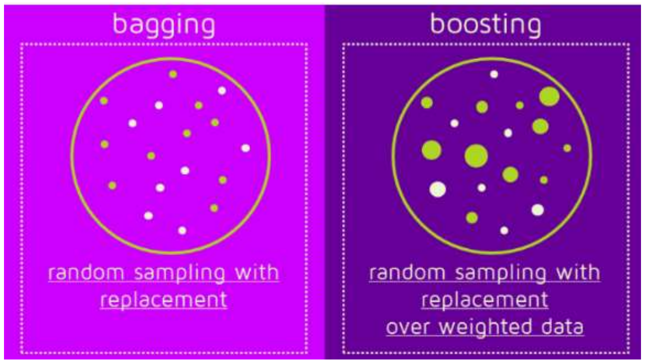
Algorithms:

Algorithms used for model selection are:

- **Linear Regression**: Linear regression is the simplest regression algorithm and I started with it as prediction of revenue is our goal. The goal of a linear regression algorithm is to minimize the cost function defined as:
  Minimize sum of squared errors:
  $$\underset{\beta_0, \beta_1}{\text{minimize}} \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$
  where m is the number of samples in the data. Gradient descent is the algorithm used to minimize the cost function. Modifying the linear regression algorithm by way of generalization, where we penalize the coefficient values of features greatly improves the performance of linear regression algorithm. **In Lasso regression** we add absolute values of coefficients to loss function and in **Ridge regression** squares of coefficients are added to the loss function.

- **Tree-based regression**: Decision trees involve segmenting the predictor space into a number of simple regions by performing a sequence of tests. A decision tree consists of nodes and leaves where each node represents a decision and each leaf is the resultant class variable. The data is split at the node by the attribute which results in most information gain. Tree based learners are robust to outliers and normalization of data is not needed. As our data us highly imbalanced, decision trees are chosen as the next algorithm for model selection. But overfitting can be a problem when there are substantial number of features**.** **Decision trees** and **Extra trees** are the algorithms used for model selection.

- **Ensemble methods**: Ensemble methods build multiple regressors on the same data and combine the output either through mean, median, mode (**Bagging**) or weighted average of weak learners **(Boosting)**.

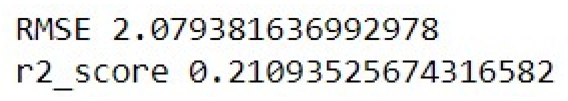

bagging
random sampling with replacement
boosting
random sampling with replacement over weighted data

1. **Random forest**, one of the simplest bagging methods works well with high dimensional data. Random forest build an ensemble of decision trees by bagging. The randomness comes from searching for the most important feature while splitting a node from a random subset of features.
2. **Xgboost and catboost**, boosting methods are used next as they build an additive model which always increases performance and they can be trained on a GPU resulting in faster train times.
   (1) **Extreme Gradient Boosting (xgboost)** is an optimized gradient boosting algorithm that is designed to be highly efficient, flexible and fast. For regression-based models, the weak learners used can be either regression or tree based. Used both tree and regression based weak learners for model selection.
   (2) **Catboost** is a gradient boosting on decision trees algorithm which uses a novel gradient boosting scheme to reduce overfitting. It can be used with non-numeric data and has GPU support. Used label encoded data and also data with categorical features for model selection.

Benchmark:

A linear regression model is chosen for benchmark using unscaled data and dropping constant columns.

Observations:



```
RMSE 2.079381636992978
r2_score 0.21093525674316582
```

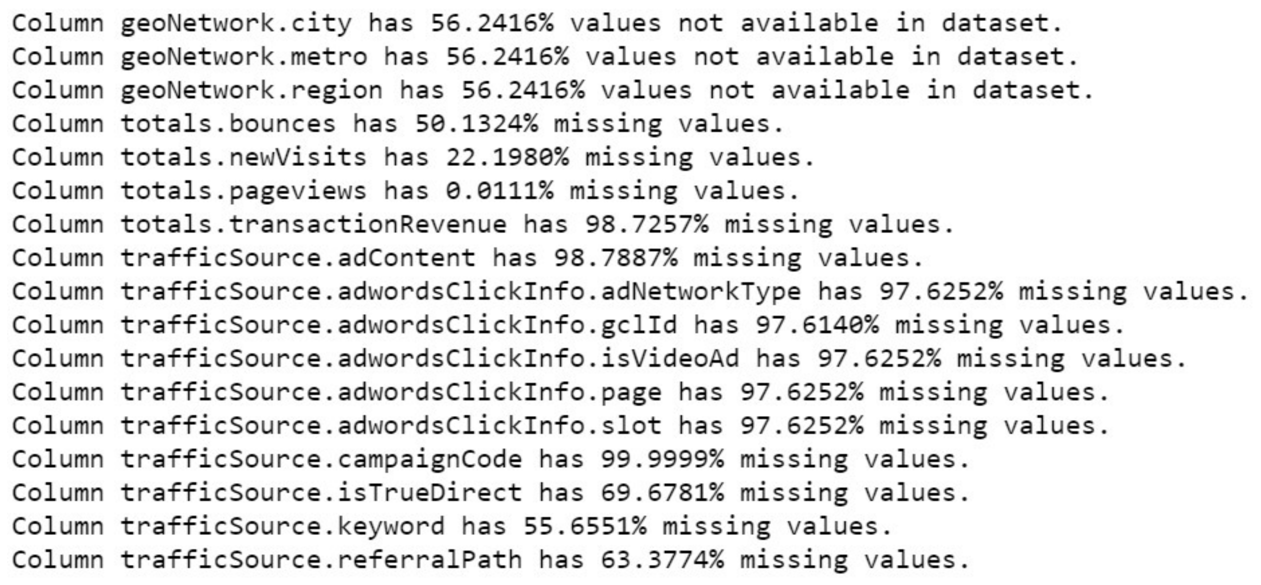
METHODOLOGY:

Data Preprocessing:

Columns which have a constant value are dropped. These columns are:

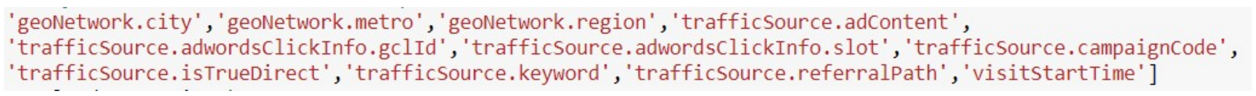
```
'socialEngagementType',
 'device.browserSize',
 'device.browserVersion',
 'device.flashVersion',
 'device.language',
 'device.mobileDeviceBranding',
 'device.mobileDeviceInfo',
 'device.mobileDeviceMarketingName',
 'device.mobileDeviceModel',
 'device.mobileInputSelector',
 'device.operatingSystemVersion',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

There are missing values in several columns. Details are as follows:



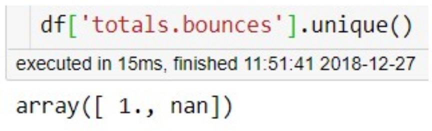
```
Column geoNetwork.city has 56.2416% values not available in dataset.
Column geoNetwork.metro has 56.2416% values not available in dataset.
Column geoNetwork.region has 56.2416% values not available in dataset.
Column totals.bounces has 50.1324% missing values.
Column totals.newVisits has 22.1980% missing values.
Column totals.pageviews has 0.0111% missing values.
Column totals.transactionRevenue has 98.7257% missing values.
Column trafficSource.adContent has 98.7887% missing values.
Column trafficSource.adwordsClickInfo.adNetworkType has 97.6252% missing values.
Column trafficSource.adwordsClickInfo.gclId has 97.6140% missing values.
Column trafficSource.adwordsClickInfo.isVideoAd has 97.6252% missing values.
Column trafficSource.adwordsClickInfo.page has 97.6252% missing values.
Column trafficSource.adwordsClickInfo.slot has 97.6252% missing values.
Column trafficSource.campaignCode has 99.9999% missing values.
Column trafficSource.isTrueDirect has 69.6781% missing values.
Column trafficSource.keyword has 55.6551% missing values.
Column trafficSource.referralPath has 63.3774% missing values.
```

Columns are either dropped or imputed with values based on their characteristics. If a column has a large number of unique values and there is no logical way to replace missing values, it has been dropped. The columns which were dropped are:

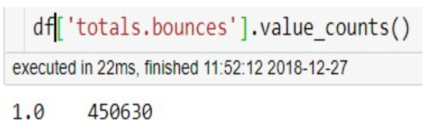


```
'geoNetwork.city','geoNetwork.metro','geoNetwork.region','trafficSource.adContent',
'trafficSource.adwordsClickInfo.gclId','trafficSource.adwordsClickInfo.slot','trafficSource.campaignCode',
'trafficSource.isTrueDirect','trafficSource.keyword','trafficSource.referralPath','visitStartTime']
```

Other columns where missing values can be replaced logically, it has been done so. For example for the column ‘totals.bounces’

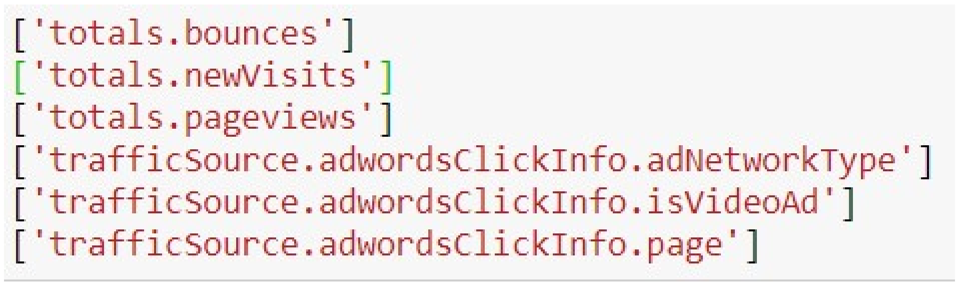


```
df['totals.bounces'].unique()
executed in 15ms, finished 11:51:41 2018-12-27
array([ 1., nan])
```



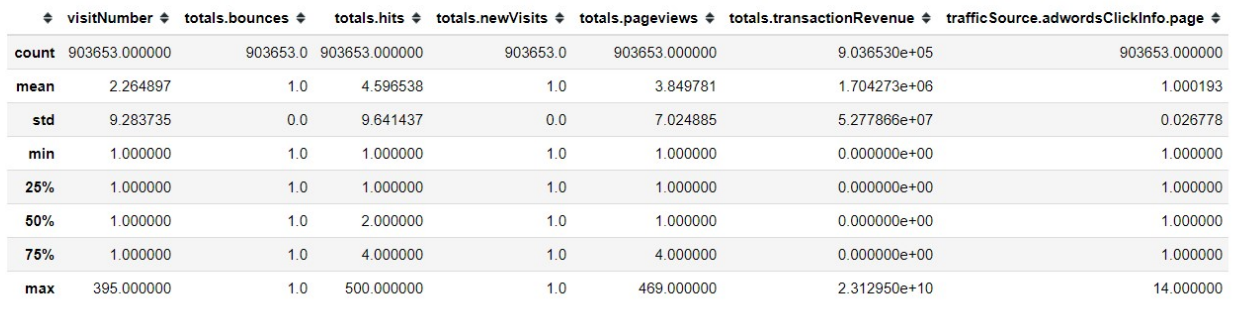
```
df['totals.bounces'].value_counts()
executed in 22ms, finished 11:52:12 2018-12-27
1.0    450630
```

Nearly 50% of the column has missing values but there is only one unique value. So the missing values have been replaced with ‘1.0’. The same has been done for the following columns:



```
['totals.bounces']
['totals.newVisits']
['totals.pageviews']
['trafficSource.adwordsClickInfo.adNetworkType']
['trafficSource.adwordsClickInfo.isVideoAd']
['trafficSource.adwordsClickInfo.page']
```

Summary statistics for the numeric columns:

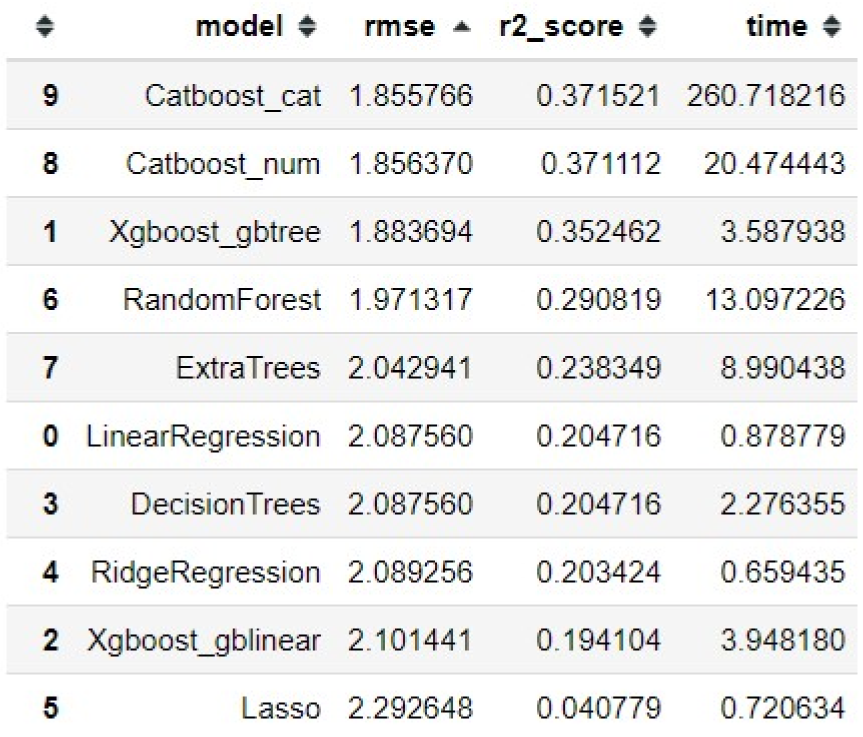


	visitNumber	totals.bounces	totals.hits	totals.newVisits	totals.pageviews	totals.transactionRevenue	trafficSource.adwordsClickInfo.page
count	903653.000000	903653.0	903653.000000	903653.0	903653.000000	9.036530e+05	903653.000000
mean	2.264897	1.0	4.596538	1.0	3.849781	1.704273e+06	1.000193
std	9.283735	0.0	9.641437	0.0	7.024885	5.277866e+07	0.026778
min	1.000000	1.0	1.000000	1.0	1.000000	0.000000e+00	1.000000
25%	1.000000	1.0	1.000000	1.0	1.000000	0.000000e+00	1.000000
50%	1.000000	1.0	2.000000	1.0	1.000000	0.000000e+00	1.000000
75%	1.000000	1.0	4.000000	1.0	4.000000	0.000000e+00	1.000000
max	395.000000	1.0	500.000000	1.0	469.000000	2.312950e+10	14.000000

The data was divided in development and validation sets based on date. The validation set is ~30% of development set. Created 3 datasets to work with different algorithms. First dataset contains data set with categorical features to use with catboost algorithm, second data set contains label encoded data to work with decision trees and ensemble methods and the third data set contains label encoded data with scaled numerical features to work with regression algorithms. Created Dmatrix with Xgboost library as xgboost with linear weak learners does not support the sklearn library.

Implementation:

Model implementation was done with the several algorithms and the corresponding R2, RMSE and computational time were calculated.



	model	rmse	r2_score	time
9	Catboost_cat	1.855766	0.371521	260.718216
8	Catboost_num	1.856370	0.371112	20.474443
1	Xgboost_gbtree	1.883694	0.352462	3.587938
6	RandomForest	1.971317	0.290819	13.097226
7	ExtraTrees	2.042941	0.238349	8.990438
0	LinearRegression	2.087560	0.204716	0.878779
3	DecisionTrees	2.087560	0.204716	2.276355
4	RidgeRegression	2.089256	0.203424	0.659435
2	Xgboost_gblinear	2.101441	0.194104	3.948180
5	Lasso	2.292648	0.040779	0.720634

Catboost with categorical features and catboost with label encoded features have the same accuracy and robustness. Selected catboost with label encoded features as the best model as it is ~13 times faster.

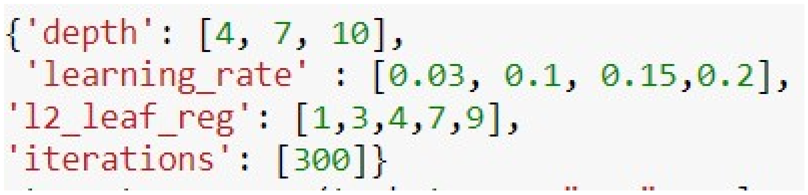
Challenges faced:

- Data cleaning and preparing multiple data sets for use with different algorithms took time.
- Though catboost with categorical features and with label encoded features performed similarly in the model selection phase, catboost with label encoded features is chosen as the best model due to lack of computational power and a powerful GPU.
- Figuring out how xgboost with linear weak learners works was a bit difficult as it does not support the scikit library. Had to use built in functions of the xgboost library after converting the data into Dmatrix.

Refinement:

Catboost with label encoded data is chosen for further parameter tuning as it was the best performing model and best R2 score.

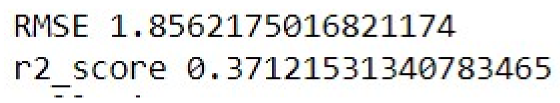
Catboost parameter tuning:



```
{'depth': [4, 7, 10],
 'learning_rate' : [0.03, 0.1, 0.15,0.2],
'l2_leaf_reg': [1,3,4,7,9],
'iterations': [300]}
```

Iterations: maximum number of trees that can be built; high value can lead to overfitting
Depth: Depth of the tree for the weak learner
12_leaf_reg: L2 regularization coefficient which is for leaf value calculation
Learning_rate: used for reducing the gradient step

Results



```
RMSE 1.8562175016821174
r2_score 0.37121531340783465
```

Model	RMSE	R2 score
Baseline model	2.079	0.211
Final Model	1.856	0.371
% Change	12	75

Final RMSE score decreased by 12% and the R2 score increased by 75% making it more accurate and robust than the baseline model. Therefore, this can be viewed as an acceptable solution to the problem statement.

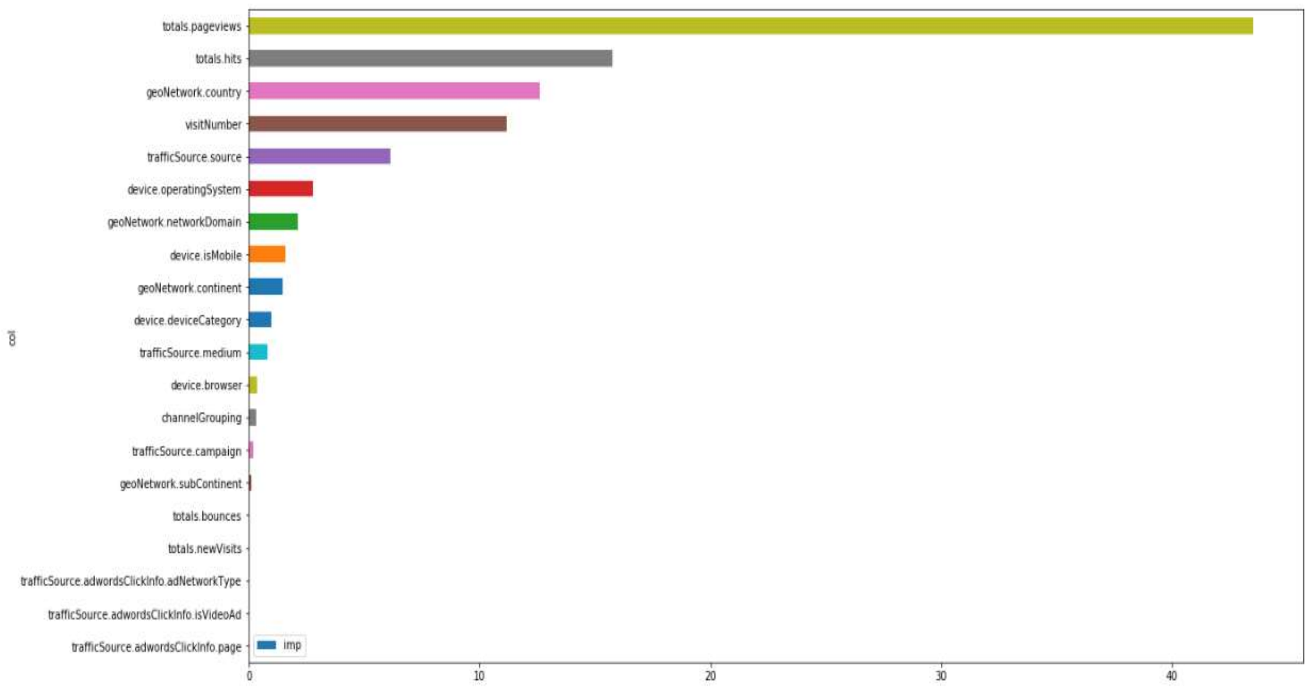
Validation of final model: Calculated the evaluation metrics with different random seed and 3-fold CV to check the robustness of the final model. The results obtained are summarized below:

Model	RMSE	R2 score
3-fold cv	1.856	0.371
Random state 999	1.855	0.371
Random state 111	1.855	0.371

This shows that the final model is robust.

Conclusion

Freeform visualization



Most important feature: totals.pageviews

Top 5 important features: totals.pageviews, totals.hits, geoNetwork.Country, VisitNumber and trafficSource.source are the 5 important features and this is corroborated by the initial EDA.

Reflection: This capstone project can be summarized as a sequence of following steps:

- Searching for a suitable problem through open repositories and Kaggle competitions that is both interesting and solvable with my current skills.
- Performing EDA to better understand the data.
- Creating a benchmark model.
- Preprocessing the data and feature selection.
- Creating different data sets to use with different algorithms.
- Model selection and best model hyper parameter tuning.
- Reporting the evaluation metrics for final model and comparing with the benchmark model.

Challenges:

- The biggest challenge with this project was loading and preprocessing of data. The data contains json values in columns, it took time to figure out how to load the data. Also, as the data had many missing values, cleaning it took a significant amount of time. Had to manually check each incomplete column description to assess the best way to deal with missing values.

- There was no improvement in the final best model before and after hyper parameter tuning. Experimented with other algorithms to fine tune the performance but their performance was no better. This can be attributed to the highly imbalanced data and shows how hard it is model with such data.

Improvement:

- Better feature selection and preprocessing of data. Had to drop a lot of columns because of missing values, domain knowledge can help with this.
- More number of hyper parameters to use in hyper parameter tuning.
- Use ensemble stacking for better performing models.

References:

- https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db
- https://www.kaggle.com/julian3833/1-quick-start-read-csv-and-flatten-json-fields
- https://www.kaggle.com/sudalairajkumar/simple-exploration-baseline-ga-customer-revenue
- https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/