# Machine Learning Project Report on Kaggle's Google Analytics Customer Revenue Prediction

**Group Members:**

| | |
|---|---|
| **Aarya Vardhan Reddy Paakaala** | **: AXP170019** |
| **Vinyas Raju** | **: VXR170005** |
| **Vishwanath D C** | **: VXD180004** |
| **Yash Jain** | **: YXJ180004** |

## Contents

## 1) Introduction and Problem description

- The goal of this project is to perform customer revenue prediction by building a model using machine learning techniques that will assist us to predict how much customer revenue is spent.
- The advantage of customer revenue prediction is that it allows your business to recognize the revenue generated rather than the units sold.
- It is evident from the customer sales record that 20% of the customer accounts for 80% of the revenue the market. Hence it becomes necessary to analyze the user traffic and promote the products accordingly.
- Here, we analyze a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict revenue per customer.
- Hopefully, the outcome will be more actionable operational changes and a better use of marketing budgets for those companies who choose to use data analysis on top of GA data.

## 2) Related work

[1] A Machine Learning Framework for Predicting Purchase by online customers based on Dynamic Pricing

Problem

- Pricing is the primary factor which drives the online purchase and affects the demand and supply of the product. The features of dynamic pricing is very important to online retailers. In this study, the researchers aim to develop a framework and techniques by applying machine learning algorithms to predict the right price purchase by the customers on e-commerce platform.

Method used

- The proposed model consists of following steps for determining the purchase behavior and pricing strategy for the online customers.

- Data Collection: It involves the collection of data from an integrated database.

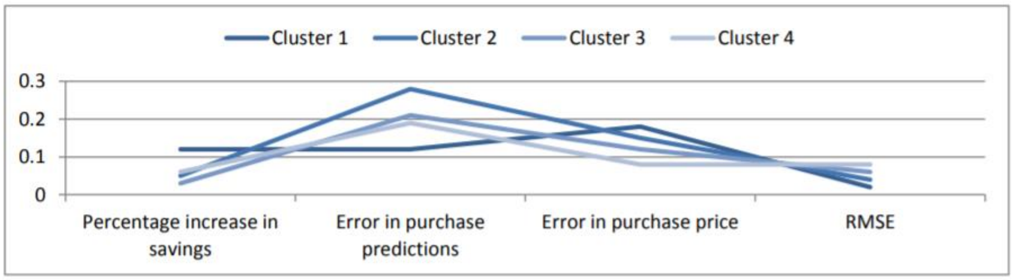
- Pre-Processing: In this step, the collected data is pre-processed and cleaned according to the prediction of the price.

- Attribute Selection: In this step, according to customer segmentation, the attributes are selected. For a new customer, Context, Purchase history, visit attribute, demographic profile, and Purchase intentions should be used as various attributes from the selected data nut in this research, they have only considered repeat customers to find a specific case for this project.

- Customer Grouping: K-means clustering algorithm is used to find the similar users to perform the customer grouping.

- Dynamic Pricing: The dynamic price range is identified for each segment of customers. Statistical and Machine learning techniques could be used to find the appropriate dynamic price range for each segment. Machine learning techniques such as supervised learning technique is preferred.

- Predictive Modeling: In this research, they have used Logistic Regression to predict whether a customer likely to purchase the product or not, given a customer group and dynamic price range.



[2] Revenue forecasting for enterprise products

Problem

- Given historical revenue for K different geographical regions across the world, this research paper aimed to predict the revenue for next quarter for each geographical region separately and worldwide aggregate as well. This could be achieved using machine learning algorithms.

Method used

- Model 1 - Time-Series Model: This model aims at constructing time-series corresponding to revenue data in each quarter for each geographical area and worldwide aggregate. Next, it aims at using functions like auto.arima, ets, stlf to construct ARIMA, ETS and STL models respectively and predict the revenue for next quarter.

- Model 2 - RandomForest Regression Model: This model uses the Random forest regressor algorithm which forecasts from the standard time-series models as mentioned in model-1 and other features such as horizon, geographical-area, and lag-features.

- Model 3 - RandomForest Regression Model with Macroeconomic indicators: It includes the macro-economic indicators such as Total Share Prices for All Shares and Current Price Gross Domestic Product (GDP) for individual geographical areas while using the random forest regressor algorithm.

**3) Dataset Description**

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

The dataset consists of multiple files.

1) Train.csv
2) Test.csv

The train and test sets contains columns which are listed below. Each row in the dataset is one visit to the store. Here we are predicting the log of total revenue (transactionrevenue) per user.

- 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 to1.
- visitStartTime - The timestamp (expressed as POSIX time).

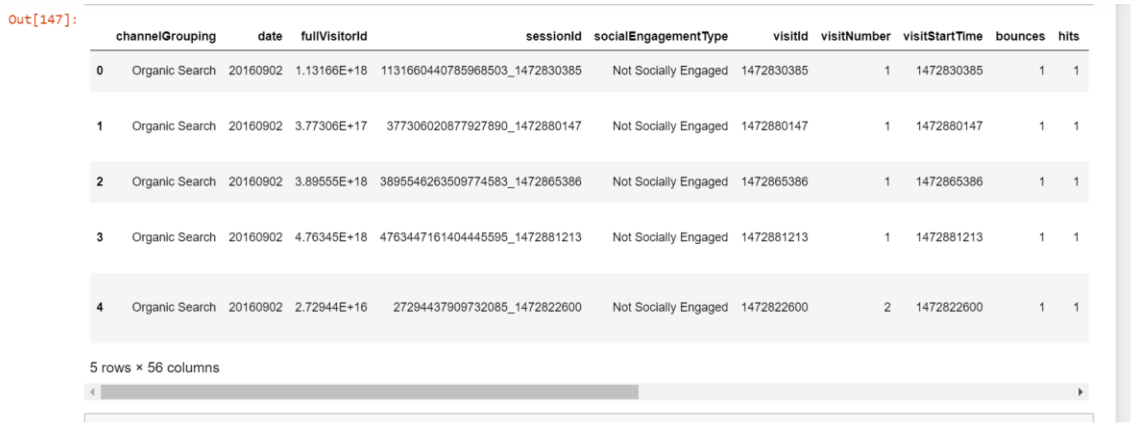
- The given data consists of the details like Visitor_ID, Total_Revenue, Visit_number and Date. Data contains information regarding customers and most of the features are JSON objects.
- Around 2% of data points have transactionrevenue. We are taking the natural log of transactionrevenue which will be our class label as per competition rule.
- The prediction value gives the revenue generated per each customer_ID.

- The Company can double estimate for advertising and marketing costs since they always tend to escalate beyond the expectations. It also helps in keeping track of direct sales and customer service time.
- We are writing the json file into a csv and then trying to convert this to dataframe on which my models can be applied on.

**4) Preprocessing Techniques**

- Read the dataset using Pandas Library and convert it into Pandas data frame.
- To convert the JSON Blobs, we will use pandas to flatten JSON objects to dataframe.
- Remove the attributes which have direct correlation or which are irrelevant.
- Fix the missing data by remove the null values.
- Use Label encoding to convert the categorical data into numerical data.
- Split the data into training and testing sets.

<u>Initial dataset</u>

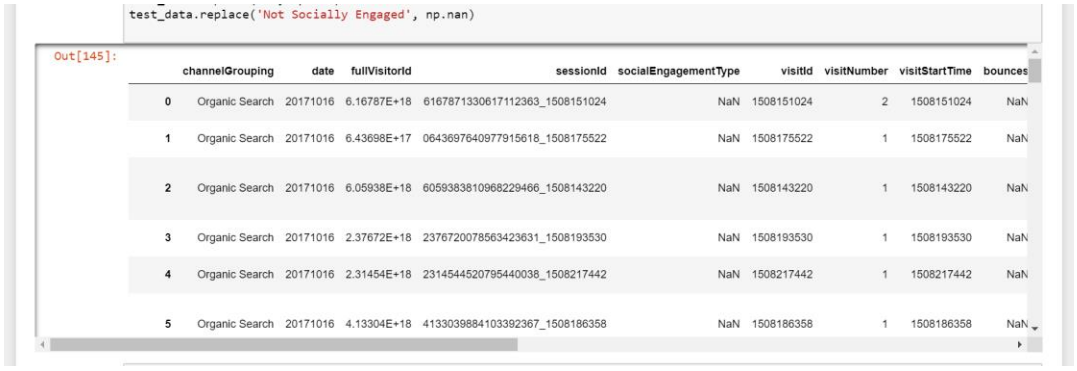


Out[147]:

| | channelGrouping | date | fullVisitorId | sessionId | socialEngagementType | visitId | visitNumber | visitStartTime | bounces | hits |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Organic Search | 20160902 | 1.13166E+18 | 1131660440785968503_1472830385 | Not Socially Engaged | 1472830385 | 1 | 1472830385 | 1 | 1 |
| **1** | Organic Search | 20160902 | 3.77306E+17 | 377306020877927890_1472880147 | Not Socially Engaged | 1472880147 | 1 | 1472880147 | 1 | 1 |
| **2** | Organic Search | 20160902 | 3.89555E+18 | 3895546263509774583_1472865386 | Not Socially Engaged | 1472865386 | 1 | 1472865386 | 1 | 1 |
| **3** | Organic Search | 20160902 | 4.76345E+18 | 4763447161404445595_1472881213 | Not Socially Engaged | 1472881213 | 1 | 1472881213 | 1 | 1 |
| **4** | Organic Search | 20160902 | 2.72944E+16 | 27294437909732085_1472822600 | Not Socially Engaged | 1472822600 | 2 | 1472822600 | 1 | 1 |

5 rows × 56 columns

<u>Replacing "Not Socially Engaged" to null</u>

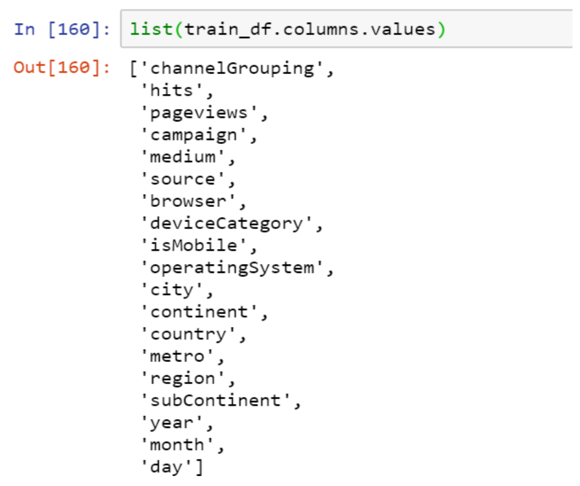


```
test_data.replace('Not Socially Engaged', np.nan)
```

Out[145]:

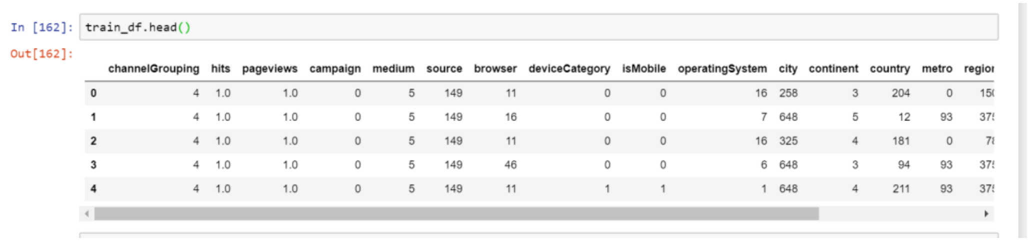
| | channelGrouping | date | fullVisitorId | sessionId | socialEngagementType | visitId | visitNumber | visitStartTime | bounces |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Organic Search | 20171016 | 6.16787E+18 | 6167871330617112363_1508151024 | NaN | 1508151024 | 2 | 1508151024 | NaN |
| **1** | Organic Search | 20171016 | 6.43698E+17 | 0643697640977915618_1508175522 | NaN | 1508175522 | 1 | 1508175522 | NaN |
| **2** | Organic Search | 20171016 | 6.05938E+18 | 6059383810968229466_1508143220 | NaN | 1508143220 | 1 | 1508143220 | NaN |
| **3** | Organic Search | 20171016 | 2.37672E+18 | 2376720078563423631_1508193530 | NaN | 1508193530 | 1 | 1508193530 | NaN |
| **4** | Organic Search | 20171016 | 2.31454E+18 | 2314544520795440038_1508217442 | NaN | 1508217442 | 1 | 1508217442 | NaN |
| **5** | Organic Search | 20171016 | 4.13304E+18 | 4133039884103392367_1508186358 | NaN | 1508186358 | 1 | 1508186358 | NaN |

## Columns after pre-processing



```
In [160]: list(train_df.columns.values)

Out[160]: ['channelGrouping',
            'hits',
            'pageviews',
            'campaign',
            'medium',
            'source',
            'browser',
            'deviceCategory',
            'isMobile',
            'operatingSystem',
            'city',
            'continent',
            'country',
            'metro',
            'region',
            'subContinent',
            'year',
            'month',
            'day']
```

## Dataset after preprocessing

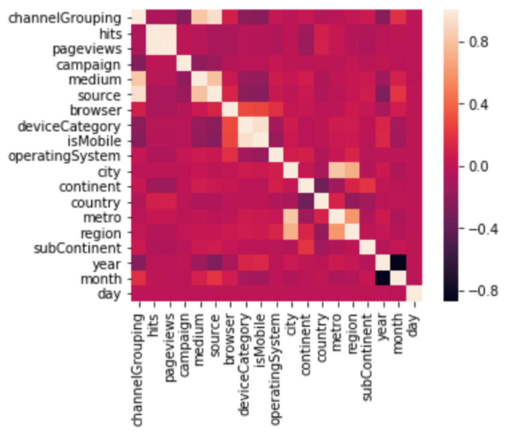


```
In [162]: train_df.head()

Out[162]:
```

| | channelGrouping | hits | pageviews | campaign | medium | source | browser | deviceCategory | isMobile | operatingSystem | city | continent | country | metro | regio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 1.0 | 1.0 | 0 | 5 | 149 | 11 | 0 | 0 | 16 | 258 | 3 | 204 | 0 | 15 |
| 1 | 4 | 1.0 | 1.0 | 0 | 5 | 149 | 16 | 0 | 0 | 7 | 648 | 5 | 12 | 93 | 37 |
| 2 | 4 | 1.0 | 1.0 | 0 | 5 | 149 | 11 | 0 | 0 | 16 | 325 | 4 | 181 | 0 | 7 |
| 3 | 4 | 1.0 | 1.0 | 0 | 5 | 149 | 46 | 0 | 0 | 6 | 648 | 3 | 94 | 93 | 37 |
| 4 | 4 | 1.0 | 1.0 | 0 | 5 | 149 | 11 | 1 | 1 | 1 | 648 | 4 | 211 | 93 | 37 |

## Correlation between the attributes using the Heat map

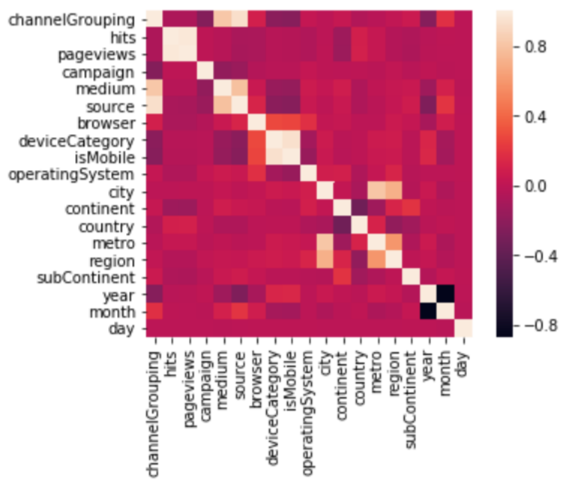
```
#print cor
sns.heatmap(cor, square = True)

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae3e9265c0>
```



Train data

```
sns.heatmap(corelation_train, square = True)
```

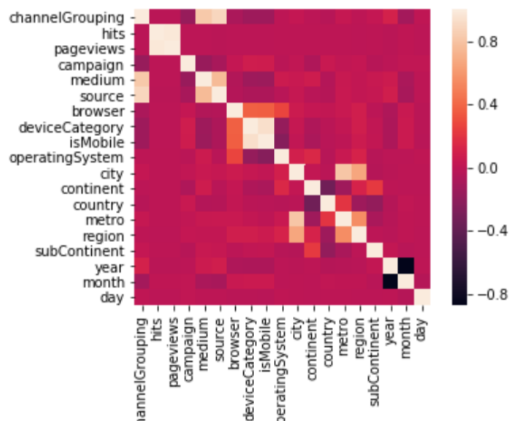
```
6]: <matplotlib.axes._subplots.AxesSubplot at 0x1b0908a1dd8>
```



Test data

```
In [167]: #heatmap to show correlation
          corelation_test = test_df.corr()
          sns.heatmap(corelation_test, square = True)
```

```
Out[167]: <matplotlib.axes._subplots.AxesSubplot at 0x1af98102ba8>
```

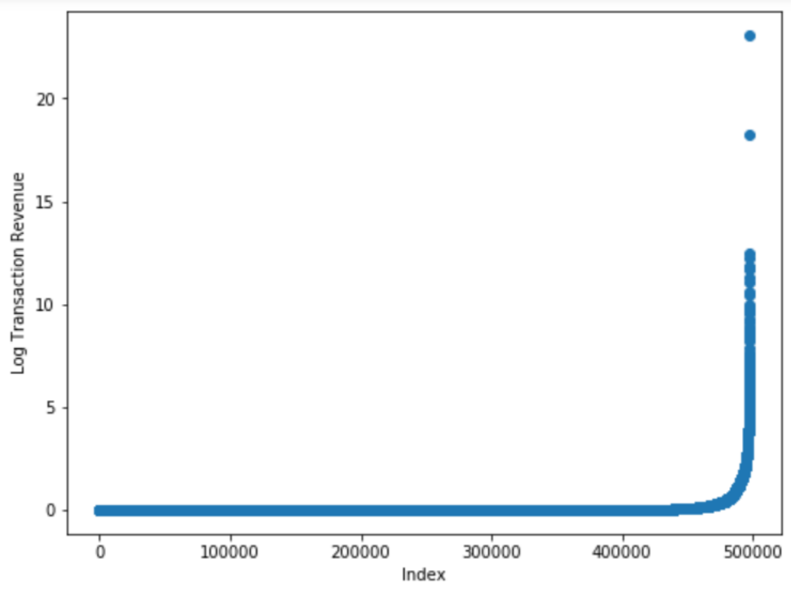


## 5) Proposed solution and methods

- We have used the packages from sklearn.metrics library for predicting the log value of the revenue.
- Some of the regression techniques like XGBoost, Random Forest and LightGBM are being considered for the experimental analysis.

### 5.1) XGBoost

- XGBoost is an optimized distributed library used for Gradient Boosting.
- It implements machine learning algorithms under gradient boosting framework.
- It provides distributed computing solution for faster training or larger datasets.
- Boosting is built on the principle that a collection of 'weak learners' can be combined to produce a 'strong learner'.
- The scoring of the competition is based on the calculation of the total revenue per user. For this, the evaluation parameter we have used is 'rmse' (Root Mean Squared Error). The training dataset is split into 20% test data and the model makes prediction using XGBRegressor.
- XGBRegressor is a Wrapper likeScikit-Learn for XGBoost, they prepare the Dmatrix and pass in the corresponding objective function and parameters. It uses gbtree as the booster by default. The parameters tuned are: eval_metric = 'rmse', verbose = '130'.
- The Model plots the Visitor vs TransactionRevenue graph where the transaction revenue is based on the predicted log revenue calculated using 'rmse' evaluation of XGBRegressor.



- Finally, a submission file is generated that contains the Predicted LogRevenue for fullVisitorId.

5.2) Random Forest

- Random forests are an ensemble learning method for classification, and regression which use a groups of decision trees to train the data.
- Random forests are trained with the bagging method.
- Random forest model can provide ranking scores for features. Higher the ranking score, the more relevant the feature is.
- Average value of out-of-bag error could be obtained to get the ranking score.
- The features among the training data are permuted and a second out-of-bag error needs to be obtained.
- Ranking score is proportional to the difference between the two out-of-bags error.
- We have used a random forest regressor. Random forest regressor algorithm is a machine learning algorithm which fits multiple decision trees on different sub-parts of the dataset.
- The random forest regressor then finds the mean of the results which improves accuracy by reducing overfitting.
- The two important parameters that we tuned in our model are n_estimators and max_depth. The number of trees in the forest set to above 70 which gives good predictions.
- The evaluation metric we are using is the Root Mean Squared Error (RMSE), calculated as follows:
- As the value of RMSE decreases, the prediction becomes more accurate.

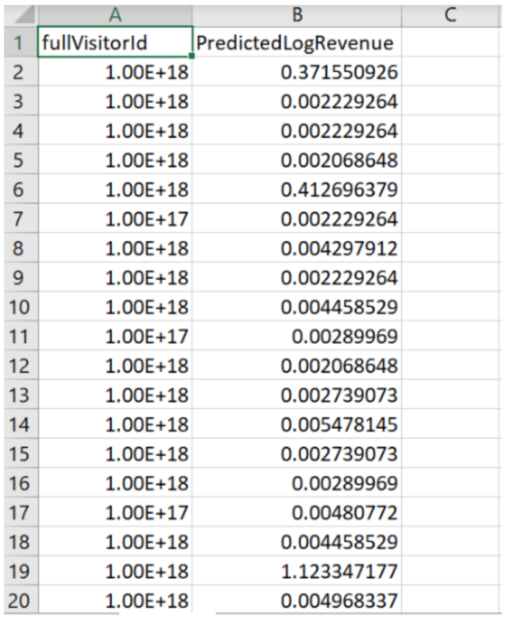
$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2$$

5.3) Light GBM

- Light GBM is gradient boosting framework which uses tree based learning algorithm.
- It grows vertically i.e. it grows leaf-wise contradict to other algorithms which grow level wise.
- It usually over fits small data and preferred for large datasets.

## 6) Experimental results and analysis

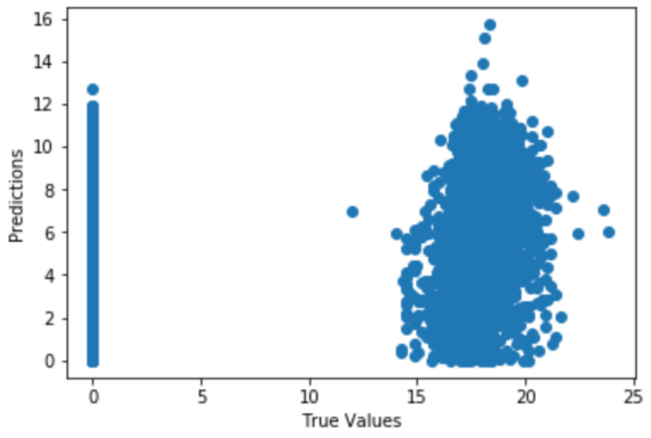
Results depicting the fullVisitorId and their corresponding PredictedLogRevenue values



| | A | B | C |
|---|---|---|---|
| 1 | fullVisitorId | PredictedLogRevenue | |
| 2 | 1.00E+18 | 0.371550926 | |
| 3 | 1.00E+18 | 0.002229264 | |
| 4 | 1.00E+18 | 0.002229264 | |
| 5 | 1.00E+18 | 0.002068648 | |
| 6 | 1.00E+18 | 0.412696379 | |
| 7 | 1.00E+17 | 0.002229264 | |
| 8 | 1.00E+18 | 0.004297912 | |
| 9 | 1.00E+18 | 0.002229264 | |
| 10 | 1.00E+18 | 0.004458529 | |
| 11 | 1.00E+17 | 0.00289969 | |
| 12 | 1.00E+18 | 0.002068648 | |
| 13 | 1.00E+18 | 0.002739073 | |
| 14 | 1.00E+18 | 0.005478145 | |
| 15 | 1.00E+18 | 0.002739073 | |
| 16 | 1.00E+18 | 0.00289969 | |
| 17 | 1.00E+17 | 0.00480772 | |
| 18 | 1.00E+18 | 0.004458529 | |
| 19 | 1.00E+18 | 1.123347177 | |
| 20 | 1.00E+18 | 0.004968337 | |

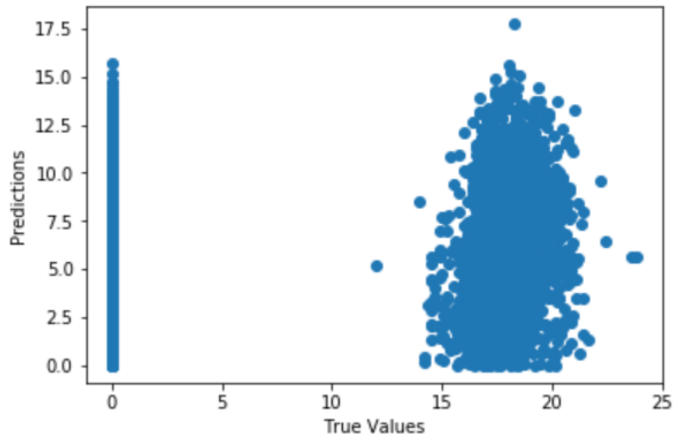
Scatterplot of XGBoost

```
In [174]: plt.scatter(Y_test, y_pred_XGB)
          plt.xlabel('True Values')
          plt.ylabel('Predictions')

Out[174]: Text(0,0.5,'Predictions')
```



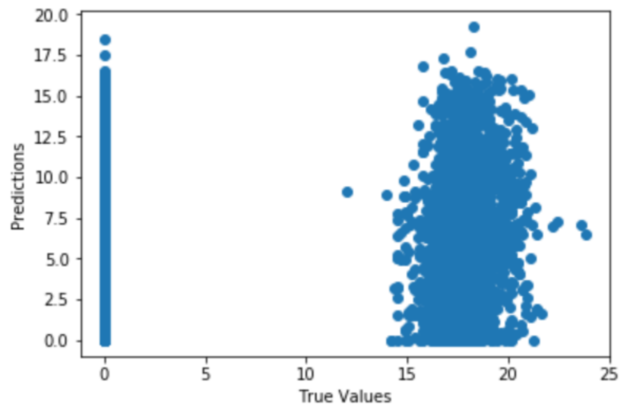
Scatterplot of LGBoost

```
In [175]: plt.scatter(Y_test, y_pred_LGB)
          plt.xlabel('True Values')
          plt.ylabel('Predictions')

Out[175]: Text(0,0.5,'Predictions')
```

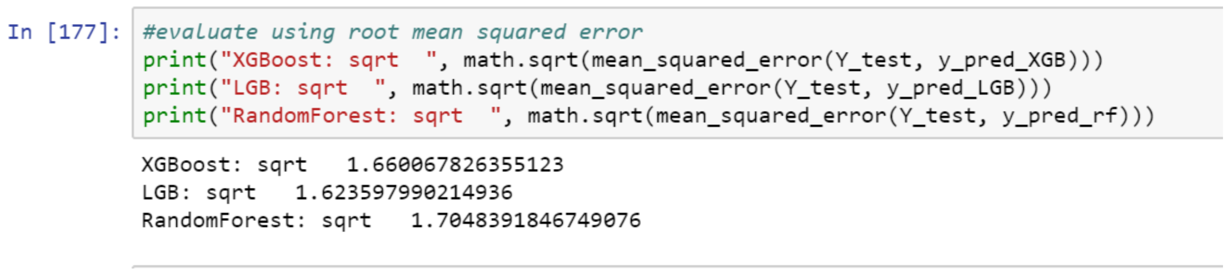


Scatterplot of Random Forest

```
In [176]: plt.scatter(Y_test, y_pred_rf )
          plt.xlabel('True Values')
          plt.ylabel('Predictions')

Out[176]: Text(0,0.5,'Predictions')
```



Evaluation Metrics for each model



```
In [177]: #evaluate using root mean squared error
          print("XGBoost: sqrt  ", math.sqrt(mean_squared_error(Y_test, y_pred_XGB)))
          print("LGB: sqrt  ", math.sqrt(mean_squared_error(Y_test, y_pred_LGB)))
          print("RandomForest: sqrt  ", math.sqrt(mean_squared_error(Y_test, y_pred_rf)))

          XGBoost: sqrt   1.660067826355123
          LGB: sqrt   1.623597990214936
          RandomForest: sqrt   1.7048391846749076
```

| Model Name | RMSE (Evaluation Metric) |
|---|---|
| XGBoost | 1.6606 |
| LGB | 1.6235 |
| Random Forest | 1.7048 |

## 7)Conclusion

- The Google customer revenue prediction dataset has been pre-processed and sound machine learning algorithms have been performed to predict the Revenue values of the visitors.
- LGB model performed the best among the 3 models used to perform prediction. It has an RMSE value of 1.62 while XGBoost and Random Forest have RMSE values of 1.66 and 1.70 respectively.
- The methods used to analyze data and perform statistical methods and machine learning methods and the results generated would be very useful to the business companies while performing customer revenue analysis.

## 8)Contribution of team members

- Pre-processing was performed together.
- Aarya and Vinyas performed Random Forest. Vishwanath performed XGBoost.
- Yash performed LGB model. All of us worked out trying different parameter tuning, to reduce the error and eventually combined our results together.

## 9)References

[1] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8017582
[2] https://pdfs.semanticscholar.org/ab2a/5df5c6db9c8fc55117ff71c44f27ba82a087.pdf
[3] https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
[4] https://arxiv.org/pdf/1603.02754.pdf
[5] http://dmlc.cs.washington.edu/xgboost.html
[6] https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
[7] http://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf