SAN DIEGO STATE UNIVERSITY

Google Store Customer Revenue Predictions

Project Report

MIS 620 Electronic Business and Big Data

From:

Amit Gupte

Alessandro Vomero

Patrick Brault

Swathika Ramesh Chandran

Outline

Executive Summary

Retail outlets are constantly searching for the best way to forecast sales so that they can optimize the amount of budget they allocate for particular business activities. The Google Store, an online retailer of Google merchandise, has provided their sales data in a Kaggle data science competition, incentivizing our team to identify the best method of forecasting future sales.

Goals

- Forecast target revenues by "Customer ID"
- Find if there is a relationship between customer revenue and method of arriving to the Google Store (i.e. direct visit, click from advertisement, etc.)
- Discover the statistical model that best predicts future revenue (lowest RMSE)

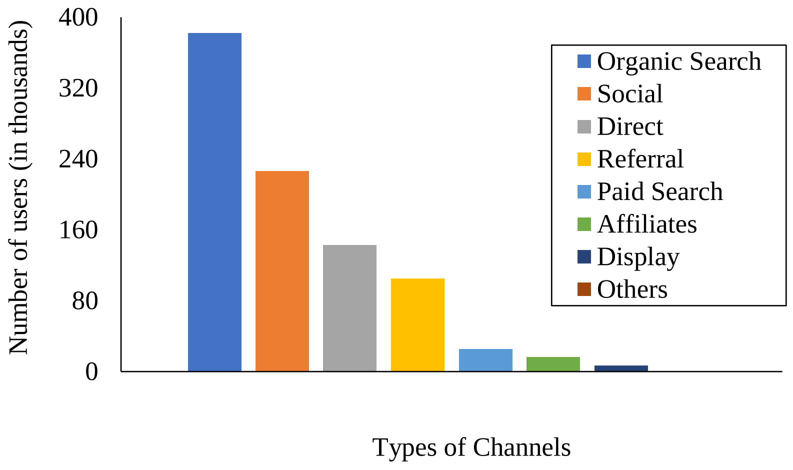


Figure 1. Barplot depicting the channels used by customers to reach the store. Helps in identifying where advertising dollars should be spent.

The provided dataset came in a JSON file which required a significant amount of cleaning before being analyzed. This included cleaning up "NA" values, handling revenue values of "0" for visitors who did not make a purchase, and engineering additional features based on aggregated data from other variables. Once the dataset was clean and normalized our group picked a handful of models that work well with predicting our quantitative dependent variable, *transactionRevenue*. The data was split 70/30 into a training set and test set; the training set was used to build our model and assist in deciding how to tune our parameters. Finally, the models were ready to be ran against our test set. Below is a graph depicting the RMSE results of our tested models:

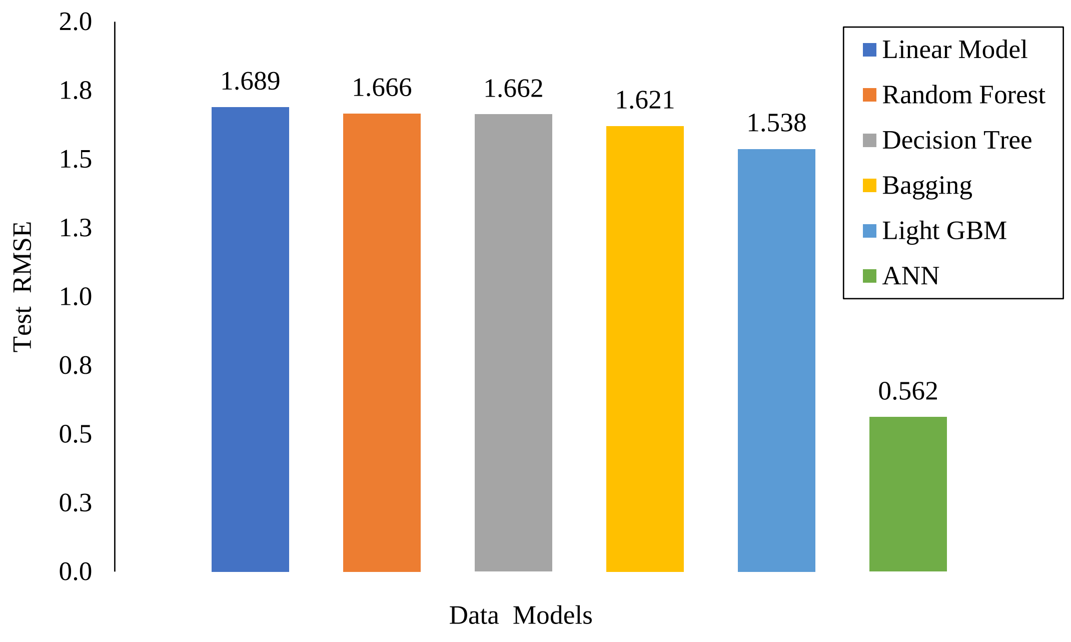


Figure 2. Column chart depicting performance of the tested models. Artificial neural networks performed best with an RMSE of 0.562.

This project gives an in-depth analysis into which customers are projected to generate the most revenue. The Google Store can use this information and match it against the key variables that predict total sales. GStore executives should be able to use the information provided to create their annual budgets and target specific market segments for advertising campaigns.

Discovery

The Google Merchandise store (also known as the GStore) is an online retail outlet selling Google swag, it opened on March 11, 2015. Like most retail companies, its marketing team has been challenged with making appropriate investment and promotional decisions. Google believes in the 80/20 rule where 80% of revenues are generated by 20% of customers; therefore, they want to identify the customers who will most likely generate the bulk of their revenue and target their marketing approaches towards them. This will help key stakeholders like upper management allocate a specific budget for marketing efforts as well as assist the marketing team in determining the funding mix that will optimize investment and advertising strategies.

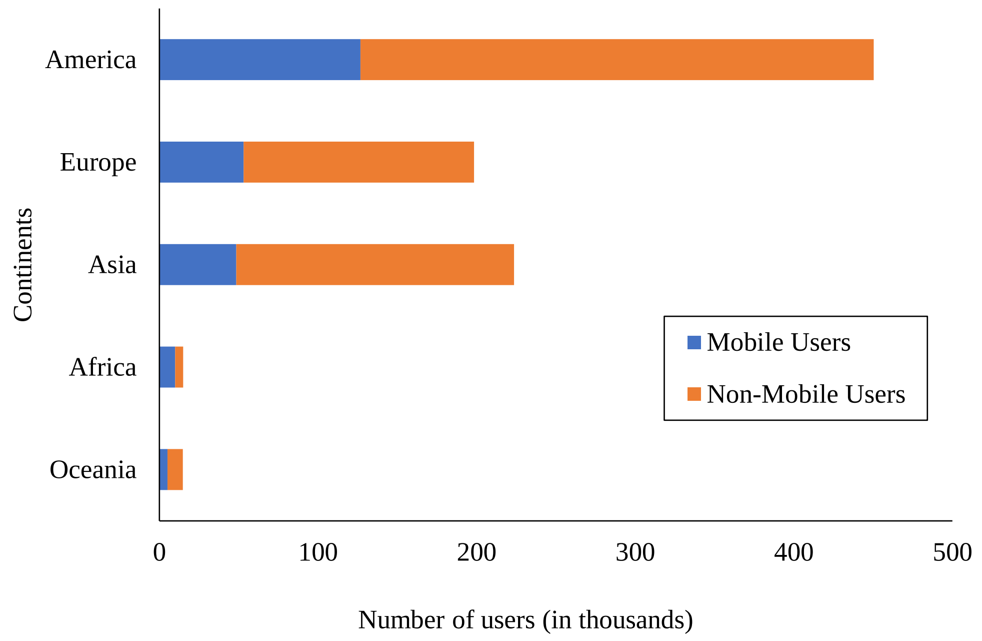
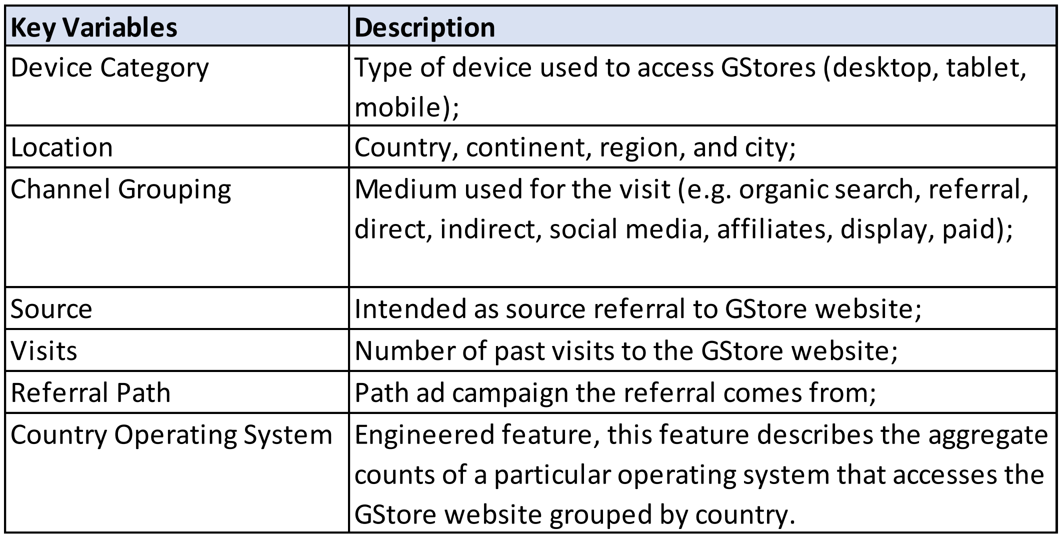


Figure 3. Bar plot representation of total users from different continents, separated by mobile and non-mobile users. The continent, "Oceania", is a geographical region comprised of Australia, New Zealand, New Guinea, and more than 1000 islands scattered over the central and southern Pacific Ocean.

Objective

The main objective for our team is to build a model that minimizes root mean squared error (RMSE) in making customer revenue predictions. To do this, ensured the data was complete and free of errors. We identified the key variables while removing the variables that were too highly correlated with others. We also used some feature engineering techniques to generate columns that were more beneficial to the analysis than the hard data provided. Finally, we determined which statistical model worked best with the data.



Key Variables	Description
Device Category	Type of device used to access GStores (desktop, tablet, mobile);
Location	Country, continent, region, and city;
Channel Grouping	Medium used for the visit (e.g. organic search, referral, direct, indirect, social media, affiliates, display, paid);
Source	Intended as source referral to GStore website;
Visits	Number of past visits to the GStore website;
Referral Path	Path ad campaign the referral comes from;
Country Operating System	Engineered feature, this feature describes the aggregate counts of a particular operating system that accesses the GStore website grouped by country.

Figure 4. Table identifying the key variables with a short description.

Hypotheses

H_o: There will be no significant prediction of future revenue based on the key variables

H_a: There will be a significant prediction of future revenue based on the key variables

H_o: Method of which customer arrived to GStore <u>does not affect</u> revenue produced (statistically significant at $p > 0.05$)

H_a: Method of which customer arrived to GStore <u>affects</u> revenue produced (statistically significant at $p < 0.05$)

Resources

The business domain for retail business such as the GStore is relatively intuitive and does not require deep knowledge of the industry to develop heuristic solutions. Our team of data scientist possesses sufficient knowledge of this business environment to produce a valuable analysis of the revenue data.

Our team is using R-Studio for cleaning and preparing the data, building models and generating analysis, an creating visualizations. MS Word and PowerPoint are also being used to communicate and present our results. In the future, we would like to automate the analysis process and produce a tableau dashboard that can quickly generate real time information so that our client can continuously update their strategies.

Data Preparation

Data Overview

Google Store has made its customer data available online as part of a Kaggle competition for coders and analysts. The dataset provided contains information from August 1, 2016 and April 30, 2018 and consists of over 1.7 million observations (customer visits to the GStore) and 10 variables in a JSON format which contained sub-columns and required flattening for data analysis purposes. The library package *"jsonlight"* was used to transform the JSON blobs into 44 columns that were much easier to read and understand.

Data Conditioning

Before any cleaning or transformation was performed, the original data was preserved as an RDS file, "train_data_flattened.rds", to compare against or look for hidden patterns that may have existed. It also aided fast data loading in memory and avoided parsing problems such as inconsistent delimiters, spurious characters, and format issues. The column types are defined explicitly for each data field and categorized as characters, variables, date and time based on the nature of the data in each column.

After examining the dataset and performing various descriptive statistics, we identified a number of columns containing values such as “not available in the demo dataset”, “not provided”, “unknown”, “not set”. These values were converted to “NA” to maintain uniformity and so they could be categorized as one factor in predicting. It was decided “NA” values would not be imputed in the independent variable columns because it would cause too much bias in our models, resulting in a models that build themselves around error rather than actual variance. However, in our dependent variable column, *transactionRevenue,* “NA” represented “no purchase”, so we imputed a value of “0”. Successively, we added 1 to each of the transaction revenue records and calculated the natural logarithm since the “0” value would have returned “undefined” as the value. This was necessary for scaling transaction revenue.

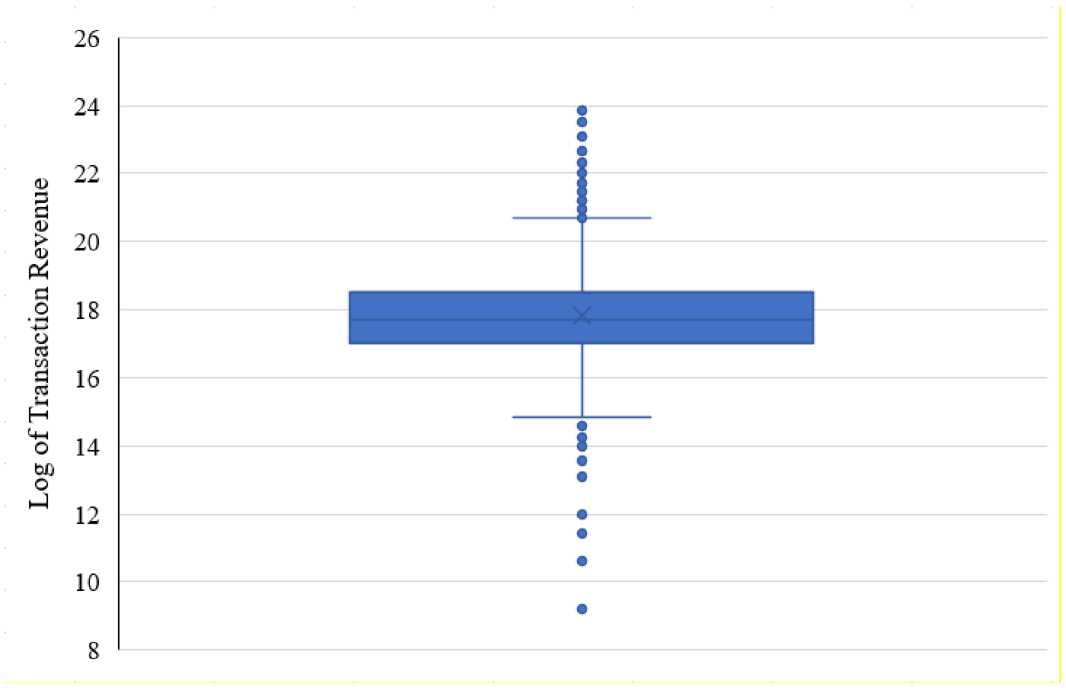


Figure 5. The boxplot shows the spread of the generated transaction revenue. The natural log of the transaction revenue is calculated and plotted above. The mean value is 17.8, whereas the minimum and maximum values are 9.2 and 23.9 respectively.

We set our correlation threshold to 0.9 and dropped any variable that had a higher correlation value when paired with another feature, however, there were very few variables that fell under this criterion. All categorical variables were converted to factors and dummy coded into numeric values. Features were engineered based on aggregation counts of number of users that are active on day of the week and hour of the day. Also, standard aggregate features of sum, max, min, and mean were calculated on *page visits* and grouped on *network domain* and *referral path* per the recommendation of a Kaggle domain expert. In total, we were able to create eleven additional features while removing eight. By combining such engineered features and adding them to the existing input variables, we were able to fine-tune the accuracy of the predictor variable. Columns that contained identifier information, namely *fullVisitorId* and *visitId*, were not selected as an input variable in prediction because they do not have an actual effect on the response.

All variables that have non-zero variance across the dataset were removed as they wouldn't contribute to model effectively. Finally, the data was converted to a data matrix before being passed to the data models for partitioning, training, and testing.

Data Partitioning

We used the "CreateDataPartition" function provided by the "caret" package to split our data into a training and test set. Though the precision decreases with a smaller test set, the accuracy of the model is greater with a larger training set; 70% was allocated to the training set and the remaining 30% was allocated to the test set.

Data Visualization

After the conditioning stage, visualizations were leveraged to view high-level patterns within the data and understand their characteristics. This helped determine the quality of our data and identify whether data contains any unexpected values or other indicators of dirty data.

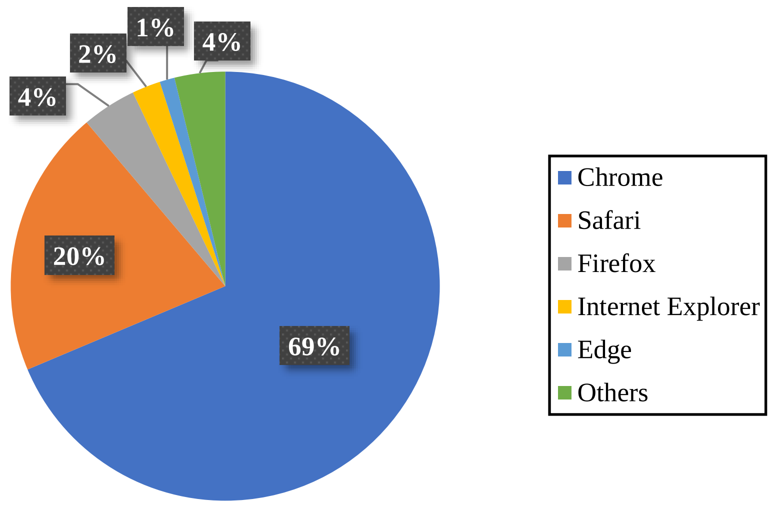


Figure 6. Pie chart representation of the percentage of usage per browser. From the figure, it is evident that Chrome is the most used browser by the customers and Safari is the second popular browser. Browser under the category of Others include: Opera, Reddit, YE etc.

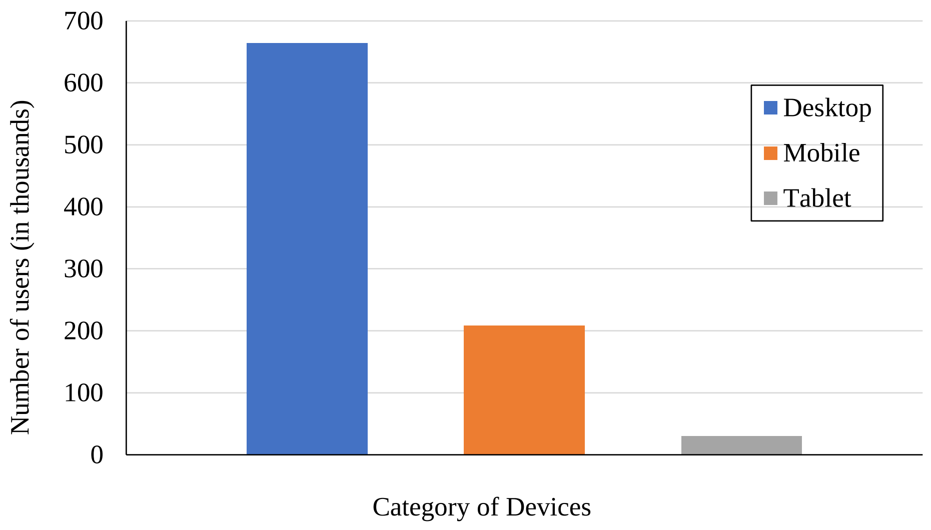


Figure 7. Barplot showing the various category of devices used by the customers to access the Gstore. About 73% of the customers use Desktop, 23% of the customers use Mobile and the rest use Tablet as their mode of accessing the store.

Model Planning

After completing the preparation phase, our team explored the data further to discover to determine which models would work best with our data. Below, we describe the methods and techniques we used to prepare for the modeling phase.

- Approximately 600,000 out of the 1.1 Million in the training set, had zero revenue. It was necessary to make sure that the folds of our cross validation had a balance of zero and non-zero values in the dependent variable. We created folds using the "createFolds" function in caret;
- We are in a regression setting and using the following metrics for our models:
    1. RMSE - Root Mean Square Error;
    2. MAE - Mean Absolute Error.
- The dependent variable was provided as part of the original dataset, therefore, our team took a supervised learning approach. Considering the type of predictors involved, the size of the training set, and the quantitative nature of our dependent variable, our team chose to use the following models:
    1. Linear Regression
    2. Decision Trees
    3. Random Forests
    4. Bagging
    5. Gradient Boosting - LightGBM
    6. Neural Networks – Keras

- Using the "Caret" package, we planned to conduct a 5 fold cross validation for the first four statistical learning models. Gradient boosting would be implemented using Light GBM and the Neural Networks model would be built with Keras.

Model Building

We built and executed the models listed in the "Model Planning" phase and evaluated their performance based on RMSE; the details of each are outlined below. Artificial neural networks with a batch size of 512 performed the best on our training set with a low RMSE of *0.3095.*

Linear Regression

We used linear regression as a baseline model for interpretability. The linear regression is a less flexible model that tries to fit the relationship between the input feature variables to the continuous outcome variable. Additionally, its key assumption is that the relationship is linear. Following the summary of the Linear Model:

Table 1

Linear Regression Coefficients – Gstore Data

Predictors	Estimate	Std. Error	T value	P value	Significance
channelGrouping	-1.11E-03	2.53E-03	-0.437	0.661755	
visitNumber	-1.60E-03	1.91E-04	-8.382	2.00E-16	***
browser	-3.80E-03	2.05E-03	-1.85	0.064271	.
operatingSystem	-1.23E-02	1.66E-03	-7.42	1.17E-13	***
isMobile	-9.93E-02	1.27E-02	-7.848	4.24E-15	***
deviceCategory	-1.90E-02	9.04E-03	-2.101	0.035671	*
continent	-3.05E-02	2.14E-03	-14.222	2.00E-16	***
subContinent	1.12E-03	3.62E-04	3.101	0.001926	**
country	4.20E-03	9.14E-04	4.593	4.37E-06	***
region	1.90E-04	1.59E-05	11.929	2.00E-16	***
metro	-2.09E-03	9.09E-05	-22.983	2.00E-16	***
city	1.09E-03	2.99E-04	3.655	0.000258	***
networkDomain	-1.27E-05	9.38E-06	-1.352	0.176492	

source	9.21E-05	3.34E-05	2.757	0.005829	**
medium	1.73E-02	2.02E-03	8.574	2.00E-16	***
referralPath	-5.08E-05	4.34E-06	-11.706	2.00E-16	***
isTrueDirect	7.15E-02	6.17E-03	11.594	2.00E-16	***
pageviews	1.12E-01	2.70E-04	413.947	2.00E-16	***
bounces	3.08E-01	3.51E-03	87.774	2.00E-16	***
sessionQualityDim	1.39E-03	4.41E-05	31.614	2.00E-16	***
browser_dev	7.97E-03	2.42E-03	3.294	0.000989	***
browser_os	1.15E-03	5.32E-04	2.159	0.030841	*
browser_chan	-3.48E-03	7.49E-04	-4.641	3.47E-06	***
chan_os	1.83E-03	4.23E-04	4.332	1.48E-05	***
country_medium	-2.18E-04	1.39E-05	-15.682	2.00E-16	***
country_source	2.85E-04	3.67E-05	7.76	8.51E-15	***
year	5.84E-02	3.13E-03	18.69	2.00E-16	***
wday	1.22E-03	8.31E-04	1.461	0.143886	
hour	-1.81E-04	3.12E-04	-0.579	0.562611	
city_browser	-3.83E-04	6.96E-05	-5.504	3.72E-08	***
city_source	1.01E-04	3.34E-05	3.033	0.002419	**
country_browser	-8.54E-04	1.01E-04	-8.498	2.00E-16	***
country_operatingSystem	1.52E-04	3.78E-05	4.03	5.58E-05	***
networkDomain_browser	8.38E-06	5.90E-06	1.419	0.155761	
wday_user_cnt	2.23E-07	5.34E-08	4.181	2.90E-05	***
hour_user_cnt	1.96E-06	1.65E-07	11.87	2.00E-16	***
mean_referralPath	-4.84E-03	3.28E-03	-1.477	0.139543	
min_referralPath	4.96E-02	6.56E-03	7.57	3.74E-14	***
max_referralPath	1.12E-03	4.47E-05	24.97	2.00E-16	***
sum_referralPath	-8.53E-08	4.86E-09	-17.567	2.00E-16	***
mean_visitNumber	-8.08E-02	2.76E-03	-29.286	2.00E-16	***
max_visitNumber	8.18E-05	3.20E-05	2.556	0.010581	*
sum_visitNumber	-7.04E-08	2.15E-09	-32.756	2.00E-16	***

Table 2

Linear Regression model results for 5-fold cross validation

RMSE	Rsquared	MAE	Resample
1.698767	0.1575123	0.4646154	Fold 1
1.691235	0.1633996	0.4667376	Fold 2
1.706207	0.1700295	0.4657585	Fold 3
1.672016	0.1527044	0.4627322	Fold 4
1.679456	0.1582779	0.4638779	Fold 5

Decision Tree

Also known as prediction trees, is a less flexible model that splits the data on the variable that will give the most information gain toward the dependent variable, in other words, the maximum decrease to RMSE (our metric). The decision tree is a greedy algorithm as it always maximizes information gain at each stage without considering future information gain profits. A pruned Decision Tree shows *Pageviews* and *country_operatingSystem* are the primary splits in predicting natural log of (*transactional_revenue* + 1), see Figure 8.

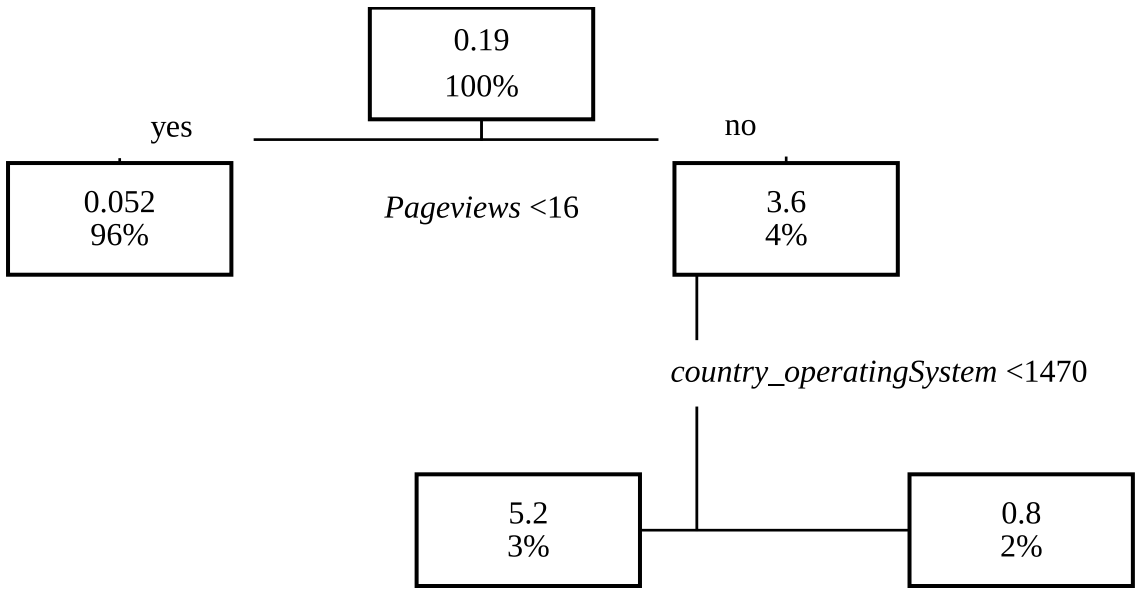


Figure 8. Pruned decision tree plot

Table 3

Decision tree model results for 5-fold cross validation

RMSE	Rsquared	MAE	Resample
1.1622459	0.2033551	0.2968985	Fold 1
1.665182	0.1844052	0.3101261	Fold 2
1.675085	0.1874603	0.3114603	Fold 3
1.634639	0.2098817	0.2995799	Fold 4
1.667272	0.1885605	0.3124219	Fold 5

Random Forests

Ensemble model that randomly uses multiple weak learners (smaller tree) to build smaller Decision Trees to predict a class. ‘M’ predictors are randomly chosen from the total pool of predictors and ‘N’ number of trees are built using random set of ‘M’ predictors every time. Each of these trees then ‘votes’ taking a decision and the most voted decision is picked as the final decision of the random forest.

Table 4

Random forest model results for 5-fold cross validation

RMSE	Rsquared	MAE	Resample
1.1688500	0.1926313	0.3232190	Fold 1
1.644976	0.1959153	0.3164610	Fold 2
1.696341	0.1593576	0.3329279	Fold 3
1.642704	0.1916511	0.3179381	Fold 4
1.677037	0.1713250	0.3236364	Fold 5

Bagging

This model is based on the concept of bootstrap aggregation. It is an ensemble model that uses multiple random samples and, on each sample, builds different models with different outputs. As a result, the outputs/predictions are averaged to minimize variance and therefore enhance model performances. Finally, models are voted in terms of best visit revenue prediction.

Table 5

Bagging model results for 5-fold cross validation

RMSE	Rsquared	MAE	Resample
1.603448	0.2218603	0.2922483	Fold 1
1.619383	0.2291292	0.2982671	Fold 2
1.622143	0.2323914	0.3006310	Fold 3
1.609305	0.2353193	0.2957630	Fold 4
1.635748	0.2255248	0.3004186	Fold 5

Light GBM (Gradient Boosting)

Gradient boosting is a machine learning technique which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion, i.e. trees are built as a sequence of iterations where each uses the previous tree splits as baseline predictors and generalizes them by allowing optimization of an arbitrary differentiable loss function. Light GBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient.

Table 6

Light GBM model training set results

RMSE	Rsquared	MAE
1.46870	0.3291515	0.255686

Artificial Neural Network (keras)

From a structural perspective, a neural network creates a network of nodes based on input predictors. Each node behaves like a neuron in the brain which weighs the predictors that are input and then the resultant weights from all nodes are assigned to predictors while building the models. Additionally, it is relevant to mention that the implemented ANN was built with the keras package. Specifically, we built basic sequential model using two layers. Successively, we experimented the model using batch size and epochs as tuning parameters. The following graphs show the training and validation RMSE of the neural network model for batch sizes of 256 and 512 sizes respectively.

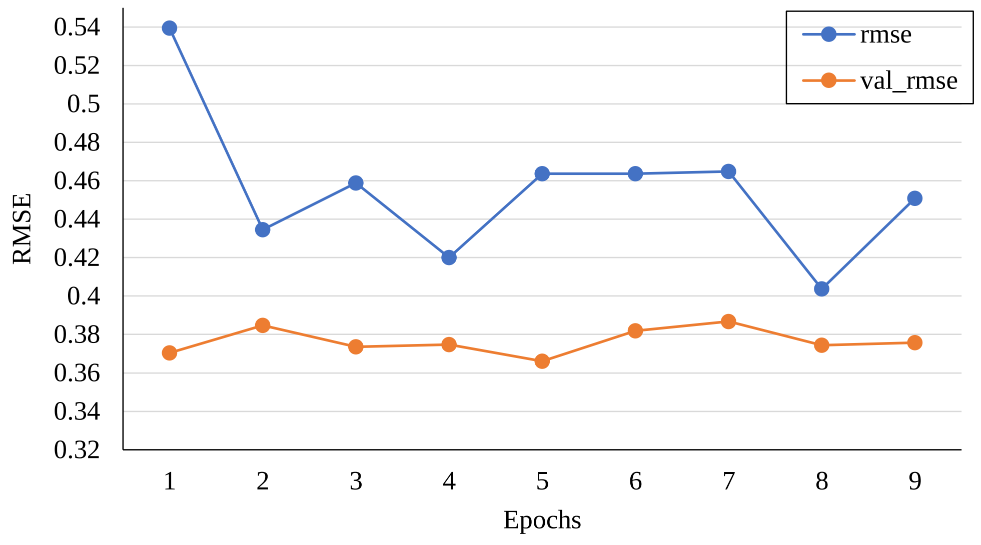


Figure 9. RMSE for neural networks, with batch size of 256

Table 7

RMSE for neural network values with batch size of 256

EPOCH	RMSE	VAL_RMSE
1	0.5394	0.3705
2	0.4345	0.3847
3	0.4588	0.3736
4	0.42	0.3747
5	0.4637	0.3661
6	0.4637	0.3819
7	0.4648	0.3868
8	0.4037	0.3745
9	0.4509	0.3758

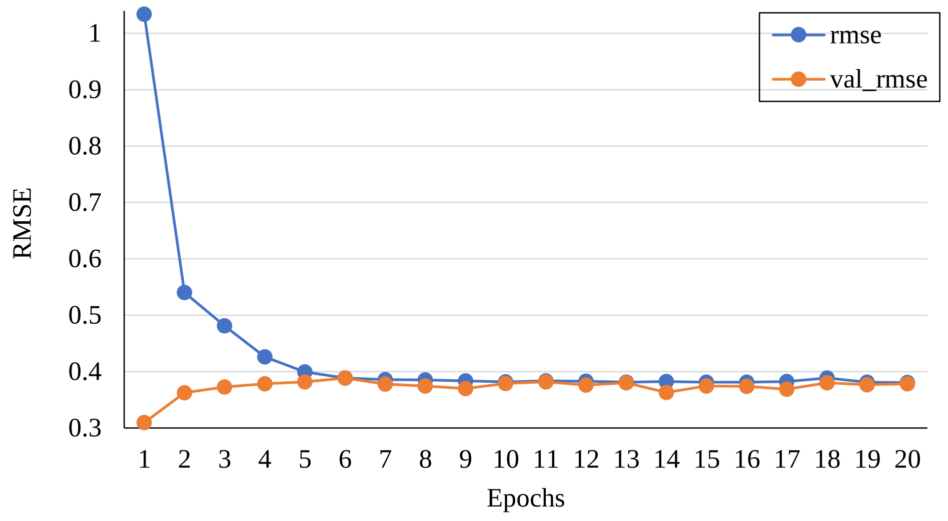


Figure 10. RMSE for neural networks, with batch size of 512.

Table 8

RMSE for neural network values with batch size of 512

EPOCH	RMSE	VAL_RMSE
1	1.0342	0.3095
2	0.5401	0.3623
3	0.4811	0.373
4	0.426	0.3784
5	0.3994	0.3818
6	0.3887	0.3885
7	0.386	0.3777
8	0.3854	0.3744
9	0.3833	0.3699
10	0.3818	0.3792
11	0.3833	0.3821
12	0.3827	0.376
13	0.3814	0.3803
14	0.3825	0.3629
15	0.381	0.3742
16	0.3815	0.3739
17	0.3823	0.3689
18	0.3885	0.3804
19	0.3812	0.3769
20	0.3805	0.3782

Results and Performance

Based on model results, we calculated Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) values to assess and measure performances and accuracy of models. All the performance metrics for choosing the final model were accessed on untouched testing set which as 30% of the initial sample size.

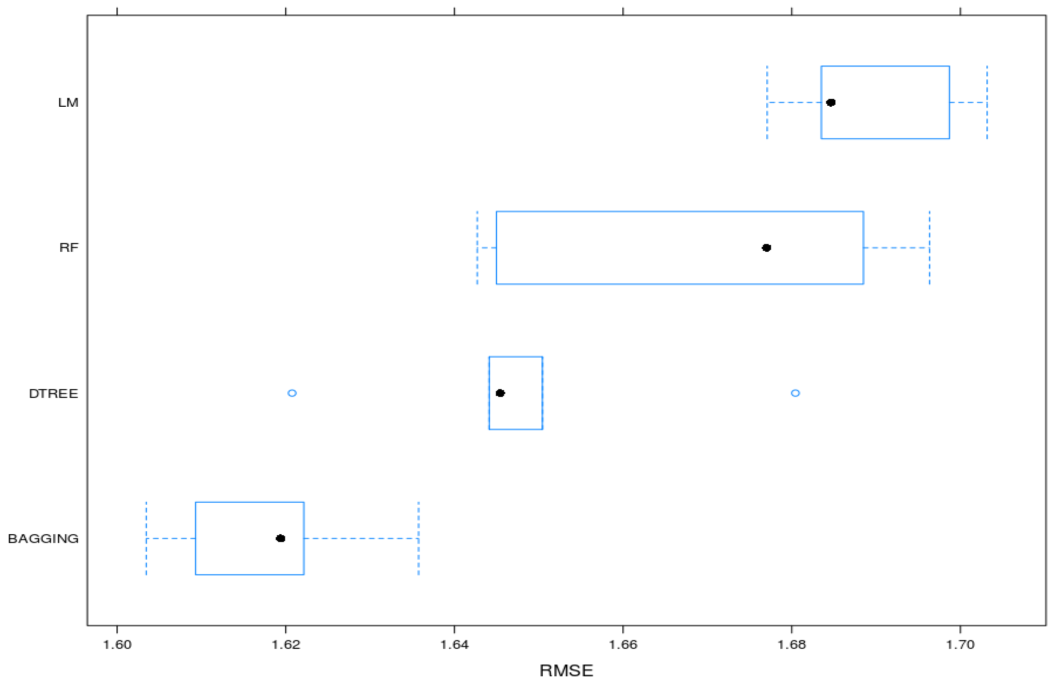


Figure 11. RMSE across folds

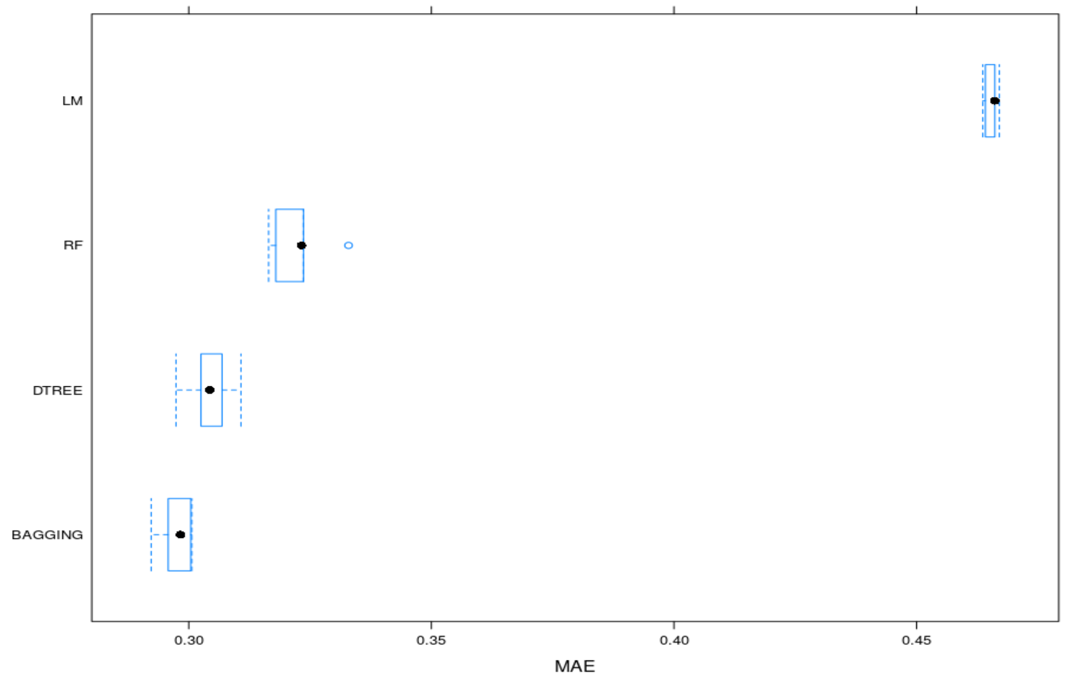


Figure 12. MAE across folds

Table 9

Light GBM model training set results

RMSE	Rsquared	MAE
1.46870	0.3291515	0.255686

Table 10

Neural Network model training set results

RMSE	Rsquared	MAE
0.5452967	0.5674557	0.191615

Test Set Results

Table 11

Data Models and their test results

Data Model	Test RMSE	RMSE in $	Test Rsquared	Test MAE
Linear Model	1.6894418	4.4164560	0.1651500	0.4646648
Decision Tree	1.6624814	4.2723780	0.1915308	0.3089065
Bagging	1.6206382	4.0563160	0.2324904	0.2979749
Random Forest	1.6660969	4.2914740	0.1936699	0.3216239
Light GBM	1.5375527	3.6531890	0.3100314	0.2681927
Artificial Neural Networks	0.5620291	0.7542284	0.4629900	0.2558987

The neural network, which is a very flexible model, was our top performer on the Test Dataset, while the second-best model was Light GBM. We would have liked to have further tuned the Artificial Neural Network parameters, like drop rate for each layer, but weren't able to do so due to time constraints; it will definitely be something we would like to look at in the future to improve model performance using the "tfruns" package to compare the runs in Keras.

Discussion and Recommendations

Based on the data analysis, our team concluded that methods Artificial Neural Networks and Gradient Boosting work well for this dataset with the context of prediction of transaction revenue. We used manual folds to make sure that RMSE is low and we aren't inducing bias in the dataset due to the high amount "0" transaction Revenue records.

Another approach that might be something to be considered in the future prospect of the project, is to primary treat this as a classification problem and predict if a visit Generates revenue "Yes/ No". And Apply separate models for only the visits that generate revenue.

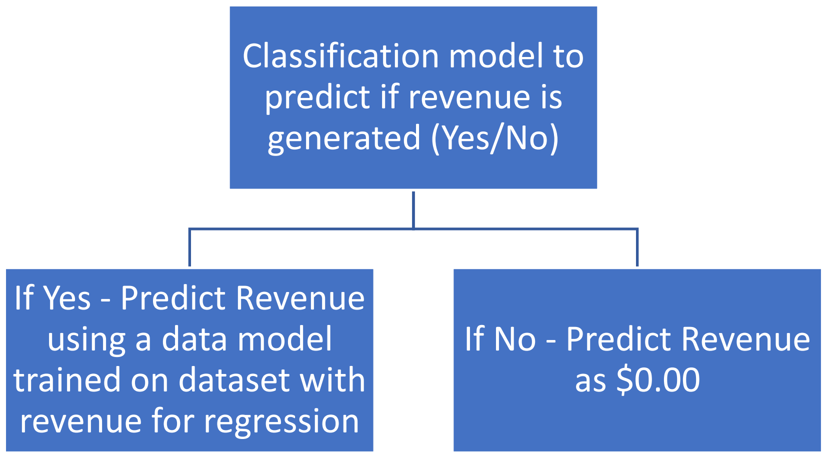

Classification model to predict if revenue is generated (Yes/No)
If Yes - Predict Revenue using a data model trained on dataset with revenue for regression
If No - Predict Revenue as $0.00

Figure 13. Another Approach to Problem

Another recommendation is potentially making ad campaigns decisions better by using the below feature importance graph and using complex features like *country_Operatingsystem*, *Day_of_Week* for visit to make targeted marketing decisions.

We also encourage the use of feature importance plot (Appendix B) to see the features that influence *transactionRevenue* generation for data driven decision making

Appendix A: R Code

```
#Loading Packages
library(tidyverse)
library(jsonlite)
library(magrittr)
library(stringr)
library(caret)
library(lubridate)
library(data.table)
library(scales)
library(ggplot2)
library(lightgbm)
library(doParallel)
library(corrplot)
library(rpart.plot)
library(rsample)
library(rattle)
library(keras)

#Defining Data Types in File for Faster Loading of 27GB file
ctypes <- cols(fullVisitorId = col_character(),
            channelGrouping = col_character(),
            date = col_datetime(),
            device = col_character(),
            geoNetwork = col_character(),
            socialEngagementType = col_skip(),
            totals = col_character(),
            trafficSource = col_character(),
            visitId = col_integer(),
            visitNumber = col_integer(),
            visitStartTime = col_integer(),
            hits = col_skip(),
            customDimensions = col_skip())


#Loading Dataset
tr <- read_csv("train_v2.csv", col_types = ctypes).

#Functions to Flatten Json Fields
flatten_json <- . %>%
  str_c(., collapse = ",") %>%
  str_c("[", ., "]") %>%
  fromJSON(flatten = T)

parse <- . %>%
  bind_cols(flatten_json(.$device)) %>%
  bind_cols(flatten_json(.$geoNetwork)) %>%
  bind_cols(flatten_json(.$trafficSource)) %>%
  bind_cols(flatten_json(.$totals)) %>%
  select(-device, -geoNetwork, -trafficSource, -totals)

#Parse Dataset to Flatten Json to Table Columns
tr <- parse(tr)

#Seperate Out Response Variable from Train Set
tr_te$transactionRevenue <- as.numeric(tr$transactionRevenue)
y <- log1p(as.numeric(tr$transactionRevenue))
y[is.na(y)] <- 0
tr$transactionRevenue <- NULL
tr$campaignCode <- NULL

#Auxilarry Functions for Processing
has_many_values <- function(x) n_distinct(x) > 1

#Defining all Values that should be considered missing data
is_na_val <- function(x) x %in% c("not available in demo dataset", "(not provided)",
                          "(not set)", "<NA>", "unknown.unknown",  "(none)")


#Data Preprocessing

tr_te <- tr %>%
  select_if(has_many_values) %>%
  mutate_all(funs(ifelse(is_na_val(.), NA, .))) %>%
  mutate(pageviews = ifelse(is.na(pageviews), 0L, as.integer(pageviews)),
      visitNumber =  visitNumber,
      newVisits = ifelse(newVisits == "1", 1L, 0L),
      bounces = ifelse(is.na(bounces), 0L, 1L),
      isMobile = ifelse(isMobile, 1L, 0L),
      adwordsClickInfo.isVideoAd = ifelse(is.na(adwordsClickInfo.isVideoAd), 0L, 1L),
      isTrueDirect = ifelse(is.na(isTrueDirect), 0L, 1L),
      browser_dev = str_c(browser, "_", deviceCategory),
      browser_os = str_c(browser, "_", operatingSystem),
      browser_chan = str_c(browser,  "_", channelGrouping),
      campaign_medium = str_c(campaign, "_", medium),
      chan_os = str_c(operatingSystem, "_", channelGrouping),
      country_adcontent = str_c(country, "_", adContent),
      country_medium = str_c(country, "_", medium),
      country_source = str_c(country, "_", source),
      dev_chan = str_c(deviceCategory, "_", channelGrouping),
      date = as_datetime(visitStartTime),
      year = year(date),
      wday = wday(date),
     hour = hour(date))



#Feature Engineering - New Combination Features

for (i in c("city", "country", "networkDomain"))
  for (j in c("browser", "operatingSystem", "source"))
    tr_te[str_c(i, "_", j)] <- str_c(tr_te[[i]], tr_te[[j]], sep = "_")



#Feature Engineering - New User Group Features Ex.No of Unique Users per Day,Hr

for (grp in c("wday", "hour")) {
  col <- paste0(grp, "_user_cnt")
  tr_te %<>%
    group_by_(grp) %>%
    mutate(!!col := n_distinct(fullVisitorId)) %>%
    ungroup()
}

#Feature Engineering - Aggregate Functions on Network Domain, Referral Path and
Number of Visits

fn <- funs(mean, min, max, sum, .args = list(na.rm = TRUE))
for (grp in c("networkDomain", "referralPath", "visitNumber")) {
  df <- paste0("sum_by_", grp)
  s <- paste0("_", grp)
  tr_te %<>%
    left_join(assign(df, tr_te %>%
                     select_(grp, "pageviews") %>%
                     group_by_(grp) %>%
                     summarise_all(fn)),  by = grp, suffix = c("", s))
}

#Final Steps in Preprocessing
tr_te %<>%
  select(-date,-dev_chan,-fullVisitorId, -visitId, -visitStartTime, -hits, -
totalTransactionRevenue) %>%
  mutate_if(is.character, funs(factor(.) %>% addNA %>% as.integer)) %>%
  mutate_if(is.integer, funs(as.numeric)) %>%
  select_if(has_many_values)

#Calculate Near Zero Variance at 95%
nzv <- nearZeroVar(tr_te)
```

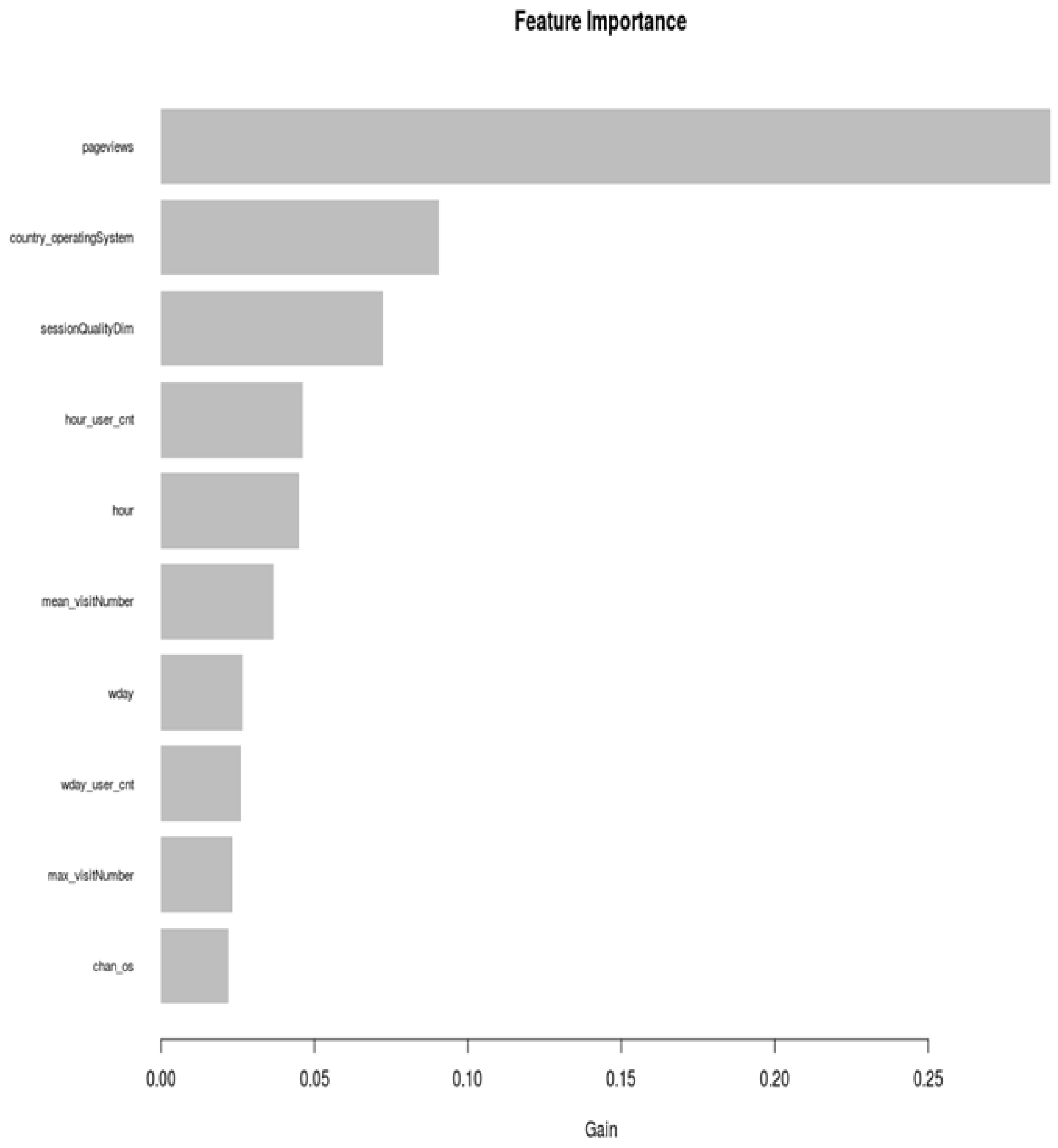
```
#Removing Columns with near zero variance

tr_te <- tr_te[,-nzv]
glimpse(tr_te)

#Create Partition and Folds
part_ind <- createDataPartition(y,p=0.7,list=FALSE)
train.data <- tr_te[part_ind,]
test.data <- tr_te[-part_ind,]

train.output <- y[part_ind]
test.output <- y[-part_ind]

folds_5=createFolds(y=train.output,k=5,returnTrain=TRUE)


#Starting Model Training

print("==== Linear Model========")

linear_model <- train(y=train.output,
                x=as.matrix(train.data),
                metric="RMSE",
                method = "lm",
                trControl = trainControl(method="cv",number=5,index=folds_5)
                )
linear_model$resample
saveRDS(linear_model,"linear_model.rds")
res_linear_model <- predict(linear_model,as.matrix(test.data))
postResample(res_linear_model,test.output)

print("======Decision Tree===========")

dtree_model1 <- train(y=train.output,
                x=as.matrix(train.data),
                metric="RMSE",
                method = "rpart",
                trControl = trainControl(method="cv",number=5,index=folds_5)
)
saveRDS(dtree_model,"dtree_model.rds")
res_dtree_model <- predict(dtree_model1,as.matrix(test.data))
RMSE(res_dtree_model,test.output)
fancyRpartPlot(dtree_model1$finalModel)
```

```
print("======Bagging===========")

bag_model <- train(y=train.output,
            x=as.matrix(train.data),
            metric="RMSE",
            method = "treebag",
            ntree = 5,
            trControl = trainControl(method="cv",number=5, index=folds_5)
)
saveRDS(bag_model,"bag_model.rds")
res_bag_model <- predict(bag_model,as.matrix(test.data))
RMSE(res_bag_model,test.output)


print("======Random Forest in Parallel===========")

rf_model <- train(y=train.output,
            x=as.matrix(train.data),
            metric="RMSE",
            method = "rf",
            ntree=32,
            nthread=32,
            trControl = trainControl(method="cv",number=5,index = folds_5)
)
saveRDS(rf_model,"rf_model.rds")

rf_model$resample

res_rf_model <- predict(rf_model,as.matrix(test.data))

postResample(res_rf_model,test.output)

print("======Light GBM===========")

#Defining Categorical Variables for LGBM optimization
categorical_feature <-
c("visitNumber","networkDomain","country","source","operatingSystem","deviceCategory"
,"region","browser","metro","city","continent"
                 ,"subContinent","channelGrouping","medium")


train <-  lgb.Dataset(data=as.matrix(train.data),label=train.output,categorical_feature
=categorical_feature)
```

```
print("-------Training LightGBM -------")
lgb_model <- lightgbm(
   data = train
  , learning_rate = .01
  , objective="regression"
  , num_leaves = 48
  , min_data_in_leaf = 30
  , max_depth = 8
  , min_child_samples=20
  , boosting = "gbdt"
  , feature_fraction = 0.8
  , metric = c('rmse','mae')
  , lambda_l1 = 1
  , lambda_l2 = 1
  , verbose = 1
  , seed = 0
  , nrounds = 60000
  , eval_freq = 100,
  , early_stopping_rounds = 200
)
lgb.get.eval.result(lgb_model,"train","RMSE")
lgb.importance(lgb_model, percentage = TRUE) %>% head(10)
res_lgbm <- predict(lgb_model,as.matrix(test.data))
postResample(res_lgbm,test.output)
tree_imp <- lgb.importance(lgb_model, percentage = TRUE)
lgb.plot.importance(tree_imp, top_n = 10, measure = "Gain")

print("---------- Neural Network ----------")
# Normalize training data
# Test data is *not* used when calculating the mean and std.

train.data <- scale(train.data)

# Use means and standard deviations from training set to normalize test set
col_means_train <- attr(train.data, "scaled:center")
col_stddevs_train <- attr(train.data, "scaled:scale")
test.data <- scale(test.data, center = col_means_train, scale = col_stddevs_train)

rmse <- function(y_pred,y_true){
  K <- backend()
  return(K$sqrt(K$mean(K$square(y_pred - y_true))))

}
```

```
build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 256, activation = "relu",
                input_shape = dim(train.data)[2]) %>%
    layer_dropout(rate = 0.75) %>%
    layer_dense(units = 256, activation = "relu") %>%
    layer_dropout(rate = 0.25) %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list(custom_metric("RMSE",rmse),"accuracy","mean_absolute_error")
  )

  model
}

model <- build_model()
model %>% summary()

print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)

history <- model %>% fit(
  as.matrix(train.data),
  train.output,
  epochs = 100,
  batch_size=256,
  validation_split = 0.2,
  verbose = 1,
  callbacks = list(callback_early_stopping(patience = 5),print_dot_callback)
)

#NN Test Output
model %>% evaluate(train.data, train.output)
test_predictions <- model %>% predict(test.data)
caret_model_resample <-
resamples(list(LM=linear_model,DTREE=dtree_model1,BAGGING=bag_model,RF=rf_model
))
bwplot(caret_model_resample, metric="RMSE")
```

Appendix B : Feature Importance Plot From LightGBM


Feature Importance
pageviews
country_operatingSystem
sessionQualityDim
hour_user_cnt
hour
mean_visitNumber
wday
wday_user_cnt
max_visitNumber
chan_os
Gain