Google Analytics Customer Revenue Prediction

GR5261 Final Project
Instructor: Zhiliang Ying

Runjie Lyu(rl3032): Coding: Data pre-processing, Exploratory data analysis, Linear Mixed model; Report Writing: Introduction, Exploratory data analysis, Linear mixed model
Zixiao Wang(zw2513): Coding: Time series model, XGBoost classification and regression models; Report Writing: Time series model, XGBoost regression model
Sitong Liu(sl4460): Coding: Exploratory data analysis, Logistic regression model, Random forests model; Report writing: Exploratory data analysis, Classification models
Shiwei Hua(sh3804): Coding: Time Series Model; Report Writing: Time series model, Performance measurement, Conclusion, Reference; Slides (all parts)
Zhenru Han(zh2348): Coding: Attempted advanced models Glmnet, Keras

1 - Project Introduction

This project targets to analyze the Google Merchandise Store (also known as G Store, where Google Swags are sold, including T-shirts, water bottles, notebooks and etc.) to predict some information related to transaction revenue. Products are priced in local currencies and the price range is between \$0.99 and \$99.9.

We collected our dataset from Google Customer Revenue Competition on Kaggle.We found that train set included 903653 observations with 36 variables and test set contained 804684 observations. Except dependent variable transaction revenue, we can divide other variables into 6 groups: Visitor Info, Visitor Num, Channel Grouping, Device, Geo Networks, and Advertisement.

2 - Main Objectives

Our main objective is to predict transaction revenue for the following 60 days and to predict transaction revenue based on each customer visiting Google Store. Before selecting models, we would perform a detailed Exploratory Data Analysis. After that, we would break down our research question into three parts.

- Firstly, we plan to forecast daily transaction revenue by using a time series model.
- Secondly, we will classify whether a customer is going to purchase some products. For this part, we will implement the logistic regression, random forests and XGBoost classification models.
- The final part is to predict the total revenue a customer is going to make by using regression models. We will use Linear Mixed Models and Boosting methods such as XGBoost regression.

To visualize our model performance, we would like to use the root mean square error as the evaluation method for time series and regression models. RMSE is defined as:

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

Also for classification models, we will compare different models with their accuracy, precision, recall and F_1 score, where F_1 score is computed by

$$F_1 = \left(\frac{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

3 - Description of Data Analysis

This section contains the description of Exploratory Data Analysis (EDA) and Data Processing based on EDA in order to help us better understand the dataset.

3.1 - Transaction Revenue

According to Missing Data Analysis (See Appendix 2), 98.7% did not make transaction revenue during their visit. This result indicates the imbalance of our target variable. In addition, revenue range is between 0.99 and 23028.54 and most revenue is lower than 1. Therefore, we apply a data transformation on it: “log1p” function transforms x to log(1+x). It converts our data to approximately normal distribution with the median 17.7. In natural log scale, most transaction revenue follows between 15 and 20 (See Appendix 3, Figure 3.1).

3.2 - Channel Grouping, Operating System, Device, and Browser

Channels are defined as how users arrive to the website. Most sessions come from “organic search”, but “referral” contributes most transaction revenue. Comparison between number of sessions and revenue indicates that desktop is still most important device for G Store online shopping and it generates much more revenue than the other two device types. There are only 7 system in the dataset: windows, macintosh, android, ios, linux, chrome os, and windows phone. Macintosh stand out because it generates much higher revenue with fewer sessions. Finally, analysis on browser gives us roughly the same views on Windows and Mac users, but it seems that Chrome is also popular on Mac because it contributes more revenue than Windows from operating system category (See Appendix 4).

3.3 - Visits, Hits, and Page Views

Customers usually prefer to take multiple views if they are interested in a product, so the page views may be an essential factor for predicting revenue transaction. From the histogram of pageviews, we find that most visitors view less than 5 pages during a session. It hardly leads to a transaction since a shopping cycle on G store needs at least 4 to 6 pages. The distribution of total revenue grouped by pageviews agrees that revenue dramatically increased when pageviews are greater than 10 (See Appendix 5). More Exploratory Data Analysis can be founded in the Appendix 3-5.

3.4 - Data Processing

Based on EDA results, we first dropped all variables with too many missing values, and then dropped several variables with high correlations. In the end, our 13 variables included all Visit ID and Time variables, Page Views, Hits, Channel Grouping, Country, Operating System, and Browser. After that, we also splitted train and test again based on whether a visit contributes to transaction revenue. In regression models, we only used data set only including customers generated revenue.

4 - Statistical Methods and Analysis

This section includes the models dealing the three research questions. To be more specific, time series model in predicting daily revenue, classification models in predicting whether a customer will purchase after looking through the page and regression models for predicting the amount of money each customer might pay based on the related information.

4.1 - Daily Revenue Prediction for the Following 60 Days

4.1.1 - Time Series Model Building

In order to predict daily revenue for the next 60 days, we came up with an idea that a time series model could fit data well based on the previous data analysis. To begin with, we started plotting

ACF and PACF to check whether daily revenue data fits some patterns. According to these plots, it was obvious that daily revenue fitted a weekly pattern. In other words, the frequency of daily revenue data is 7. After having such an information, we tried to fit a seasonal ARIMA model. Based on the lowest AIC, the most suitable model is SARIMA(1,0,1)*$(2,1,0)_7$ model.

4.1.2 - Cross Validation

After building the time series model, we performed cross validation based on our seasonal ARIMA model. The idea for cross validation is to use first n days to fit model and the $(n+1)_{th}$ day to compute error. After this step, we used the first (n+1) days to fit model and the $(n+2)_{th}$ day to find the corresponding error. We did this step by step and had a data set for these cross validation errors. We calculated RMSE by using this cross validation method and used it to predict the test error. For training error, we used all the data to fit the seasonal ARIMA model and get the residuals to compute RMSE. As a result, the cross validation error is 2.290954, and the training error is 2.227386, which shows that the model is relatively suitable and is not overfitting.

4.1.3 - Prediction for the Following 60 Days

To predict the next 60 days revenue, we used a rolling window as our data set to predict. To be more specific, we used the last 120 days to predict the revenue of the first following day. After having the first prediction, we added this prediction to the rolling window and removed the first observation from the rolling window. In this way, we would have a new 120 observation to predict the second day revenue. We repeated such a process and got the 60-day predictions. This method gave us a much better 95% confidence interval compared with the confidence interval directly using the whole dataset to predict 60 days at a time (See Appendix 6).

4.2 - Analysis of Whether an Individual Would Make a Purchase

To answer the second research question, “Whether an individual would make a purchase or not? ”, there are three steps that have been involved. Firstly, the imbalanced dataset with

$$\frac{Revenue\ transaction\ made}{No\ Revenue\ made} \approx \frac{1}{50}$$

has been processed. Secondly, logistic regression, random forests, and XGBoost classification models have been applied for analysis. Finally, all three models was measured the performance in test dataset.

4.2.1 - Imbalanced Dataset Processing

In order to build the classification models, we firstly converted the amount of revenue transaction into a binary variable, 1 (a revenue transaction has been made) and 0 (no revenue has been made) in both train and test datasets. After converting the variable, we recognized that the dataset is highly imbalanced between a revenue transaction made and no revenue made. Thus, we rebalanced the train dataset in to 1:10 instead of 1:50, and used original test dataset for performance measurement. We also tried 1:5 and 1:15 for model building, but the ratio of 1:10 gave us the best result.

4.2.2 - Model Building

After data processing, we performed the baseline model, logistic regression model, and improvement models, random forests and XGBoost classification models in the train dataset. For these models, we worked on the binary response variable, whether a purchase is made or not, and explanatory variables: Channel Grouping, Date, Visit Number, Visit Start Time, Browser, Operating System, Country, Hits, Page Views. (For logistic regression model, we dropped variables Country and Operating System for model building since there are too many dummy variables in model building process, which not only took too much time, but also resulted in a relatively worse classification.) Having the models based on the train dataset, all three models was tested in the original test dataset to check the performance.

4.2.3 - Performance Measurement

Since the confusion matrices returned close values for three models, we further measured the accuracy, precision, recall and F_1 score for all models (See Appendix 7).

Although among the confusion matrices, logistic regression model has correctly predicted the most individuals did not make purchases. However, it performed worstly on precision, recall and F_1 score values in performance measurement table. In this way, logistic regression model was the weakest model to predict whether a person would make a purchase or not, and it should not be used for the research question. It is hard to determine the best model between random forests and XGBoost classification models since they have very similar results. Therefore, random forests and XGBoost classification models could both be used in exploring whether an individual would make a purchase in the future.

4.3 - Analysis of the Total Revenue an Individual Would Make

4.3.1 - Linear Mixed Effects Model

The first model we implemented is linear mixed effects model. It is a simple extension of linear regression but allows to both fixed and variable effects. Therefore, it works better on categorical data. We started with treating Visit ID as the intercept and used this model as the benchmark. Then we added one variable into our formula each time. Finally we got 8 models in total. To compare model performance, we introduced ANOVA table and selected our best model with the lowest AIC. Linear Mixed Model 6 that included Page Views, Hits, Visit number, Channel Grouping, Browser, and Operating System has AIC Score 26130. Fitting this model on train set, we got its RMSE as 0.5878892. However, predicting this model on test set received a much higher RMSE, 1.05369. The results imply a potential overfitting.

4.3.2 - XGBoost Regression Model

The second model we tried to use is a regression tree method, XGBoost regression model. The idea of XGBoost regression method is to use decision tree ensembles. The tree ensemble model consists of a set of classification and regression trees. In other words, this method first classifies inputs and use their average as the output value for regression purpose. As the number of training rounds became more and more, the tree would always adjust their conditions for determining new classification. In this way, a regression model would be built. In our case, we noticed that some variable might have highly correlated or even have including relationship. Thus, we used Page Views, Hits, Visit number, Channel Grouping, Browser, Country and Operating System for prediction to prevent our prediction from misclassification. We set the number of rounds for boosting as 1000 for training, and it gave us an RMSE value 1.016699. Using this model on test set, the RMSE is 1.065387, which is pretty close to the training error. This shows that this model is much better than the previous one and we decided to use XGBoost regression model to predict personal purchase amount.

5 - Conclusions

To answer our research questions in the beginning part, we have these following conclusions based on our models:

- SARIMA(1,0,1)*(2,1,0)$_7$ would produce the best predictions on daily revenue in the future 60 days.
- Random forests model and XGBoost classification model would more accurately determine customer behavior, which give the prediction for customers who might purchase products in Google Store.
- XGBoost regression model would be evaluated as the best model based on RMSE for predicting transaction revenue amount per customer.

For further thinking of this project, since our data is highly imbalanced and there are too many levels of categorical data, we would think further on the improvements on data balancing and dimension reduction.

6 - Acknowledgement

We want to express our sincere gratitude to Professor Ying and Teaching Assistants George and Jitong. Thanks for their patience, help, and guidance on this project.

7 - Reference

- Google Merchandise Store, shop.googlemerchandisestore.com/.
- Google Analytics Customer Revenue Prediction, Google, Dec. 2017, www.kaggle.com/c/ga-customer-revenue-prediction/overview/description.
- Hyndman, Rob J, and George Athanasopoulos. Forecasting: Principles and Practice. Monash University, 2016.Chapter: Evaluating forecast accuracy
- “Introduction to Boosted Trees.” XGBoost, 2016, xgboost.readthedocs.io/en/latest/tutorials/model.html.
- Box, G.E.P., and D.R. Cox. *An Analysis of Transformations*. University of Wisconsin, Birkbeck College, University of London, 1964, www.ime.usp.br/~abe/lista/pdfQWaCMboK68.pdf.

Appendix 1: Independent Variables Introduction

After processing train data, we finally obtained 36 variables. All independent variables can be divided into 6 groups: visitor info, visit number, channel, geo networks, devices, and advertisement.

1. Visitor Info: Full Visitor ID, Session ID, Visitor ID, Visit Number, Visit Start Time, Date
2. Visit Num: Visits, Hits, Pageviews, Bounces, New Visits
3. Channel: Channel Grouping, Campaign, Source, Keyword, Is True Direct, Referral Path
4. Geo Networks: Continent, Sub Continent, Country, Region, Metro, City, Network Domain
5. Device: Browser, Operating System, Is Mobile, Device Category
6. Advertisement: AD Content, and ADword Click Info Page, ADword Click Info Slot, ADword Click Info Gclid, ADword Click Info AdNetworkType, and ADword Click Info IsVideoAd

Since not all variables have been clearly interpreted, we only analyzed some of variables that potentially might be important features.

Variables	**Data type**	**Description**	**Level**
Date	Quantitative	The date on which a customer visited the store	20160801 - 20170801
Full Visitor Id	Categorical	A unique identifier for each user of the store	Length : 903653
Session Id	Categorical	A unique identifier for each visit to the store	Length: 903653
Visit Number	Quantitative	The session number for the user	1-395
Visit Time	POSIXct	The time on which a customer started to visit the store	ex: “2016-09-02 15:33:05”
Browser	Categorical	The browser category that a visitor using	Chrome; Safari; Internet Explorer; Edge; Firefox; Opera Mini; Safari (in-app)
Operating System	Categorical	The operating system that an user accessing the page	Windows; Android iOS; Macintosh; Chrome OS; Linux
Is Mobile	Binary	An indicator to show whether a customer visted the store from mobile devices	True; False
DeviceCategory	Categorical	The device that the a visitor using	desktop; tablet; mobile
Geo Networks	Categorical	These sections contain information about geography of a visitor, it can be further down into continents, sub	An example of a visitor from Madrid can be like: Europe, Southern Europe,

		continents, countries, regions, metro and cities.	Spain, Community of Madrid, Madrid.
Network Domain	Categorical	The info of network providers	ex: dodo.net.au
Hits	Quantitative	The hit times for a visitor	1-500
Page Views	Quantitative	Number of page that a visitor views	1-469
Bounces	Quantitative	A *bounce* is a single-page session on the site. Bounce rate is defined as the percentage of all sessions on your site in which users viewed only a single page and triggered only a single request to the Analytics server.	0,1
Transaction Revenue	Quantitative	The target variable that we need to predict, which is the revenue that a customer made from a single visit.	0-23028.54
Source	Categorical	This section contains info from which the session orginated	ex: Google, baidu
Medium	Categorical	An indicator to categorize source path	ex: organic, referral
Keyword	Character	This section contains keywords that a visitor used in the search engine	ex: buy google souvenirs
Is True Direct	Binary	A binary indicator that displayed whether a visit is direct or through referral path.	True; False
Ad content	Categorical	This section displayed the content that visitors clicked to access G store	ex: Google Merchanise Collection

Table 1.1: The table shows descriptive statistics of variables in the dataset.

Appendix 2: Missing Data Analysis

In order to better understand the train dataset, checking the missing value would be important. There are 98.7% users did not make the transaction while looking the product pages. In addition, all advertisement variables have missing rate around 97.6%. This result suggests to remove advertisement variables from the set. Similarly, “keyword”, “campaign”, and “referral path” from channel group also have high missing rate between 63.4% and 96.2%. Also, detailed information on geography of visitors such as “city” and “region” miss approximately 60%, which implies a better analysis of transaction revenue on country level.

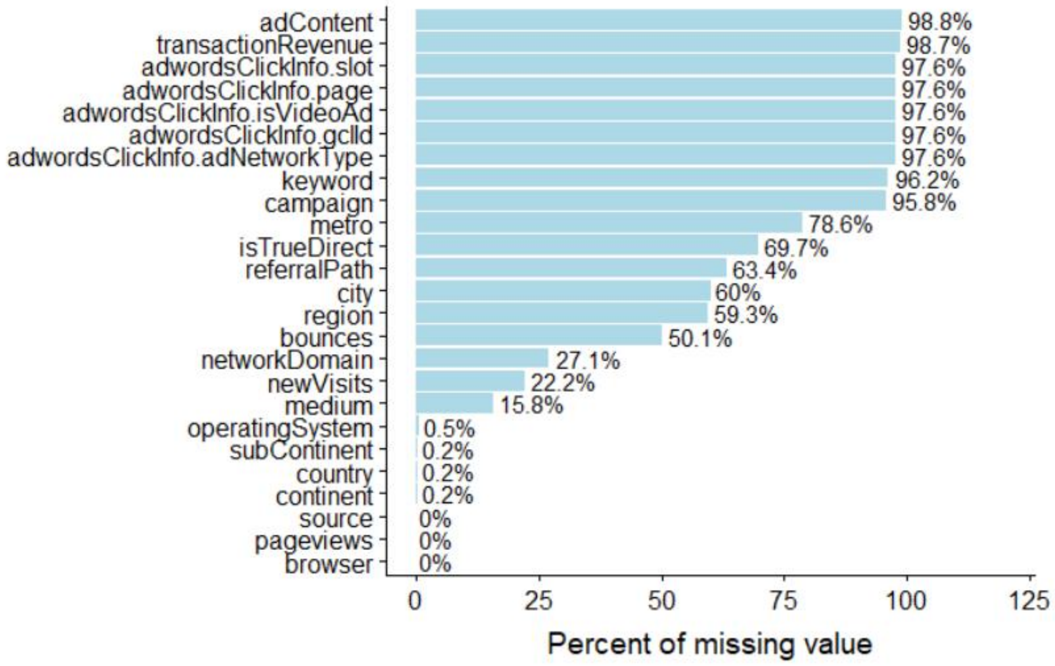


Figure 2.1 The above figure indicates the percentage of missing value, there are 98.7% of users did not make the transaction.

Appendix 3: Transaction Revenue

The original transaction revenue distribution indicates that most revenues locate between 0.99 to 99.9. It matched with our preliminary analysis of G Store's product prices, but the revenue distribution challenges our prediction. Thus, we transform the transaction revenue variable into log scale through "log1p" function in R. In the following table and histogram, log transaction revenue presents an approximately normal distribution.

Variable	Minimum	1st Quantile	Median	Mean	3rd Quantile	Maximum
Transaction Revenue	0.01	24.9	49.5	133.7	107.7	23028.54
Log Revenue	9.2	17.0	17.7	17.8	18.5	23.9

Table 3.1 Comparison of transaction revenue and log transaction revenue

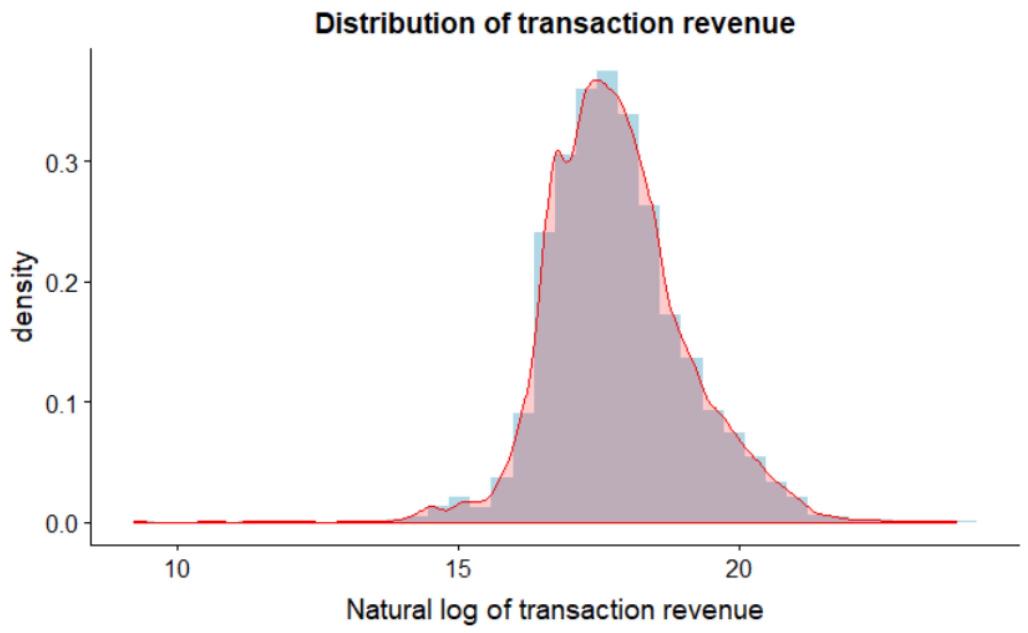


Figure 3.1 the above figure is roughly a normal distribution of natural log of transaction revenue with mean equals 17.8 and median equals 17.7.

For transaction revenue, "NA" value can be treated as 0 since "NA" means no revenue made during the single visit. After we convert all "NA" to 0, the distribution also shift to the left (See Table 3.2).

Variable	Minimum	1st Quantile	Median	Mean	3rd Quantile	Maximum
Log Revenue	0.00	0.00	0.00	0.2271	0.00	23.8644

Table 3.2 Log Revenue after converting all NA to 0.

Although the daily transaction data are very volatile, it seems that a “high and low” pattern is regular over the year. Using the “smooth” function from ggplot2 in R, we observe a relatively stable trend from August 2016 to August 2017.

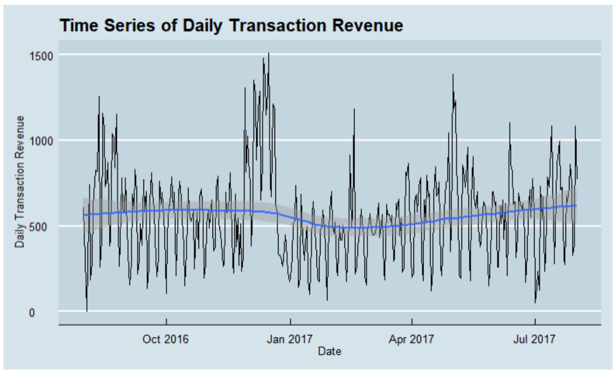


Figure 3.2 Time Series of Daily Transaction Revenue

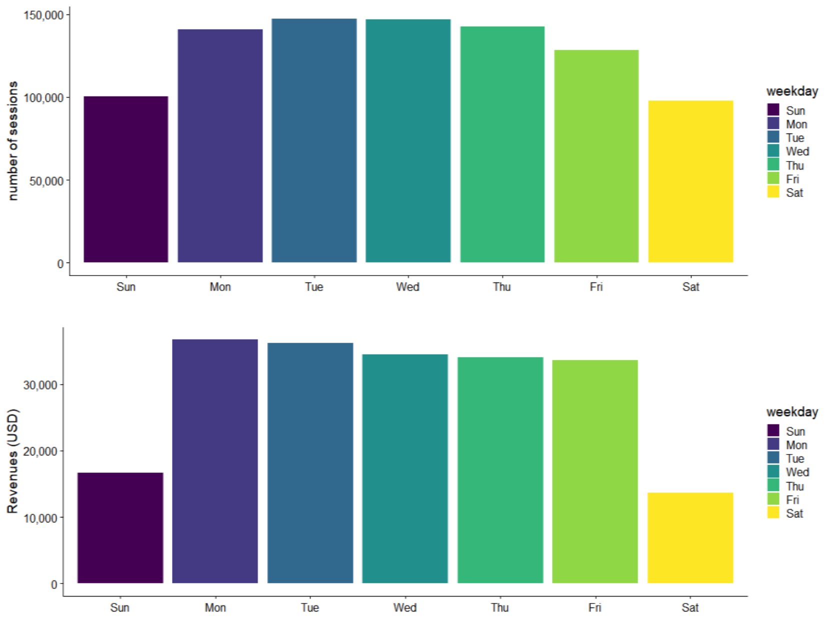


Figure 3.3 Comparison plot between the number of sessions and revenue by weekday

We further analyze the transaction revenue by Time Series model. In addition, we compare the trend between number of sessions and revenue. The weekday comparison present that session number and purchase are higher through Monday to Friday than the weekend. From this relationship, we think that most customers purchased Google Swag at G Store are not for individual interest. They might purchase T-shirts or water bottles for the whole office.

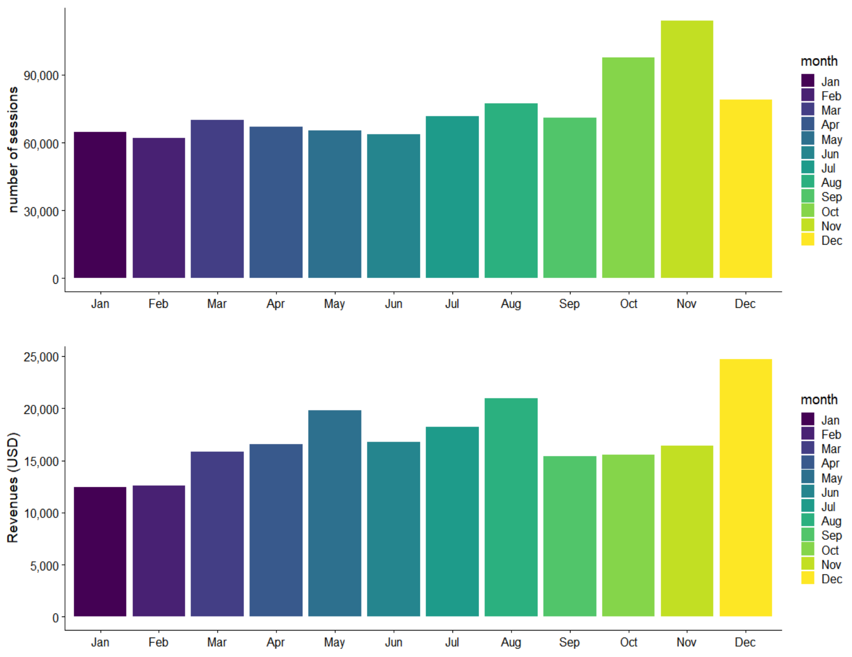


Figure 3.4 Comparison plot between the number of sessions and revenue by weekday

From the above figure, we find that the number of sessions are close to uniform distribution every month except November. Similarly, revenue in December is higher than monthly average. It is reasonable to assume that purchase close to the end of year is prepared for event celebrations such as Thanksgiving or Christmas.

Appendix 4: Channel Grouping, Operating System, Device, and Browser

Channels is one of the most important factor feature since it is defined by Google Analytics as how a customer come to the G Store website. It related to other variables such as “source”, “referral path” in the “channel grouping” group (See Appendix 1). Organic Search means that a customer arrive the G Store from a search engine; Social is related to any social media websites or apps; Referral means that a customer come from G Store’s advertisement on other websites.

Organic Search is the main access to the G Store and this channel also contribute essential amount of revenue. Social path hardly can generate revenue even though it contributes around half number of sessions than Organic Search. It demonstrates that customer are less interested in placing orders after they redirect from the social media.
However, the referral path outperforms any other channels: it generates most transaction revenue. This result indicates that referral path is the most useful channel for selling products from G Store.

Device category contains desktop, mobile, and tablet. According to Figure 4.3 and 4.4, we observe that most people prefer to use desktop to visit products and generate transaction revenues. On the other side, customers rarely placed orders on mobile and tablet devices since they are not as user-friendly as computers.

In addition to device category usage, the operating system usage could also be an important factor when predict the revenue transaction. There are seven operating systems that have been recorded, windows, macintosh, android, ios, linux, chrome os, and windows phone. Although most people use windows to visit the system, macintosh users actually makes significantly more revenue transaction. Furthermore, Figure 4.5 and 4.6 also confirm that mobile systems are less attractive to customers who plan to purchase products from G Store.

Analysis on browser almostly leads to the same result as that on device and operating system. The main reason is that “Safari” visitors also used macintosh operating system, but “Chrome” outperforms “Windows” from operating system category on transaction revenue. This result let us believe that most Macbook users also Chrome as the dominant browser.

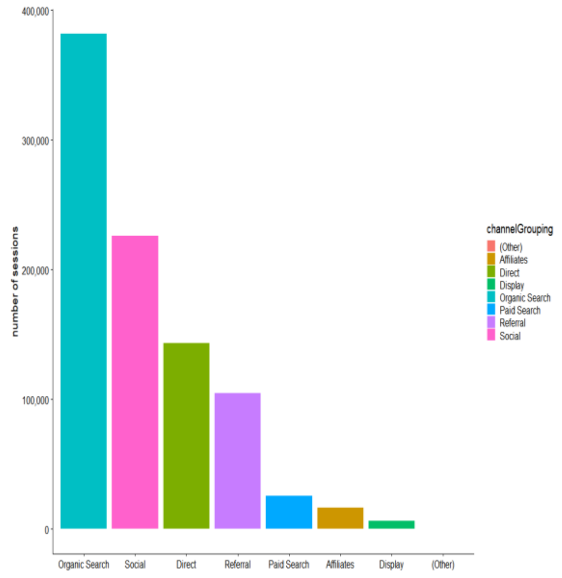


Figure 4.1 Number of sessions by channel grouping

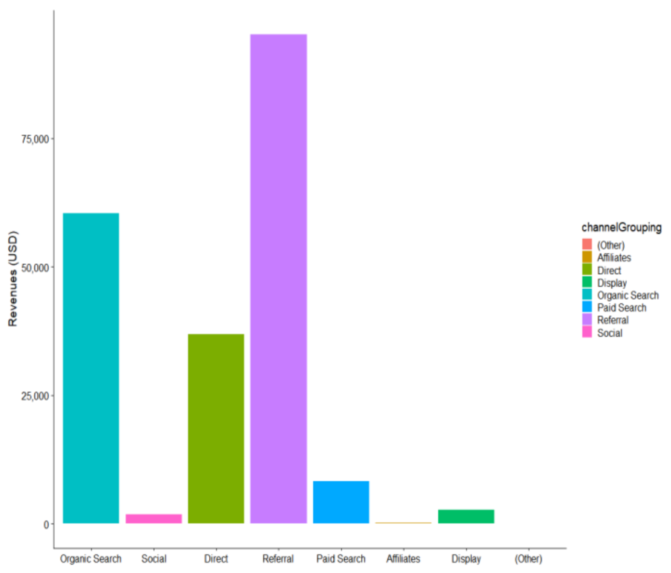


Figure 4.2 Revenue by channel grouping

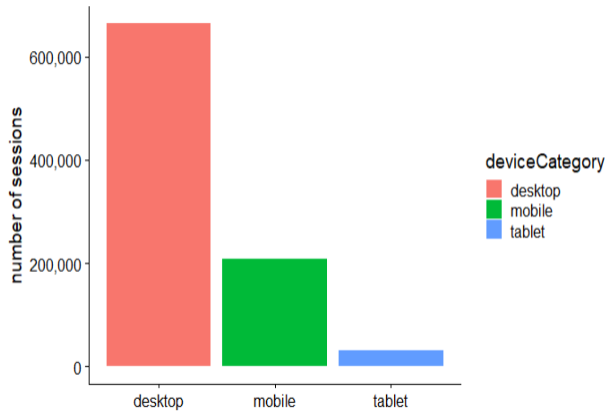


Figure 4.3 Number of sessions by device category

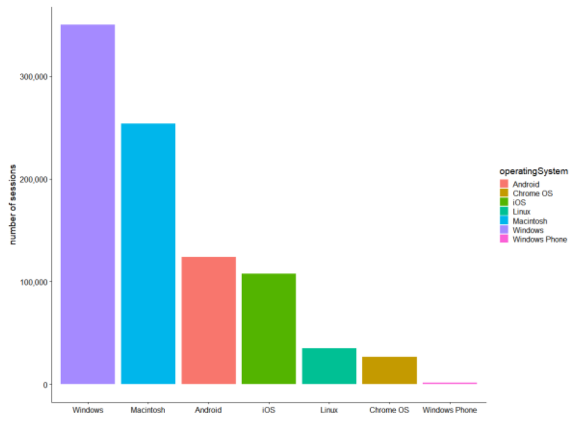


Figure 4.5 Number of sessions by operating system

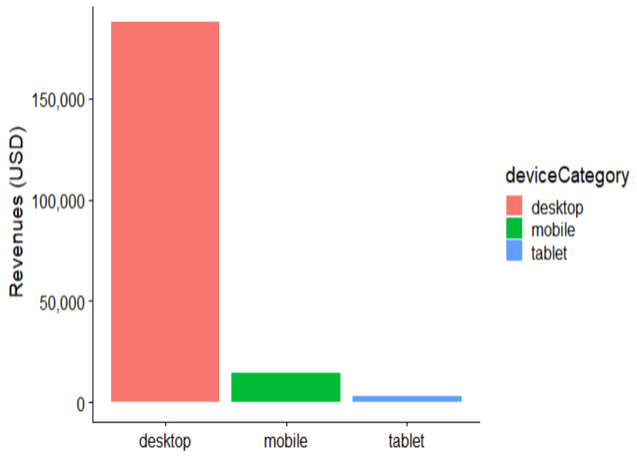


Figure 4.4 Revenue by device category

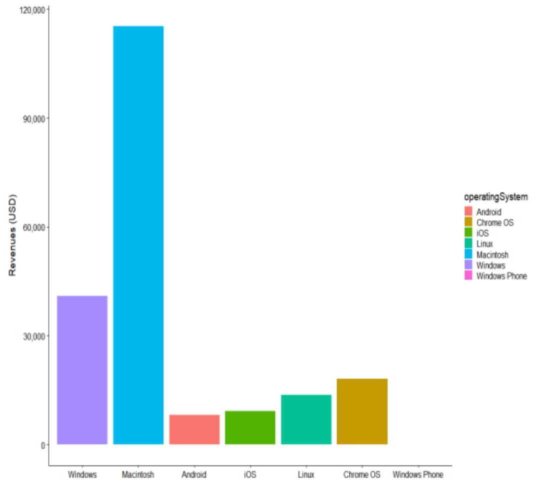


Figure 4.6 Revenue by operating system

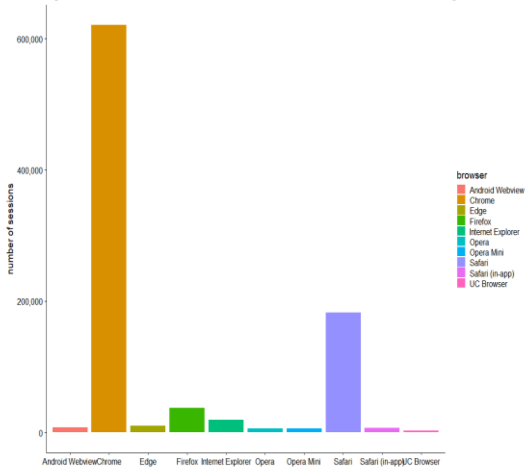


Figure 4.7 Number of sessions by browser

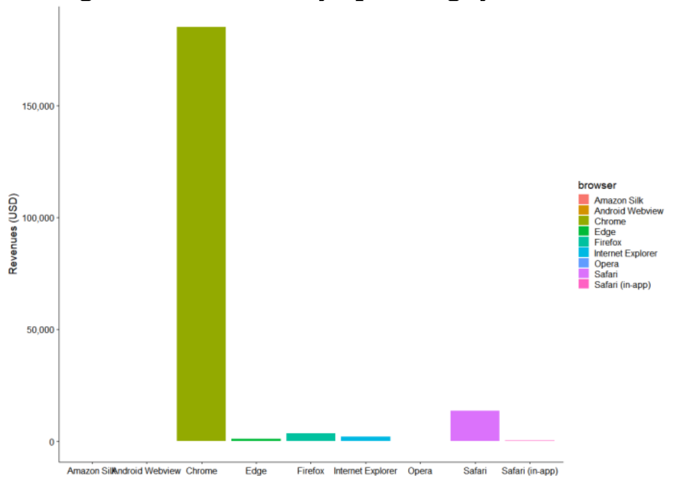


Figure 4.8 Revenue by browser

Appendix 5: Visits, Hits, and Page Views

The following figure shows the trend of daily visit in a year, from August 2016 to August 2017. The trend is roughly smoothy, except the date from October 2016 to January 2017, there is a obvious convex curve. At the end of 2016, between November 2016 to December 2016, the daily visits increases significantly. Thus, we believe there were more people visits the product at the end of 2016.

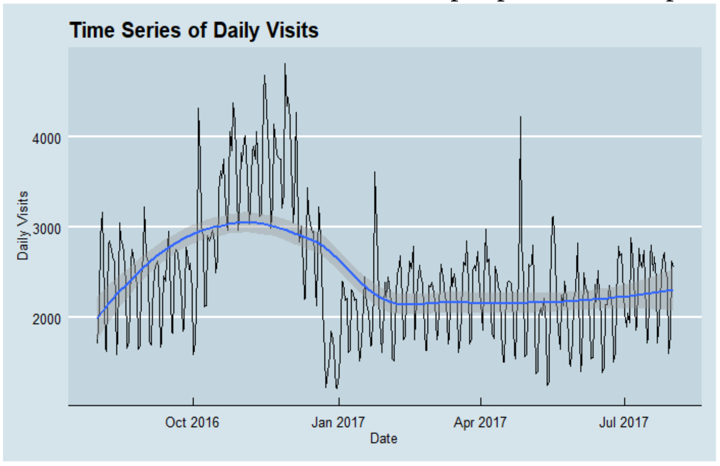


Figure 5.1 Time Series of Daily Visits

The following figure shows the trend of daily hits in a year, from August 2016 to August 2017. The trend decreases significantly from 2016 to 2017. Therefore, we believe there are less and less customers that can be attracted.

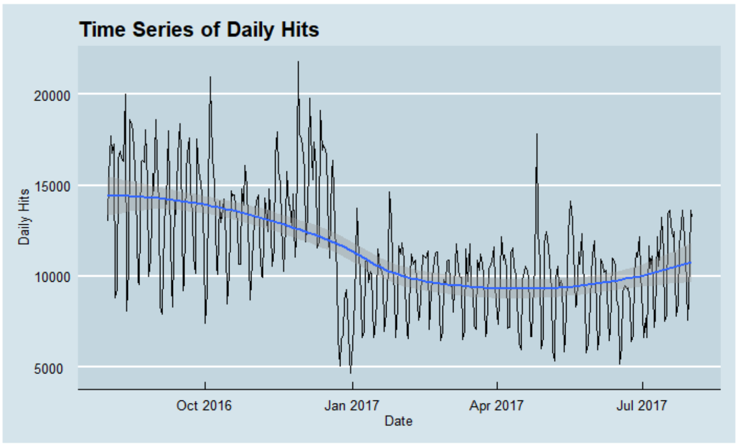


Figure 5.2 Time Series of Daily Hits

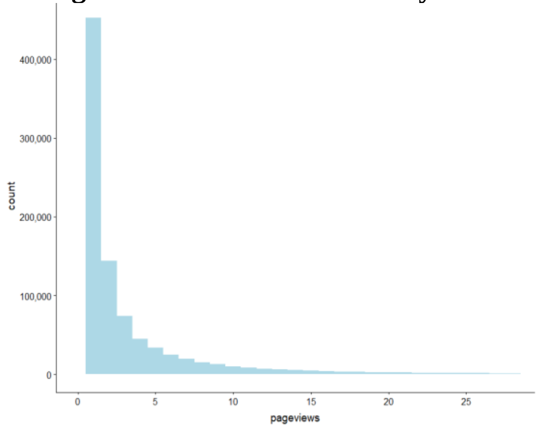


Figure 5.3 Histogram of Page Views

The above figure shows that the pageviews of customers. The right tail distribution indicates that most people view the product one to ten times.

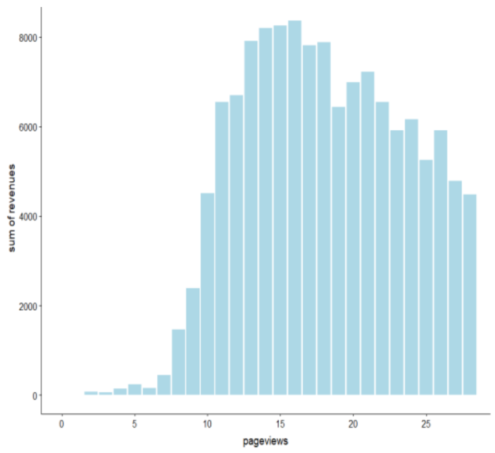


Figure 5.4 Histogram of pageviews with sum of revenue

The above figure shows the relationship between page views and revenue transaction. The histograms shows that people who view the page more than 10 times highly likely to make the transaction. If we only count pageviews with any transaction revenue, it leads to the same distribution (See Figure 5.5).

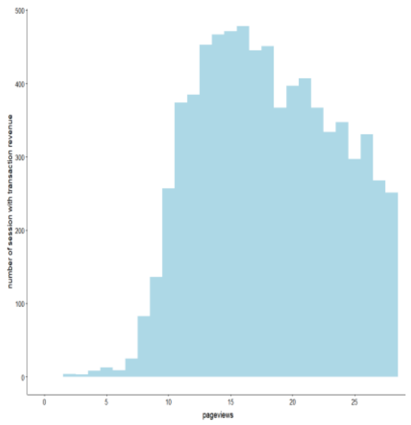


Figure 5.5 Histogram of pageviews with nonzero transaction revenue

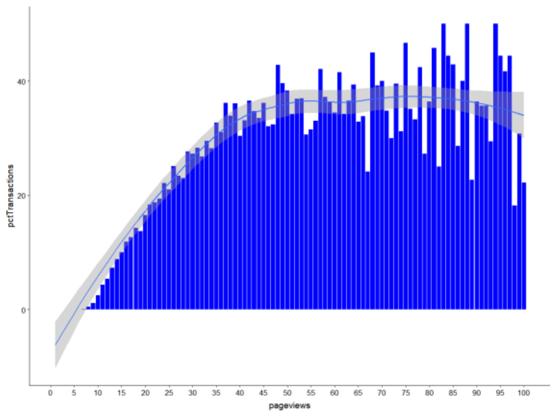
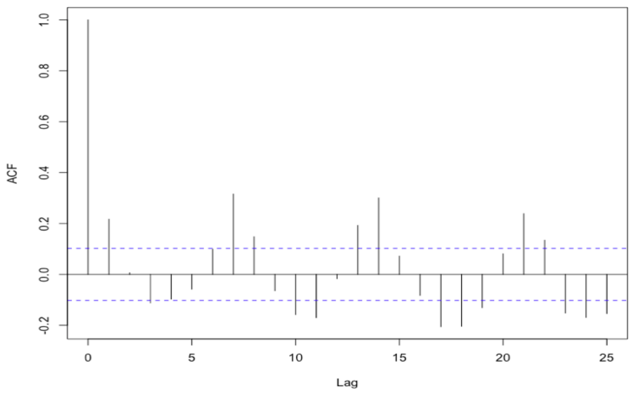


Figure 5.6 Transaction Percent for each category of Page Views

Appendix 6: Time Series Plots



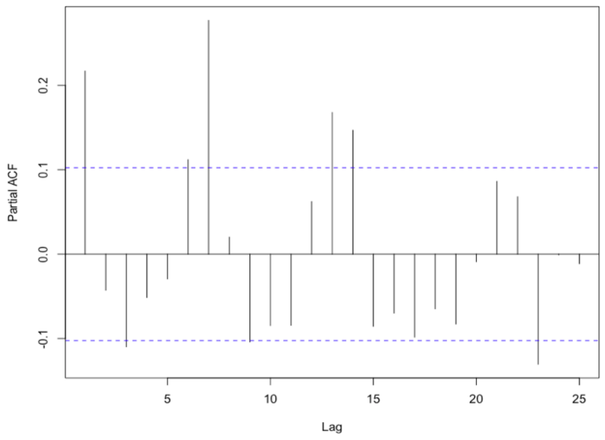


Figure 6.1 ACF Plot

Figure 6.2 PACF Plot

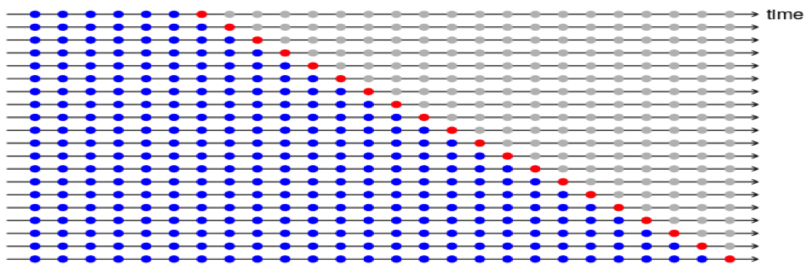
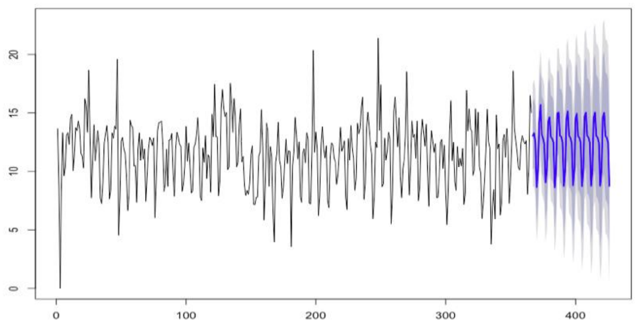


Figure 6.3 Cross Validation Illustration Diagram



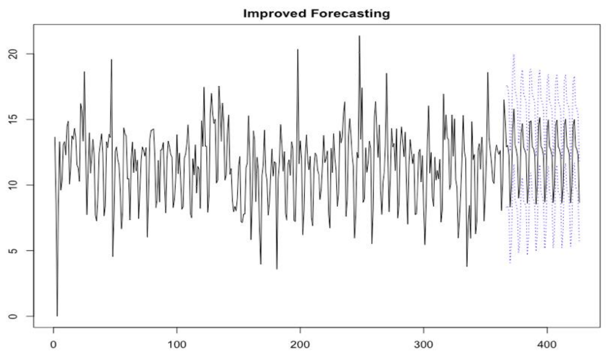


Figure 6.4 Predictions with 95% Confidence Interval

Appendix 7: Classification Models Results

Model		No Revenue	Transaction Made
Logistic Regression	No Revenue	219,477	1,093
	Transaction Made	3,558	1,786
Random Forests	No Revenue	218,201	128
	Transaction Made	4,834	2,751
XGBoost Classification	No Revenue	218,386	141
	Transaction Made	4,649	2,738

Table 7.1: *the table serves confusion matrices for logistic regression, random forests and XGBoost classification models. Random forests has correctly predicted revenue transaction made the most, and XGBoost classification model has very close result as random forests. Although logistic regression model has predicted correctly no revenue made the most, the amount that correctly predicted of revenue transaction made for logistic regression model is significantly less than other two models.*

Model	Accuracy	Precision	Recall	F_1 Score
Logistic Regression	97.94%	33.42%	62.03%	43.43%
Random Forests	97.80%	36.26%	95.55%	52.58%
XGBoost Classification	97.88%	37.07%	95.10%	53.34%

Table 7.2: *the above table serves the values of performance measurement for logistic regression, random forests and XGBoost classification models. Overall, logistic regression model has the worst results on precision, recall and F_1 score compared to random forests and XGBoost classification models. Also, random forests and XGBoost classification have very close results.*

Code:

~~~
library
```{r}
library(data.table)
library(jsonlite)
library(readr)
library(dplyr)
library(tidyr)
library(magrittr)
library(lubridate)
library(purrr)
library(ggplot2)
library(gridExtra)
library(countrycode)
library(highcharter)
library(ggExtra)
library(skimr)
library(cowplot)
library(ggthemes)
library(scales)
```

##############################Read Data##############################

```{r}
gtrain <- read_csv("train.min.csv")
gtest <- read_csv("test.min.csv")
```

Data Preprocessing
convert date column from numeric to data class
```{r}
gtrain$date <- as.Date(as.character(gtrain$date), format='%Y%m%d')

# convert visitStartTime to POSIXct
gtrain$visitStartTime <- as_datetime(gtrain$visitStartTime)
```

str() and summary()
```{r}
summary(gtrain)
str(gtrain)
```



Missing Values
```{r}
options(repr.plot.height=4)
NAcol <- which(colSums(is.na(gtrain)) > 0)
NAcount <- sort(colSums(sapply(gtrain[NAcol], is.na)), decreasing = TRUE)
NADF <- data.frame(variable=names(NAcount), missing=NAcount)
NADF$PctMissing <- round(((NADF$missing/nrow(gtrain))*100),1)
NADF %>%
  ggplot(aes(x=reorder(variable, PctMissing), y=PctMissing)) +
~~~

```
  geom_bar(stat='identity', fill='lightblue') + coord_flip(y=c(0,120)) +
  labs(x="", y="Percent of missing value") +
  geom_text(aes(label=paste0(NADF$PctMissing, "%"), hjust=-0.1))
```

############################Data Exploration#####################

```{r}
time_range <- range(gtrain$date)
print(time_range)
```

transaction revenue

```{r}
rev_range <- round(range(gtrain$transactionRevenue, na.rm=TRUE), 2)
print(rev_range)
```

```{r}
summary(gtrain$transactionRevenue)
```

```{r}
gtrain %>%
 ggplot(aes(x=transactionRevenue, y=..density..)) +
 geom_histogram(fill='lightblue', na.rm=TRUE, bins=40) +
 geom_density(aes(x=transactionRevenue), fill='red', color='red', alpha=0.2, na.rm=TRUE) +
 labs(
   title = 'Distribution of transaction revenue',
   x = 'Natural log of transaction revenue'
 )
```

```{r}
revenue <- gtrain$transactionRevenue
revenue[is.na(revenue)] <- 0
summary(revenue)
```

```{r}
date_wise <- gtrain  %>%
group_by(date)  %>%
summarise(daily_visits = sum(visits, na.rm = TRUE),
daily_hits = sum(hits, na.rm = TRUE),
daily_pageviews = sum(pageviews, na.rm = TRUE),
daily_bounces = sum(bounces, na.rm = TRUE),
daily_newVisits = sum(newVisits, na.rm = TRUE),
daily_transactionRevenue = sum(transactionRevenue, na.rm =TRUE)
        )
```

```{r}
ggplot(date_wise,aes(date,daily_visits)) + geom_line() +
theme_economist(dkpanel=TRUE) +
```

~~~
labs(title = "Time Series of Daily Visits",
   x = "Date",
   y = "Daily Visits") +
geom_smooth() -> p1
```

```{r}
ggplot(date_wise,aes(date,daily_hits)) + geom_line() +
theme_economist(dkpanel=TRUE) +
labs(title = "Time Series of Daily Hits",
   x = "Date",
   y = "Daily Hits") +
geom_smooth() -> p2
```

```{r}
ggplot(date_wise,aes(date,daily_newVisits)) + geom_line() +
theme_economist(dkpanel=TRUE) +
labs(title = "Time Series Daily new Visits",
   x = "Date",
   y = "Daily new Visits") +
geom_smooth() -> p3
```

```{r}
ggplot(date_wise,aes(date,daily_bounces)) + geom_line() +
theme_economist() +
labs(title = "Daily Bounces Trend",
   x = "Date",
   y = "Daily Bounces") +
geom_smooth() -> p4
```

```{r}
plot_grid(p1,p2,p3,p4, ncol = 2)
```

Bounce Rate
```{r}
date_wise$daily_bouncerate <- (date_wise$daily_bounces / date_wise$daily_visits) * 100
ggplot(date_wise,aes(date,daily_bouncerate)) + geom_line() +
theme_economist(dkpanel=TRUE)+
labs(title = "Daily Bounce Rate Trend",
    subtitle = "Bounce Rate = Bounces / Visits ",
   x = "Date",
   y = "Daily Bounce Rate in %") +
geom_smooth()
```

```{r}
ggplot(date_wise,aes(date,daily_transactionRevenue)) + geom_line() +
theme_economist(dkpanel=TRUE) +
labs(title = "Time Series of Daily Transaction Revenue",
    #subtitle = "Bounce Rate = Bounces / Visits ",
   x = "Date",
~~~

```
   y = "Daily Transaction Revenue") +
geom_smooth() -> p1
#plot_grid(p1)
```

```{r}
#Note: I have not included reordering of x in this function. First of all, I don't want to reorder the workday and month plots.
#Second: Manual reordering gave me the opportunity to order the Revenues in the same order as first plot (descreasing sessions). See for instance Channel Grouping.
#The adjusted functions to display plots with flipped x and y (see section 2.4.2 the source/medium dimension) includes reordering.

plotSessions <- function(dataframe, factorVariable, topN=10) {
   var_col <- enquo(factorVariable)
   dataframe %>% count(!!var_col) %>% top_n(topN, wt=n) %>%
   ggplot(aes_(x=var_col, y=~n, fill=var_col)) +
   geom_bar(stat='identity')+
   scale_y_continuous(labels=comma)+
   labs(x="", y="number of sessions")
   #+theme(legend.position="none")
   }

#also creating a function to plot transactionRevenue for a factorvariable
plotRevenue <- function(dataframe, factorVariable, topN=10) {
   var_col <- enquo(factorVariable)
   dataframe %>% group_by(!!var_col) %>% summarize(rev=sum(transactionRevenue)) %>% filter(rev>0) %>% top_n(topN, wt=rev) %>% ungroup() %>%
   ggplot(aes_(x=var_col, y=~rev, fill=var_col)) +
   geom_bar(stat='identity')+
   scale_y_continuous(labels=comma)+
   labs(x="", y="Revenues (USD)")
   #+theme(legend.position="none")
   }
```

```{r}
gtrain$transactionRevenue[is.na(gtrain$transactionRevenue)] <- 0
```

```{r}
gtrain$weekday <- wday(gtrain$date, label=TRUE)
#date_wise$weekday <- wday(date_wise$date, label=TRUE)
#gtest$weekday <- wday(test$date, label=TRUE)
str(gtrain$weekday)
```

```{r}
options(repr.plot.height=4)
w1 <- plotSessions(gtrain, weekday)
w2 <- plotRevenue(gtrain, weekday)
grid.arrange(w1, w2)
```

```{r}
```

~~~
options(repr.plot.height=4)

gtrain$month <- month(gtrain$date, label=TRUE)
#test$month <- month(test$date, label=TRUE)


m1 <- plotSessions(gtrain, month, 12)
m2 <- plotRevenue(gtrain, month, 12)
grid.arrange(m1, m2)
```

```{r}
options(repr.plot.height=6)

#adding reordering of x manually
sessionOrder <- gtrain %>% count(channelGrouping) %>% top_n(10, wt=n) %>% arrange(desc(n))
sessionOrder <- sessionOrder$channelGrouping

c1 <- plotSessions(gtrain, channelGrouping) + scale_x_discrete(limits=sessionOrder)
c2 <- plotRevenue(gtrain, channelGrouping) + scale_x_discrete(limits=sessionOrder)
grid.arrange(c1,c2)
```

```{r}
options(repr.plot.height=5)
d1 <- plotSessions(gtrain, deviceCategory)
d2 <- plotRevenue(gtrain, deviceCategory)
grid.arrange(d1,d2)
```

```{r}
options(repr.plot.height=5)

sessionOrder <- gtrain %>% filter(operatingSystem != "(not set)") %>% count(operatingSystem) %>%
top_n(7, wt=n) %>% arrange(desc(n))
sessionOrder <- sessionOrder$operatingSystem

o1 <- plotSessions(gtrain %>% filter(operatingSystem != "(not set)"), operatingSystem, 7) +
scale_x_discrete(limits=sessionOrder)
o2 <- plotRevenue(gtrain, operatingSystem) + scale_x_discrete(limits=sessionOrder)
#grid.arrange(o1, o2)
```
```{r}
options(repr.plot.height=5)

d1 <- plotSessions(gtrain, browser)
d2 <- plotRevenue(gtrain, browser)
d1
#grid.arrange(d1, d2)
```

```{r}
options(repr.plot.height=4)
#sessions with more than 28 pageviews all have frequencies of less than 1,000. Since these are hardly visible, I
am excluding them.
~~~

~~~~
#excluding 100 pageview NAs

p1 <- gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=28) %>%
ggplot(aes(x=pageviews)) +
   geom_histogram(fill='lightblue', binwidth=1) +
   scale_y_continuous(breaks=seq(0, 500000, by=100000), label=comma) +
   scale_x_continuous(breaks=seq(0, 28, by=5)) +
   coord_cartesian(x=c(0,28))

p2 <- gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=28) %>% group_by(pageviews) %>%
   ggplot(aes(x=pageviews, y=transactionRevenue)) +
   geom_bar(stat='summary', fun.y = "sum", fill='lightblue') +
   scale_x_continuous(breaks=seq(0, 28, by=5)) +
   coord_cartesian(x=c(0,28)) + labs(y="sum of revenues")
grid.arrange(p1, p2)
```
```{r}
gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=28 & transactionRevenue>0) %>%
ggplot(aes(x=pageviews)) +
   geom_histogram(fill='light blue', binwidth=1) +
   scale_x_continuous(breaks=seq(0, 28, by=5)) +
   coord_cartesian(x=c(0,28)) +
   labs(y='number of session with transaction revenue')
```



```{r}
p1 <- gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=28 & transactionRevenue>0) %>%
group_by(pageviews) %>%
   ggplot(aes(x=pageviews, y=transactionRevenue)) +
   geom_bar(stat='summary', fun.y = "mean", fill='blue') +
   scale_x_continuous(breaks=seq(0, 28, by=5)) +
   coord_cartesian(x=c(0,28)) + labs(y="mean of revenues") +
   geom_label(stat = "count", aes(label = ..count..), y=0, size=2)
p2 <- gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=28 & transactionRevenue>0) %>%
group_by(pageviews) %>%
   ggplot(aes(x=pageviews, y=transactionRevenue)) +
   geom_bar(stat='summary', fun.y = "median", fill='blue') +
   scale_x_continuous(breaks=seq(0, 28, by=5)) +
   coord_cartesian(x=c(0,28)) + labs(y="median of revenues") +
   geom_label(stat = "count", aes(label = ..count..), y=0, size=2)
grid.arrange(p1, p2)
```


```{r}
gtrain$transaction <- ifelse(gtrain$transactionRevenue > 0, 1, 0)

gtrain %>% filter(!is.na(gtrain$pageviews) & pageviews <=100) %>% group_by(pageviews) %>%
summarize('sessions'= n(), 'transactions'= sum(transaction),
'pctTransactions'=round(x=((transactions/sessions)*100),digits=1)) %>%
   ggplot(aes(x=pageviews, y=pctTransactions)) +
   geom_bar(stat='identity', fill='blue') +
   scale_x_continuous(breaks=seq(0, 100, by=5)) +
   geom_smooth()
```
~~~~

~~~~
---
####################################title: Time Series ##############################
---

```{r}
library(ggplot2)
library(forecast)
library(plotly)
library(ggfortify)
library(tseries)
library(gridExtra)
library(readr)
library(MASS)
```

## Load the Data

```{r}
gtrain_new <- read_csv("train.min.csv")
```

## Compute Daily Revenue

```{r}
date_wise <- gtrain_new  %>%
group_by(date)  %>%
summarise(daily_transactionRevenue = sum(transactionRevenue, na.rm =TRUE))
```

## Model Preparation

```{r}
acf(date_wise$daily_transactionRevenue)
pacf(date_wise$daily_transactionRevenue)
```

Based on this plots, it could be observed that there is a seasonal pattern at lag 7, which makes sense since it
could be a weekly pattern.

```{r}
# Convert data to time series based on observation
TRDts <- ts(date_wise$daily_transactionRevenue, frequency = 7)
plot.ts(date_wise$daily_transactionRevenue, main = "Daily Transaction Revenue")
```

```{r}
# Note that the third observation is 0, which could not be used in this test. Thus, we drop it for the test.
ntD=1:length(TRDts)
bcTransD = boxcox(TRDts[-3]~ntD[-3])
lambdaD = bcTransD$x[which(bcTransD$y == max(bcTransD$y))]
lambdaD
```

```{r}
dataTrans=(TRDts)^(0.3)
~~~~

~~~~
ts.plot(dataTrans)
var(dataTrans)

# Reference: https://www.ime.usp.br/~abe/lista/pdfQWaCMboK68.pdf
```

## Model Selection

```{r}
# Select model based on AIC
auto.arima(dataTrans)
```

```{r}
fit1 <- arima(dataTrans, order=c(1,0,1), seasonal=list(order=c(2,1,0), period=7))
fit1
```

```{r}
# Check assumptions for residuals
# Test for independence of residuals
Box.test(residuals(fit1), type="Ljung")

plot(residuals(fit1))

# Histogram
hist(residuals(fit1),main = "Histogram")
# q-q plot
qqnorm(residuals(fit1))
qqline(residuals(fit1),col ="blue")
```

## Cross Validation

```{r}
# Cross validation test error RMSE
# We compare the RMSE obtained via time series cross-validation with the residual RMSE
data_new <- c(dataTrans)
fit_new <- function(x, h){
  forecast(arima(x, order=c(1,0,1), seasonal=list(order=c(2,1,0), period=7)), h=h)
}
e <- tsCV(data_new, fit_new, h=1)

# Cross Validation Error (Predicted Test Error)
sqrt(mean(e^2, na.rm=TRUE))

# Training Error
sqrt(mean(residuals(arima(data_new,   order=c(1,0,1),   seasonal=list(order=c(2,1,0),   period=7)),   h=1)^2,
na.rm=TRUE))
```

## Prediction

```{r}
# First attempt
~~~~

~~~
fit_new = arima(ts(dataTrans), order=c(1,0,1), seasonal=list(order=c(2,1,0), period=7))
fcast_new <- forecast(fit_new,h=60)
plot(fcast_new, main=" ")

#Improvement of Forecasting
dataNew <- NULL
dataUpper <- matrix(nrow = 60,ncol = 2)
dataUpper[,1] <- c(367:426)
dataLower <- matrix(nrow = 60,ncol = 2)
dataLower[,1] <- c(367:426)
dataNew <- as.numeric(dataTrans)
length(dataNew) <- 426

for(i in 1:60){
  fit_new = arima(ts(dataNew[(247+(i-1)):(366+(i-1))], frequency = 7), order=c(1,0,1),
seasonal=list(order=c(2,1,0), period=7))
  fcast_new <- forecast(fit_new, h=1)
  dataNew[366+i] <- fcast_new$mean
  dataUpper[i,2] <- fcast_new$upper[2]
  dataLower[i,2] <- fcast_new$lower[2]
}

dataCI <- matrix(NA, nrow = 426, ncol = 4)
dataCI[,1] <- c(1:426)
dataCI[,2] <- dataNew
dataCI[367:426,3] <- dataUpper[,2]
dataCI[367:426,4] <- dataLower[,2]
colnames(dataCI) <- c("date", "Reveune", "upper", "lower")

ts.plot(dataNew, xlab = "", ylab = "", main = "Improved Forecasting")
lines(dataUpper, col="blue", lty = 3)
lines(dataLower, col="blue", lty = 3)

# ggplot(data.frame(dataCI), aes(x = date, y = Reveune)) +
#   geom_line(color = "blue", size = 0.05) +
#   geom_ribbon(data=dataCI,aes(ymin=lower,ymax=upper),alpha=0.3) +
#   theme_minimal() +
#   ylim(low=0,high=20)
```

To get the prediction of revenue, we need to transform data into the original form.

```{r}
#(fcast_new$mean)^(10/3)
(dataNew[367:426])^(10/3)
```


---
############################title: Imbalanced Dataset Processing####################
---



```{r}
#gtrain <- read_csv("gtrain.csv")
revenue_convert<-0
~~~

~~~
gtrain<-data.frame(gtrain,revenue_convert)
gtrain$revenue_convert[gtrain$transactionRevenue >0]<-1

train_data_Y<-gtrain[gtrain$revenue_convert==1,]
train_data_N<-gtrain[gtrain$revenue_convert==0,]
sample_size_Y <- floor(0.75*nrow(train_data_Y))
sample_size_N <- floor(0.75*nrow(train_data_N))
set.seed(222)
train.ind_Y <- sample(seq_len(nrow(train_data_Y)), size = sample_size_Y)
train.ind_N <- sample(seq_len(nrow(train_data_N)), size = sample_size_N)
train_Y <- train_data_Y[train.ind_Y,]
train_N <- train_data_N[train.ind_N,]
train_sample_N<-sample_n(train_N,10*nrow(train_Y))
test_Y <-train_data_Y[-train.ind_Y,]
test_N<-train_data_N[-train.ind_N,]
test_sample_N<-sample_n(test_N,5*nrow(test_Y))
train_new<-rbind(train_Y,train_sample_N)
#test_new<-rbind(test_Y,test_sample_N)
test_new<-gtest

```
###########################Logistic Model#################
```{r}
train_new$channelGrouping<-as.factor(train_new$channelGrouping)
train_new$operatingSystem<-as.factor(train_new$operatingSystem)

test_new$channelGrouping<-as.factor(test_new$channelGrouping)
test_new$operatingSystem<-as.factor(test_new$operatingSystem)

glm.fit<-           glm(revenue_convert~.-X1-transactionRevenue-fullVisitorId-visitId-sessionId-country-
operatingSystem, data =train_new , family = binomial)
summary(glm.fit)
coef(glm.fit)
glm.probs <- predict(glm.fit,test_new[test_new$operatingSystem!='9',],type="response")
# test the predictions and the real data
glm.pred <- rep("0",nrow(test_new))
glm.pred[glm.probs>0.5] = "1"
conf<-table(glm.pred, test_new$revenue_convert)

log.acc<-(conf[1,1]+conf[2,2])/sum(conf[,])
log.acc
log.prec <- conf[2,2]/sum(conf[2,])
log.prec
# Recall (true pos/true pos + false neg)
log.rec <- conf[2,2]/sum(conf[,2])
log.rec
# F1 Score
log.f1 <- 2*(log.rec*log.prec)/(log.rec+log.prec)
log.f1

#################################XGBoost Model###########################
grid_default <- expand.grid(
  nrounds = 1000,
  max_depth = 10,
  eta = 0.3,
~~~

~~~
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)
xgbtree <- train(factor(revenue_convert) ~ .-X1-transactionRevenue-fullVisitorId-visitId-sessionId,
                 data = train_new,
                 tuneGrid = grid_default,
                 method = "xgbTree",
                 trControl = trainControl(method = "cv"))
xgb.pred <- predict(xgbtree, newdata = test_new)
conf = table(xgb.pred, test_new$revenue_convert)
xgb.acc<-(conf[1,1]+conf[2,2])/sum(conf[,])
xgb.acc
xgb.prec <- conf[2,2]/sum(conf[2,])
xgb.prec
# Recall (true pos/true pos + false neg)
xgb.rec <- conf[2,2]/sum(conf[,2])
xgb.rec
# F1 Score
xgb.f1 <- 2*(xgb.rec*xgb.prec)/(xgb.rec+xgb.prec)
xgb.f1
xgbtree

```
######################Random Forest Model#######################################
```{r}
library(randomForest)
set.seed(100)
memory.limit(100000)
#train_new<-read_csv("train_new.csv")
train_new$revenue_convert<-as.factor(train_new$revenue_convert)
test_new$revenue_convert<-as.factor(test_new$revenue_convert)
model_randomforest<-randomForest(revenue_convert~.-X1-transactionRevenue-fullVisitorId-visitId-
sessionId,data =train_new, mtry=5,importance=TRUE, type='classification',na.action = na.pass)
print(model_randomforest)

#test_new<-read_csv("test_new.csv")
pred <- predict(model_randomforest, test_new)

conf <- table(pred, test_new$revenue_convert,dnn=c("Prediction", "Actual"))
rf.acc<-(conf[1,1]+conf[2,2])/sum(conf[,])
rf.acc
```
########################Performance Measurement#################
```{r}
# Precision (true pos/true pos + false pos)
prec <- conf[2,2]/sum(conf[2,])
prec
# Recall (true pos/true pos + false neg)
rec <- conf[2,2]/sum(conf[,2])
rec
# F1 Score
f1 <- 2*(rec*prec)/(rec+prec)
f1
~~~

~~~
eval.sum <- data.frame(method = c("RF", "Logit", "Xgb"),
                    accuracy = c(rf.acc, log.acc, xgb.acc),
                    precision = c(prec, log.prec, xgb.prec),
                    recall = c(rec, log.rec, xgb.rec),
                    f1score = c(f1, log.f1, xgb.f1))
```
#######################Linear Mixer Model###########################
```{r}
tr <- read.csv("train_new.csv")
```

```{r}
tr <- tr[,-1]
tr <- tr[,-14]
summary(tr$transactionRevenue)
```

```{r}
m_lmm0 <- glmer(transactionRevenue ~ (1|fullVisitorId), data = tr)

bg_var <- summary(m_lmm0)$varcor$fullVisitorId[1]
resid_var <- attr(summary(m_lmm0)$varcor, "sc")^2

summary(m_lmm0)
```

```{r}
m_lmm1 <- update(m_lmm0, transactionRevenue ~ pageviews + (1|fullVisitorId))
m_lmm2 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + (1|fullVisitorId))
m_lmm3 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + visitNumber + (1|fullVisitorId))
m_lmm4 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + visitNumber + channelGrouping +
(1|fullVisitorId))
m_lmm5 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + visitNumber + channelGrouping +
browser + (1|fullVisitorId))
m_lmm6 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + visitNumber + channelGrouping +
browser + operatingSystem + (1|fullVisitorId))
m_lmm7 <- update(m_lmm0, transactionRevenue ~ pageviews + hits + visitNumber + channelGrouping +
browser + operatingSystem + country + (1|fullVisitorId))

anova(m_lmm0, m_lmm1, m_lmm2, m_lmm3, m_lmm4, m_lmm5, m_lmm6, m_lmm7)

#
```

```{r}
pred_lmm <- predict(m_lmm7)
RMSE(tr$transactionRevenue, pred_lmm)
```



```{r}
te <- read_csv("test_new.csv")
```
~~~

```{r}
RMSE(te$transactionRevenue, pred_lmm)
```

```
######################XGBoost Regression#########################
paid <- train_new[train_new$revenue_convert == 1,]
paidtest <- test_new[test_new$revenue_convert == 1,]

xgbmodel <- model.matrix(transactionRevenue ~ .-X1-transactionRevenue-fullVisitorId-visitId-sessionId, data
= paid)
xgbtest <- model.matrix(transactionRevenue ~ .-X1-transactionRevenue-fullVisitorId-visitId-sessionId, data =
paidtest)

xgb.final <- xgboost(data = xgbmodel,
            label = paid$transactionRevenue,
            eta = 0.01,
            max_depth = 4,
            nrounds = 1000,
            eval_metric = "rmse",
            objective = "reg:linear"
)
pred.xgb <- predict(xgb.final, newdata = xgbtest)
sqrt(mean((paidtest$transactionRevenue - pred.xgb)^2))
```