Revenue Prediction:

How much customers will spend?

Chenxin Guo cg633

Jianqiu Lu jl3846

Yanxin Xu yx533

Qiyue Zou qz298

December 15, 2018

Abstract.

Analyzing shopping behavior of customers can help the service providers offer differentiated information to customers and improve their shopping experience. In this project, we use a dataset which contains about 1 million recorded visiting entries of an online merchandise store to study the purchasing behaviors of visitors. We choose the best classification and regression pair models based on the mean square error and ROC curve results. We finally build a combined hybrid model that can precisely predict whether the users will buy the products and how they much will spend on this online store.

1 Introduction

As eShopping becoming a more popular way to get what people need these days, online sellers found that the 80/20 rule true also on their business: only a small percentage of customers produce most of the revenue. The dataset provides comprehensive visiting information of the online Google Merchandise Store for more than $900,000$ recorded visits, and we are challenged to analyze it and predict revenue per visit.

In general, to predict the revenue, our idea is to divide the problem into three steps:

1. Construct some classification models to divide the visits with and without revenue

2. Build several regression models, training only on the data with (or might with) revenue, to predict how much revenue a single visit is probably to produce behaviors.

3. Combine the classification and regression models to get the final model.

Model outputs y are scored by Mean Squared Error on the natural log of the actual summed revenue ($0.000001) value plus one:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \log(\text{transactionRevenue}_i + 1))^2$$

In this report we first explore the data set, drop useless information in the data and explore the correlation between the features and the label. Then for classification we use Logistic regression and two tree based methods (Random Forest and XGBoost) to predict whether the customer spent money on the website. Cross validation are used for selecting hyper parameters for each model and we compare their performance by Mean Square Error.

For regression we try linear regression (including feature selection), tree based methods (bagging and boosting) and neural network solution. We hope to get a much smaller mean square error compared to the variance of the data.

Then we use the best models from both classification (XGboost) and regression (ridge regression) to build a combined model, and improve the result by using predicted data to train the regression model. The new hybrid model has a MSE of 1.11, which shows a good performance. We formally write down the algorithm we choose to use in the fifth part.

2 Exploratory Data Analysis

2.1 Data Cleaning

The original dataset is provided by Google through the Kaggle website (The analyze is based on data of the first version, which is not available anymore).[1] It contains 903653 observations with 12 columns, with 4 of them are in json format. After parsing these columns, we get a dataset with 54 features. 19 of them have constant values, which give no information for our prediction, so we just remove them.

As for other features, we can observe many missing values in the data set (Fig. 1). In the plot, transactionRevenue, newVisits, IsTrueDirect and bounces are exceptions: missing means 0 revenue. For some other features like "keyword" and "adContent", missing can either represents data missing or no such data. Indicated by the figure we know that most of the visits (almost 99%) brought no revenue. For the features with high proportion of missing or unclear meaning, we drop them from our data set. For those with low missing proportion, we filled the NAs by a specific value – we can always do that because all features in this situation are categorical variables.

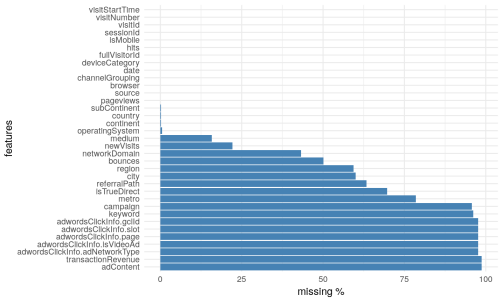


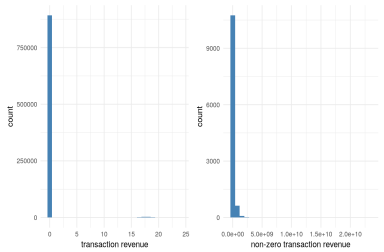
Figure 1: Missing Value Proportion

There is a special feature in our data, which is time and date. The data is stored in POSIX format and we transform them into time and date separately using the local time zone of the customers. We further transform the time into different levels: Morning, Noon, Afternoon, Evening, Night and Late Night, with a prior information that people may have different behavior in different time slots.

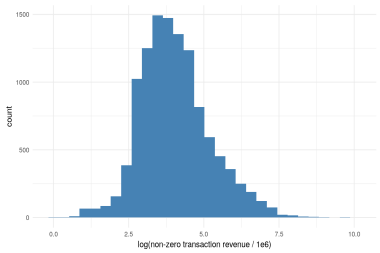
2.2 Dataset Exploring

We are interested in our target variable. For modelling log-transformed target will be used. Only 1.27% of all visits have non-zero revenue(see Fig. 2(a)). The natural log of non-zero revenue seems to have a log-normal distribution with positive skewness (Fig. 2(b)). User from Social and Affiliates channels did never make profit. Referral is the most profitable channel. And most of the revenues are generated by new users rather than existed users(Fig. 2(d)). Though we have the information of the visit date, there seems

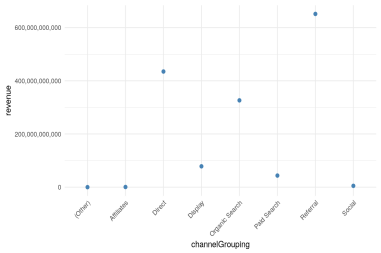
to have no consistent relationship between date and revenue (Fig. 2(e)). So we will no longer reverse the date information while training. However, time, especially the hour, will be remained since we believe it has a high correlation with the revenue.



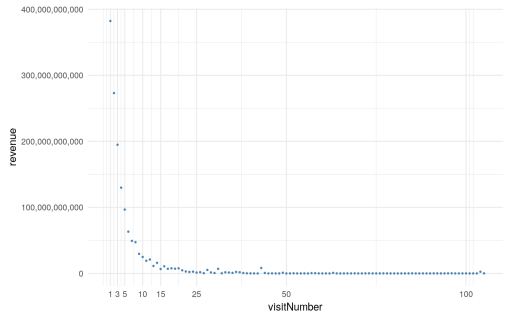
(a) Revenue Count



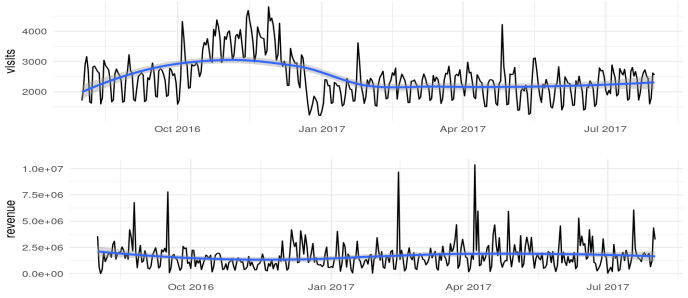
(b) Non-0 Revenue Count



(c) Revenue vs. Channel

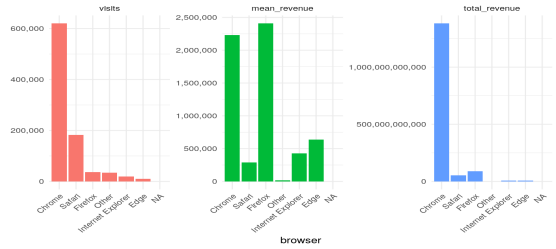


(d) Revenue vs. Visits

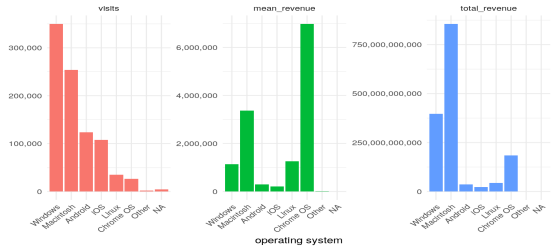


(e) Revenue vs. Date

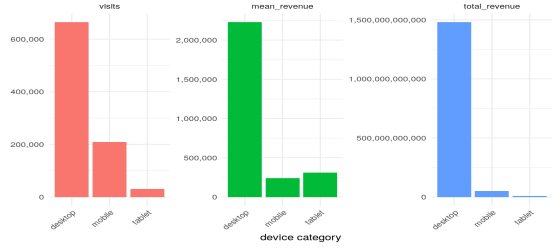
Figure 2: Revenue Related Plots 1



(a) browser



(b) system



(c) device

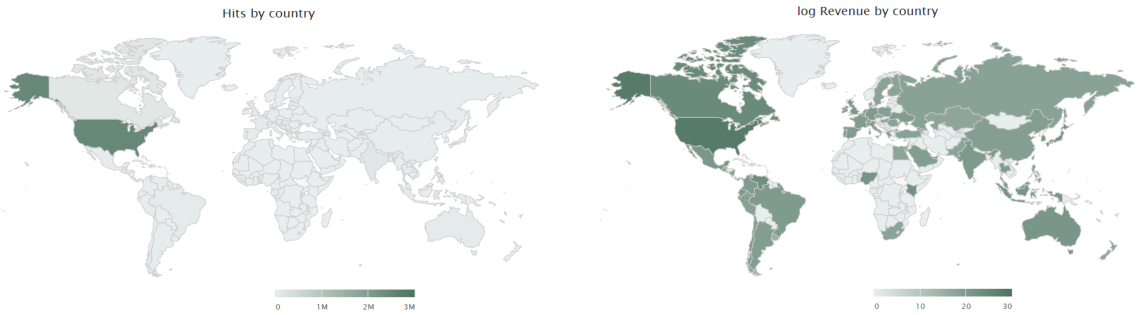
Figure 3: Revenue, Devices, Browsers and others

We also have huge amount of information about the users' browsers and devices. generally there seems to have strong relationship between the mean revenue and them (Fig. 3): visitors using chrome (which is

the most popular browser) and firefox contributes to higher mean revenue, and visitors using Mac have more power on purchasing compared to those using Windows. What's more, most of the revenue came from desktop devices, though mobile device occupied a quarter of all visits.

We can divide all visitors by their country information(Fig. 4). It shows that revenue came from all continents, with US having the highest total amount. Also, most of the hits came from US, showing the main market of Google Store is actually US rather than the whole world. Therefore the information of country might be useful in prediction revenue, but we are not sure about it.

At last we discuss the correlations between revenues and our features. Most of the features are categorical so we reencode them as one hot encoder. ID columns are dropped and we get the histogram 5(a). The correlation coefficients are concentrated around zero, but there are two exceptions. They are listed in Table 1.



(a) Country vs Hit

(b) Country vs Revenue

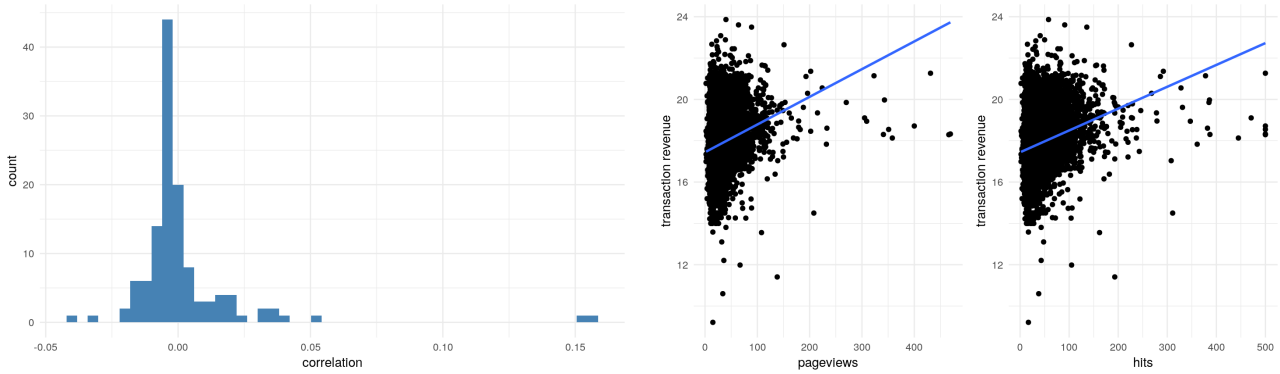
Figure 4: Country, Revenue and hits

Table 1: Features having high correlation with Revenue

Feature	rho
pageviews	0.1555894
hits	0.1543326

The relation ship of log revenue with these features are visualized in Fig. 5(b). We can find weak positive relationship. This phenomenon gave evidence that these features could play important role in a statistical model.

From the analysis above, we can know that the data we have provides meaningful information about the transaction revenue, showing it is possible to predict the revenue using the data we have. Therefore we start building our statistical model.



(a) All features (b) Pageviews and Hits

Figure 5: Correlations between Revenue and Features

3 Classification

In this part, we are working on separating the data into two disjoint part – one part having transaction revenue and the other not. In the classification step we replace the transaction revenue column with a binary variable showing whether revenue is made. We use 80% of all data on training and the remaining 20% data for testing. Since the data is highly unbalanced between data with and without revenue (almost 60:1), in all the following approaches we duplicated the data without revenue in training 60 times to make the data balance.

3.1 Approach 1: Logistic Regression

After doing all these pre-processing work, we start the logistic regression fitting with a maximum iteration of 100. In the prediction model we have some features with low p-value, part of which are shown in Table 2.

Table 2: Features having low p value in Logistic Regression

Feature	Estimate	p value
(Intercept)	-7.35	<1e-10
operatingSystemChrome OS	1.73	<1e-15
operatingSystemLinux	1.65	<1e-15
operatingSystemMacintosh	1.93	<1e-15
operatingSystemWindows	1.38	<1e-15
pageviews	0.50	<1e-15
isTrueDirect	0.31	<1e-15
source-youtube	-3.77	<1e-15

We can find some interesting phenomenon in the table, like people are more like to make deals with their computer rather than mobile devices, and from where people come to visit the site can have great

influence on their behavior!

Figure 6 shows the ROC of our LR model. By observing the ROC curve, we choose the threshold equal to 0.5, which has the lowest error rate at the ROC, and calculate the absolute error in the validation set. The overall validation error is 5.6%, with a 3.7% False Negative Rate and 5.6% False Positive Rate. Since a high classification error rate will strongly influence the regression accuracy, we hope to build a more precise model.

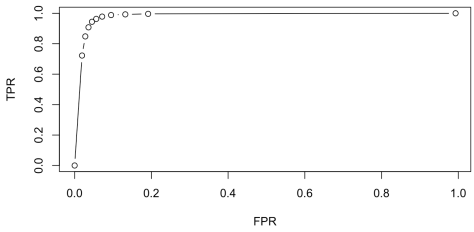


Figure 6: ROC of LR Model

3.2 Approach 2: Tree Based Methods

3.2.1 Random Forest

Compared to other methods, random forests algorithm is robust against overfitting problem. We still wanted to explore the best value of the number of sub-features(ntry). After using cross validation, we would like to choose 12 as the number sub-features. The final result is shown in Table 3.

Table 3: Testing Result using RF model, ntry=12

Testing	Real		Training	Real	
Prediction	False	True	Prediction	False	True
False	180681	1272	False	727301	0
True	1191	1120	True	300	576511

The testing error of random forest is 1.34% and training error is even as low as 0.02%! However, the FNR of the predictions on test set does not perform very well, which is higher than 1/2. In this case, we believe the error of random forest is mainly caused by high bias so that may not be the model we want to use in classification.

3.2.2 XGBoost

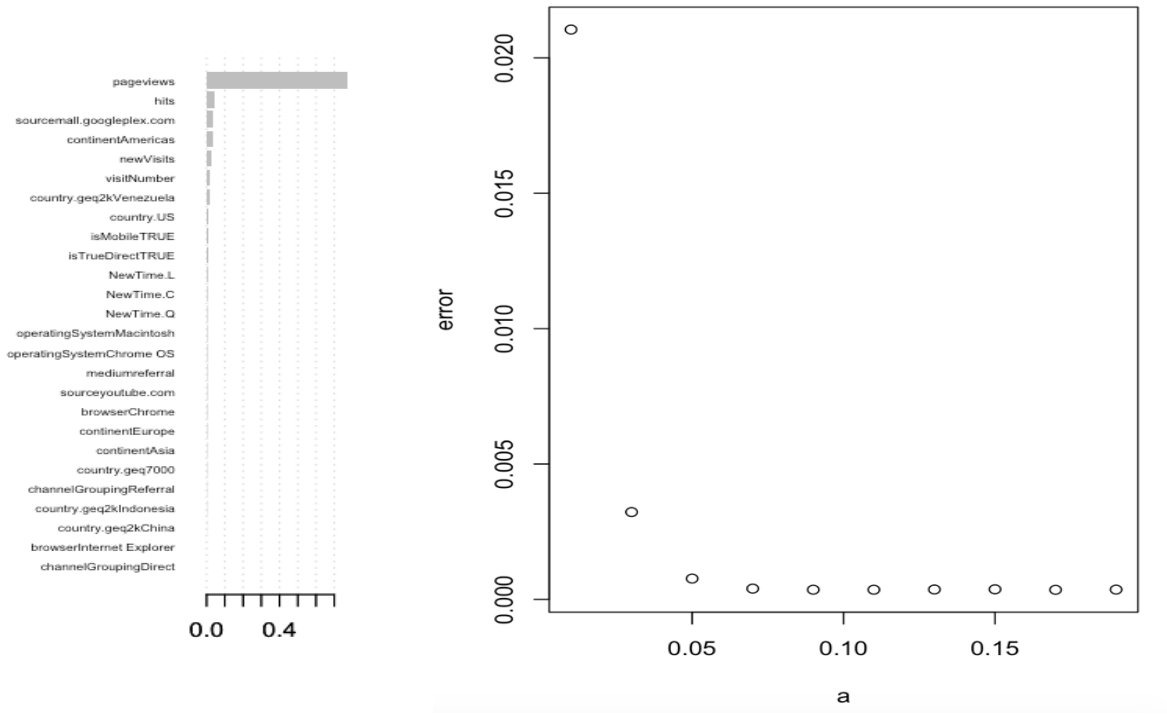
Among tree based methods, bagging can help reduce variance and boosting can help reduce bias. As we have plenty of training data, variance is not as important as bias, so we prefer to use boosting. XGBoost is a popular boosting method with high training speed, and we will apply this model to our data.

Firstly, XGBoost manages only numeric vectors, so we transform the categorical variables into dummy ones. Besides, there are some important tune parameters in building the XGBoost model, for example,

ETA control the learning rate in gradient descend and a lower *ETA* is more prone to find the global optimum of the model; max_depth means the depth of the tree. Therefore, we performed the Cross Validation to find the best learning rate (*ETA*) ranging from 0.01 to 0.2 with 100 intervals. From the Figure 7(b), we can see that the validation error decreases with the increasing of learning rate but after step 7, the error increase a little bit. Therefore, we choose *ETA* equals 0.07. The result shows that the lowest test error rate is 2.87%, and the detail is shown in Table 4. It is surprising that compared to Random Forest, XGBoost is less overfitting because of the lower FNR and have similar TPR and TNR. Though the error rate is higher, we believe this is a better classification result.

Table 4: Testing Result using XGBoost, nround=1000

Testing	Real		Training	Real	
Prediction	False	True	Prediction	False	True
False	176696	110	False	707909	366
True	5176	2282	True	19692	576145



(a) Feature Importance

(b) Validation Error

Figure 7: Training Result of XGBoost

In addition, Fig. 7(a) shows the feature importance. According to the plot, we can see that some features greatly affect the prediction of revenue, for example: hits, pageview, visit-time and the hour time. This importance is almost the same as the previous picture introduced in the data exploration part.

3.3 Comparison

Comparing all three methods we used, we found that the tree based models(Random Forest and XGBoost) performed better than the linear model (Logistic Regression). Though Random Forest have even lower error rate than XGBoost, it misclassified too many positive samples in the test set, and therefore is not totally better than XGBoost. We will keep these two methods and compare their final performance in Section 5.

4 Regression

After the classification, we then extract the rows whose profit is bigger than zero and fit them with regression supervised machine learning. The data with profit now only have 11843 rows. Since the profit of google store is 80/20, we know that only a small number of customers spent large amounts of money. It is therefore better to take log of those profits.

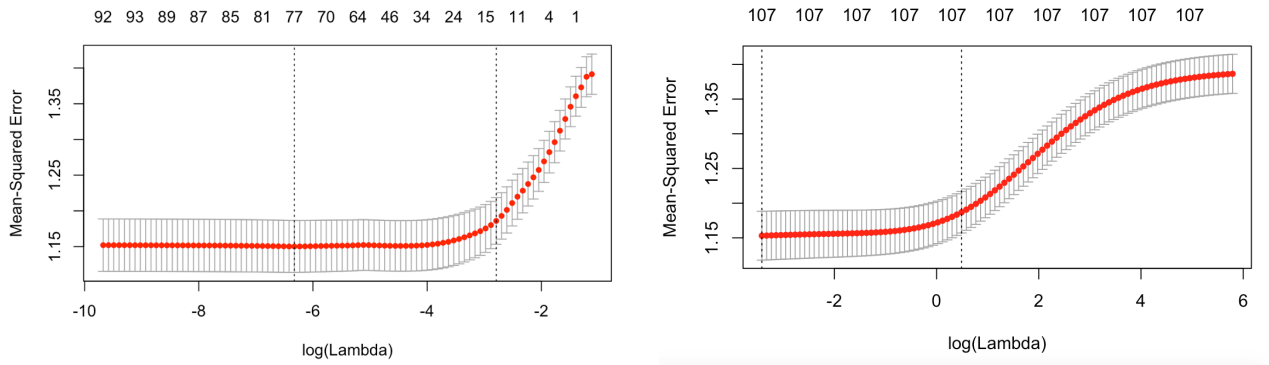
We use mean square error to evaluate the performance of each model. The variance of the data with profits is 1.445991. Any model with a lower validation error is acceptable, and model with higher error rate is not better than the average prediction.

4.1 Approach 1: Linear Regression & Features Selection

Firstly, we do a linearly regression model to have a general understanding of our data. The testing MSE is around 1.735, which is even higher than the real variance. Besides, among 53 features, only some of them are significant. Therefore, we wanted to do the features selection and find the useful and interpretable ones to rebuild a model. Considering the computation expense, we chose to use forward stepwise selection rather than best subset selection. The result shows that apart from the features like *hit*, *pageviews*,*new-visit*, which are also important in classification part, there still have some significant features like *isMobile* and *browser*.

Besides the hard selection method, we also used soft selection method: regularization. There are two types of regularization to achieve smaller flexibility: lasso and ridge. Appropriate choose of the tune parameter, λ, can manage the bias-variance trade off very well.

In order to find the optimal λ, we do the cross validation of ridge regression and plot the CV MSE



(a) CV for Ridge Regression

(b) CV for Lasso Regression

Figure 8: Cross Validation on Linear Regression

vs $log(\lambda)$. From Figure 8(a), we know that when λ equals 0.001789537 the corresponding CV error is the lowest and equals 1.147567. Besides, we also do the similar cross validation for lasso regression and the CV error achieve its lowest value (> 1.15) when lambda is 0.0326147.

4.2 Approach 2: Tree Based Methods

4.2.1 Random Forest

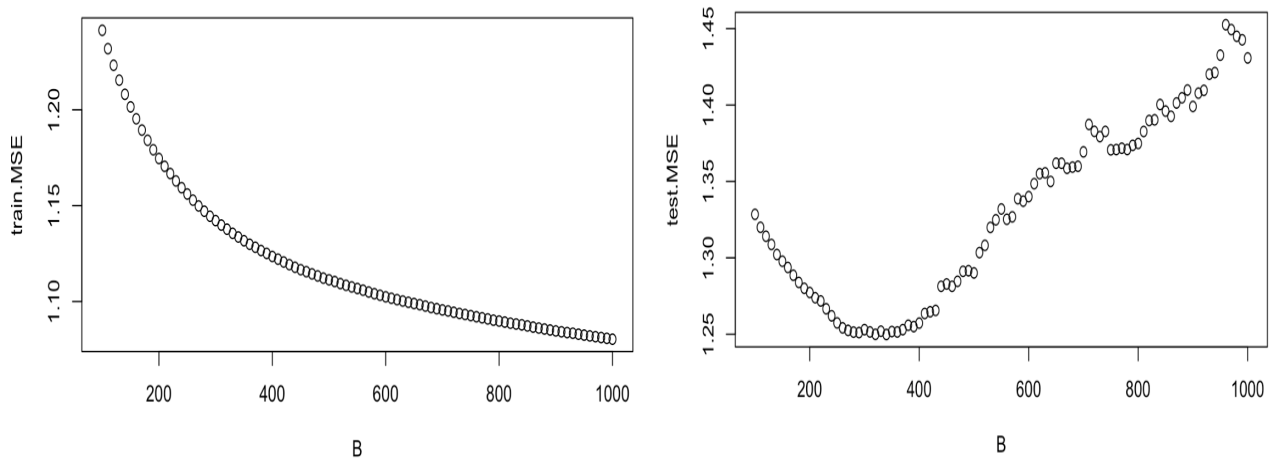
Based on the previous analysis, we know that there are some features highly correlative in our data. Therefore, we chose to use random forests because it could still perform well for highly correlated features when tune parameters are proper.

Similar with the previous approach to find the tuning parameter, we chose 5 as the number of sub-features and the testing MSE is 1.6760. According to the features importance plot, the result is almost the same as linear regression. However, random forest model seems to have higher bias.

4.2.2 Boosting

In order to reduce bias, we use boosting, but we spent more effort on tuning parameters: number of trees (B), tree's complexity/depth and shrinkage rate λ. Large B will cause overfitting, so we would like to try cross validation to find the optimal B. The CV plot for test and training error is shown in Figure 9.

In test MSE plot, the test error reaches the lowest point at B=340, with error rate equals 1.249811. Then we trained the model with the whole training data with B=340 and the final test error is 1.24.



(a) (b)

Figure 9: Cross Validation on Boosting

Besides boosting, we also tried XGBoost in regression analysis and XGBoost got the best test error which equals 1.196031.

To sum up, XGBoost error rate is the lowest among the tree algorithms. From the perspective of accuracy: XGBoost> boosting> random forest.

4.3 Approach 3: Deep Neural Network

As been clarified in many latest papers, a deep neural network, with reasonable network structure and parameters, can fit any function arbitrarily well. That is, neural network can be trained to process prediction tasks in high accuracy with enough data. Therefore, we built a simple deep sequential neural network with 5 hidden layers and one output layer. All neighboring layers are fully connected (Structure shown as Table 5. We dropped some features with huge amount of catagories and missings like region, metro,

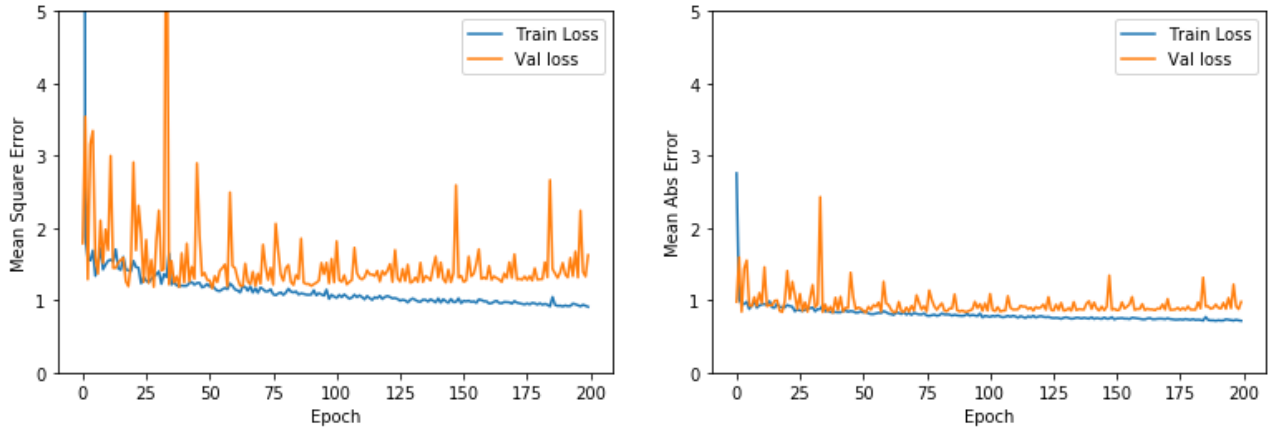
Table 5: Neural Network Structure

Layer (type)	Output Unit #	Param #
Input(Dense)	740	.
1 (Dense)	2048	1517568
2 (Dense)	512	1049088
3 (Dense)	1024	525312
4 (Dense)	512	524800
5 (Dense)	32	16416
Output (Dense)	1	33

Total params: 3,633,217

city, referralPath, keyword, networkDomain, and replace the time by time periods, and then used one-hot encoder for all catagory variables. After these steps we got 740 features in total.

We randomly choose initial parameters, and training our neural network using Adam Optimizer several times. One of the training results after 200 epoch is shown in Fig. 10. The training error is generally going down, while test error has a trend of going down at first and then vibrate in a unstable level. We choose the model with lowest validation loss, which is 1.16364 as the result after 52 epochs. The corresponding training loss is 1.2208. After that, though training error continues going down, there is no positive effect on validation loss, suggesting overfitting happens. We may need different features or better net structure to enhance to result.



(a) Mean Square Error after each Epoch

(b) Mean Absolute Error after each Epoch

Figure 10: Neural Network: Training Results

4.4 Comparison

Each model above gave us a smaller MSE compared to predicting the average revenue, but the difference of their MSE is not significant. To compare their performance, we train these model on the same training set and ridge regression still have the lowest error rate. Therefore we will use Ridge Regression ($\lambda = 0.001789537$) for the regression model in the whole prediction.

5 Wrap Up Results

The final evaluation of our model will be in this form:

- Separate the data into two half (60%+40%) by visit time.

- Balance the non-revenue data and with-revenue data in training set by duplicating the with-revenue data 60 times.
- Train the classification model using the training set, and train the regression model use the with-revenue data in training set.
- Predict by the classification model, then predict the revenue for the positive samples by the regression model. Calculate the MSE.

The final result is shown in Table 6. We are already have the prior expectation that random forest will perform badly on TPR, however its bad performance(TPR=2.86%) in testing is still out of our expectation. Though random forest has a lower test error for classification, but it almost predicted all samples to be negative in test set, so we do not believe it is better than XGBoost. Therefore, XGBoost has a lowest test error in the classification period.

Table 6: Testing Result using the best two model

Model	RF+Ridge	XGBoost+Ridge
Classification:		
Training Error	0.01%	1.60%
Test Error	1.44%	2.36%
Regression Test Error		
for negative-predicted sample	4.4125	1.1327
TPR	2.86%	75.2%
FPR	0.05%	2.03%
Regression:		
Training Error(MSE)	1.177	1.177
Test Error(MSE)	171.9413	203.156
Regression training with predicted data:		
Training Error*(MSE)	1.177	1.1409
Test Error*(MSE)	0.442	0.5285
Combined Model:		
Test Error(MSE)	4.567	7.35
Test Error*(MSE)	4.426	1.11

However, the higher FPR on XGBoost makes bad predictions in regression. If we use the true positive data in training set to build ridge regression model, we will have a training error of 1.177 but a much higher test error of 203.156! This high test error is caused by the false positive samples predicted in the first step. Therefore, XGBoost failed to be the better one and got a higher test error in regression. For the whole combined models on the metric of MSE, random forest plus ridge regression performed better.

To improve our model performance, instead of using real positive data to train the ridge regression model, we use predicted positive data to do that. Both XGBoost and random forest have 100% TPR and

low FPR on training set, which helps introduce negative data into the model and make better prediction on false positive data(Result using this method is marked with "*" in Table 6). This changing had tremendous magic on our prediction and made the prediction on false positive data much more reliable. In this way it is obvious that a model with higher TPR will be appropriate in our problem under the metric of MSE.

Data: X_{train},$Revenue_{train}$,X_{test}
Result: Prediction of log revenue of X_{test}
Training:
Balance the positive and negative sample in training data by duplicating the fewer one;
Train the classification XGBoost model using the training set with parameters: tree_depth = 7, eta = 0.07, nround = 1000 ;
Make prediction on training data and select the predicted positive data. Use them to train the ridge regression model with $\lambda = 0.001789537$;
Stack this two model;
Predicting:
if *classify*(X_{test}) $= -1$ **then**
    Return 0;
else
    Return $Regress(X_{test})$*;*
end

Algorithm 1: Revenue Predicting Algorithm

Stacking up all above, the best model(algorithm) we choose to use is shown as Alg. 1. The minimum MSE we get from our algorithm is 1.11. The log revenue variance of all data is 4.01. Therefore our algorithm has a great performance.

6 Conclusion and Future Study

In this study we analyzed the data of customer behavior from Google Store, and built a mixed model combining XGBoost classification and ridge regression to predict shopping behavior of every visit, which is the best model among all we have tried. Using the mixed model we can predict the log revenue with an average error of about 1, considering zero revenue visits behavior and non-zero ones are highly separative.

However, it is still hard to specifically tell how much money they will spend, since we know nothing about what they are looking for. Google now provides a new version of the data set, which contains what items visitors look through and how long they stay at each page. The richer information can help make

better prediction of revenue. Next we can use these data to do future analysis, expecting to have a lower error.

On the other hand, in this study we assumed that all visits are independent and not considering relationships between visitors. For example, one might explore what he wanted to buy on his smartphone but then order by using his desktop. We can group people by their visit start time, visitor id and other information, and then consider these visits as one visit. That may help to build better predicting model.

Finally, people always wish to get prediction on what people *will* do rather than what they *are* doing. We can do further study on using existed data to predict the revenue that one customer will made in the future months. In this way Google can draft different advertising price to better serve the customers and merchants, which makes a win-win business.

Reference

Data Source: Google Analytics Customer Revenue Prediction — Kaggle. *https://www.kaggle.com/c/ga-customer-revenue-prediction*

R Packages: leaps, xgboost, randomForest, e1701, glmnet, glm, gbm

Document of xgboost: *https://cran.r-project.org/web/packages/xgboost*

Python packages: numpy, pandas, sklearn, tensorflow, matplotlib.pyplot, imblearn