Google Analytics Customer Revenue Prediction

Adithya Gautam (agb160630), Vignesh Iyer (vxm154230)

Introduction

The Pareto principle (also known as 80/20 rule or the law of vital few) states that, for many events, roughly 80% of the effects comes from 20% of the causes. This rule has been proven true for many business- only a small percentage customers produce most of the revenue. As such, marketing teams are challenged to make appropriate investments in their promotional strategies.

Our goal here, is to analyze a Google Merchandise Store (also known as GStore, where Google swag is sold) customer dataset to predict revenue per customer using various Machine Learning algorithms like neural network, random forests, gradient boosted trees and XG boosting. Also, we are going to compare the algorithms and analyze which algorithms gives a better overall prediction.

Related work

A similar project to this G-store project is the DengAI project. These projects are similar in that the both involve in predicting to some degree an unknown quantity; potential revenue in our case and dengue fever cases in DengAI. A quick glance at both datasets reveals that there will be noise variables to eliminate and valuable attributes to identify in order to produce training and testing sets that will be capable of producing efficient models. The DengAI case involved the use of a gradient boosting regressor that allows it to predict the cases of dengue. Ideally to get mean squared error you would have to produce some amount of know target variable. However for this competition we assumed the revenue will be similar to the past given data and therefore were able to use library models to get error metrics.

Datasets and Features

Dataset Fields	Datatype	Dataset Field Description
fullVisitiorId	string datatype	unique identifier for each user of the Google Merchandise store.
channelGrouping	string datatype	channel through which the user came to the store
date	date datatype	date on which the user visited the store
device	string datatype	specification of the device used to access the store
geoNetwork	string datatype	information of geographic data of the user
socialEngagementType	boolean datatype (0 or 1)	either “socially engaged” or “not socially engaged”

sessionId	Integer datatype	unique identifier for this visit to the store
totals	float datatype	aggregate values across different sessions
trafficSource	integer datatype	information about the traffic source of the session
visitId	integer datatype	unique identifier for the session and unique to the user
visitNumber	integer datatype	session number for particular user
visitStartTime	POSIX time datatype	timestamp

Table 1: Data field with description of each field

The training set contains user transactions from August 1st 2016 to April 30th 2018 and the testing dataset contains user transactions from May 1st 2018 to October 15th 2018. There are totally 1783 instances on the training set and 1090 instances on the testing dataset.

We are predicting the natural logarithm of the sum of all transactions per individual (customer). This prediction will be for all customers purchasing from Google Merchandise Store from 1st December 2018 to 31st January 2019. For every user, the target is:

$$Y_{\text{user}} = \sum_{i=1}^{n} Transaction_{\text{user (i)}}$$

$$\text{Target}_{\text{user}} = \log_e(Y_{\text{user}} + 1)$$

Pre-processing techniques

Raw data is prone to noise, missing values and inconsistency. These may affect our final prediction. To overcome this issue, we pre-process the raw data more useful and understandable information. Data pre-processing can be divided into four parts;

Data cleaning : Data to be analyzed might be incomplete, contain errors or outliers or might have some discrepancies. If a missing value is noted, we filter out all the missing values. We can then smoothen the data (i.e.) remove random error (noise) from our data. For this we can fit our data to a regression function. Multiple linear regression finds the best line to fit two or more variables, so that we can predict one variable using other variables. This helps us to smooth out the noise from our data.

Smoothing: Removing noise from the data (using regression or binning to smoothen the data).

Dimensionality reduction. Complex data analysis may take a very long time, making it infeasible analysis. Here irrelevant, weakly relevant and redundant attributes are detected and removed from the original dataset. Encoding methods like PCA (Principal Component Analysis) are used to reduce size of the dataset. In our case, the data which are constant (i.e.) ones which does not have any impact on our predictions are considered as noise and removed from the dataset. The data which are removed are:

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

Methods

In our analysis, we are going to implement neural network to predict customer revenue of Google Merchandise Store and then we compare the performance of neural network with methodologies like random forest, gradient boosting and XG boosting to study and analyze the accuracy of each learners.

I. Neural Network

A perceptron is an unit linear classifier that only works with linear separable data. A neural network is combination of multiple perceptron in multiple layers. This neural network is a powerful learner and provides a robust approach to approximating real-valued or discrete-valued target functions. Neural network can be used in these kind of predictions because of its distributed representation, learning ability, generalization ability, parallelism and high fault tolerance. In general, neural network has the ability to determine the function solely based on its inputs and the ability to produce reasonable outputs for inputs that are not used to train them.

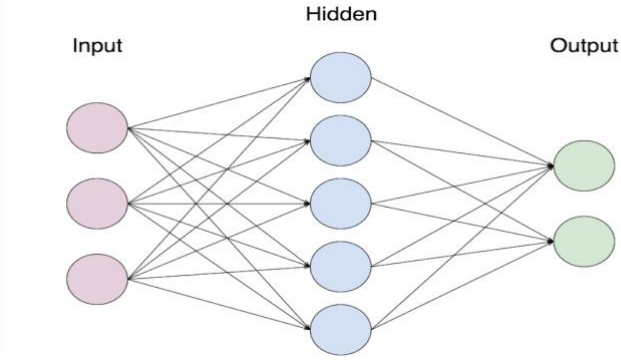

Fig 1: Neural Network with one hidden layer with five hidden unit

For every neuron unit in the network, there is an activation function which is applied to weighted sum of inputs to produce an output for that neuron. Majority of the neural network use sigmoid function. The advantage of a sigmoidal function is that it has a bounded range (i.e.) never reaches maximum or minimum and is monotonically increasing (i.e.) the derivate of the function is always positive.

$$\text{Sigmoid function} \qquad s(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d(s(x))}{dx} = -\frac{1}{(1+e^{-x})^2}(-e^{-x})$$

$$= \frac{1}{1+e^{-x}}\left[1 - \frac{1}{1+e^{-x}}\right]$$

$= s(x)\,[1-s(x)]$ (derivate of the sigmoid function is a positive quantity)

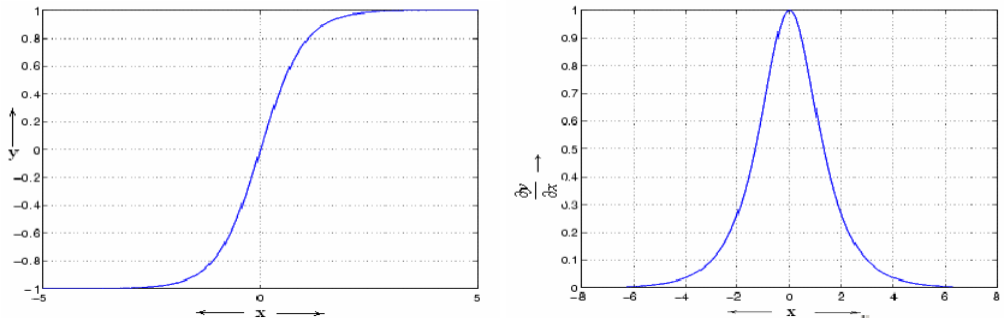

Fig 2: Sigmoid function (left) and derivative of sigmoid function (right)

The weights in neural network are very vital for determining its function. Training a network is an act of presenting the network with some sample data and modifying the weights to better approximate the desired function. The two types of training are supervised training and unsupervised training. In supervised training, response of the network to the inputs is measured (weights are modified to reduce the difference between the actual and desired outputs). In unsupervised training, the network adjusts its own weights so that similar inputs cause similar outputs.

II. Random Forest

Random forest is a flexible, easy to use algorithm that produces, even without hyper-parameter tuning, great results most of the time. It is widely used algorithm because of its simplicity and the fact that it can be used for both regression and classification tasks. Random Forest is a supervised learning algorithm, which creates forest and makes it somehow random. In other words, the algorithm builds multiple decision trees and merges them together to get a more accurate and a stable prediction.

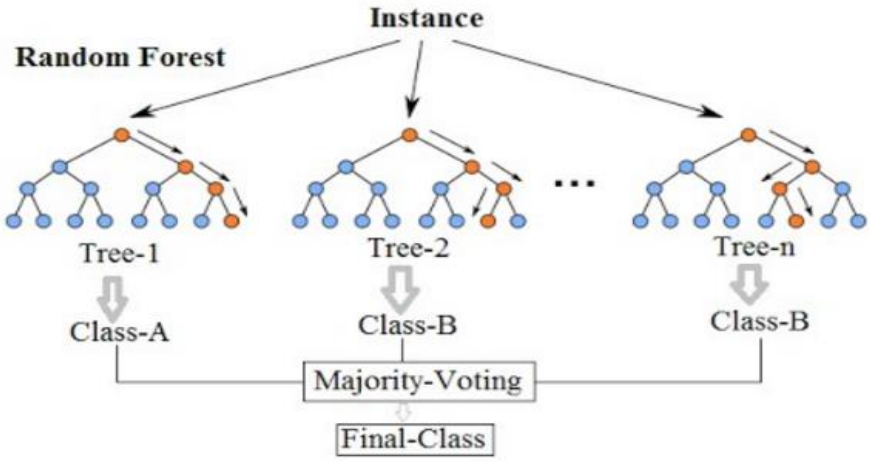

Fig 3: Decision trees grouped to form random forest

Random Forest has nearly the same hyperparameters as decision tree or a bagging classifier. It adds additional randomness to the model while growing decision trees. Instead of searching for the most vital feature while splitting a node, random forest algorithm searches for the best feature among a random subset of features. This results in a better model. Another nice quality of the random forest is that it is easy to measure the relative importance of each feature on the prediction. By looking at feature importance, one can decide on the features that would not contribute enough or nothing to the prediction. This is very important, because in machine learning with more number of features, the model might likely to suffer from overfitting problem. Hyperparameters are important in random forest, because they are either used to increase the predictive power of the model or make the model much more faster. For increasing the predictive power, we can adjust the n_estimator hyperparameter, which is just the number of decision trees the algorithm builds before taking the averages of the predictions. Also, max_features is another hyperparameter which can cantor the maximum number of features the algorithm is allowed to try in an individual tree. For increasing the speed of the model, min_sample_leaf which determines the minimum number of leaves that are required to split an internal node. Lastly, oob_score can be used in order to preserve one third of the training data which can be used to evaluate the performance of the algorithm. These samples are called out-of-the bag samples. It is similar to leave-one-out cross validation, but no additional computational burden goes along with it.

III. Gradient Boosting Algorithm

When predicting target variable using any machine learning technique, the main difference in actual and predicted values are either noise, variance or bias. An ensemble helps to reduce these effects. An ensemble is collection of predictors that comes together to give final prediction. Boosting is an ensemble technique in which predictors are made sequentially, not independently. The gradient boosting algorithm a prediction model in form of an ensemble of weak prediction. Let us consider mean square error (MSE);

Loss function MSE = $\sum(y_i - y_i^p)^2$, where y_i= ith target value and y_i^p = ith prediction value
$y_i^p = y_i^p + \alpha * \delta (\sum(y_i - y_i^p)^2/\delta y_i^p \Rightarrow \quad y_i^p = y_i^p - \alpha * 2 * \sum(y_i - y_i^p)$ where, α is the learning rate and $\sum(y_i - y_i^p)$ is the sum of residuals.

By using gradient descent, and updating the predictions based on learning rate, one can find the value that minimizes the MSE (loss function), which is our ultimate goal. Here, the prediction is updated such that the sum of residuals becomes close to zero and prediction is close to the actual value. In gradient boosting the central idea is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it a better model. At one stage, the residuals may not have any pattern that could be modeled, which implies that the algorithm should be stopped. Otherwise, it can lead to overfitting issues.

Experimental Results and Analysis

For each regressor, we ran experiments using support vector regressor (SVR), random forest regressor, gradient boosting and finally a multilayer perceptron. We are also implementing GridSearchCV for tuning the parameters for each regressors and the results are tabulated below.

I. Neural network (best parameters)

Parameter name	Best parameter value
Hidden layer size	(500,300,200)
Activation	Logistic
Solver	SGD
Alpha	0.001
Learning rate	0.001
Max iterations	1000
Random state	None
Tolerance	1e-05
Verbose	False
Momentum	0.9
Epilson	1e-07

II. Random forest (best parameters)

Parameter name	Best parameter value
N estimator	100
Criterion	mse
Max depth	None
Min sample split	4
Min sample leaf	40
Max features	20
max leaf nodes	30
Bootstrap	True
Oob_score	True
Random state	1

III. Gradient Boosting (best parameters)

Parameter name	Best parameter value
Loss	Ls
Learning rate	0.001
N estimator	100
Min sample split	7
Min sample leaf	20
Max depth	5
max leaf nodes	30
Min impurity split	1e-9
Oob_score	True
Max features	10
Alpha	0.5

IV. Support Vector Regressor (best parameters)

Parameter name	Best parameter value
kernal	poly
Degree	5
Gamma	auto_deprecated
Penalty parameter (C)	0.6
Tolerance	0.01
Epsilon	0.5
Max iteration	200
Verbose	False

Learner	Mean Square Error	Explained Variance	R-square Score	Mean Absolute Error

Neural network	4.1514e-18	0.0	0.0	2.0375e-09
Random forest	0.0	1.0	1.0	0.0
Gradient boosting	0.0	1.0	1.0	0.0
Support vector	0.0	1.0	1.0	0.0

From the above table of regression performance, it is evident that multilayer perceptron regressor does the worst fitting. The mean square is considerably high, but mostly the r-square score and variance explained by the regression fit are both 0. This translates into saying that the variability of data is not explained by the fitted model and also the prediction results are not so accurate. When it comes to ensemble technique (i.e.) both random forest and gradient boost regressors, the squared error and absolute error are 0, implying the performance is very good. Also, it could the fit is very accurate in these cases, since both regressors have a r-square score of 1.0 (best prediction accuracy) and variability is also 1.0 (fit explains all the variability in the dataset).

Finally, support vector performance is as same as the ensemble technique and it can be seen that this is sensible since the kernel used for support vector is a polynomial kernel and also the degree of polynomial is set high (please refer the best parameter table above).

	fullVisitorId	PredictedLogRevenue
0	0000059488412965267	0.0
1	0000118334805178127	0.0
2	0000174453501096099	0.0
3	0000397214032106948	0.0
4	0000421850492821864	0.0
5	0000955020478109523	0.0
6	0000998759681663618	0.0
7	0001059349366430257	0.0
8	0001251773762451924	0.0
9	0001269799921883506	0.0
10	000140003762682805	0.0
11	0001499588478181249	0.0
12	0001521034365414265	0.0
13	0001522706438705020	0.0
14	000184052201924153	0.0
15	0001875931252987660	0.0
16	0001936363130317831	0.0
17	0002032890182827978	0.0
18	0002261098850688772	0.0
19	0002585859302677487	0.0
20	0002710340700180818	0.0
21	0002716572822023371	0.0
22	0002767095999940001	0.0
23	0002790207422855765	0.0
24	0002898210441062509	0.0
25	0002982451707782308	0.0
26	0003221164989695564	0.0
27	0003383691915264439	0.0

28	0004109950587123497	0.0
29	0004420062338776803	0.0
...	...	...
83071	999733032197512248	0.0
83072	9997386018919084520	0.0
83073	9997410095465539131	0.0
83074	999745775113794640	0.0
83075	99976789209401933	0.0
83076	9997870411663962231	0.0
83077	9998298038378056727	0.0
83078	9998324633121536664	0.0
83079	9998453538864140422	0.0
83080	9998586078553303691	0.0
83081	9998593574150070182	0.0
83082	99986329012823178	0.0
83083	9998706979086672199	0.0
83084	9998714718001469075	0.0
83085	9998739239824248627	0.0
83086	9998871186081510973	0.0
83087	9998875558770947556	0.0
83088	9998948869605353323	0.0
83089	9998953829285454941	0.0
83090	9999099829057411345	0.0
83091	9999138145553675515	0.0
83092	999914072337251353	0.0
83093	9999205271965164893	0.0
83094	9999250019952621738	0.0
83095	9999425102935415824	0.0
83096	999942696836758605	0.0
83097	9999664171331798815	0.0
83098	9999686339684184799	0.0
83099	9999803509476553670	0.0
83100	9999941518946450908	0.0

[83101 rows x 2 columns]

The above table, contains the full visitor ID in our test set and predicted natural log of their total revenue for each user in the google merchandise store.

Conclusions

In summary, we are using various regressors to predict the natural logarithm of total revenue are each user of the Google Merchandise Store and we use RMSE (root mean squared error) as an evaluation measure and lower the RMSE, more accurate our prediction will be. Based on the results, we can say that MLP regression does not give very high accuracy just because of the high mean squared error obtained. Whereas, regressors like random forests and gradient boosting gives better results by giving lower mean squared error, which translates into better overall revenue prediction for each customers. This makes sense since, random forests and gradient boosting are ensemble techniques which uses ensemble of weak predictors to improve the overall accuracy of the prediction. Also, support vector regressor gives a lower mean squared error when used with a non-linear kernel.

References

[1] Prince Grover. "Gradient Boosting from scratch", ML review blog, medium.com

[2] Dilek Penpece and Orhan Emre Elma. "Predicting Sales Revenue by Using Artificial Neural Network in Grocery Retailing Industry", International Journal of Trade, Economics and Finance, Vol. 5, No. 5, October 2014

[3] Amita Gajewar . "Revenue Forecasting for Enterprise Products", proceedings of International Symposium on Forecasting 2016

[4] Hyndman RJ and Khandakar Y. "Automatic time series forecasting: the forecast package for R", Journal of Statistical Software

[5] Serasa Experian. "Revenue Score - Forecasting Credit Card Products with Zero Inflated Beta Regression and Gradient Boosting"

[6] Mariana Listiani. "Support Vector Regression Analysis for Price Prediction in a Car Leasing Application", Master Thesis, Information and Media Technology, Hamburg University of Technology

[7] Niklas Donges. "An introduction to random forest algorithm", Towardsdatascience.com

[8] scikit learn documentation and tutorials "https://scikit-learn.org/stable/documentation.html"