Title
Click Prediction with Machine Learning Tools

Permalink
https://escholarship.org/uc/item/0cf772s3

Author
Ding, Fan

Publication Date
2018

Peer reviewed|Thesis/dissertation
Click Prediction
with Machine Learning Tools

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Fan Ding

2018
ABSTRACT OF THE THESIS

Click Prediction
with Machine Learning Tools

by

Fan Ding
Master of Applied Statistics
University of California, Los Angeles, 2018

Professor Yingnian Wu, Chair

In this paper, we will explore multiple machine learning tools with their applications in the industry of advertisement technology. Companies like Sabio Mobile Inc. aim to provide platforms for advertisers to get their ads published and target their users with higher accuracy. With transaction log files offered by Sabio Mobile Inc., we will train several statistical models to predict whether or not a user will click on a certain ad.
The thesis of Fan Ding is approved.

Nicolas Christou
Qing Zhou
Yingnian Wu, Committee Chair

University of California, Los Angeles
2018
# TABLE OF CONTENTS

1 Introduction ......................................................................................... 1

2 Problem Defining .................................................................................. 3

3 Data ........................................................................................................ 4
   3.1 Data Summary .................................................................................. 4
   3.2 Data Processing ............................................................................... 4

4 Methodology .......................................................................................... 8
   4.1 Logistic Regression .......................................................................... 9
   4.2 Neural Network .............................................................................. 10
   4.3 Random Forest ............................................................................... 12
   4.4 Gradient Boosting Machine (GBM) .................................................. 13
   4.5 Extreme Gradient Boost (XGBoost) ............................................... 15

5 Data Implementation and Model Training ........................................... 16
   5.1 Logistic Regression .......................................................................... 17
   5.2 Neural Network ............................................................................... 19
   5.3 Random Forest ............................................................................... 21
   5.4 Gradient Boosting Machine (GBM) .................................................. 22
   5.5 Extreme Gradient Boost (XGBoost) ............................................... 24

6 Result Comparison and Discussion ...................................................... 26
   6.1 Comparing Logistic Regression and Neural Network ...................... 26
   6.2 Comparing Three Advanced Tree Algorithms ................................. 26
LIST OF FIGURES

4.1 Activation Function Summary ........................................ 11
4.2 Activation Function Plot ........................................... 11

5.1 Logistic Regression Hyperparameter Adjustment ..................... 17
5.2 Logistic Regression Prediction Result ................................ 18
5.3 Neural Network Hyperparameter Adjustment ......................... 19
5.4 Neural Network Prediction Result .................................... 20
5.5 Random Forest Hyperparameter Adjustment ......................... 21
5.6 Random Forest Prediction Result ..................................... 22
5.7 Gradient Boosting Machine Hyperparameter Adjustment ........... 22
5.8 Gradient Boosting Machine Prediction Result ....................... 23
5.9 Extreme Gradient Boost Hyperparameter Adjustment ............... 24
5.10 Extreme Gradient Boost Prediction Result ......................... 24

6.1 Model Effectiveness Comparison ..................................... 27
LIST OF TABLES

3.1 Variable Name and Explanation ............................................. 5
3.2 Variable Name and Explanation (cont.) .................................... 6
3.3 Variable Type and Sample Value ............................................. 7
CHAPTER 1

Introduction

Advertising technology, also known as ad tech, refers to the implementation of different software solutions to various aspects in advertising, including audience targeting, data collection, decision making and ad delivery [1]. Starting from the year of 1993, ad tech keeps developing and gradually changes the advertising ecosystem. In 1993, the first banner ad was published; the advertising system was very simple since the advertiser made direct contact with the publisher [1]. Two years later, intermediary agents first occurred to connect advertising side and publishing side of ads and formed a so called “ad network” [1]. In the following decade, this ad network pattern continued to develop and ad exchanges appeared, served as platforms to bring both buying and selling parties of the advertising inventory together to a common marketplace [2]. Correspondingly a bidding system had been invented to pricing the inventory. In the year of 2008, real-time bidding (RTB) system was introduced to the marketplace, which allows more inventory sources to be published and takes the ad tech to a new era [2].

Along with the evolution of advertising technology, people in the industry rely more and more heavily on machine learning tools to provide efficient solutions for various issues. This paper will provide an example of such problem solving process.

Machine learning is defined as “a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty” [3]. From a statistical perspective, machine learning tools are based on the probability theory, which is to deal with uncertainty coming in multiple forms: best prediction to make about the future, optimized probabilistic models to apply, further measurement to conduct, etc [3]. In other words, the role of machine
learning is to help people draw patterns from the past and use those patterns to make better decisions in the future, reducing the influence of unknown as much as possible.

There are two main types of machine learning: supervised learning and unsupervised learning.

Supervised learning (SL), also known as predictive approach, is defined to “learn a mapping from inputs $x$ to outputs $y$, given a labeled input-output pairs” [16]. To implement this approach, the data need to contain both features or attributes as predictors and output variables as responses. Based on the responses, there are two major subtypes in supervised learning. If values in the output variable is discrete from a finite set, the problem is known as classification or pattern recognition, while if they are from a infinitely continuous value set, the problem is called regression [16].

Unsupervised learning, on the other hand, is defined as a descriptive approach to find patterns the data, dealing with the data that do not contain any predetermined output [16]. In supervised learning, we derive models to generalize patterns or distributions of the response variable, and test the goodness of each model by comparing our predicted output values to the observed ones. However, there are neither specific patterns to look for, nor obvious error metrics to apply for unsupervised learning approaches due to the lack of response variables in the data [16].
CHAPTER 2

Problem Defining

In advertising, a demand-side platform (DSP) is an interface that uses RTB process to bid for available inventories on behalf of advertisers [3]. When a publisher sends an ad request to a RTB exchange, the RTB exchange correspondingly sends a bid request to all subscribed DSPs; those DSPs will then respond with a predetermined bidding price [3]. Through second bid price auction, the RTB exchange decides the winner to be the DSP with highest bidding price, and that ad from the advertiser served by the winning DSP, will be instantly displayed on a publishers site for a user to see [3]. The whole process takes place within 120 milliseconds, so the user will see the ad as soon as he or she interacts with the publishers site, or in other words, in real time.

In such a RTB advertising market, a DSPs aim is to maximize the revenue for both its advertisers and itself [4]. A prevalent pricing model for the DSPs is CPC meaning cost per click; they charge advertisers for every click on their ads from users. So in order for DSPs to boost profit, it is crucial to increase number of clicks on ads from advertisers they provide service to.

A solution for this is to target the user groups that are more likely to click on a certain ad before bidding, to increase the potential number of clicks. In this case, some in-advance estimation on whether or not a user will click on a certain ad will be helpful.

To achieve this goal, we apply machine learning tools to train some statistical models for click prediction.
CHAPTER 3

Data

3.1 Data Summary

I use a transaction log file from Sabio Mobile Inc.. Every time a bid is generated, a record will be created and kept named a transaction log, containing the users information and interactions he or she makes with the ad. From the transaction log file, we can see if the user clicks the ad and use it as a further indication of his or her interest.

The dataset contains 16 predictors, 1 response variable click and 50000 observations.

The information about variables is shown in Table 3.1, Table 3.2 and Table 3.3.

3.2 Data Processing

To increase the running speed of each model, we change all variables with type string to type factor indicated by integers; this way we will significantly reduce the size of our data so that the time to run each model will significantly shrink. In addition, we remove all NAs from the dataset before we start to train the model.
<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campaign Id</td>
<td>A campaign is a series of advertisements that share a single theme. Each campaign has a unique id associated with it.</td>
</tr>
<tr>
<td>Site ID</td>
<td>A site is a website or an app that ads are delivered on. Each site has a unique id associated with it.</td>
</tr>
<tr>
<td>Media ID</td>
<td>Advertising media means various channels through which advertising is done. Each medium has a unique id associated with it.</td>
</tr>
<tr>
<td>Language</td>
<td>The language type of the ad.</td>
</tr>
<tr>
<td>Secure</td>
<td>Binary; 1 for secure http traffic and 0 for non-secure http traffic.</td>
</tr>
<tr>
<td>Inventory Source</td>
<td>An inventory source is a RTB exchange sending bid request.</td>
</tr>
<tr>
<td>Region ID</td>
<td>A region id is a integer representing a state in the United States.</td>
</tr>
<tr>
<td>DMA</td>
<td>DMA stands for designed marketing area, representing different marketing areas.</td>
</tr>
<tr>
<td>Zip Code</td>
<td>Zip code in string type.</td>
</tr>
<tr>
<td>DNT</td>
<td>DNT stands for do not track, which is a web browser privacy setting for not tracking the users.</td>
</tr>
<tr>
<td>Media Type</td>
<td>Media have multiple types including online media, print media, mobile media, etc, and each one might have subtypes within it.</td>
</tr>
</tbody>
</table>
Table 3.2: Variable Name and Explanation (cont.)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Variable Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos</td>
<td>The position of the ad inventory.</td>
</tr>
<tr>
<td>Site Or App</td>
<td>Binary; 1 for web site and 2 for app.</td>
</tr>
<tr>
<td>User Agent Make</td>
<td>An user agent identifies the agent, including information on browser software and operating system, that are making the request; make means the software name.</td>
</tr>
<tr>
<td>User Agent Model</td>
<td>A string indicating models for the devices.</td>
</tr>
<tr>
<td>Carrier</td>
<td>A carrier is a service provider which supplies data service to mobile phone and tablet devices.</td>
</tr>
<tr>
<td>Click</td>
<td>Binary output; 1 for click and 0 for non-click.</td>
</tr>
<tr>
<td>Variable Name</td>
<td>Variable Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Campaign Id</td>
<td>Integer</td>
</tr>
<tr>
<td>Site ID</td>
<td>String</td>
</tr>
<tr>
<td>Media ID</td>
<td>String</td>
</tr>
<tr>
<td>Language</td>
<td>String</td>
</tr>
<tr>
<td>Secure</td>
<td>Integer</td>
</tr>
<tr>
<td>Inventory Source</td>
<td>Integer</td>
</tr>
<tr>
<td>Region ID</td>
<td>Integer</td>
</tr>
<tr>
<td>DMA</td>
<td>Integer</td>
</tr>
<tr>
<td>Zip Code</td>
<td>String</td>
</tr>
<tr>
<td>DNT</td>
<td>Integer</td>
</tr>
<tr>
<td>Media Type</td>
<td>Integer</td>
</tr>
<tr>
<td>Pos</td>
<td>Integer</td>
</tr>
<tr>
<td>Site Or App</td>
<td>Integer</td>
</tr>
<tr>
<td>User Agent Make</td>
<td>String</td>
</tr>
<tr>
<td>User Agent Model</td>
<td>String</td>
</tr>
<tr>
<td>Carrier</td>
<td>String</td>
</tr>
</tbody>
</table>
CHAPTER 4

Methodology

To choose proper machine learning models to train, we need to check the dataset we use. Since our data contain the binary output variable click, it is appropriate to apply supervised learning for classification to train the model. The two main types of algorithm that will fit are neural network and the tree algorithm.

Before we try neural network, we can use logistic regression to fit the data first. It is a simple parametric method that is easy to apply and understand, so it is one of the most commonly used approach to solve the click prediction problem. Since the neural network model can be regarded as a multilayer version of logistic regression, we assume that it will generate better result than logistic regression. Therefore we do neural network next to compare the performances of these two approaches. Besides, the tree algorithm for classification is also powerful in predicting categorical responses. To get a higher accuracy rate, we apply three advanced tree algorithms: random forest, gradient boosting machine and extreme gradient boost.

To test the goodness of each model, we focus on prediction accuracy, and also pay attention to efficiency. Among multiple approaches, we prefer ones that provide large numbers of correct prediction with less complexity at the same time.

Finally we do comparison among all methods mentioned above. In addition to comparing four machine learning techniques, we want to see how much improvement it possibly makes from logistic regression to neural network, and which tree algorithm works the best.
4.1 Logistic Regression

This is the most common method people use to predict click. Logistic regression, also called logit regression or logit model, is a parametric regression method for predictive analysis. It estimates probability of occurrence of a categorical response based on one or more predictors [5].

There are two basic types of logistic models. Binary logistic regression model is used when the dependent variable is a dichotomous response such as Yes/No or Success/Fail, while independent variables can be either continuous or categorical [5]. Usually to generalize the result, we substitute two categories of response variable with 0 and 1. If the dependent variable is comprised of more than two levels, the model is known as multinomial or nominal logistic regression model [5]. Specifically, when the multilevel dependent variable is an ordered response such as Light/Medium/Dark, the model is called ordinal logistic regression model. It is considered to be more powerful because it takes into account the ordering of the response.

Since the topic of this paper is to predict a two-level variable, we will focus on binary logistic regression.

To construct a logistic regression, we define odds first. We assume \( p \) is the probability that the value of outcome \( Y \) is 1, then \( 1 - p \) will be the probability that it will be 0. The corresponding odds is defined by

\[
\text{odds} = \frac{p}{1 - p} = \frac{\text{probability}(y = 1)}{\text{probability}(y = 0)}
\] (4.1)

Then we perform a logit transformation and linearly regress the natural log of odds on the independent variables:

\[
\text{logit}(y) = \ln(\text{odds}) = \ln\left(\frac{p}{1 - p}\right) = \beta_0 + \beta_1 * x_1 + \cdots + \beta_k * x_k
\] (4.2)

where each \( x \) is one predictor and each \( \beta \) is the corresponding coefficient.
A logistic regression coefficient can be interpreted as the amount of change in log odds when the corresponding predictor increases by one unit.

After the linear regression, we transform the log odds back to probability $p$ by taking antilog of equation (4.2) on both sides:

$$
p = P(Y = 1|X_1 = x_1, \ldots, X_k = x_k) = \frac{1}{1 + e^{-\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k}}
$$

(4.3)

and use $p$ to predict the value of the dependent variable $Y$ [5]. If $p$ is larger than a certain threshold, we predict the value of $Y$ as 1, otherwise we predict the value of $Y$ as 0.

With logistic regression algorithm, our goal is to find the best fitted model to generate a relationship between click and all predictive variables, and then make prediction of whether the outcome is a click or not given those predictors.

### 4.2 Neural Network

Neural network can be regarded as a multilayer network of logistic regression unit. Since this model contains more layers with deeper and more complex structure, we assume that it is more powerful in prediction than one-layer parametric logistic regression. After the data implementation of both models, we will compare their performance to check if this statement is true.

Neural network approach is a nonlinear model used for tasks such as prediction and classification.

The network consists of three layers: one input layer, one hidden layer possibly including multiple sublayers and one output layer, each containing individual but interconnected elements called neurons, nodes or units [6]. The process for neural network to work is that the hidden layer first receives information from input layer, then the nodes in the hidden layer assign a weight to each input unit, add the weight-and-input multiplications and passes the sum to an activation function [6]. The activation function, also known as transfer function,
is applied to the weighted sum of inputs to produce the resulting output of neural network [7]. The commonly used activation functions include linear, step, logistic (sigmoid), tanh and rectified linear unit (ReLU).

The following two tables provide a summary of each activation function.

There are various types of neural network including feedforward neural network, convolutional neural network (CNN), recursive neural network (RNN) and recurrent neural network (RNN). For our dataset, we only consider the feedforward neural network, also known as multi-layer perceptron. It is the most straightforward type of neural network; it feeds information from front to back, moves from the input layer through hidden layer(s) directly to the output layer without forming cycles. In order to compare this result with logistic regression, we choose sigmoid function as the activation function, so that the neural network we implement is “a series of logistic regression models stacked on top of each other” [16].
4.3 Random Forest

Classification and regression tree (CART) model is a decision tree algorithm used for regression or classification prediction. It refers to “recursively partitioning the input space, and defining a local model in each resulting region of input space” [16]. Each CART unit can be represented by a binary tree with one leaf per region. Each leaf node corresponding to a segment of values for the independent variable and each leaf gives a prediction for the dependent variable value. CART algorithm is different from the logistic regression method. Logistic regression algorithm makes classifications based on the relationship between response and predictors, whereas for decision tree, the idea is to divide the dataset into smaller sections according to certain rules, until a small enough set is reached that data points in it fall under the same label.

With classification tree algorithm, we use impurity to choose split variable as well as their thresholds, and determine the predicted label (category) of $y$ based on the proportion of $y$ in each region of $x$. We assign $x$ with some random threshold, split the data and calculate the impurity. After repeatedly go through this process, we choose a split point with the minimum impurity.

The impurity is expressed by a Gini index function as following:

$$G = \sum_{k=1}^{K} (p_{m,k} \ast (1 - p_{m,k}))$$  \hspace{1cm} (4.4)

where $m$ is the region that $x$ is split into and $p_{m,k}$ is the proportion of training instances with category $k$ in region $m$. If a node is completely pure, meaning all data points belong to the same category, $G=0$. The more mixed of $Y$ values in the node, the larger $G$ will be [8].

We stop the process when all cases in a node is pure with identical value, or the node size is smaller than a certain limit we set, then we assign each node a value as the predicted response.

Random forest is a improved version of the tree algorithm which creates a forest with
a large number of trees. Each tree works as a weak classifier with relatively high variance, therefore the more trees included in the model, the more accuracy the prediction will have. To explain the algorithm, we add one step before growing a tree in CART algorithm: we repeatedly do random sampling on both observations and predictors with replacement, then grow a large number of trees using samples above consisting of a forest. The following is a complete illustration of the procedure [9]:

1. Randomly select $k$ features from total of $k$ features where $k$ is less than $m$;
2. With $k$ features, calculate the best split point using the method in CART and do splitting;
3. Repeat steps 1 and 2 until reaching a stopping point where the node size equals the predetermined minimum value or the values in each node become identical;
4. Build forest with a large number of trees by repeating steps 1 to 3.

Following this procedure, each tree will work as a weak classifier containing only a random segment of variables and observations. By continuously creating trees, the accuracy of the combined classifier will gradually increase and finally the total variance of this forest will be much lower than one tree. Thus, the random forest algorithm is robuster and more accurate than each individual decision tree.

4.4 Gradient Boosting Machine (GBM)

Boosting method is similar to random forest algorithm as in both of these two serve as stronger versions of the CART algorithm. The difference between them is, random forest improves the model performance by resampling the data, while boosting method achieves the same goal by reweighing the data.

Boosting is currently one of the most powerful and widely spread method in the field of machine learning. It refers to a combination of simple classifiers which converts weak learners to strong learners with better performance [10]. A weak learner is defined to be one whose performance is at least slightly but strictly better than a random chance, while a
strong learner provides classifications with much smaller error probability [10]. So the idea of boosting is to combine a number of poor-performing weak learners such as individual decision trees to form a powerful classifier. The algorithm assigns a weight to each observation, giving more on hard-to-classify records while putting less on wellhandled ones [11]. It initiates by placing equal weight on each observation then predicting. If the prediction is incorrect based on the earlier learner, it reassign higher weight to those incorrectly predicted records [11]. It keeps adding learners and iterates with a new random sample each time, until it reaches a stopping point that the lower limit of the node size is met or our ideal accuracy is achieved [11]. Predictions of all weak classifiers are combined through a weighted majority vote to produce the final prediction; the weight on each value is generated from previous iterative procedure.

Gradient Boosting Machine (GBM) is a generalized boosting framework. It improves the model by training on the remaining errors of the strong learner instead of training on a newly sampled distribution [12]. At each iteration, it computes the residuals from the previous classification and fit a weaker learner to those residuals, then a new learner is built by predicting the loss after each step until the error is smaller a certain threshold [12].

The algorithm of gradient boosting machine is as following:

\[
f(x) = \hat{f}(x) + \beta \ast h(x)
\] (4.5)

where at each iteration, \( \hat{f}_x \) is the previous guess, \( h_x \) is the base learner and \( \beta \) is the weight for the newly added learner. The learner here means each tree formed in each loop. The product \( \beta \ast h_x \) denotes the step at each iteration.

The boosting method is to solve the function

\[
\{\beta, h(x)\} = \arg\min_{\{\beta, h(x)\}} \sum (L(y_i, \hat{f}(x_i) + \beta \ast h(x_i)))
\] (4.6)

The term \( L(y_i, \hat{f}(x_i) + \beta \ast h(x_i)) \) is the error term and different error types have their own expressions. For this paper, we use L2 regularization which gives least squares error, so
\[ L = \sum(y_i - (\hat{f}(x_i) + beta \ast h(x_i)))^2 \]. Each time we add the step term to update the function \( \hat{f}_x \), and the final goal is to minimize the error that current \( \hat{f}_x \) cannot fit.

### 4.5 Extreme Gradient Boost (XGBoost)

Extreme gradient boost (XGBoost) can be regarded as an improved version of gradient boosting machine. It uses Newton’s method rather than traditional gradient descent to optimize the loss function, also called the objective function, through both first and second derivatives.

First recall that the equation of Second Taylor Expansion is

\[
f(x_0 + \Delta x) = f(x_0) + f’(x_0) \ast \Delta x + \frac{1}{2} \ast f''(x_0) \ast \Delta x^2 \quad (4.7)
\]

Similarly the objective function of XGBoost will be:

\[
L = \sum_{i=1}^{n} L(y_i, f(x_i)) = \sum_{i=1}^{n} (L(y, f(x_i))) + L'(y_i, \hat{f}_x) \ast \Delta x_i + \frac{1}{2} \ast L''(y_i, \hat{f}_x) \ast (\Delta f(x_i))^2
\]

\[
(4.8)
\]

We define \( g_i = L'(y_i, \hat{f}_x), a_i = L''(y_i, \hat{f}_x) \) and \( T(x_i) = \Delta f(x_i) \), then our goal is to grow trees to minimize the objective function of \( g_i T(x_i) + a_i T(x_i)^2 \). As a result, we get the optimal weight of each leaf on the tree, similar to the beta in GBM approach.
CHAPTER 5

Data Implementation and Model Training

We will train 5 models mentioned above to fit the data.

Before implementing the model, the whole dataset needs to be split into two subsets, with 60% of observations in the training set and the rest 40% in the testing set.

To train and test the model, we use cross-validation approach which is defined as “a model validation technique for assessing how the result of a statistical analysis will generalize to an independent dataset” [13]. Training set is used to learn about the dataset and to build the model, whereas testing set is an independent dataset to assess the performance of the generated model as well as to obtain the performance characteristics such as accuracy. According to the evaluation metrics of each run, we tune the hyperparameters of the model to improve the result. Hyperparameters are variables related to the algorithm configuration instead of the training data itself [14]. There are different types of hyperparameters and each one plays an unique role in improving the model accuracy.

Three metrics are used to assess the model performance: confusion matrix, ROC curve and AUROC. Confusion matrix, in machine learning, is one way to visualize the performance of the classification model. It is a table containing the actual observations and predicted classifications, with number of records both correctly and incorrectly predicted. We can calculate the correct percentage dividing number of correct predictions by the total number of predictions and use this as a measure of accuracy of the model. Receiver operating characteristic curve, also known as ROC curve, is a curve indicating “the diagnostic ability of a binary classifier system” [15]. Its x-axis is false positive rate, meaning the ratio of number of false positives (a positive result that should actually be negative) to total number of predictions. Its y-axis is the true positive rate, meaning the ratio of number of true
positives (a positive result that should actually be positive as well) to total number of predictions. A tradeoff exists: false positive rate and true positive rate changes toward the same direction, so when we increase false positive rate, true positive rate will also increase. The curve illustrates “the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one” [17]. We prefer a point with relatively smaller false positive rate and larger true positive rate at the same time. The corresponding AUC, also known as AUROC, is the area under the ROC curve. The larger the AUC, the more accurate the model and the prediction.

The following part of this paper is to train each model using our dataset with hyperparameter tuning processes.

5.1 Logistic Regression

<table>
<thead>
<tr>
<th>Hyperparameter adjustment</th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>0</td>
<td>0.01</td>
<td>1E-05</td>
</tr>
<tr>
<td>alpha</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.1: Logistic Regression Hyperparameter Adjustment

The goal of adding hyperparameters alpha and lambda in the model are to perform regularization and avoid overfitting. Alpha controls the allocation between L1 and L2 regularizations. When alpha = 0, the model applies L1 regularization while when alpha = 1, the model is applying L2 regularization. For this paper, we use L2 regularization for all penalties, so we do not change alpha value of 1. Lambda is a non-negative shrinkage parameter representing regularization strength. The larger lambda is, the more penalty will be introduced into the model, and therefore the more the coefficients will be reduced toward 0. It is important to tune lambda to provide a proper amount of regularization of the model, so that the model will not overfit and keep strong predictive power at the same time.
From figure 5.2 we can tell that the third run with lambda equal to 1e-5 gives the best result. The confusion matrix indicates that in the third run, 22 out of 9943 observations in the testing set are predicted incorrectly, which gives the highest accuracy of 0.997. From its ROC plot, when false positive rate becomes 0.2, true positive rate reaches 0.8 which is highest among all three. In addition, the third area under ROC of 0.849 is the largest. Taking all three metrics into consideration, the third setting with lambda of 1e-5 gives the best result.

<table>
<thead>
<tr>
<th></th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>confusion matrix table</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 Error</td>
<td>0 9921 11 0.001108</td>
<td>0 9912 20 0.002014</td>
<td>0 9922 10 0.001007</td>
</tr>
<tr>
<td>1 23 10 0.696970</td>
<td>22 11 0.666667</td>
<td>21 12 0.636364</td>
<td></td>
</tr>
<tr>
<td><strong>Totals</strong> 9944 21 0.003412</td>
<td><strong>Totals</strong> 9934 31 0.004215</td>
<td><strong>Totals</strong> 9943 22 0.003111</td>
<td></td>
</tr>
</tbody>
</table>

**AUROC**
- first run: 0.7775678
- second run: 0.79527
- third run: 0.8494032

Figure 5.2: Logistic Regression Prediction Result
5.2 Neural Network

<table>
<thead>
<tr>
<th>Hyperparameter adjustment</th>
<th>Initial run</th>
<th>Second run</th>
<th>Third run</th>
<th>Fourth run</th>
<th>Fifth run</th>
<th>Sixth run</th>
</tr>
</thead>
<tbody>
<tr>
<td>learning rate</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
<td>0.001</td>
<td>0.01</td>
</tr>
<tr>
<td>hidden layer size</td>
<td>[10,10]</td>
<td>[10,10]</td>
<td>[20,20]</td>
<td>[20,20]</td>
<td>[20,20]</td>
<td>[20,20]</td>
</tr>
<tr>
<td>number of iterations</td>
<td>10</td>
<td>100</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 5.3: Neural Network Hyperparameter Adjustment

While training the neural network model, we try to improve the model through hyperparameter adjustment based on the evaluation metrics. The 3 hyperparameters we choose to change are learning rate, hidden and epochs. Learning rate, as a regularization parameter, shrinks the contribution of predictors. Smaller learning rate results in less overfitting and slower convergence, so the model will be more stable. Hidden layer size means the number of hidden layers along with the number of nodes in each layer. Number of iterations is how many times the dataset will be iterated.
From figure 5.4 we can tell that the second run gives the best result. Its accuracy of 0.997 and AUROC of 0.901 are both high. As for the tradeoff, when false positive rate approaches 0.3, true positive rate gets close to 1 which is the highest among all six runs. We will also take the efficiency of each model into account to determine the best model setting. When we increase hidden layer size or number of iterations, there will be an obvious boosting in
job running time, so we prefer model2 with less number of hidden layers. Considering both accuracy and efficiency of the model, we conclude that the second configuration is the best in predicting click with our dataset.

5.3 Random Forest

<table>
<thead>
<tr>
<th></th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of trees</td>
<td>50</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>maximum depths of each tree</td>
<td>10</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 5.5: Random Forest Hyperparameter Adjustment

For random forest model, the hyperparameters we tune are number of trees and maximum of each tree. Maximum depths of each tree means the largest number of layers of each tree. As those two parameters increase, the model becomes more complex, and we want to check if the more complex model provides better result.
From figure 5.6, we find that the first run gives the best result. Even though it contains least numbers of trees among all three runs, and each tree is also not as deep as trees in the other two forests, it still provides an accuracy of 0.998 and an AUROC of 0.900, both of which are the highest compared to other two configuration settings.

### 5.4 Gradient Boosting Machine (GBM)

<table>
<thead>
<tr>
<th></th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of trees</td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Depth of each tree</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.1</td>
<td>0.05</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Figure 5.7: Gradient Boosting Machine Hyperparameter Adjustment
Similar as before, we change number of trees, depth of each tree and learning rate to improve the predictive power of the model. During this process, the model becomes more and more complicated each time with a slower rate of convergence.

<table>
<thead>
<tr>
<th>Confusion Matrix Table</th>
<th>Initial run</th>
<th>Second run</th>
<th>Third run</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 0 157</td>
<td>0 1 57 20</td>
<td>0 0 1 51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ROC</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>False positive rate</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>True positive rate</td>
<td>0.82</td>
<td>0.82</td>
<td>0.82</td>
</tr>
</tbody>
</table>

| AUROC                   | 0.7809384   | 0.8474252  | 0.8927621 |
| Accuracy                | 0.997       | 0.997      | 0.998     |

Figure 5.8: Gradient Boosting Machine Prediction Result

From the ROC plot and AUROC values, the last run outperforms all other runs. When false positive rate equals to 0.3, the third run gives highest true positive rate among all three, 0.82. From the confusion table, the same conclusion can be drawn since in the last run, the accuracy increases to 0.998.
5.5 Extreme Gradient Boost (XGBoost)

Same as the previous analyzing procedure, we train the model with hyperparameter modification. Three parameters to change are: number of iterations, maximum depth of each tree and eta. Eta controls the learning rate to prevent overfitting and make the boosting process more conservative.

<table>
<thead>
<tr>
<th></th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of iterations</td>
<td>10</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>maximum tree depth</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>eta</td>
<td>0.5</td>
<td>0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

**Figure 5.9: Extreme Gradient Boost Hyperparameter Adjustment**

For three runs, we gradually increase the number of iterations and maximum depth of trees, while decrease eta. From the AUROC values, there are not much difference between

<table>
<thead>
<tr>
<th></th>
<th>first run</th>
<th>second run</th>
<th>third run</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUROC</td>
<td>0.88666623</td>
<td>0.8856305</td>
<td>0.8818495</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.997</td>
<td>0.997</td>
<td>0.997</td>
</tr>
</tbody>
</table>

**Figure 5.10: Extreme Gradient Boost Prediction Result**

For three runs, we gradually increase the number of iterations and maximum depth of trees, while decrease eta. From the AUROC values, there are not much difference between
first two runs. However, the first run gives a better tradeoff as that when the false positive rate reaches 0.2, its true positive rate is around 0.8. As for the accuracy, there are more correct predictions being made in the first run than other two runs. In addition, the confusion matrix indicates that 19943 out of 20000 predictions are correct, which is the highest among all three.
CHAPTER 6

Result Comparison and Discussion

In this session we will put together all the prediction results we have to compare their predictive power. We choose the best performance of each model to compare.

6.1 Comparing Logistic Regression and Neural Network

The prediction results when applying logistic regression and neural network models indicate that:

1. both models reach a rate of accuracy as high as 0.997;
2. all six AUROCs from neural network are above 0.85 with a maximum value of 0.90, while the largest AUROC from logistic regression is less than 0.85;
3. with false positive rate of 0.4, neural network provides a true positive rate of almost 1 but logistic regression is only capable of 0.8.

Based on these three criteria, neural network has stronger predictive power.

6.2 Comparing Three Advanced Tree Algorithms

From prediction results of random forest, Gradient Boosting Machine and XGBoost,

1. three methods work equally well in generating accuracy as high as 0.997;
2. all three algorithms are capable of an AUROC of 0.89;
3. at a 0.8 true positive rate level, three models suffer from a similar false positive rate close to 0.3.
According to this comparison, three advanced tree algorithms are equally powerful in general.

### 6.3 Comparing All Five Models

Since the topic of this paper is to predict whether or not a click will occur, it is important not only to gain a high overall prediction accuracy, but also to capture more click events because they matter more as to drive profit for DSP’s. To be more specific, if a non-click is mis-predicted as click, no harm would come when including the corresponding user as a target of a certain type of ads. On the other hand, if we are unable to capture a click and predict it as non-click, it is highly possible that the DSP miss the chance to target a user who has a large probability to contribute clicks. This mis-classification will cause a potential loss for DSP. Therefore, to judge the effectiveness of all five models, it is crucial to compare accuracy in predicting the occurrence of clicks ignoring non-clicks.

![Figure 6.1: Model Effectiveness Comparison](image)

<table>
<thead>
<tr>
<th></th>
<th>Logistic Regression</th>
<th>Neural Network</th>
<th>Random Forest</th>
<th>GBM</th>
<th>XGBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>overall accuracy</td>
<td>0.997</td>
<td>0.997</td>
<td>0.998</td>
<td>0.998</td>
<td>0.997</td>
</tr>
<tr>
<td>accuracy for click</td>
<td>0.303</td>
<td>0.303</td>
<td>0.364</td>
<td>0.456</td>
<td>0.455</td>
</tr>
<tr>
<td>AUROC</td>
<td>0.849</td>
<td>0.901</td>
<td>0.900</td>
<td>0.893</td>
<td>0.887</td>
</tr>
</tbody>
</table>

From figure 6.1 we can tell that the Gradient Boosting Machine performs the best since it has the highest rate of both overall accuracy and accuracy for click event prediction. Also it has AUROC as high as 0.893, which is not much different from the highest AUROC value of 0.901. In contrast, logistic regression has the lowest overall accuracy, click prediction accuracy and AUROC, so among all five models, it is still the ideal one. This comparison result is reasonable since the algorithm for logistic regression is not as sophisticated as other four machine learning tools so that it might not be as powerful.
In summary, all four machine learning techniques work equally well through a criterion of overall accuracy of prediction, and they all beat the performance of logistic regression. Taking into account the accuracy to predict click instead of both click and non-click, the performance of Gradient Boosting Machine and XGBoost are better. If we further look into AUROC and consider the model effectiveness in general, Gradient Boosting Machine is the best.
CHAPTER 7

Conclusion

Machine learning as a data analysis method, keeps providing efficient software solutions to advertising industry. This paper focus on one instance of machine learning application in advertising, which is to accurately predict click event. Based on the data format and output type, we apply four models in supervised learning for binary classification which are neural network, random forest, gradient boosting machine and extreme gradient boost. In order to measure the effectiveness of machine learning techniques, we construct a commonly used model of logistic regression as a contrast.

From this analysis, we find out that machine learning is a powerful tool for click prediction. Among four machine learning methods, all of them work well with high accurate rate in general. When it comes to capturing the actual click events among a huge number of non-click instances, Gradient Boosting Machine performs the best.

In addition, as an advanced algorithm combining multiple layers of logistic regression unit, neural network is assumed to provide a better performance than a simple parametric logistic regression model; from the prediction results we have, this statement is proved to be true. In fact all four machine learning models beat the performance of logistic regression.
CHAPTER 8

Further Research Interest

Since XGBoost approach is an advanced version of Gradient Boosting Machine, it is interesting that from our model, GBM outperforms XGBoost. To further look into this, we need to come up with a metric to determine the significance of the difference between these two prediction result. If this difference is not statistically significant, then we can ignore it and conclude that these two models work equally well in click prediction.

Besides, if we implement those models on larger datasets with more variables, we need to add a step in advance called feature selection. We look into each individual predictor and test their correlation with the response variable; for those variables which are barely correlated with our output variable click, we remove them to make our models less complicated while keeping their efficiency. For logistic regression there is an index called information value. It gives each variable a score to indicate how much valuable information each one contains when used to predict the output. For machine learning tools, we can also filter out less contributed variables through a criteria-based procedure using AIC and BIC.

Also, in order to achieve the final goal of increasing profit for DSPs, our research focus will be “what to do next with our prediction result”. The major step to take is user group targeting. It requires unsupervised learning for clustering, which is the second main type in machine learning as mentioned in the introduction section of this paper. After assigning all users with various labels on their interests and preferences, we can finally provide a principle for DSPs to follow while doing the audience targeting.
REFERENCES


