Team Avazu Naïve Bayes Model for Advertisement CTR Prediction in Avazu Kaggle Competition 2014

Ian Chin, Samuel Lau, Shangyu Zhao
github.com/ucb-stat-157/Avazu

Abstract
As part of a class project and online competition, we produce a Naive Bayes model to predict the click through rate of digital advertisements. In creating our model, we compute general as well as engineered features. Our model results in 0.49 for log-loss over the test set and 0.43 over the private validation test.

Introduction
The Kaggle competition that ended February 2015 was one in which teams competed in predicting the whether advertisements on devices were clicked or not. The context of the competition revolves around the likelihood of users on a given device clicking on an ad that appears via the application being used or the website he or she is browsing. This likelihood, termed the click-through-rate of ads, is a common problem among businesses that generate revenue from advertisements being clicked on their app or site. By creating models that successfully predict whether users will click an ad given certain features or characteristics, these types of companies or organizations can increase the productivity and revenue of their ads.

Data (s3://stat157-uq85def/home/sclau/data/)
The data, which documents certain characteristics for each advertisement instance over 11 days, originated from Avazu, an international advertising program platform. The data is split into a training and test set: The training set, which covers the first ten days, includes whether the advertisement was clicked or not; the test set, which covers the final day, excludes this information. Each hour in the data contains roughly 200,000 instances, for a total of approximately 48,800,000 observations in the training data. The test data consists of 4,577,464 instances. For each instance, the data includes categorical information on the hour, banner position (the position of the ad), the site or app used, the device, as well as nine anonymized variables.

Process Overview
In creating a model, we split the training data into a sub-training set that includes the first eight days and a validation set that includes the ninth and tenth days of the training data.

As the validation set acts as a proxy for the test set during the training and tuning process of the model, it is important to construct a validation set that closely mimics the test set in structure. If the validation set has a structure very different to the test set, the model may not be as predictive. Since the train data are observations from Oct 21, 2014 to Oct 30, 2014 and the test data is from Oct 30, 2014, we naturally think it is most appropriate to construct the validation set as a whole day. In contrast, a random sampling method would shuffle the days, thereby failing to maintain the integrity and characteristics of a whole day. Taking into account that having only

any one day from the train to be the validation set may be arbitrary and biased, we decide to use the last two days of the training data to be our validation.

As mentioned above, the training data has around 200K observations each hour. After examining the click-through-rate for the sub-train and validation sets, (which is 0.17 for both), we believe that our validation set resembles the sub-train and test.

General Features - Categorical Variables

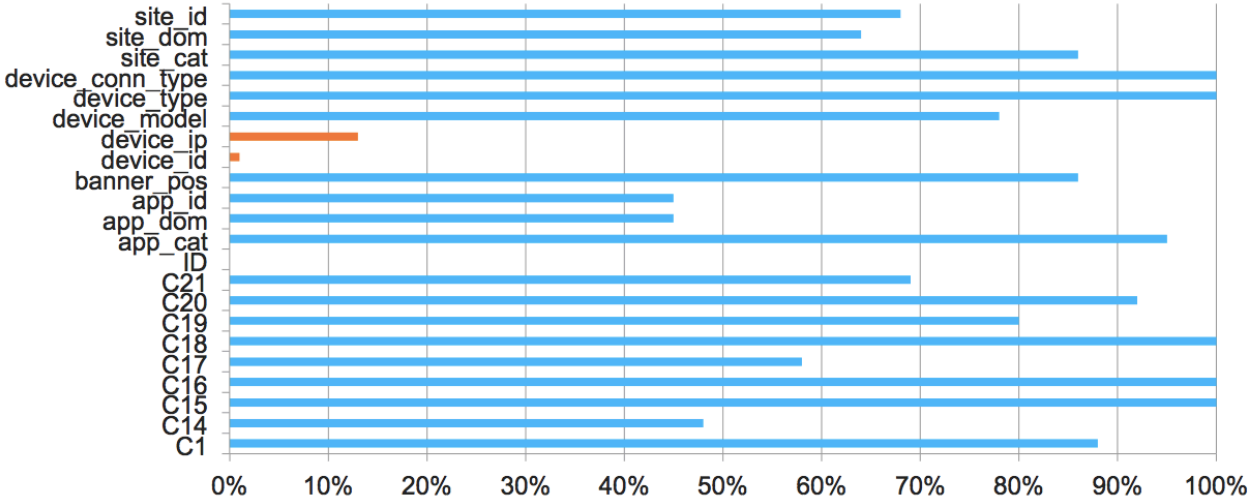
There are 24 variables in the train data and 23 variables in the test (where the test data excludes the click/no click column).

- id: identifier
- Click: 0/1 for non-click/click
- Hour: YYMMDDHH
- C1: Anonymized
- Banner_pos: Position of the advertisement
- Site_id: id of the website where ad is shown
- Site_domain: domain of the website where ad is shown
- Site_category: category of the website where ad is shown
- App_id: id of the app where ad is shown
- App_domain: domain of the app where ad is shown
- App_category: category of the app where ad is shown
- Device_id: id of the device on which the ad is shown
- Device_ip: the ip address of the device
- Device_model: model of device (eg. iphone4, iphone5, etc)
- Device_type: type of the device (eg. computer, phone, tablet, etc)
- Device_conn_type: website connection method
- C14 -C21: Anonymized

Notice that in any one observation, the site feature and app feature is exclusive to each other, because the viewer can only either see the ad from the browser or the app. So when the viewer sees the advertisement from the app, the site features should show 'Null'. However, since the data is all coded to conceal the linguistic information, we cannot directly identify the Null values. Using the fact that site features and app features are exclusive to each other, we successfully identified the Null values by counting which site and app features show up the most and whether the sum of them equals to all the observations.

For all the variables, we checked the overlap between the train and the test and plotted the overlap percentage [figure 1]

As shown in the graph, device_id and device_ip have a very limited overlap, whereas some of the other features have close to full overlap.



[figure 1]

Engineered Features

Advertisement Size:

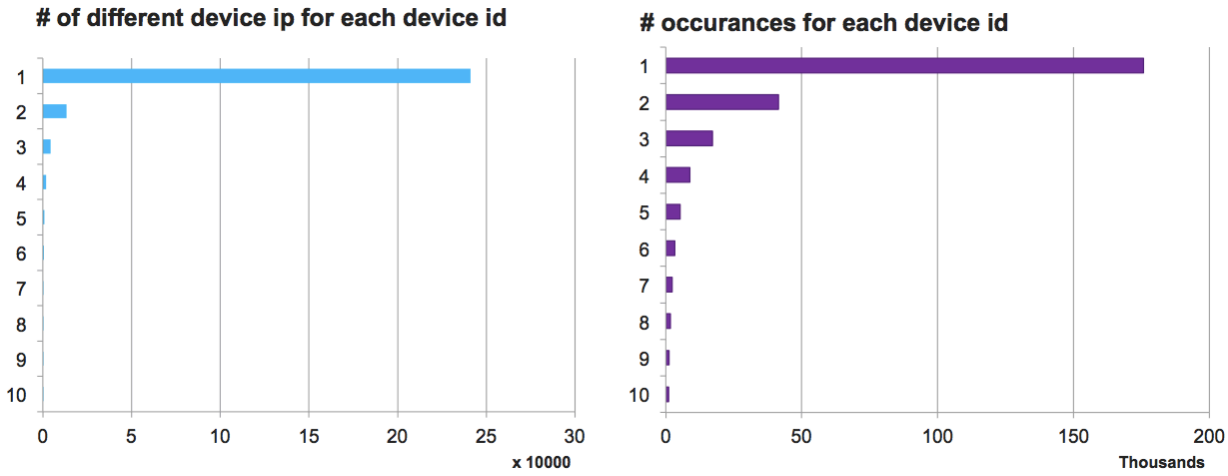
Each instance includes two variables on the height and width of the advertisement. Rather than treating these two variables independently, we choose to categorize the advertisement sizes into the different possible height/width combinations. In total, there are ten observed combinations of ad sizes in the training data, nine of which are observed in the test data.

Number of Unique Device IPs for Each Device ID

In attempting to examine the user activity level in predicting the click through rate, we use the number of unique device IPs observed for each device ID in a given day as a proxy. With a higher number of unique device IPs for a given day, the feature measures how many different locations the device is being used at over a given day. The distribution of the feature is presented in the left panel of Figure 2. There are not surprisingly many device IDs attributed with one device IP. Nonetheless, there are quite a few device IDs that are attributed to well over one device IPs.

Number of Unique Device ID Observations

Another feature that attempts to proxy for user activity is the more direct approach of measuring how many times a given device ID is observed in a day. A device ID that has a high number of observations is intuitively more active. The distribution of the feature is shown in the right panel of Figure 2. The distribution is highly similar to that of the interaction between device IPs and IDs.



[Figure 2]

Naive Bayes

The model we use in predicting the click through rate is a Naive Bayes model, which calculates the probability of a instance being clicked conditional on the features that are included using Bayes Theorem. It assumes independence of features in its calculation, a strong assumption that may not always be the case.

$$p(C|F_1,\ldots,F_n) = \frac{p(C)\ p(F_1,\ldots,F_n|C)}{p(F_1,\ldots,F_n)}$$

In training our model, we test a few different models with different features included. Although our full model includes all the features, most of our models include only a subset of variables that we believe are more relevant in predicting the click through rate.

Unknown Threshold

Due to data limitations, we are bound to observe certain categories of features that appear rarely in the training data or that appear in the test data but not at all in the training data. If untreated, these categories' conditional probabilities would compute to either 0 or very close to 0, swinging our prediction rates under the false assumption that our data completely captures these rare categories' attributes. To deal with this issue, we group, for each feature, the categories that appear less times than a threshold parameter that we set. This group of categories is then treated as its own category, with the number of instances being the sum of these rare categories' instances. For any feature category in the test data that is unobserved in our training set, we categorize this observed test category as the feature's "strange" category.

Smoothing Parameter

When a given category in a feature is observed to have only clicks attributed to it and zero no-clicks, or vice versa, the formula for Naive Bayes would either make the numerator 0 or the numerator equal to the denominator, resulting in a strong prediction of either 1 or 0. Rather than making such strong (and likely false) predictions, we utilize an additive smoothing parameter to prevent the conditional probability of either a click or no-click given the category from

becoming 0. That is, we use the following smoothing parameter formula when computing each category's conditional probabilities:

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d)$$

where alpha is set to 1 in our model, and d is the unique number of categories in a given feature.

Evaluation of Model Results: log-loss, kaggle submission and leaderboard

In evaluating the results of our model, we use the following log-loss function, which is the criteria required by Kaggle:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Note that a prediction of 0.5 for each instance would result in a log-loss of 0.69. Our full model, which includes all the available features results in a log-loss value of over 0.75, varying with the unknown threshold parameter. We believe the result for our full model is poor due to overfitting of the predictions. In moving forward, we include only a subset of variables we think are the most relevant. The results for each model over the validation set are summarized below. Using Model 3 in predicting the actual test data, we obtain a value of 0.49, and is posted on the Kaggle leaderboard under "Stat-157," with a position of 521.

Model	Site_cat	App_cat	Banner_position	Device_type	ad_size	device_ip	user activity	Result
1					---	---	---	0.436
2						---	---	0.432
3							---	0.430
4								0.544

Difficulties in the Project

- Brainstorming features to engineer, lack of "similarity index"
- Identifying the "null" values from coded data
- Identifying anonymized variables
- Overfitting occurs when include too many variables

Conclusion

In our project, we run a Naive Bayes model to predict the click through rate of digital advertisements over a full day using Avazu data. Our best model is a rather simple model, including only a few core variables that seem to be most attributed to predicting whether a user clicks an ad or not. Under the validation set that we created, we get a log-loss score of 0.430, and a value of 0.49 under the test set. Given more time, we could potentially improve our model by brainstorming better engineered features or perhaps trying a different model such as decision tree or logistic regression.