Challenge Report

Mozhdeh, Naghmeh, Rozita, Samim

January 2015

Abstract

In this report we tried to find a model to fit the Kaggle competition for Avazu Click-Through rate prediction. First we explain how to load the data and what attributes to choose for building models. Then we built a number of models and evaluated them. In the end we found that the Naïve Bayes is the best model that fits our data.

1 Loading data into R

We took the first 5000 records of the original data as a CSV file and read it into R.

```
ds = read.csv("new_train.csv")
ds[] <- lapply(ds, factor)
target <- "click"
```

After some inspection we found that we can ignore some attributes. `hour` attribute is constant in this portion of data.

```
summary(ds$hour)

:     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
: 14100000 14100000 14100000 14100000 14100000 14100000
```

So we decided that this is not good portion of data and all of our models may be wrong or inaccurate. So we took first four million records of data using `ff` package of R. It took about half an hour to load the data.

```
library(ff)
table <- read.csv.ffdf(file = 'train.csv', nrows = 4000000)
ds <- data.frame(table)
```

Some of the models were so heavy that our computers could not use them with so many records of data. Therefore some models were evaluated using the 5000-records data, and some were evaluated using the 4-million data.

2 GBM

We used *Gradient Boosting* to build a sense of how important each attribute is so that we can ignore it in our models. Before using gradient boosting we ignore `device_ip` and `id`, as we concluded that they are not relevant to the target variable. This model were evaluated using the 5000-record data.

```
ignore <- c(
    "id",
    "device_ip",
    ) # Coloumns to ignore
vars <- setdiff(names(ds), ignore)
inputs <- setdiff(vars, target)
form <- formula(paste(target, "~ ."))
nobs <- nrow(ds)
train <- sample(nobs, 0.7*nobs)
test <- setdiff(seq_len(nobs), train)
actual <- ds[test, target]
data <- ds[train,vars]
```

Then we build our model by using `gbm` package.

```
library(gbm)
GBM_model = gbm(form,data = data,n.trees = 10000,distribution = "gaussian", cv.folds=2)
summary(GBM_model)

                             var      rel.inf
device_model        device_model 91.112123429
site_id                  site_id  5.990171556
C14                          C14  2.014844931
device_id              device_id  0.871112833
site_domain          site_domain  0.006012711
app_id                    app_id  0.005734540
C1                            C1  0.000000000
banner_pos            banner_pos  0.000000000
site_category      site_category  0.000000000
app_domain            app_domain  0.000000000
app_category        app_category  0.000000000
device_type          device_type  0.000000000
device_conn_type device_conn_type  0.000000000
C15                          C15  0.000000000
C16                          C16  0.000000000
C17                          C17  0.000000000
C18                          C18  0.000000000
C19                          C19  0.000000000
C20                          C20  0.000000000
C21                          C21  0.000000000
```

3 Data Pre-processing

By looking at how important each attribute is, we decided to ignore less important attributes to make the models more accurate.

```r
ignore <- c(
    "hour",
    "id",
    "device_ip",
    "C1",
    "banner_pos",
    "site_category",
    "app_domain",
    "app_category",
    "device_type",
    "device_conn_type",
    "C15",
    "C16",
    "C17",
    "C18",
    "C19",
    "C20",
    "C21"
    ) # Coloumns to ignore
vars <- setdiff(names(ds), ignore)
inputs <- setdiff(vars, target)
form <- formula(paste(target, "~ ."))
nobs <- nrow(ds)
train <- sample(nobs, 0.7*nobs)
test <- setdiff(seq_len(nobs), train)
actual <- ds[test, target]
data <- ds[train,vars]
```

Here is the structure of the data after data pre-processing.

```r
str(data)

 $ click       : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 2 2 1 1 ...
 $ site_id     : Factor w/ 276 levels "02d5151c","030440fe",..: 32 150 32
 $ site_domain : Factor w/ 234 levels "00e1b9c0","0150cc3e",..: 222 188 22
 $ app_id      : Factor w/ 210 levels "00848fac","03528b27",..: 197 39 197
 $ device_id   : Factor w/ 574 levels "004270bf","017c59a6",..: 395 26 395
 $ device_model: Factor w/ 841 levels "00b08597","00b1f3a7",..: 353 563 10
 $ C14         : Factor w/ 216 levels "375","377","380",..: 52 193 47 110
```

Let us explore our data a little. Displaying distribution of data based on $\text{site}_{\text{category}}$ for all data and clicked data. For both all data and clicked data major site category is 28905ebd:

```r
table(ds$site_category)

0569f928 110ab22d 28905ebd 335d28a8 3e814130 50e219e0 72722551 75fa27f6
      35        1     1909       57      604     1244       12       11
76b2941d a818d37a bcf865d9 c0dd3be3 f028772b f66779e6
     116        1        1        3      994       12
```

Displaying distribution of data based on $\text{app}_{\text{category}}$ for all data and clicked data. For both all data and clicked data major app category is 07d7df22:

```
table(ds$app_category)


 07d7df22 09481d60 0f2161f8 4ce2e9fc 75d80bbe 8ded1f7a cef3e649 d1327cf5
     3955        1      751        4        6       66       70        5
 f95efa07 fc6fa53d
      141        1
```

4 SVM

This model were evaluated using the 5000-record data. First we can use the tune function to determine our constants in using SVM.

```
library(e1071)
tuned <- tune.svm(form, data = data, gamma = 10^(-6:-1), cost = 10^(1:2))
summary(tuned)

Parameter tuning of svm:

- sampling method: 10-fold cross validation

- best parameters:
 gamma cost
 1e-06   10
```

Using the constants above we can train our model.

```
model  <- svm(form, data = data, gamma = 10^(-6:-1), cost = 10)
```

Here is the confusion matrix of our model

```
svmPred <- predict(model, ds[test,vars])
tab <- table(pred = svmPred, true = ds[test,target])
print(tab)

     true
 pred    0    1
    0 1244  256 a b
    1    0    0 c d
```

These are the results of confusion matrix, which shows that this model has high accuracy but very low precision, so makes the model not ideal.

$$TP = d/(c+d) = 0$$

$$FP = b/(a+b) = 256/1244+256 = 0.17$$

$$TN = a/(a+b) = 0$$

$$FN = c/(c+d) = 0$$

$$AC = (a+d)/(a+b+c+d) = 1244/1244+256 = 0.82$$

$$P = d/(b+d) = 0$$

5 Naïve Bayes

This model were evaluated using the 4-million-record data.

```
library(e1071)
classifier <- naiveBayes(data[train, vars], data[train, target])
table(predict(classifier, data[test, vars]), data[test, target])

         0      1
  0 693115    308    a b
  1    953 145381    c d
```

These are the results of confusion matrix, which shows that this model high accuracy and can be accepted. It has both high precision and accuracy together.

$$TP = d/(c+d) = 145381/146334 = 0.99348$$

$$FP = b/(a+b) = 308/693423 = 0.00044$$

$$TN = a/(a+b) = 693115/693423 = 0.99955$$

$$FN = c/(c+d) = 953/146334 = 0.00651$$

$$AC = (a+d)/(a+b+c+d) = 838496/839757 = 0.99849$$

$$P = d/(b+d) = 145381/145689 = 0.99788$$

6 kNN

This model were evaluated using the 5000-record data.

```
library(RWeka)
classifier <- IBk(form, data = data, control = Weka_control(K = 2, X = TRUE))
evaluate_Weka_classifier(classifier, numFolds = 10)

=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances        2866               81.8857 %
Incorrectly Classified Instances       634               18.1143 %
Kappa statistic                          0.0876
Mean absolute error                      0.2578
Root mean squared error                  0.379
Relative absolute error                 90.5826 %
Root relative squared error            100.4847 %
Coverage of cases (0.95 level)          96.9429 %
Mean rel. region size (0.95 level)      85.3143 %
Total Number of Instances             3500

=== Confusion Matrix ===

    a    b   <-- classified as
 2811   88 |    a = 0
  546   55 |    b = 1
```

These are the results of confusion matrix. It has good accuracy but a low precision.

$$TP = d/(c+d) = 55/(546+55) = 0.09$$

$$FP = b/(a+b) = 88/(2811+88) = 0.3$$

$$TN = a/(a+b) = 2811/(2811+88) = 0.96$$

$$FN = c/(c+d) = 546/(546+55) = 0.90$$

$$AC = (a+d)/(a+b+c+d) = (2811+55)/(2811+88+546+55) = 0.81$$

$$P = d/(b+d) = 55/(88+55) = 0.38$$

7 Decision Tree

This model were evaluated using the 5000-record data.

```
library(party)
ctree <- ctree(form , data=data)
table(predict(ctree) , data$click)

        0    1
   0 2899  601 a b
   1    0    0 c d
```

These are the results of confusion matrix. It has a some how good accuracy but a very low precision.

$$TP = d/(c+d) = 0$$
$$FP = b/(a+b) = 601/(2899+601) = 0.17$$
$$TN = a/(a+b) = 2899/(2899+601) = 0.82$$
$$FN = c/(c+d) = 0$$
$$AC = (a+d)/(a+b+c+d) = (2899+0)/(2899+601) = 0.82$$
$$P = d/(b+d) = 0$$

8 Loss Function

As Kaggle wanted, all the models and prediction should be evaluated against the logarithmic loss function. After building our models we found out that Kaggle wanted the prediction for every click as a probability. To be able to have probabilities we should see our target variable as a numerical variable. But unfortunately we built our data and model using our target variable as a binary variable. So we cannot calculate the logarithmic loss of our models.

9 Conclusion

The best model in these models were the Naïve Bayes model. We choose this model as the best model based on its high accuracy and precision together. And also because of its good time costs of algorithms as we were able to build the model using our four-million-record data.