Radim Kašpárek, Samuel Kožuch,
Štěpán Svoboda, Jan Hynek
Quantitative Marketing

Assignment: CTR prediction

1 Introduction

In this assignment we compared the performance of two different methods for predicting CTR. The first one used was Naive Bayes, the second was logistic regression. Because of the size of the data, both algorithms were modified to allow for out-of-core (*online*) learning. It was confirmed that both algorithms perform very well, however, the Naive Bayes was significantly faster. It also does not necessarily need the one-hot-encoding of categorical variables, therefore it is possible to use online learning without much workaround. In the end we used the one-hot-encoding for ease of implementation.

2 Dataset

The dataset used was obtained from Kaggle, it was used in 2014 in a competition aimed at prediciting the click through rate (CTR) of Avazu advertisments. The data file contained 45M rows and 22 columns, detailed description of the columns is under this link. The dataset was ordered by time, all columns were of type 'text' and the format was '.csv'.

3 Methodology

3.a Data preparation

At first, the dataset was shuffled. As it is reasonable to assume that the click through rate varies between days, it is necessary to shuffle the dataset if the online learning should be used. This was done by using Linux bash command 'shuf'. As the next step, the columns were turned into integers representing labels, as all the variables were categorical. Secondly, the 'hours' column containing the timestamp was converted into column 'hours_clean' containing only the hour (the information about year and day was stripped of). Thirdly, the correlation between variables was examined. It was found that there are two pairs of nearly-perfectly correlated variables, therefore one in both these pairs was deleted (C1 and C14 were deleted). Last but not least, the id column was deleted since it has no predictive value and we don't necessarily need it as all the algorithms used preserve the ordering of the data.

3.b Expected CTR

The aim is to predict the CTR of the ad. Therefore, for every observation the probability of a click was predicted (this was done using method 'predict_proba' in the sklearn.SGDClassifier). It was decided to predict the probability of an event occurring (click) instead of a simple binary label prediction since this is a standard approach in machine learning. Having predicted the probability of a click for every observation in the train set, these were averaged and the result was interpreted as the expected CTR of an ad.

The loss function used for benchmarking the models is simply the difference between the expected (predicted) CTR and the true CTR of the test set. As it was decided to use the aforementioned approach to predicting CTR, it does not really matter how the algorithm predicts classes for specific observations, the only metric of interest is the expected (predicted) CTR.

Therefore, the loss function used for benchmarking the models reflects this lack of interest in predictions for specific observations and utilizes only the expected CTR.

3.c Logistic Regression

The logistic regression requires that the categorical variables are one-hot-encoded. However, this poses a problem when working with online learning algorithms as it is very unlikely that all possible values are encountered in the first batch. Current method used to deal with this problem is feature hashing, however, we did not implement this as it would be very time-consuming due to the lack of experience with this method. We decided to use a subset of data, specifically a datafile containing 4.5M randomly chosen rows. These were one hot encoded via sklearn.preprocessing.OneHotEncoder class and resulting dataset was saved as a sparse matrix. This sparse matrix was later divided into 45 equally large parts and the logit model was gradually trained on these parts. Specifically, the model was trained on 269 (6 * 45 - 1) subsamples, randomly sampled with replacement, the last subset of 100K rows was used as the test sample. After each iteration the CTR for the test sample was predicted and saved. Finally, these predicted CTRs were plotted. We used the mean of predicted probabilities of click as the estimated CTR.

The logit model was specified as follows. The algorithm used was sklearn.SGDClassifier with loss function set to 'log'. The penalty function used was 'l1' as it leads to sparse solutions (most coefficients are driven to zero). This preference of sparse solutions is reasonable as it substitutes pruning. The alpha parameter, the constant that multiplies the regularization term, was kept at the default value 0.0001. The learning rate was set to 'optimal' (default setup), i.e. $\eta = 1.0/(\alpha * (t + t_0))$, where $t = n\_samples * n\_iter$ and t_0 is chosen based on heuristic proposed by Leon Bottou (Bottou, 2004). Finally, the parameter 'warm_start' was set to true to allow for online learning.

3.d Naive Bayes

In our case we can choose from two implementations of Naive Bayes algorithm, the so-called Bernoulli and Multinomial Naive Bayes. The Bernoulli one uses one-hot-encoding and processes the data in the same way as logit. The Multinomial Naive Bayes uses the counts of our categorical variables and needs to be aggregated before it is used. For simplicity and ease we used the same data preparation for both cases and used only the Bernoulli Naive Bayes.

The CTR plot and predicted CTR are done in the same way as in the logit model described above.

The model itself was trained using Python's $sklearn.naive\_bayes.BernoulliNB$. We didn't use any prior since we have enough data and not enough prior knowledge/experience to set a reasonable prior. We also used the default additive smoothing parameter.

4 Results

4.a Logistic Regression

The logistic regression showed convergence approximately after 150 iterations. The colour on the plot denotes 45 iterations ($\sim$ one epoch). As can be noticed, the benefit of the last $\sim$ 90 iterations is not as high. This was expected, as the number of unique subsamples is only 45 and therefore the information in these is probably already utilized after $\sim$ 180 iterations. This

approach is also rather time consuming as the 180 iterations take 1min 31s ± 3.9 s (mean ± std. dev. of 5 runs).

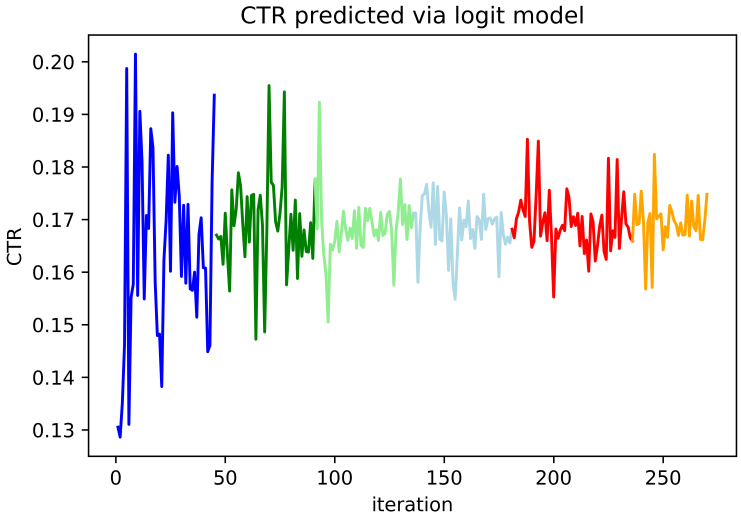

Figure 1: CTR predicted by logit model.

4.b Naive Bayes

The results for Naive Bayes became after about 50 iterations nearly constant and even before were very close to the actual values. Naive Bayes is thus much less time-consuming since we do not need to train it for as long even though the actual training time for the entire 180 iterations was close to the logistic regression. The real time-saver is the fact that we are good to go nearly from the beginning since even the first training run is pretty much spot-on in comparison with logit.

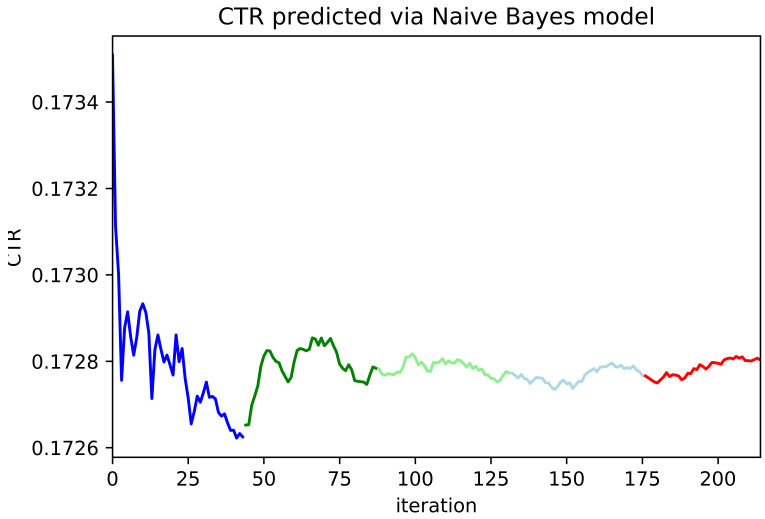

Figure 2: CTR predicted by Naive Bayes

4.c Comparison

We can see from the two comparison figures that logit once trained performs somewhat better but overall more volatile but Naive Bayes works well from the beginning and delivers consistent and precise results even though in the long run doesn't do better than the logit.

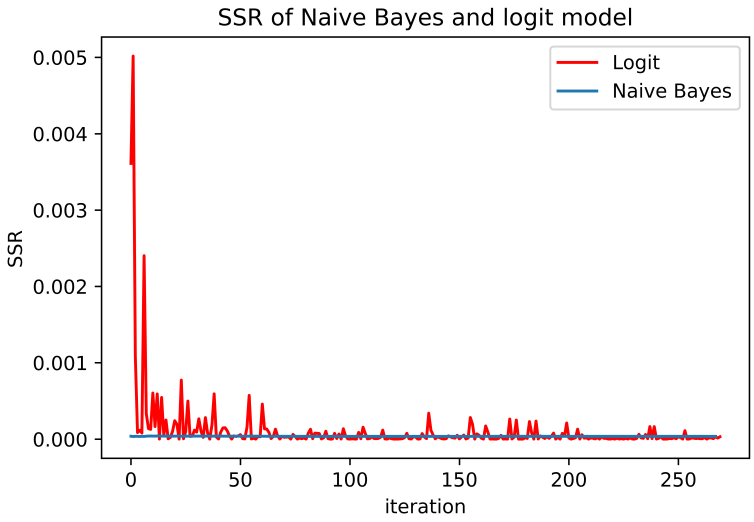

Figure 3: Squared difference in CTR predicted for the test set by Naive Bayes (Logistic regression) and the true CTR in the test set

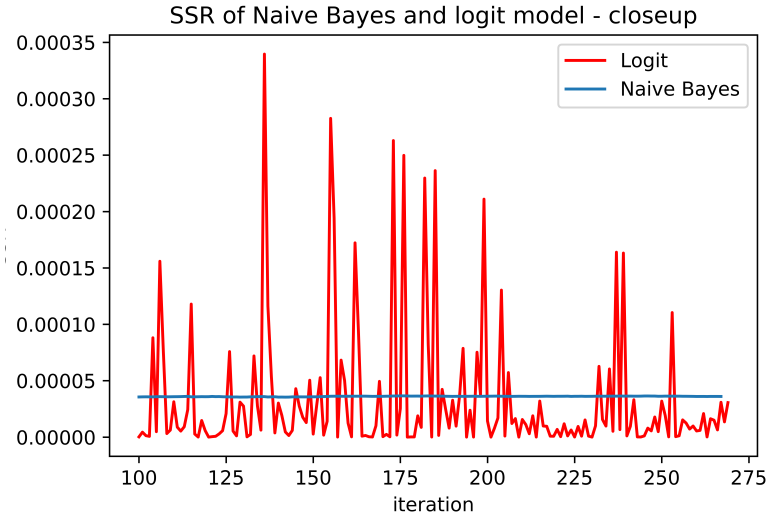

Figure 4: Squared difference in CTR predicted for the test set by Naive Bayes (Logistic regression) and the true CTR in the test set

5 Conclusion

We would like to conclude with a quick summary of our work and a comparison of the used methods. We tried to utilise all the data at our disposal and thus dropped only variables

showing high multicollinearity. Limited by computational and time constraints we decided to use only a subset of our data and took 10% of the shuffled dataset. The convergence visible in the results supports our decision.

Regarding the comparison of the two methods we can see that Naive Bayes gives us very precise results nearly immediately (the error is of order 0.01%). The SGD approach is very different since it takes a lot of time to converge and even then we can see a lot of noise in comparison with NB (the error is of order 0.1%). So even though the actual training phase of Naive Bayes is of comparable length to SGD we can save substantial time by using the data set fewer times or possibly only once which could lead in the case of SGD to very imprecise results.

References

Bottou, L. (2004). Stochastic learning. In *Advanced lectures on machine learning*, pages 146–168. Springer.