COEN 281 Patt Recog & Data Mining

# Project Report

## Problem: Outbrain Click Prediction
## Goal: Create high accuracy model for click prediction

Group 8
Team Member: Tzuping Chen, Wen-Chieh Tai, Hsuan-Chih Chen, Wen-Han Chang
Project Source Code: https://github.com/hschen0307/DM

### I. Introduction

Predicting advertisements click is a large-scale learning problem that is the core of the several-billion online advertising business. Google, Facebook, and other top-level Internet advertising companies are striving to improve their advertising system and make more profit day by day. Currently, Outbrain, the web's leading content match platform, presented a motivating challenge on Kaggle. Outbrain pairs relevant content with interested readers in about 250 billion personalized advertisement every month across several thousands of sites. In this competition, participators are asked to predict which advertisement in a content are likely to be click on by its viewer. Improving Outbrain's advertising algorithm will help users more satisfy Outbrain's recommender system.

### II. Dataset for this problem

The dataset includes a sample of users' page views and clicks, as observed on multiple publisher sites in the United States between June/14/2016 and June/28/2016. Each viewed page, clicked advertisements, and other information related to content are recorded in these documents.

The dataset consists of numerous sets of content which is served to a specific user in a specific context. Each context is given a display_id. In each such set, the user has clicked on at least one advertisement. The answers of the clicked advertisement in the test set are not given. Our task is to predict the clicked advertisement in each context.

Each user in the dataset is assigned a unique id (uuid). A user can view a document (document_id), which is a web page. On each document, a set of ads (ad_id) are displayed. Each ad was created by a campaign (campaign_id) pushed

by an advertiser (advertiser_id).

**page_views.csv** is a log of users visiting documents.

- ✧ uuid
- ✧ document_id
- ✧ timestamp (ms since 1970-01-01 - 1465876799998)
- ✧ platform (desktop = 1, mobile = 2, tablet =3)
- ✧ geo_location (country>state>DMA)
- ✧ traffic_source (internal = 1, search = 2, social = 3)

**clicks_train.csv** is the training set, showing which of a set of ads was clicked.

- ✧ display_id
- ✧ ad_id
- ✧ clicked (1 if clicked, 0 otherwise)

**promoted_content.csv** provides details on the ads.

- ✧ ad_id
- ✧ document_id
- ✧ campaign_id
- ✧ advertiser_id

**documents_meta.csv** provides details on the documents.

- ✧ document_id
- ✧ source_id (Domain Name, e.g. www.bbc.com)
- ✧ publisher_id
- ✧ publish_time

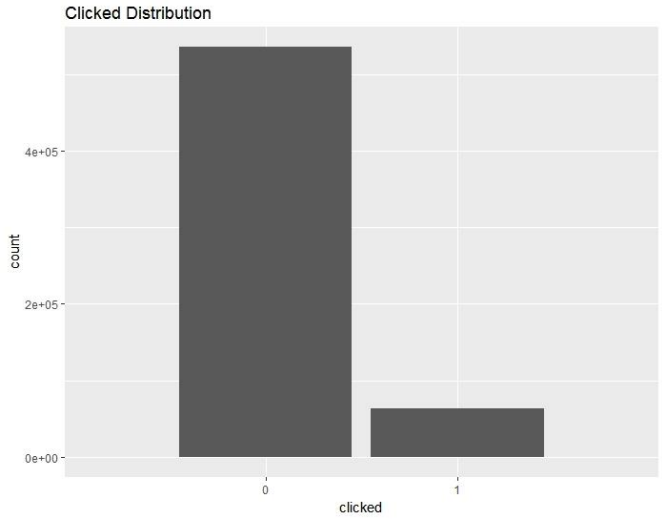
**documents_topics.csv** provide Outbrain's confidence level between document and topic in each respective relationship.

- ✧ document_id
- ✧ topic_id
- ✧ confidence_level

## III. Data Pre-Processing

We utilize database to merge several datasets into a single table to reduce the complexity of data. To realize this, we store and insert the all data into a database and then connect the database in R for further processing. We also shrink the dataset down to 600000 rows to prevent memory crash in R. Last, we

found out the dataset is unbalance since too many values are 0 in the clicked attribute. To solve this problem, we create a dataset with 50% of clicked and 50% of non-clicked data from the original dataset. The final dataset has around 160000 rows. You can see our implementation in the code.




From above image, we can see that the clicked attribute is unbalanced in the original dataset.

## IV. Data Analysis

In general, we try different combinations of features to figure out which feature is highly correlated to results. For K-NN, our baseline model, we keep it simple by using the popularity of each advertisement to predict the result.

## V. Training & Validation preparation

We divided training data set to a smaller one (80% training set / 20% validation set) for validation. The validation set is used to evaluate the performance of a specific model and compare it with other same type's models to determine hyperparameters and model properties.

## VI. Models that we implemented

### Baseline models

We choose **K-NN**, an easy understanding and implementation model, for our baseline model which helps us measure how much accuracy is improved by our advanced models. We also build a trivial model which makes predictions by random to see whether K-NN works or not.

### Advanced models

We adopt **Logistic regression** and **Random forests** algorithm for our advanced

models. Logistic regression is a regression model which outputs probabilities of predictions from a bunch of categorical variables by using a logistic function. Random forests algorithm is an ensemble method which enhances prediction accuracy by constructing multiple decision trees at training time and making strong predictions through voting or linear combination of predictions from each tree.

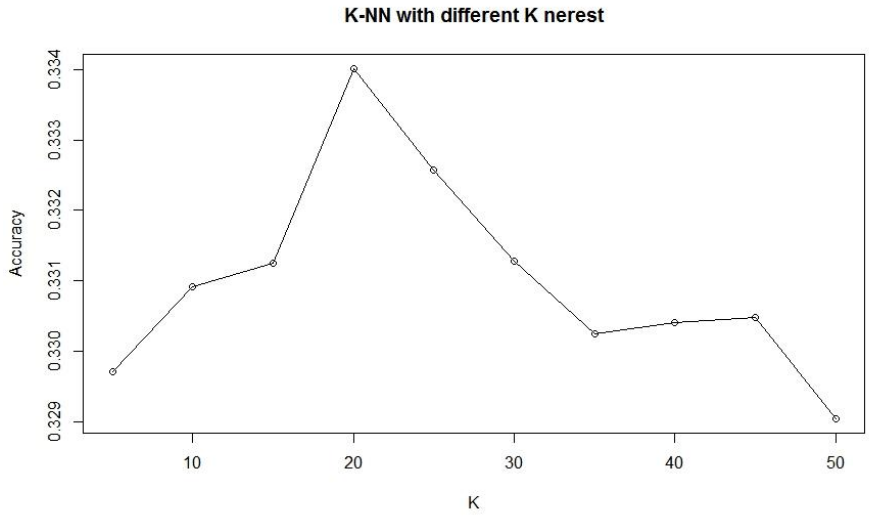
## VII. Result

### Baseline models

| Method | Feature Used | Accuracy |
|---|---|---|
| Random Guess | none | 23.94% |
| K-NN | clicked record and view | 33.40% |

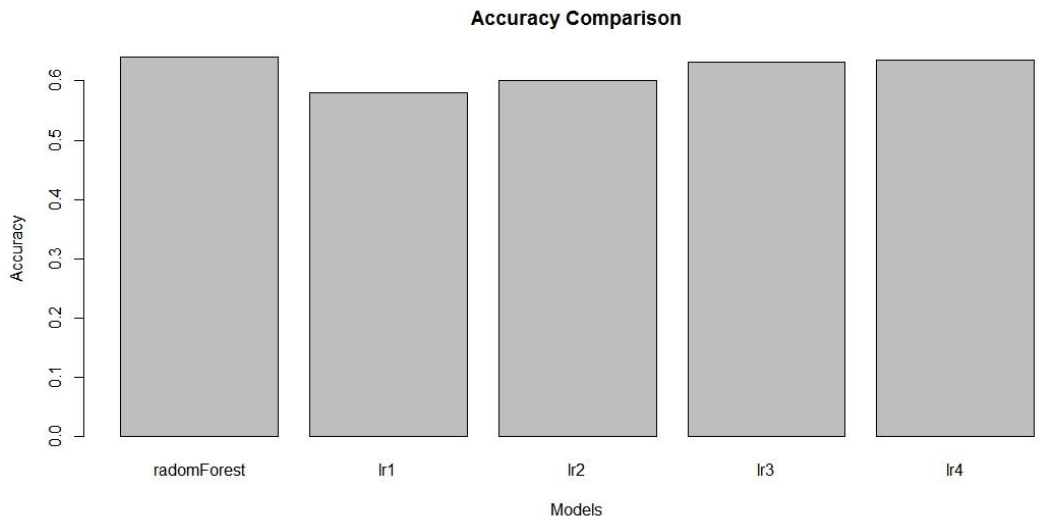
From above table, we can see that K-NN model works well and improves about 10% upon Random guess.

To find the optimal hyperparameter K, we give different values to K. As we can see from the following Image, the optimal value for K is 20.



**Advanced models**

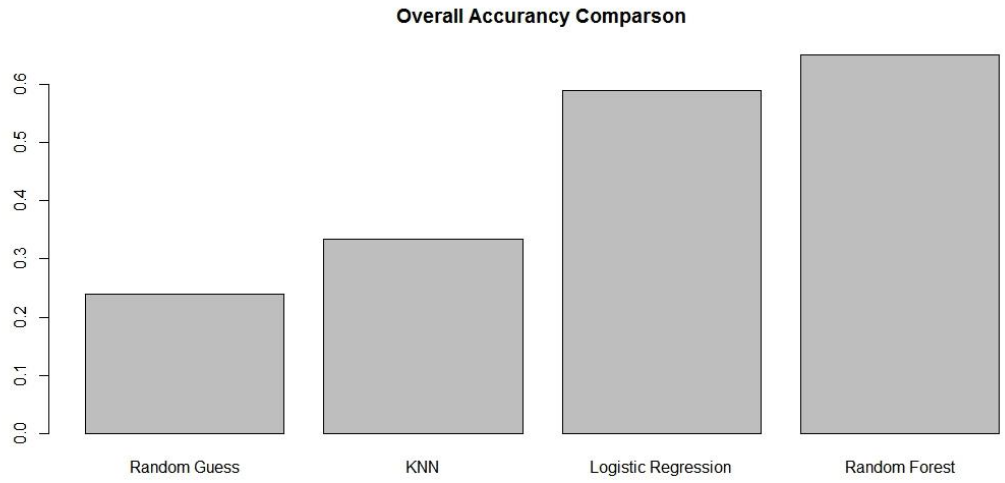
| Method | Feature Used | Accuracy |
|---|---|---|
| Logistic Regression 1 | clicked and confidence_level | 57.92% |
| Logistic Regression 2 | clicked, confidence_level, and topic_id | 60.00% |
| Logistic Regression 3 | clicked, confidence_level, topic_id, and advertiser_id | 63.09% |
| Logistic Regression 4 | clicked, confidence_level, topic_id, advertiser_id, and campaign_id | 63.47% |
| Random Forests | clicked, confidence_level, and topic_id | 64.78% |




From above image, we can see that all advanced models perform better than the baseline models. In addition, Random forests have the highest accuracy among the advanced models.

**Overall Comparison**

| Method | Accuracy |
|---|---|
| Random Guess | 23.94% |
| K-NN | 33.40% |
| Logistic Regression | 57.92% |
| Random Forests | 64.78% |




From the graph of overall accuracy comparison, we can see that Logistic regression and Random forest work well than Random guess and K-NN algorithm. Moreover, Random forest still occupied the top position.

## VIII. Lesson that we learnt from this project

First, we learnt how to merge several datasets into one dataset to reduce the complexity of data. To do this efficiently, we learnt to use database to process datasets and connect it in R. We also learnt to adopt over-sampling technic to fix the unbalanced datasets. For data analysis, although we found that we couldn't do too much in this part, we still try our best to figure out the relations between features and predictions. For validation part, we practiced the technic learnt from class which is separating a single dataset to two smaller datasets, one for training and the other one for validation. For modeling, we learnt the details of implementing several machine learning algorithms that we acquired in class. Overall, we greatly appreciated obtaining this opportunity to exercise what we learn from class because practice makes perfect.

## IX. Reference

Python - sklearn library
(http://scikit-learn.org/stable/modules/classes.html)
Python - scipy library
(https://docs.scipy.org/doc/scipy/reference/)
R - Logistic Regression
(https://www.tutorialspoint.com/r/r_logistic_regression.htm)
R - Random Forest
(https://www.tutorialspoint.com/r/r_random_forest.htm)

Database - DIY: A PostgreSQL database server setup anyone can handle (http://www.techrepublic.com/blog/diy-it-guy/diy-a-postgresql-database-server-setup-anyone-can-handle/)

Database - Import CSV file into PosgreSQL Table (http://www.postgresqltutorial.com/import-csv-file-into-posgresql-table/)

K-Nearest Neighbors Algorithm (https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

Logistic Regression (https://en.wikipedia.org/wiki/Logistic_regression)

Random Forest Algorithm (https://en.wikipedia.org/wiki/Random_forest)