Short summary report

Goal

The goal of the task is successful prediction of the CR (marked with 1 in the ‘label’ column of the datasets).

Dataset

Three datasets have been provided (2 campaigns, 1 creative).

Alias	Type	Filename	Impressions	Conversions	Ratio [%]
DS1	Campaign	campaign_768874	828065	2342	0.0028
DS2	Campaign	campaign_821471	268710	6230	0.0232
DS3	Creative	creative_5188417	454731	10323	0.0227

Exploratory analysis

All code for exploratory analysis, various plots, tests, benchmarking is contained in *‘exp_analysis.ipynb’* at corresponding github repo.

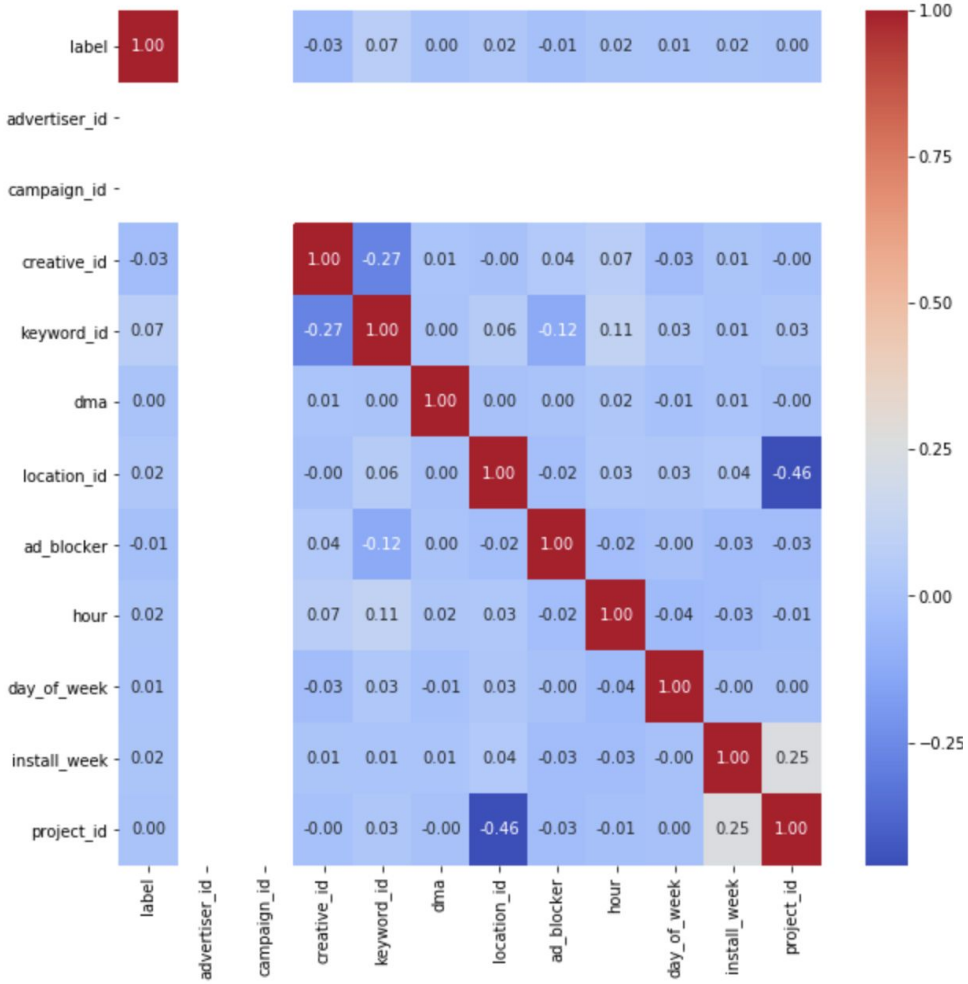
List of all columns

['label', 'advertiser_id', 'campaign_id', 'creative_id', 'keyword_id',
'country_code', 'state', 'dma', 'organization', 'browser_ver',
'platform', 'network', 'location_id', 'request_tld', 'ad_blocker',
'hour', 'day_of_week', 'install_week', 'content_category_ids',
'url_category_ids', 'project_id']

Numerical

	label	advertiser_id	campaign_id	creative_id	keyword_id	dma	location_id	ad_blocker	hour	day_of_week	install_week	project_id
count	268710.000000	268710.0	268710.0	2.687100e+05	2.673940e+05	268710.000000	2.687100e+05	268710.000000	268710.000000	268710.000000	268710.000000	268710.000000
mean	0.023185	276.0	821471.0	1.159905e+07	9.839641e+07	615.315504	5.658788e+04	0.098057	14.463652	3.893130	408.854103	123.292293
std	0.150491	0.0	0.0	2.900026e+04	7.188922e+07	132.978032	2.296055e+05	0.297393	5.260805	1.991392	32.384240	58.798611
min	0.000000	276.0	821471.0	1.159121e+07	1.609200e+04	0.000000	0.000000e+00	0.000000	0.000000	1.000000	309.000000	0.000000
25%	0.000000	276.0	821471.0	1.159121e+07	1.170112e+07	524.000000	5.479000e+03	0.000000	10.000000	2.000000	393.000000	115.000000
50%	0.000000	276.0	821471.0	1.159121e+07	1.522403e+08	602.000000	6.344000e+03	0.000000	15.000000	4.000000	419.000000	146.000000
75%	0.000000	276.0	821471.0	1.159121e+07	1.666210e+08	709.000000	6.805000e+03	0.000000	19.000000	6.000000	432.000000	167.000000
max	1.000000	276.0	821471.0	1.170859e+07	1.671625e+08	881.000000	1.141576e+06	1.000000	23.000000	7.000000	456.000000	192.000000

Cross-correlation among numerical columns



Categorical

	country_code	state	organization	browser_ver	platform	network	request_tld	content_category_ids	url_category_ids
count	268710	267108	268710	268710	268710	268710	268710	268710	268598
unique	1	56	4896	86	11	155	24845	256	823
top	US	CA	Comcast Cable	Chrome 65	Windows 10	W127	open.spotify.com	0	20006
freq	268710	32071	51538	65911	107580	38230	18007	246071	31352

Data wrangling

There are some numerical variables which are actually categorical, so they're converted to categorical (e.g. *ad_blocker*). For some variables binning was used as it showed better results (e.g. *day_of_week*). All categorical variables are afterwards treated with one-hot-encoding approach. *'request_tld*" column was pre-treated with binning the rare occurring values in order to reduce the dimensionality.
content_category_ids, *url_category_ids* are specific in terms that they represent a list of ids. Treating them as a regular category variables would reflect in great dimensionality and improper representation. All values are extracted in a list which served as a list of new columns to be created. Every occurrence of an id in the list would yield a +1 in the corresponding column for that particular impression.
Some variables are dropped completely due to being constant or not giving any value (e.g. (*country_code, advertiser_id, campaign_id*)

NaN values

The amount of NaN values is around 1%, following the same distribution of the whole sample.

Training / Modeling

Prepared dataset is split into 80/20 ratio (train/test). Train set was later used in a cross fold validation for each of the algorithms. Stratified k fold was used in order to preserve the original class (im)balance. Grid Search was used for hyperparameter tuning of the models. Test set is saved for comparison between the algorithms.

Due to goal of the task (**binary classification**), nature of the dataset (**highly imbalanced classes**) and need for fast/efficient training (**large dataset, local machine**) following algorithms were considered: Logistic Regression, Naive Bayes (Bernoulli, due to high number of one-hot-encoded categories) and Random Forest. In production model if the resources are readily available, we can build a randomized search of the parameters, followed by a more focused grid search.

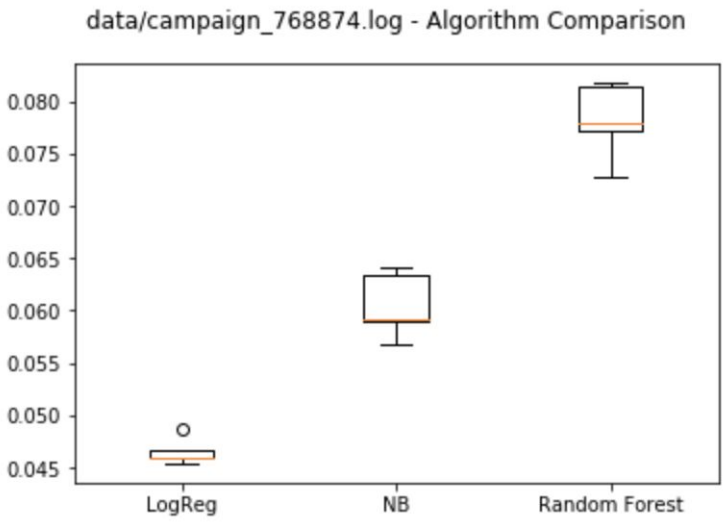
Evaluation metrics

Standard classification metrics for evaluating the model performance include Accuracy, Precision, Recall (Sensitivity) etc.
Due to highly imbalanced class representation (<3% minority class) in the dataset, some other metrics like F1 score or ROC curve (AUC) are more suitable.

The **F1 score** was used as a metric for comparison of algorithms. From a business perspective, it can be discussed what we want to achieve. For example, in terms of maximizing the conversion rate for a limited number of impressions, we can focus on Precision etc.

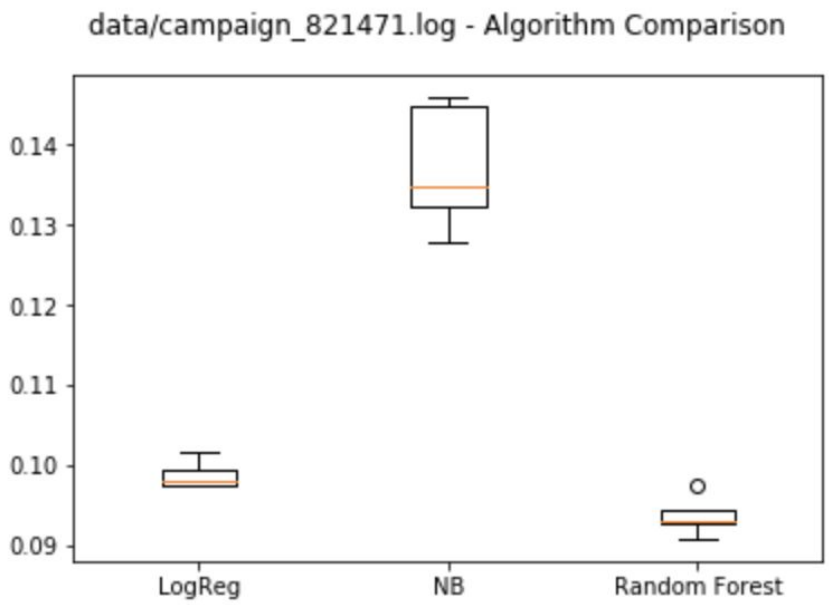
Results

DS1



Algorithm	**Train F1 score (n=5, mean)**	**Test F1 score**
Logistic Regression	0.0465	0.0458
Naive Bayes	0.0605	0.0612
Random Forest	0.0781	0.0868

DS2



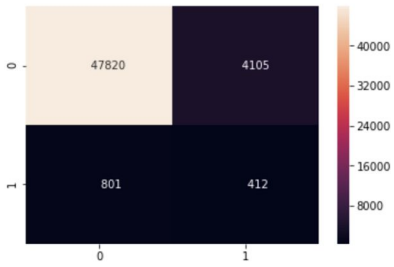
Algorithm	**Train F1 score (n=5, mean)**	**Test F1 score**
Logistic Regression	0.0987	0.1008
Naive Bayes	0.1370	0.1438
Random Forest	0.0936	0.0944

single algorithm - sample report

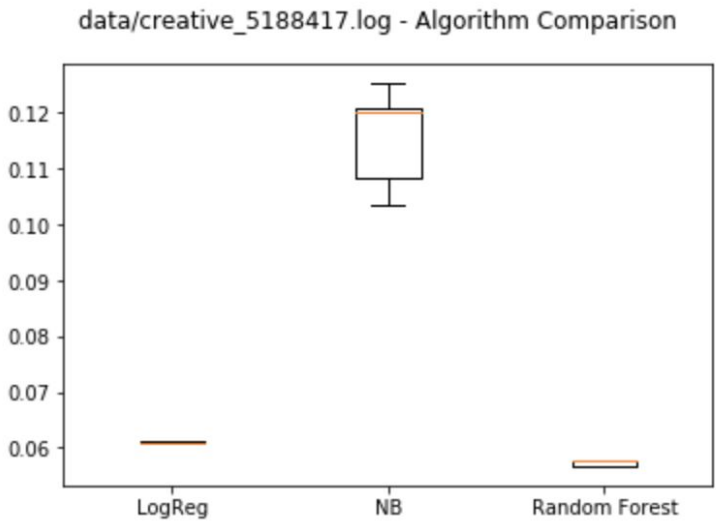
```
model:  NB
model params:  {'class_prior': (0.85, 0.15)}
--------------
scoring method:  f1
train scores:  [0.1278559889222248, 0.14468690702087286, 0.13475836431226765, 0.14583811052510892, 0.1321444901691815]
train scores, mean:  0.13705677218993112
test score:  0.14380453752181502
--------------
test set - classification report:
             precision    recall  f1-score   support

          0      0.984     0.921     0.951     51925
          1      0.091     0.340     0.144      1213

avg / total      0.963     0.908     0.933     53138
```



DS3



Algorithm	**Train F1 score (n=5, mean)**	**Test F1 score**
Logistic Regression	0.0610	0.0613
Naive Bayes	0.1156	0.1237
Random Forest	0.0574	0.0556

Application/Script

The application/script was developed in Python (Python3). It can be executed from a command line *python cr_optimization.py dataset_file.log*. The script will automatically run the 3 predefined algorithms and choose the best concerning the f1 score. The results are printed on the standard output and the best model is saved locally as a file.

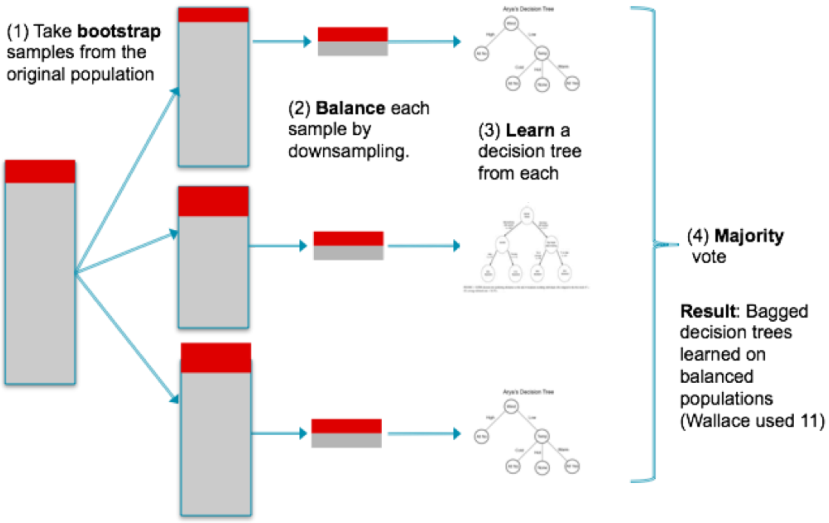
Library dependencies:
pandas, numpy, seaborn, matplotlib, scikit-learn, pickle

Concerning the exploratory analysis part, work-in-progress version of the file is also available on github (exp anaysis.ipynb).

Dependencies:
Jupyter notebook installation: http://jupyter.org/install
Library dependencies are the same as for the python script.

Future ideas

- Advanced undersampling/oversampling techniques (SMOTE): https://github.com/scikit-learn-contrib/imbalanced-learn
- Undersampling with majority vote



- Libffm for CTR prediction https://github.com/guestwalk/libffm
- Clustering
    - Collecting data for the displayed ads themselves (images, text, videos, banner analysis, sentiment analysis etc.)
    - Collecting data about users, tracking impressions (like fb, google etc.)
- Recommendation system (ads data, user data)
- Interesting papers
    - http://cs229.stanford.edu/proj2016/report/JacobKong-Predicting%20Which%20Recommended%20Content%20Users%20Click-report.pdf
    - https://gking.harvard.edu/files/0s.pdf
    - http://quinonero.net/Publications/predicting-clicks-facebook.pdf
    - http://sci2s.ugr.es/sites/default/files/ficherosPublicaciones/1422_2011-Galar-IEEE_TSMCc-Ensembles.pdf