DIGIX Implementation Instruction

# Strategy of Implementation

* *Reading dataset with chunks and downcasting to fit the memory.*
* *Target encoding with smoothing.*
* *SGD model with mini-batches.*
* *class\_weights to balance classes.*

# Implementation Description

## Runtime

*We implement our experiments on online Kaggle notebooks which provides*

* *Intel(R) Xeon(R) CPU @ 2.30GHz 16 cores*
* *16 GB Ram*
* *5GB harddisk*

*Ram and harddisk were main constraints in our experiments so we had to implement datatype conversions and compressions in several notebooks. Also, we had limited options on machine learning models such as SGD, passive-aggressive and neural networks which work on mini batches.*

## Experiment process

* *We read whole train (42M) dataset with chunk size of 10K and apply downcast to reduce the size in memory.*

[*https://www.kaggle.com/hakkoz/ctr-2-read-data*](https://www.kaggle.com/hakkoz/ctr-2-read-data)

* *We implemented target encoding on columns by using a custom function which smooths standard target encoding with global mean of a column.*
* *We dropped uid and pt\_d columns on train dataset.*
* *We shuffle the dataset and split it to 40M for train and rest (~2M) for test purposes.*
* *We produce train dataset in several notebooks due to hard disk limitations of Kaggle platform (only 5GB).*

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-0*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-0)

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-1*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-1)

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-1-5*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-1-5)

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-2*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-2)

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-2-5*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-2-5)

[*https://www.kaggle.com/aysenur95/ctr-train-test-split-3*](https://www.kaggle.com/aysenur95/ctr-train-test-split-3)

[*https://www.kaggle.com/aysenur95/ctr-train-test-split-3-5*](https://www.kaggle.com/aysenur95/ctr-train-test-split-3-5)

[*https://www.kaggle.com/hakkoz/ctr-6-train-test-split-4v2*](https://www.kaggle.com/hakkoz/ctr-6-train-test-split-4v2)

[*https://www.kaggle.com/aysenur95/ctr-train-test-split-4-5*](https://www.kaggle.com/aysenur95/ctr-train-test-split-4-5)

* *We chose SGD model of Scikit Learn with default parameters and feed it with batches of 10K.*
* *For every batch we used class\_weight parameter to balance classes.*
* *After evaluating our model on our test dataset (with AUC score of 70%), we refit our model on whole training set (~42M) and export the model.*

[*https://www.kaggle.com/hakkoz/ctr-6-split-sgd-batch-class-weight*](https://www.kaggle.com/hakkoz/ctr-6-split-sgd-batch-class-weight)

* *By using this model, we implement prediction on submission dataset test\_data\_B.csv. For this step we used mean values to fill NA values which are produced by target encoding because of newly encountered values.*

[*https://www.kaggle.com/hakkoz/ctr-7-predict-submission-datasets*](https://www.kaggle.com/hakkoz/ctr-7-predict-submission-datasets)

* *We didn’t use any cross validation or hyper parameter tuning technique for this contest due to computational constraints of online platforms.*
* *We didn’t perform any of feature engineering techniques also.*
* *We also tried Decision Tree, XGBoost, catboost and lightGBM with several parameters but they didn’t work out due to memory errors.*