ASSIGNMENT 2

CS5304 - FEATURE ENGINEERING FOR MACHINE LEARNING

1. Data Set Details

For this task, we'll be using the Display Advertising Challenge dataset from Criteo. This can be obtained at `https://s3-eu-west-1.amazonaws.com/criteo-labs/dac.tar.gz` and it is roughly 4.5GB.

The data consists of two partitions, **train** and **test**. We'll only be using the **train** partition in this assignment. As described at Criteo, this dataset consists of 13 integer features and 26 categorical features, along with a binary label. The training data set has about 48 million examples. Your task is to transform the features in this dataset in order to make it suitable to build and evaluate various click prediction models on it.

We'll now detail the steps you need to perform to prepare the data for modeling. Optionally, you can continue to build models and evaluate them. It is not required for this assignment but we highly encourage you to do it so that you get a sense for the entire process.

2. ETL, Feature Extraction, and Data Preparation

Before building any model, you need to understand the features that are available to you and plan out how you are going to use them in your model. This is a very important part of the Data Science process. Data Scientists spend a large portion of their time understanding the data, and preparing the experiments. The actual model running part is usually pretty straightforward. To help you with understanding the features, you'll compute some summary statistics, look at some plots etc to get a sense for the data set. After that, you'll partition the data into training, validation, and test sets. You'll then perform any normalizations using only the information gleaned from the **train** partition. Please note that any feature engineering, data transformations etc should be done using information only from the **train** partition. Using the validation or test partitions for this purpose is tantamount to **cheating**.

2.1. **Partitioning the Data.** In order to keep things manageable, we'll partition the data and deal only with a small portion of the data. First, split the entire train.txt file into two portions: 2 million rows on which you'll do all your assignment work and the remaining 46 million rows that you'll use only if you choose to do the optional modeling part.

From these 2 million rows, create 3 partitions: **train1M**, **validation250k**, and **test750k**. Randomly partition 1 million rows for training, 250k for validation and the remaining for testing. For debugging purposes, you may want to further split the 3 partitions into small chunks while writing your models and verifying that they are working correctly. We usually recommend a sample size of about 1000-10,000 rows for such purposes.

2.2. **Summary Statistics.** For all the 39 features, you should compute histograms and plot them. In your assignment, submit a few examples from both the integer numerical features and categorical features. However, your code should contain logic to show that you computed all 39 histograms.

Additionally, compute summary statistics for each of the **integer** columns: You should compute at least **Mean** and **Standard Deviation**. Sometimes, computing **Skew** and **Kurtosis** is useful, if you suspect that the data is not normally distributed.

2.3. **Normalization and Feature Engineering.** Using information only from **train1M**, you should standardize the Integer variables so that they are zero-mean/unit-variance.

From your summary statistics, decide which of the 26 categorical features you are going to drop from modeling experiments. Again, use only the **train1M** partition to make your choices. Your assignment submission should include rationalization for each of the features that you decide to drop. For the features that you selected, figure out a way to handle the categorical features. You can do one-hot encoding, retain only the top few of distinct values of a categorical feature (by using the histogram information) and then perform one-hot encoding. You can also compute a rate value for the categorical feature and use that rate value instead. Or you can do something more elaborate such as computing TF*IDF estimates and choosing the categorical values for a feature based on these TF*IDF estimates. Choose one approach for all categorical features and do the same. If time permits, you can experiment with other techniques but it is not required for the assignment.

Please note that the dataset contains many missing values. Treat all -1 and missing integer feature values as 0. For categorical features, treat missing value as another token and transform it as you would another other categorical feature value using onehot, rate, or TF*IDF method.

Once you have selected the features you are going to use and their normalization approach, you should apply this transformation on both **validation250k** and **test750k** data sets. Your code will be evaluated to see the following:

- Does it partition the **train.txt** file into 3 parts and are they of their expected sizes?
- Did you calculate your summary statistics and feature transformations only on the **train1M** partition and apply them to the remaining two partitions?
- Do the statistics and transformations you applied agree with what we'd expect to see for each of those features

We expect you to submit the following files on CMSX:

- **train_ids.txt** which is your 1M training partition. Should contain the row numbers that you selected from the original train.txt file, zero-based numbering.
- **validation_ids.txt** Your 250K validation partition. Same format as above.
- **test_ids.txt** Your 750K test partition. Same format as above.

- **features.txt** A file containing a list of feature columns, one per row, for all the features (both categorical and integer) that you ended up selecting. Please note that 0-12 will be considered as integer features and 13-38 will be considered categorical.
- **preprocess.txt** a file indicating the method you chose to handle the categorical features. Should be one of **onehot, rate, tfidf**. Any other value will be ignored and you'll lose points.
- **assign2.py** Your Python code that clearly demonstrates your working code for doing the above ETL and Feature Engineering tasks.

3. Grading Rubric

You'll be graded according to the following:

- Correctness of data splitting: 3 points
- Correctness of integer statistics: 3 points
- Visualizing all features: 2 points
- Encoding Categorical features: 5 points
- Principled approach for dropping / retaining features: 2 points

4. Optional Modeling and Evaluation

Once you have done feature engineering, you should build at least 2 models. You should test out Logistic Regression and Random Forest. For each model, compute the AUC of the ROC curve on both the validation set and test set.

5. Optional Model Running in Production

Once you have successfully built the models and built the ROC curves, you'll take one of the models and write code to score the remaining 46 million rows. You should run this code on your local machine. This will require you to load your data into your code, do the necessary normalizations and feature engineering steps from above and then score the data with one of the models you built.