

# DForest: A Dimensionality-aware Indexing for High-Dimensional Similarity Search

The technical report: <https://www.techrxiv.org/doi/full/10.36227/techrxiv.23550093.v1>

The storage files for our dataset are all in CSV format and are located in the path `/data_set/dataid/dataid.csv`. If your dataset format is different from ours, you can either write a custom input functions to replace the functions of our code or convert the file format to CSV. In our CSV files, each row represents a vector, which corresponds to a data point in the dataset. For example, the storage format of a dataset with  $n$  points and  $d$  dimensions is as follows:

$$\begin{matrix} p_{11}, p_{12}, p_{13}, \cdots, p_{1d} \\ p_{21}, p_{22}, p_{23}, \cdots, p_{2d} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ p_{n1}, p_{n2}, p_{n3}, \cdots, p_{nd} \end{matrix}$$

We are using Python 3.10 to run our Python code, and we need to have the `scikit-learn` package installed. We are using `CodeBlocks 20.04` with `GCC 8.1.0` to compile our C++ code, with the following compilation parameters: `-O3 -Wall -ffast-math`.

We use `create_queryset` to generate a noisy query set. We support generating query sets with two distributions: uniform and Zipf. The filename format for the query set generation is `dataid_queryid.csv`, where `queryid` represents the distribution type followed by the number of query points.

For example, if the dataset ID is `audio` and the query set is generated with the `uniform` distribution and contains 100 query points, the file name would be `audio_uniform100.csv`.

Similarly, if the query set is generated with the `Zipf` distribution and contains 50 query points, the file name would be `dataid_zipf50.csv`.

First, use `deal_dataset.py` to process the dataset. Parameter is `dataid`. After processing, the transformed dataset will be generated in the directory `/data_set/dataid` as `/data_set/dataid/dataid_afterpca.csv`. The retained information when reducing to a specific number of dimensions will be stored in `/data_set/dataid/bao1iu_info.csv`. For example:

```
python deal_dataset.py audio
```

Second, we use `deal_queryset.py` to process the query set by performing a change of basis. Parameters are `dataid` and `queryid`. For example:

```
python deal_queryset.py audio unifrom1000
```

Finally, use `main` or `main_storage` to perform range query and k-nearest neighbor (kNN) operations. Parameters are `dataid`, `queryid`, `e`, `[page_size]`, `dataset_trans_time`, `queryset_trans_time`, the number of  $r, r_1, r_2, \dots, r_{num}$ , and `rou`. The value of  $k$  can be directly set in the code. By default, the values are set as 10, 20, 30, 40, and 50.

```
./main audio uniform1000 42516 267 0.0008 5 46798 60475 67635 76067 86026  
./main_storage audio uniform1000 42516 32768 267 0.0008 5 46798 60475 67635 76067  
86026
```