Fuzzy SQL

Release 0.1.8

Samer Kababji @ EHIL

Oct 21, 2022

CONTENTS

CHAPTER

ONE

INSTALLATION

Install Fuzzy SQL using pip:

```
(.venv) $ pip install fuzzy-sql
```

Check out *Usage* for further information.

The package includes the necessary dependencies.

CHAPTER

TWO

USAGE

2.1 fuzz_tabular()

The core function for generating random queries is `fuzz_tabular()`. Here is a description of the function:

`fuzzy_sql.fuzzy_sql.`**`fuzz_tabular`**(*n_queries*, *query_type*, *real_file_path*, *metadata_file_path*, *syn_file_path='None'*, *run_folder='None'*, *printme=False*, *db_path='fuzzy_sql.db'*)

The function generates random queries for the input tabular datasets.

Parameters

- **n_queries** (*int*) – The number of random queries to be generated
- **query_type** (*str*) – The type of queries to be generated and can be 'single_agg', single_fltr', 'twin_agg', 'twin_fltr', or 'twin_aggfltr'
- **real_file_path** (*path*) – The full path to the real data csv file including the file extension.
- **metadata_file_path** (*path*) – The full path to the metadata json file including the file extension.
- **syn_file_path** (*path*) – The full path to the synthetic data csv file including the file extension or 'None' if random queries are desired for single dataset.
- **run_folder** (*str*) – The full path for the your output folder or 'None', which will save the output reports to the current folder.
- **printme** (*logical*) – Set it to True if an html report is desired. The report lists random records of all the generated queries.

Returns

A dictionary of all generated random queries.

Return type

dictionary

Here is an example how to generate 10 queries for a single dataset:

```
queries=fuzzy_sql.fuzz_tabular(10,,"single_fltr","path/to/file/X_real.csv", "path/to/
↪file/X_metadata.json")
```

Below is another example to generate 100 aggregate queries simultaneously applied to both real and synthetic input datatsets:

```
queries=fuzzy_sql.fuzz_tabular(100,"twin_agg","path/to/file/X_real.csv", "path/to/file/X_
↪metadata.json", "path/to/file/X_syn.csv")
```

Note: Windows users may need to add 'r' before the path string they pass to the function. This will force treating windows backslashes as literal raw character. For instance, pass: `r"C:\path\to\file\X_real.csv"`

2.2 make_table()

`fuzzy_sql.fuzzy_sql.`**`make_table`**(*table_name: str*, *df: DataFrame*, *db_conn: object*)

Imports the input dataframe into a database table. All dots in the variable names will be replaced by underscores.

Parameters

- **`table_name`** – The intended name of the table in the database.
- **`df`** – The input data
- **`db_conn`** – Database (sqlite3) connection object

2.3 load_csv()

`fuzzy_sql.fuzzy_sql.`**`load_csv`**(*file_path: Path*) → DataFrame

Reads the input csv file.

Parameters

`file_path` – The input file full path including the file name and csv extension.

Returns

The pandas dataframe in 'unicode-escape' encoding. Any "`" is deleted in the data.

2.4 TABULAR_QUERY()

`class` `fuzzy_sql.tabular_query.`**`TABULAR_QUERY`**(*db_conn*, *real_tbl_name: str*, *metadata: dict*)

This is a class used for generating random SELECT queries for a table residing in sqlite database.

Once called, various query parameters (listed below) are constructed typically in forms of dictionaries. The value entries of the dictionaries represent discrete probabilities for better model versatility. However, these probabilities can be modified only by accessing the constructor.

AGG_OPS: Aggregation functions

LOGIC_OPS: Logic Operations

NOT_OP_STATE: NOT operation state i.e. used or not used.

CAT_OPS: Value comparison operations for nominal variables.

CNT_OPS: Value comparison operations for continuous variables.

CAT_VAL_BAG: A bag of possible values for nominal variables.

CNT_VAL_BAG: A bag of possible values for continuous variables.

DT_VAL_BAG: A bag of possible values for date variables.

Parameters

- **`db_conn`** (*connection object created using sqlite3.connect()*) – A connection to the database that contains the table to be subjected to random queries.

- **real_tbl_name** (*String*) – The name of the table to be randomly queried.
- **metadata** (*Dictionary*) – A dictionary that includes table's variable names (i.e. column names) as keys and types of variables as values. THey types shall be restricted to: 'continuous', 'data' and 'nominal'. Any table shall have at least one nominal variable.

INDEX