Fake Data Generation

Hari Kishor Chintada
DATA 515 Project

Why Fake Data Generator?

DATA IS (GOLD) KING

ALL ML PROJECTS NEEDS DATA

HARD TO GET REAL DATA WITH INCREASE IN PRIVACY AND SECURITY CONCERNS

COMPANIES ARE RESTRICTING CUSTOMER DATA USAGE

LOAD TEST – NEED LARGE VOLUME OF DATA

Objective

The objective of this project is to create a package to generate a random dataset using configuration.

Flexibility/Configurability: Simple option to provide configuration:, so users need flexibility to generate data based on their conditions.

Extensibility: Support custom data input: Each field needs different data, for example, medical data is different from sales data. So, it is helpful if the package supports bringing their own data.

Manage relation between data columns

Generate pandas data frame: Simple to output or re structure in the required format

Targeted Users

STUDENT

DEVELOPER/PROGRAMMER

DATA SCIENTISTS

ANALYST/RESEARCHER

Implementation

My primary objective is to create a project using bellow concepts I learned from Data 515 class.

- **Object oriented design:** Created data generator class and defined various methods in it.
- **General purpose deep modules**: Depending on user input, the data generator performs various tasks and returns requested data.
- **Separation of concern:** Created modules to process data from data frames and process regular expressions
- **Information hiding:** Most of the complex processing logic is hidden from the user

Implementation

Use concepts I learned from Data 515 class

Object oriented design: Created data generator class and defined various methods in it

General purpose deep modules: Depending on user input, the data generator performs various tasks and returns requested data.

Separation of concern: Created modules to process data from data frames and process regular expressions

Information hiding: Most of the complex processing logic is hidden from the user

Error handling & Unit tests

Libraries

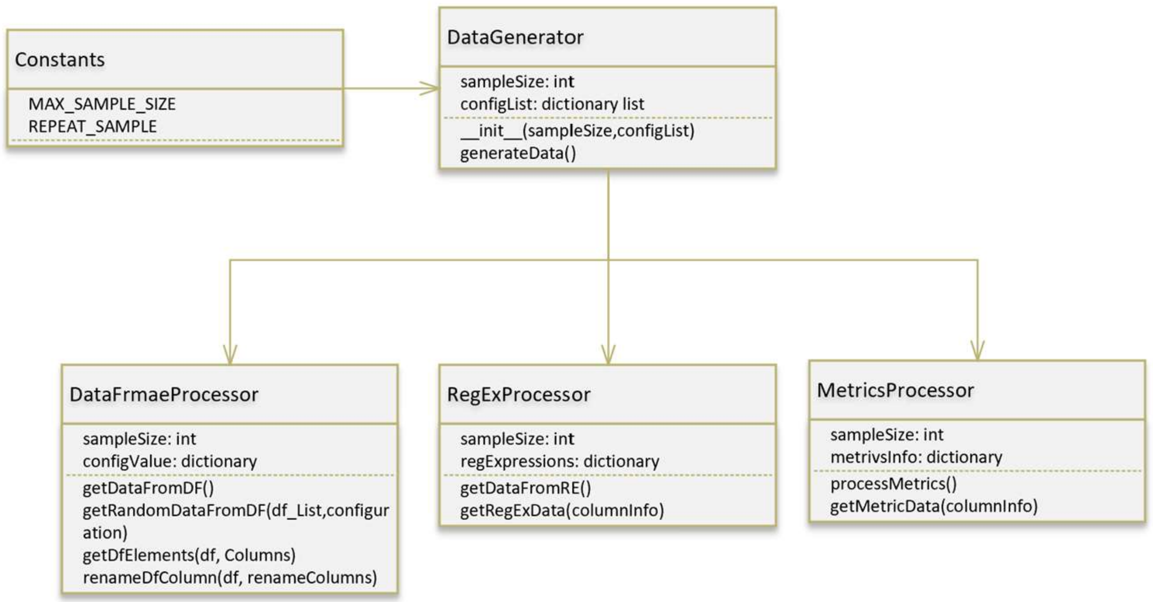
Following libraries/modules are used in this project.

pandas: To use pandas data frame and operations on it

random: To generate random value or sample

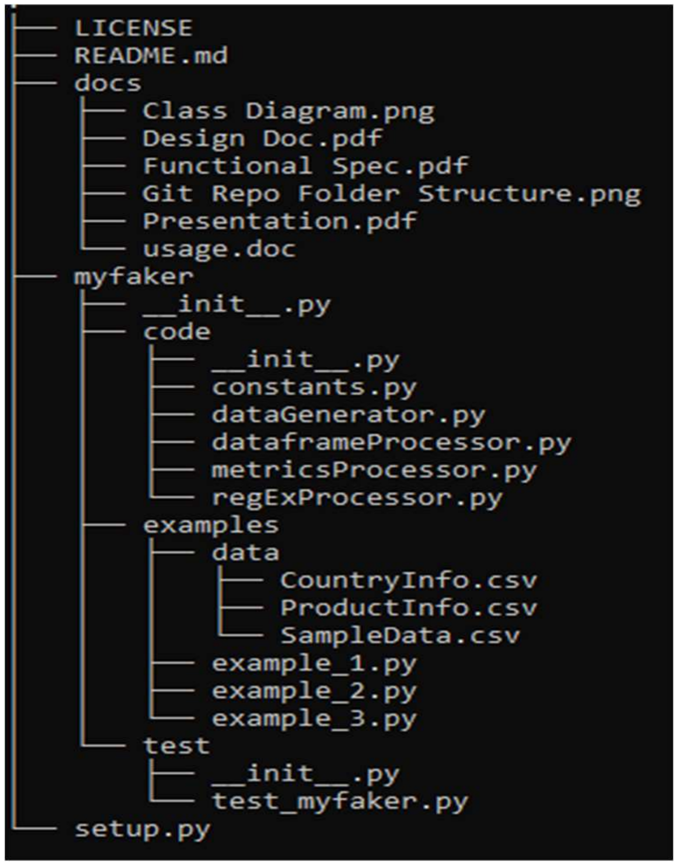
rsts: To generate random string using simple regular expression

- Class diagram



Git Repo Structure

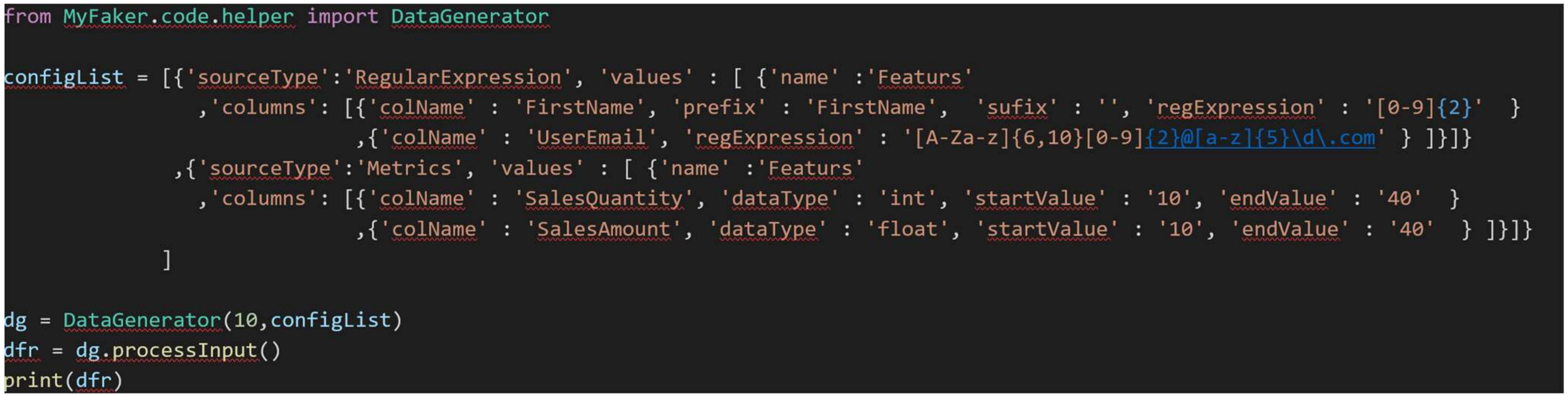
- https://github.com/HarikcUW/myfaker



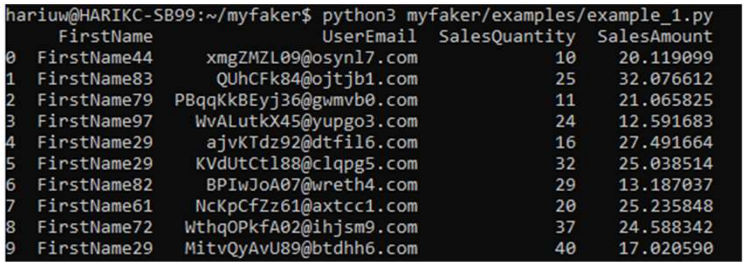
```
├── LICENSE
├── README.md
├── docs
│   ├── Class Diagram.png
│   ├── Design Doc.pdf
│   ├── Functional Spec.pdf
│   ├── Git Repo Folder Structure.png
│   ├── Presentation.pdf
│   └── usage.doc
├── myfaker
│   ├── __init__.py
│   ├── code
│   │   ├── __init__.py
│   │   ├── constants.py
│   │   ├── dataGenerator.py
│   │   ├── dataframeProcessor.py
│   │   ├── metricsProcessor.py
│   │   └── regExProcessor.py
│   ├── examples
│   │   ├── data
│   │   │   ├── CountryInfo.csv
│   │   │   ├── ProductInfo.csv
│   │   │   └── SampleData.csv
│   │   ├── example_1.py
│   │   ├── example_2.py
│   │   └── example_3.py
│   └── test
│       ├── __init__.py
│       └── test_myfaker.py
└── setup.py
```

Example#1

Generate random data using regular expression. In this scenario no input data is required.
Refer example_1.py in /myfaker/examples/



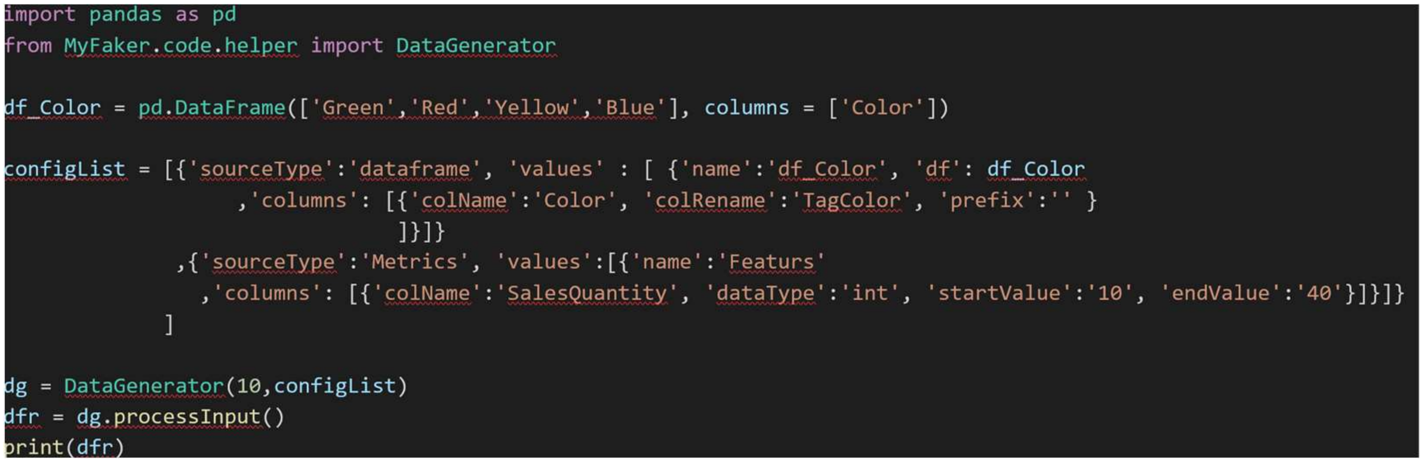
```
from MyFaker.code.helper import DataGenerator

configList = [{'sourceType':'RegularExpression', 'values' : [ {'name' :'Featurs'
                ,'columns': [{'colName' : 'FirstName', 'prefix' : 'FirstName',  'sufix' : '', 'regExpression' : '[0-9]{2}'  }
                             ,{'colName' : 'UserEmail', 'regExpression' : '[A-Za-z]{6,10}[0-9]{2}@[a-z]{5}\d\.com' } ]}]}
              ,{'sourceType':'Metrics', 'values' : [ {'name' :'Featurs'
                ,'columns': [{'colName' : 'SalesQuantity', 'dataType' : 'int', 'startValue' : '10', 'endValue' : '40'  }
                             ,{'colName' : 'SalesAmount', 'dataType' : 'float', 'startValue' : '10', 'endValue' : '40'  } ]}]}
             ]

dg = DataGenerator(10,configList)
dfr = dg.processInput()
print(dfr)
```



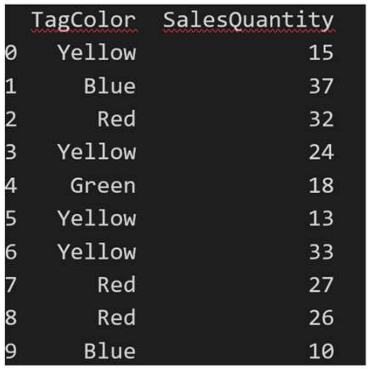
```
hariuw@HARIKC-SB99:~/myfaker$ python3 myfaker/examples/example_1.py
     FirstName                UserEmail  SalesQuantity  SalesAmount
0  FirstName44      xmgZMZL09@osynl7.com             10    20.119099
1  FirstName83       QUhCFk84@ojtjb1.com             25    32.076612
2  FirstName79  PBqqKkBEyj36@gwmvb0.com             11    21.065825
3  FirstName97     WvALutkX45@yupgo3.com             24    12.591683
4  FirstName29      ajvKTdz92@dtfil6.com             16    27.491664
5  FirstName29     KVdUtCtl88@clqpg5.com             32    25.038514
6  FirstName82      BPIwJoA07@wreth4.com             29    13.187037
7  FirstName61     NcKpCfZz61@axtcc1.com             20    25.235848
8  FirstName72    WthqOPkfA02@ihjsm9.com             37    24.588342
9  FirstName29    MitvQyAvU89@btdhh6.com             40    17.020590
```

Example#2

Generate random data using data frame. In this scenario user can get data from a file or create a list and convert it to data frame.



```python
import pandas as pd
from MyFaker.code.helper import DataGenerator

df_Color = pd.DataFrame(['Green','Red','Yellow','Blue'], columns = ['Color'])

configList = [{'sourceType':'dataframe', 'values' : [ {'name':'df_Color', 'df': df_Color
                   ,'columns': [{'colName':'Color', 'colRename':'TagColor', 'prefix':'' }
                                ]}]}
              ,{'sourceType':'Metrics', 'values':[{'name':'Featurs'
                ,'columns': [{'colName':'SalesQuantity', 'dataType':'int', 'startValue':'10', 'endValue':'40'}]}]}
             ]

dg = DataGenerator(10,configList)
dfr = dg.processInput()
print(dfr)
```



```
  TagColor  SalesQuantity
0   Yellow             15
1     Blue             37
2      Red             32
3   Yellow             24
4    Green             18
5   Yellow             13
6   Yellow             33
7      Red             27
8      Red             26
9     Blue             10
```

Comparison

Comparison module: https://github.com/joke2k/faker

- Many shallow modules
- Attribute relation is not maintained, each attribute is independent
- RegEx support is not available, always need source data

Lessons Learned, Limitations & Future Work

Lessons Learned:

- Design is harder than coding
- Continuous build integration with unit test helps to detect issues quickly

Limitations:

- rstr module doesn’t support all regular expressions, any unsupported complex expression required own implementation
- Not using distributes design, so result data set size & volume depends on user system capacity

Future Extensions:

- Simplify input configuration
- Support data generation using simple input like (categorical, categorical, categorical, int, int, float)
- Extend to generate how many distinct values should generate for each categorical feature
- For Metrics – right now we generate random number, extend it to support any specific distribution