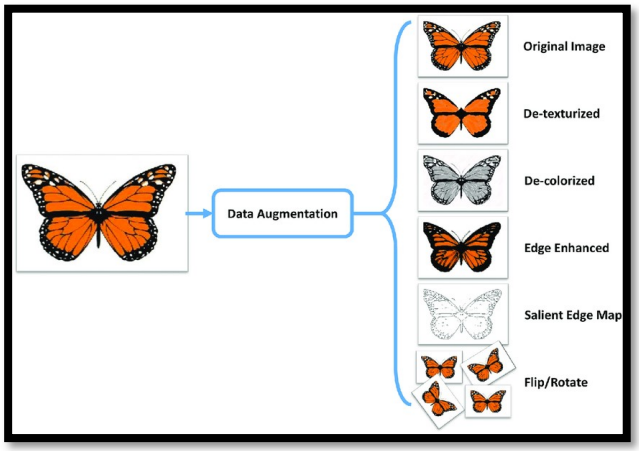
SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Software Engineering Project



Data Augmentation

K029 – Arijit Kulkarni

K025 – Parshwa Jani

K010 – Jeet Desai

SVKM'S NMIMS Deemed to be UNIVERSITY | MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

Complete title:

Generating Synthetic Data using Data Augmentation: A Comparative Analysis of GAN, SMOTE TOMEK, and DataSynthesizer Algorithms

Abstract:

In the realm of software engineering and data science, the generation of synthetic data has emerged as a vital tool for enhancing the performance of machine learning models, especially in scenarios with limited or sensitive data availability. This project focuses on the implementation and comparative analysis of synthetic data generation algorithms: Conditional Tabular GAN (CTGAN), Synthetic Minority Over-sampling Technique with Tomek Links (SMOTE TOMEK), and DataSynthesizer, all implemented in Python.

The study begins by addressing the challenge of imbalanced datasets, a common issue in machine learning applications. The SMOTE TOMEK algorithm, a combination of SMOTE for oversampling and Tomek links for undersampling, is employed to balance the datasets, resulting in a more robust training dataset for our machine learning models.

The Conditional Tabular GAN (CTGAN) algorithm is then used to generate synthetic data. GANs have shown remarkable success in generating data that preserves the statistical characteristics of the original dataset. CTGAN, specifically designed for tabular data, is applied to create synthetic data samples that mimic the original data distribution.

DataSynthesizer, the third algorithm explored in this project, is employed to generate synthetic data while maintaining privacy and data utility. This library employs differential privacy techniques to ensure that sensitive information is not exposed during the data generation process, making it an ideal choice for scenarios where data privacy is of paramount importance.

The project involves conducting a comprehensive comparative analysis of the three algorithms to assess their effectiveness in generating synthetic data for various

real-world datasets. Evaluation criteria include data quality, utility, and the ability to enhance machine learning model performance. Additionally, a thorough statistical analysis is performed to compare the synthetic data with the original data, ensuring that the generated data closely matches the statistical characteristics of the real-world data.

This research aims to shed light on the strengths and weaknesses of each algorithm, providing valuable insights into their applicability in different scenarios. The findings will serve as a reference for data scientists and software engineers seeking to make informed decisions when selecting a synthetic data generation technique based on the specific requirements of their projects. The project not only contributes to the growing body of knowledge in the field of synthetic data generation but also highlights the importance of choosing the right algorithm to improve the robustness and performance of machine learning models.

Introduction: Our project focuses on the application of three distinct data synthesis algorithms to generate synthetic data for different datasets. These algorithms include CTGAN and SMOTE-TOMEK, both utilized on the "insurance.csv" dataset, and the DataSynthesizer algorithm applied to the "adult_ssn.csv" dataset. The primary objective of this project is to explore the effectiveness of these algorithms in creating synthetic datasets that closely resemble the original data. Synthetic data generation holds significance in various domains, such as privacy preservation, data augmentation, and machine learning model testing, making it a critical area of study within data science and software engineering.

Background: The field of synthetic data generation has gained prominence in recent years due to the growing need to protect sensitive information while facilitating data-driven processes. Privacy concerns, data sharing, and the scarcity of labeled data are common challenges that can be addressed by generating synthetic datasets. In this project, the application of CTGAN, SMOTE-TOMEK, and DataSynthesizer algorithms provides an opportunity to explore how well these methods mimic the characteristics of the original datasets, ultimately contributing to advancements in data privacy and machine learning applications.

Project Scope: The project's scope encompasses the application of three distinct data synthesis algorithms to two unique datasets: "insurance.csv" and "adult_ssn.csv." After synthesizing the data, a critical component of the project involves conducting a thorough comparison to determine the similarity between the synthetic data and the original datasets. This comparative analysis aims to assess the practical utility of the generated data for various applications, including machine learning model training, while maintaining data privacy and security.

Project Components: The project comprises three core components. First, it involves data preprocessing and algorithm selection, where the raw datasets are prepared and the CTGAN, SMOTE-TOMEK, and DataSynthesizer algorithms are applied. Second, data synthesis takes place, generating synthetic datasets from the original data. Finally, the comparison and evaluation component is critical, as it involves the assessment of how closely the synthetic datasets resemble the original ones. This component is essential to validate the utility and reliability of synthetic data for specific use cases.

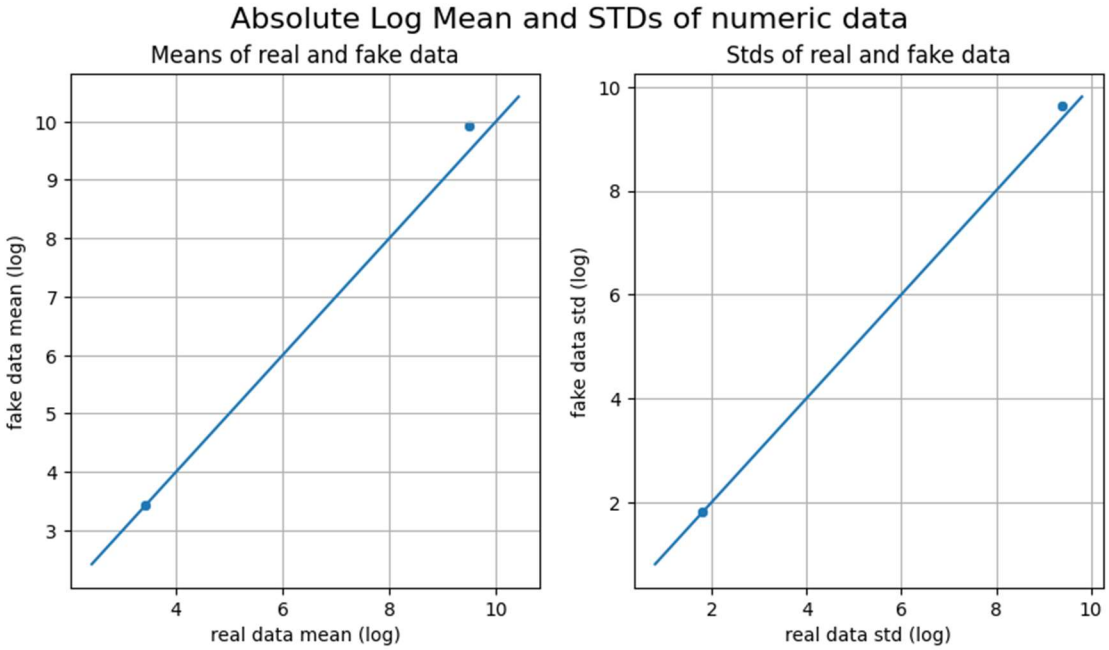
User Interaction: User interaction in this project is primarily focused on project setup, algorithm configuration, and the evaluation of comparison results. The project requires user input for selecting the datasets to be synthesized and specifying algorithm parameters. Additionally, users are involved in the interpretation and validation of comparison results to determine the suitability of the synthetic data for their intended applications. This interaction ensures that the project aligns with user needs and provides valuable insights for decision-making in data-related tasks.

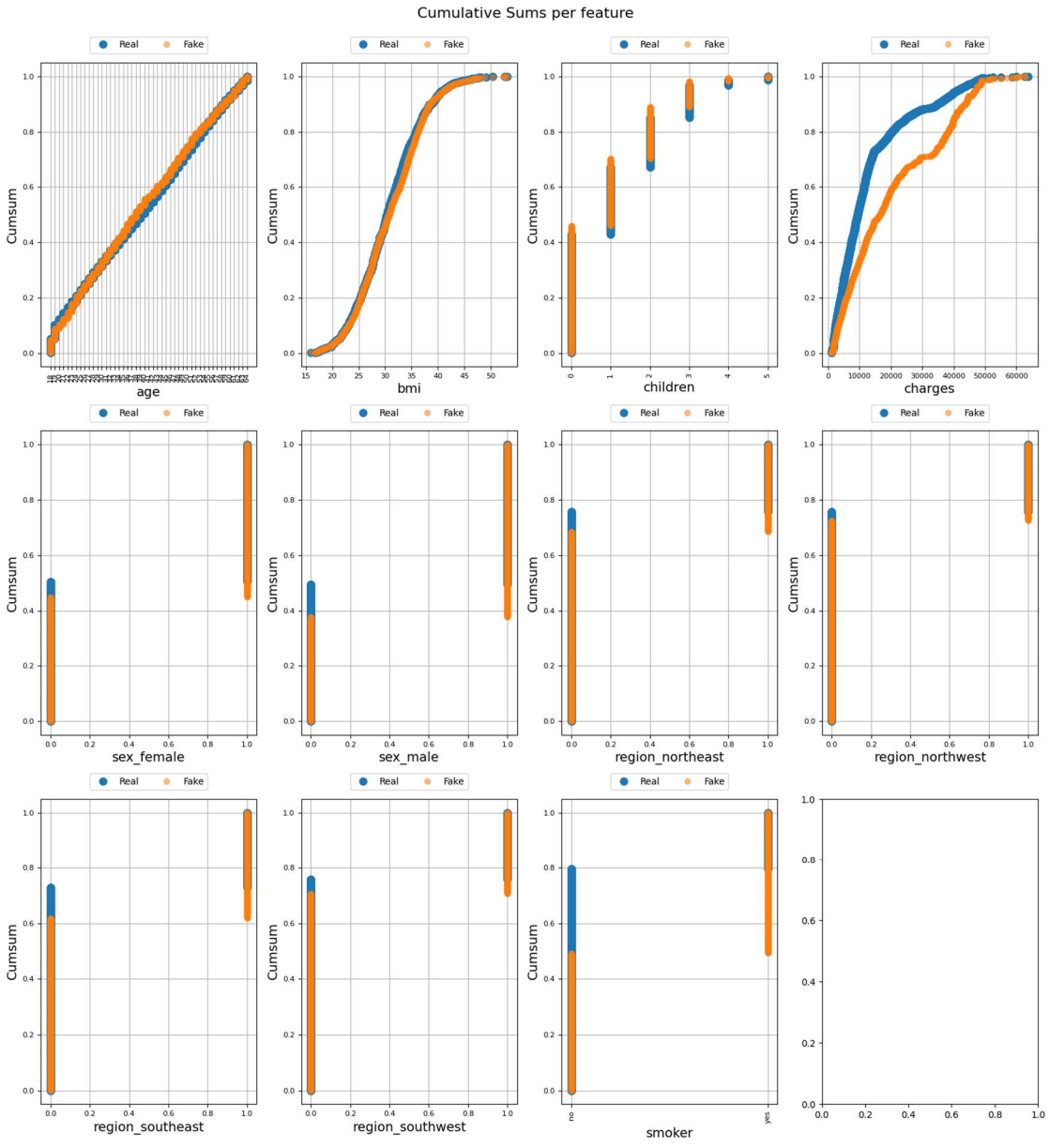
CODE FOR SMOTE TOMEK:

```
import pandas as pd
from imblearn.combine import SMOTETomek
from table_evaluator import TableEvaluator

# Read the csv file into a dataframe
df = pd.read_csv('insurance.csv')
categorical_features = ['age','sex','children','smoker','region']
# Separate the features and the target variable
X = df.drop('smoker', axis=1)
y = df['smoker']

# Encode the categorical variables using one-hot encoding
X = pd.get_dummies(X, columns=['sex', 'region'])

# Create a SMOTE TOMEK object with random_state=42
smt = SMOTETomek(random_state=42)

# Apply SMOTE TOMEK to generate synthetic data
X_res, y_res = smt.fit_resample(X, y)

# Create a new dataframe with the synthetic data
df_res = pd.concat([X_res, y_res], axis=1)

# Save the new dataframe as a csv file
df_res.to_csv('insurance_smote_tomek.csv', index=False)
```

```
import pandas as pd
from table_evaluator import TableEvaluator
categorical_features = ['age','children','smoker']
df = pd.read_csv('insurance_coded.csv')
df_syn = pd.read_csv('insurance_smote_tomek.csv')
print(df.shape, df_syn.shape)
table_evaluator =  TableEvaluator(df, df_syn, cat_cols= categorical_features)
table_evaluator.visual_evaluation()
```

```
(1338, 11) (2060, 11)
```

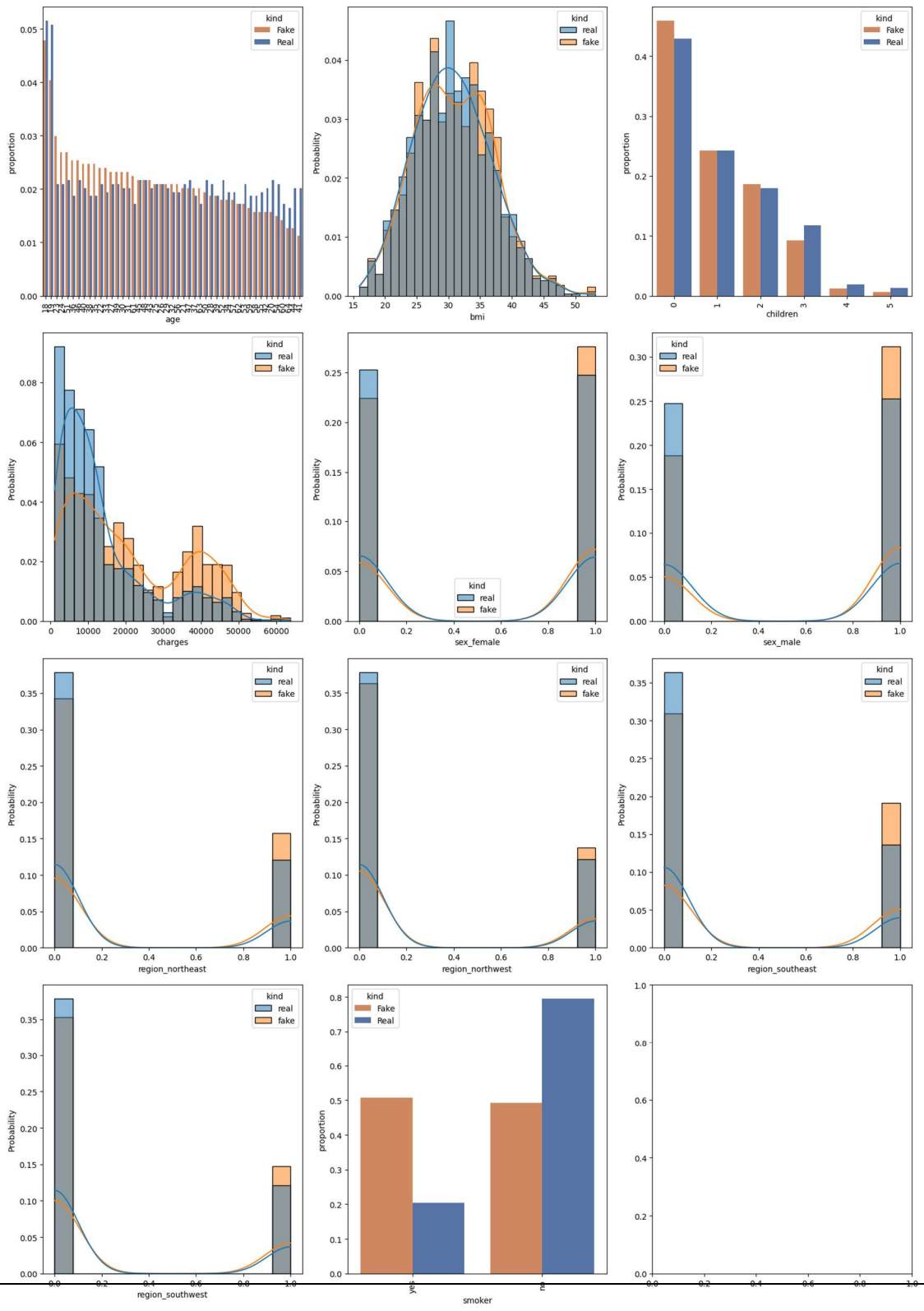

Absolute Log Mean and STDs of numeric data
Means of real and fake data
fake data mean (log)
real data mean (log)
Stds of real and fake data
fake data std (log)
real data std (log)


Cumulative Sums per feature
Real
Fake
Cumsum
age
bmi
children
charges
sex_female
sex_male
region_northeast
region_northwest
region_southeast
region_southwest
smoker
no
yes

SVKM'S NMIMS Deemed to be UNIVERSITY

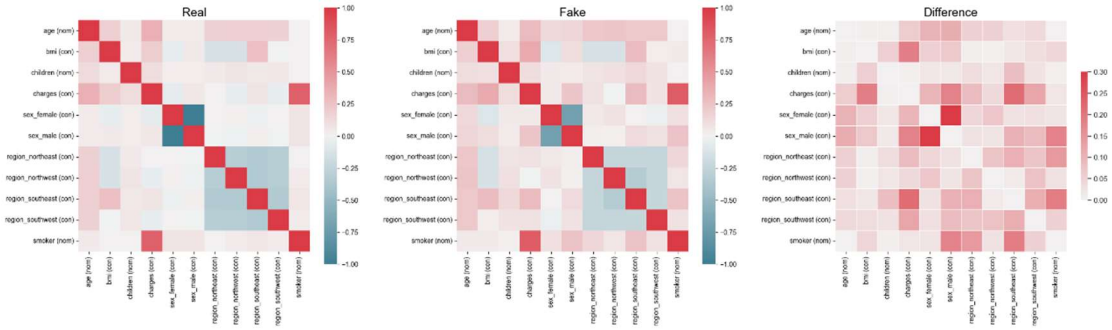
MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

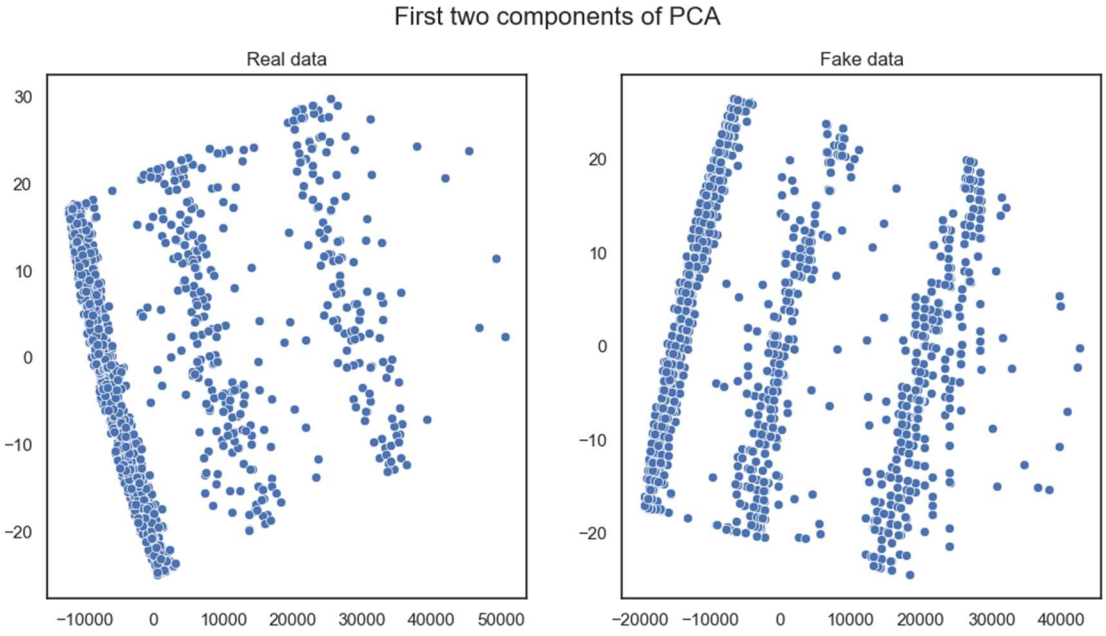
Distribution per feature

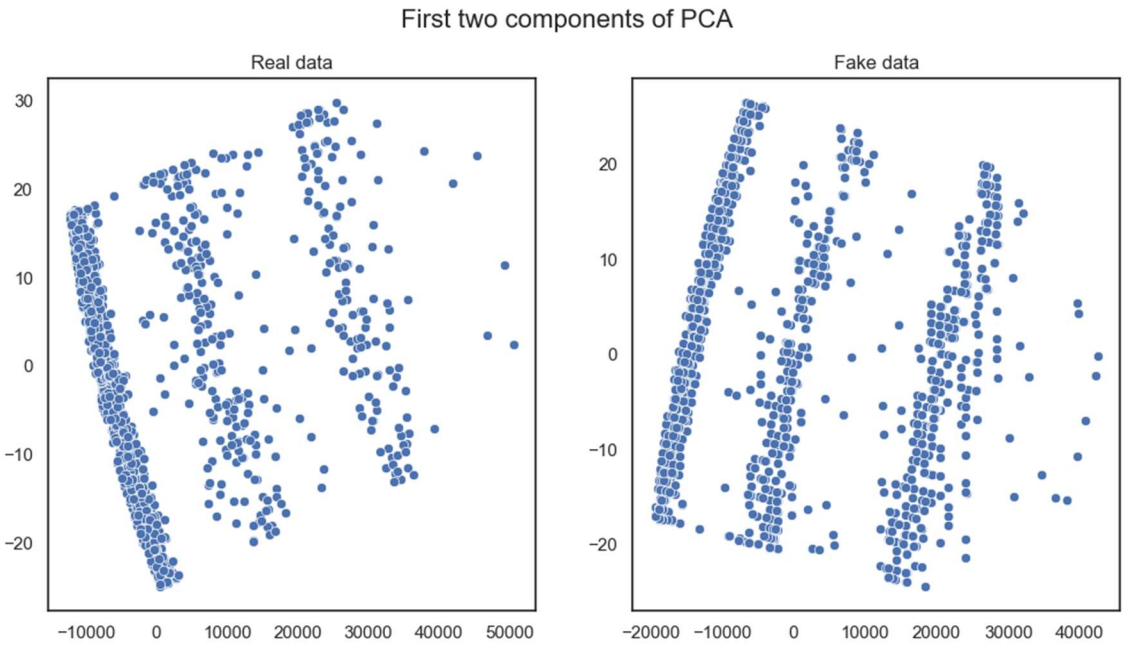


SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING






First two components of PCA
Real data
Fake data
30
20
10
0
−10
−20
−10000
0
10000
20000
30000
40000
50000
20
10
0
−10
−20
−20000
−10000
0
10000
20000
30000
40000

CODE FOR CTGAN:

```
import pandas as pd
data = pd.read_csv('./insurance.csv')
```

```
data
```

```
      age     sex     bmi  children smoker     region      charges
0      19  female  27.900         0    yes  southwest  16884.92400
1      18    male  33.770         1     no  southeast   1725.55230
2      28    male  33.000         3     no  southeast   4449.46200
3      33    male  22.705         0     no  northwest  21984.47061
4      32    male  28.880         0     no  northwest   3866.85520
...   ...     ...     ...       ...    ...        ...          ...
1333   50    male  30.970         3     no  northwest  10600.54830
1334   18  female  31.920         0     no  northeast   2205.98080
1335   18  female  36.850         0     no  southeast   1629.83350
1336   21  female  25.800         0     no  southwest   2007.94500
1337   61  female  29.070         0    yes  northwest  29141.36030
```

```
[1338 rows x 7 columns]
```

"'Next, we define a list with column names for categorical variables. This list will be passed to the model so that the model can decide how to process these fields."'

```
categorical_features = ['age','sex','children','smoker','region']
```

#Model training

"'Next, we simply define an instance of CTGANSynthesizer and call the fit method with the dataframe and the list of categorical variables.

We train the model for 300 epochs only as the discriminator and generator loss becomes quite low after these many epochs."'

```
from ctgan import CTGAN

ctgan = CTGAN(verbose=True)
ctgan.fit(data, categorical_features, epochs = 200)
```

```
Epoch 1, Loss G:  1.7139,Loss D: -0.0040
Epoch 2, Loss G:  1.7498,Loss D: -0.0197
Epoch 3, Loss G:  1.7268,Loss D: -0.0198
```

.

```
.

.
Epoch 197, Loss G:  0.0499,Loss D: -0.0114
Epoch 198, Loss G:  0.0642,Loss D: -0.0597
Epoch 199, Loss G: -0.0499,Loss D: -0.0317
Epoch 200, Loss G: -0.0920,Loss D: -0.0607
```

#Synthetic data generation

```
samples = ctgan.sample(1000)

samples
```

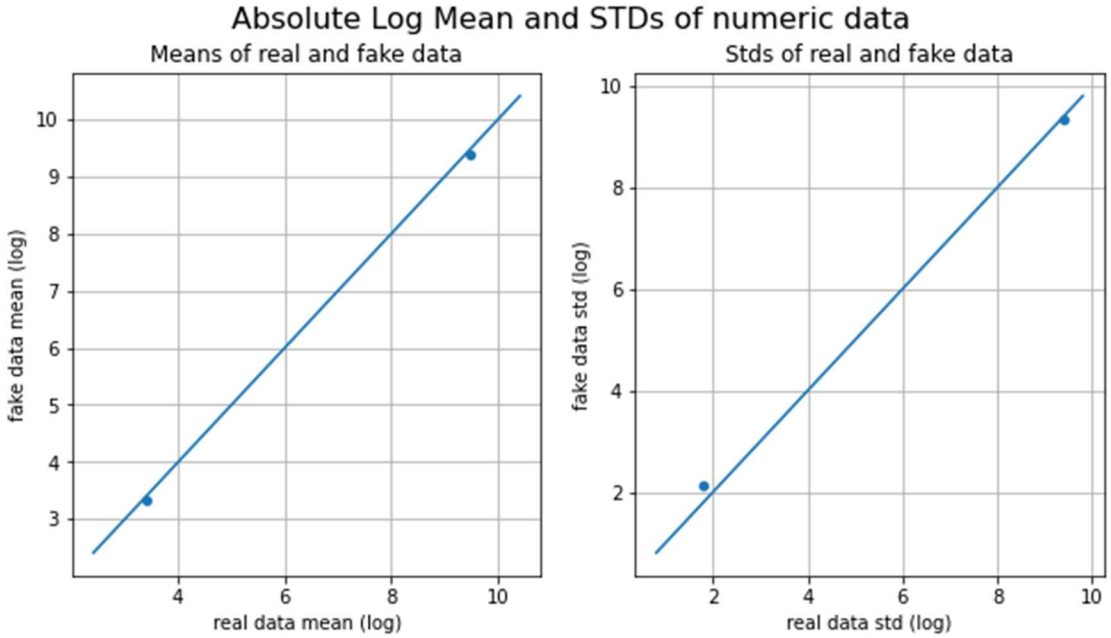
	age	sex	bmi	children	smoker	region	charges
0	28	male	23.843002	2	no	northwest	5395.416825
1	30	female	19.287026	0	no	northeast	15324.961520
2	32	male	39.435769	2	yes	southwest	7777.564509
3	19	female	25.590652	3	no	southeast	10820.135281
4	31	female	37.862905	1	no	northwest	6453.324656
..	...	...	...	...	...	...	...
995	23	male	20.232958	0	no	southeast	12494.436981
996	21	female	31.894804	1	yes	southeast	4511.261236
997	34	male	19.953252	3	no	northeast	1500.385825
998	55	male	22.062920	2	no	southwest	-2326.721959
999	24	female	16.703888	0	no	southeast	357.234286

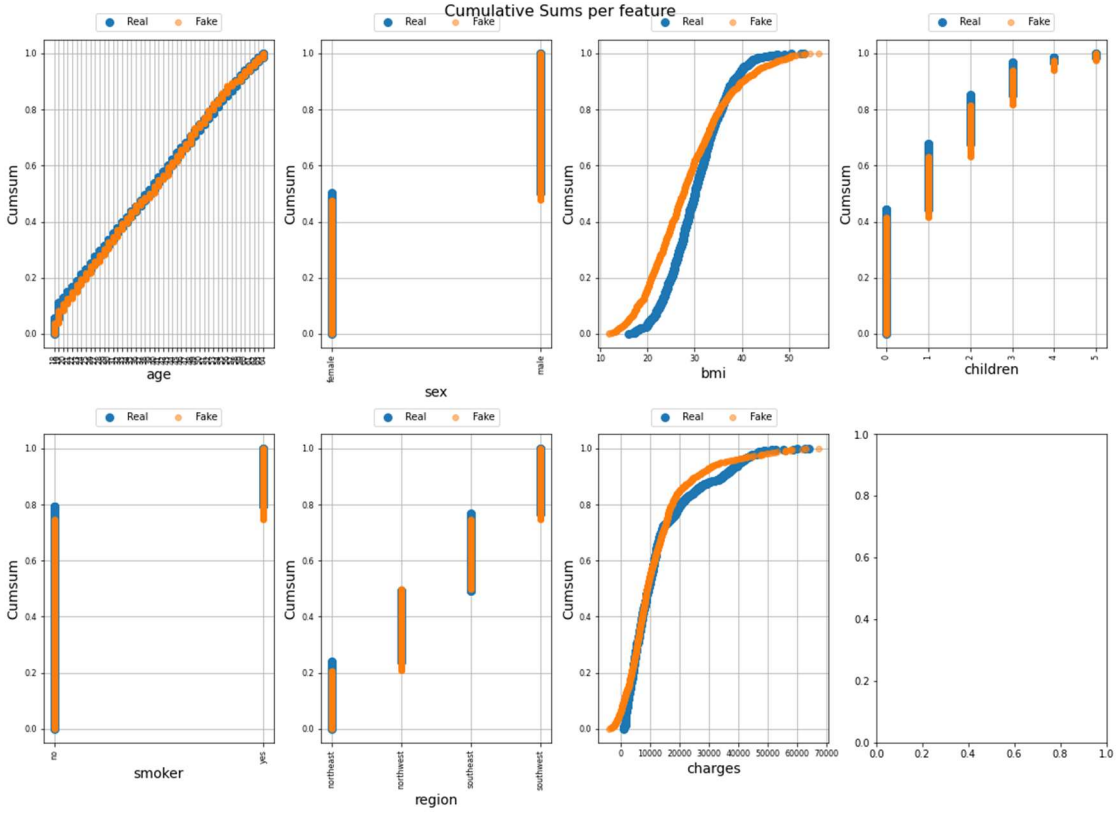
[1000 rows x 7 columns]

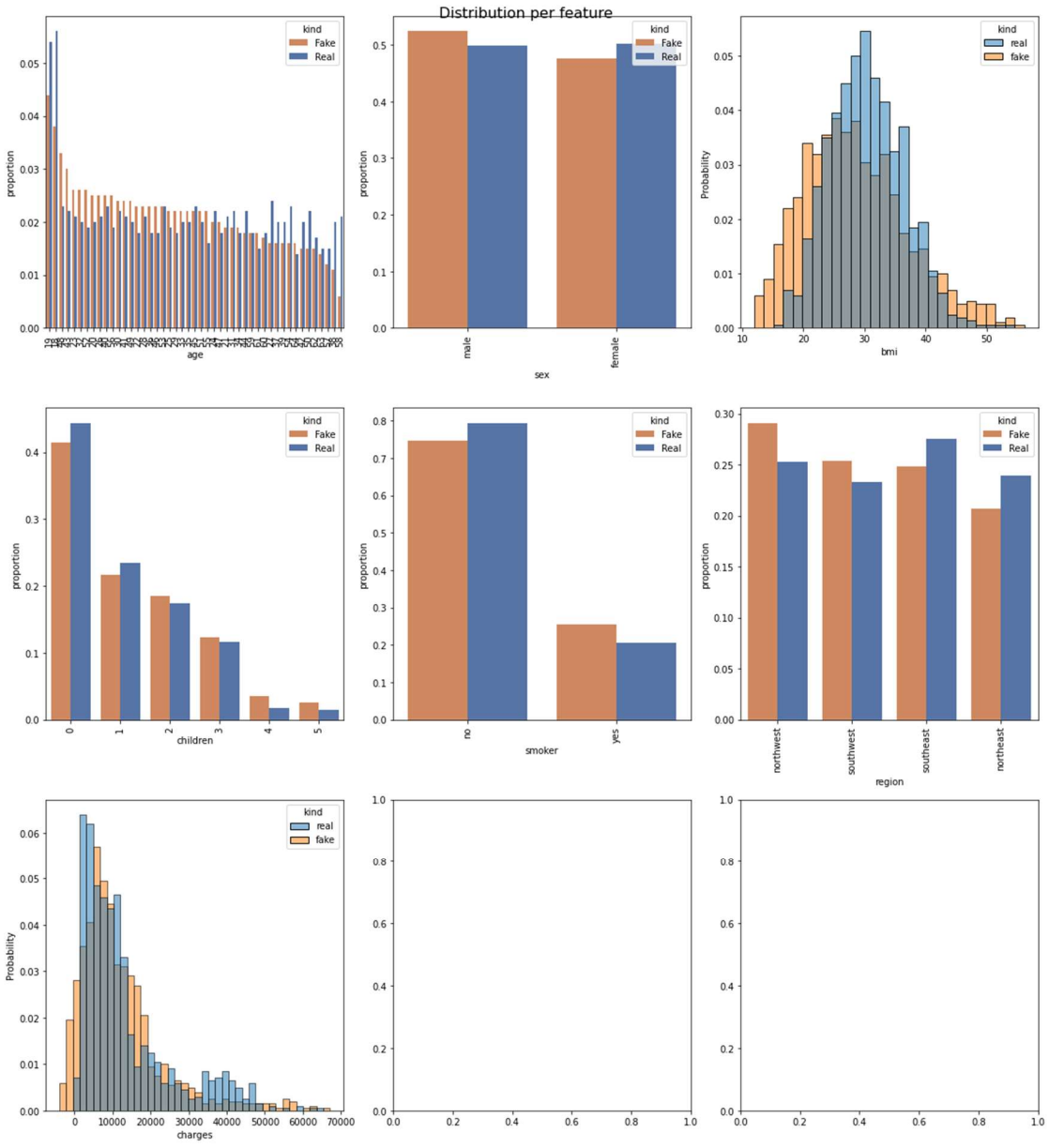
#Evaluation

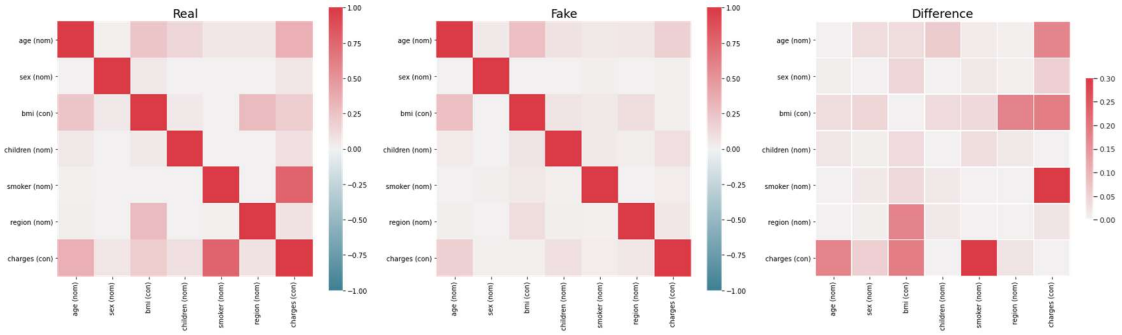
```
from table_evaluator import TableEvaluator

print(data.shape, samples.shape)
table_evaluator =  TableEvaluator(data, samples, cat_cols=
categorical_features)

table_evaluator.visual_evaluation()
```

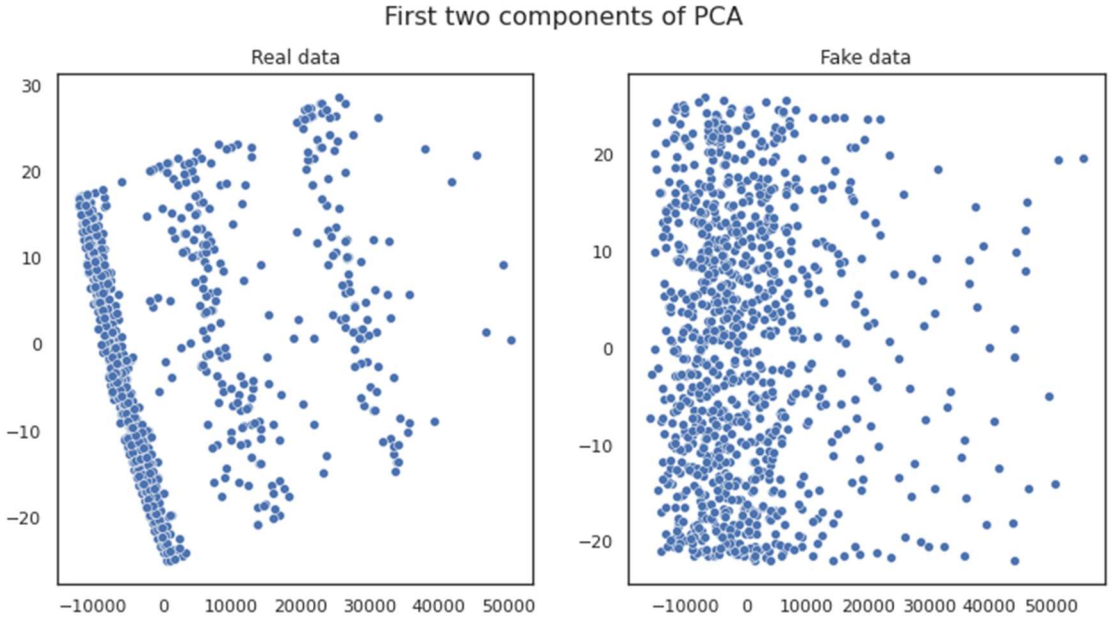
(1338, 7) (1000, 7)


Absolute Log Mean and STDs of numeric data
Means of real and fake data
fake data mean (log)
real data mean (log)
Stds of real and fake data
fake data std (log)
real data std (log)


Cumulative Sums per feature
Real
Fake
Cumsum
age
sex
bmi
children
smoker
region
charges
female
male
no
yes
northeast
northwest
southeast
southwest


Distribution per feature
kind
Fake
Real
real
fake
proportion
Probability
age
sex
male
female
bmi
children
smoker
no
yes
region
northwest
southwest
southeast
northeast
charges


Real
Fake
Difference
age (nom)
sex (nom)
bmi (con)
children (nom)
smoker (nom)
region (nom)
charges (con)


First two components of PCA
Real data
Fake data

CODE FOR DataSynthesizer:

DataSynthesizer

Step 1 import packages

```
from DataSynthesizer.DataDescriber import DataDescriber
from DataSynthesizer.DataGenerator import DataGenerator
from DataSynthesizer.ModelInspector import ModelInspector
from DataSynthesizer.lib.utils import read_json_file,
display_bayesian_network

import pandas as pd
```

Step 2 user-defined parameteres

```
# input dataset
input_data = 'adult_ssn.csv'
# location of two output files
mode = 'independent_attribute_mode'
description_file = f'description.json'
synthetic_data = f'sythetic_data.csv'

# An attribute is categorical if its domain size is less than this threshold.
# Here modify the threshold to adapt to the domain size of "education" (which
is 14 in input dataset).
threshold_value = 20

# specify categorical attributes
categorical_attributes = {'education': True}

# specify which attributes are candidate keys of input dataset.
candidate_keys = {'age': False}

# Number of tuples generated in synthetic dataset.
num_tuples_to_generate = 32561 # Here 32561 is the same as input dataset, but
it can be set to another number.
```

Step 3 DataDescriber

1. Instantiate a DataDescriber.

1. Compute the statistics of the dataset.
2. Save dataset description to a file on local machine.

```
describer = DataDescriber(category_threshold=threshold_value)
describer.describe_dataset_in_independent_attribute_mode(dataset_file=input_d
ata,

attribute_to_is_categorical=categorical_attributes,

attribute_to_is_candidate_key=candidate_keys)
describer.save_dataset_description_to_file(description_file)
```

Step 4 generate synthetic dataset

1. Instantiate a DataGenerator.
1. Generate a synthetic dataset.
2. Save it to local machine.

```
generator = DataGenerator()
generator.generate_dataset_in_independent_mode(num_tuples_to_generate,
description_file)
generator.save_synthetic_data(synthetic_data)
```

Step 5 compare the statistics of input and sythetic data

The synthetic data is already saved in a file by step 4. The ModelInspector is for a quick test on the similarity between input and synthetic datasets.

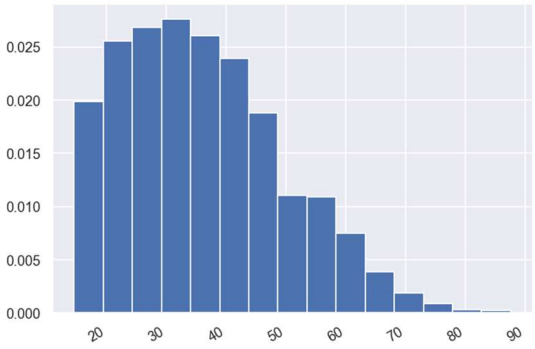
5.1 instantiate a ModelInspector.

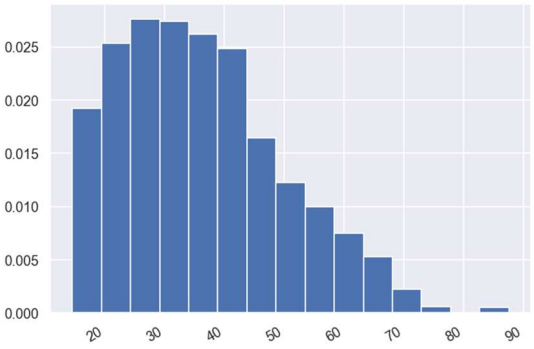
It needs input dataset, synthetic dataset, and attribute description.

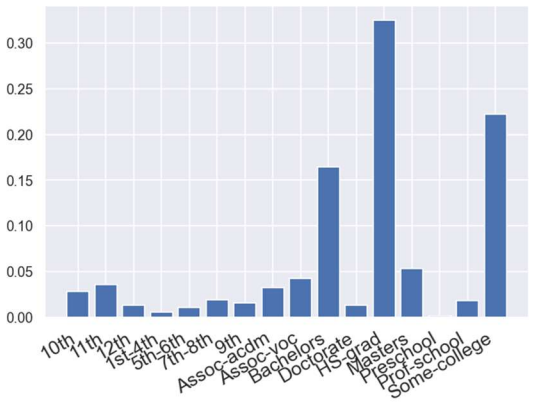
```
# Read both datasets using Pandas.
input_df = pd.read_csv(input_data, skipinitialspace=True)
synthetic_df = pd.read_csv(synthetic_data)
# Read attribute description from the dataset description file.
attribute_description =
read_json_file(description_file)['attribute_description']

inspector = ModelInspector(input_df, synthetic_df, attribute_description)
```

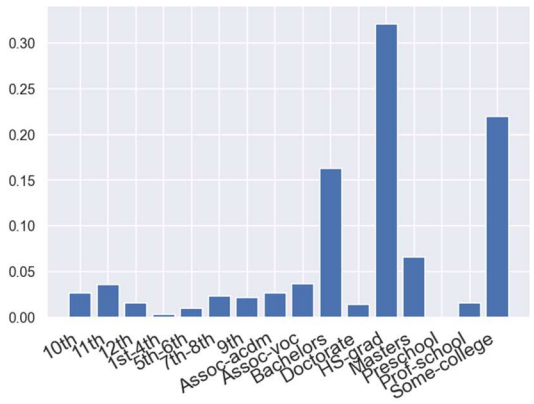
5.2 compare histograms between input and synthetic datasets.

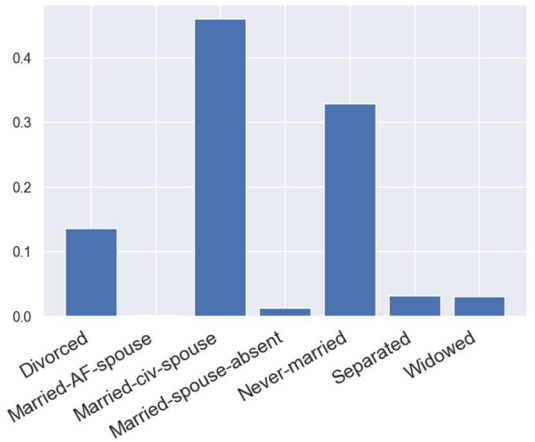
```
for attribute in synthetic_df.columns:
    inspector.compare_histograms(attribute)
```

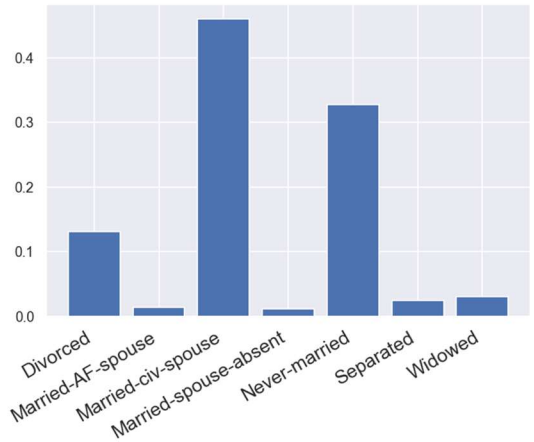

0.025
0.020
0.015
0.010
0.005
0.000
20
30
40
50
60
70
80
90

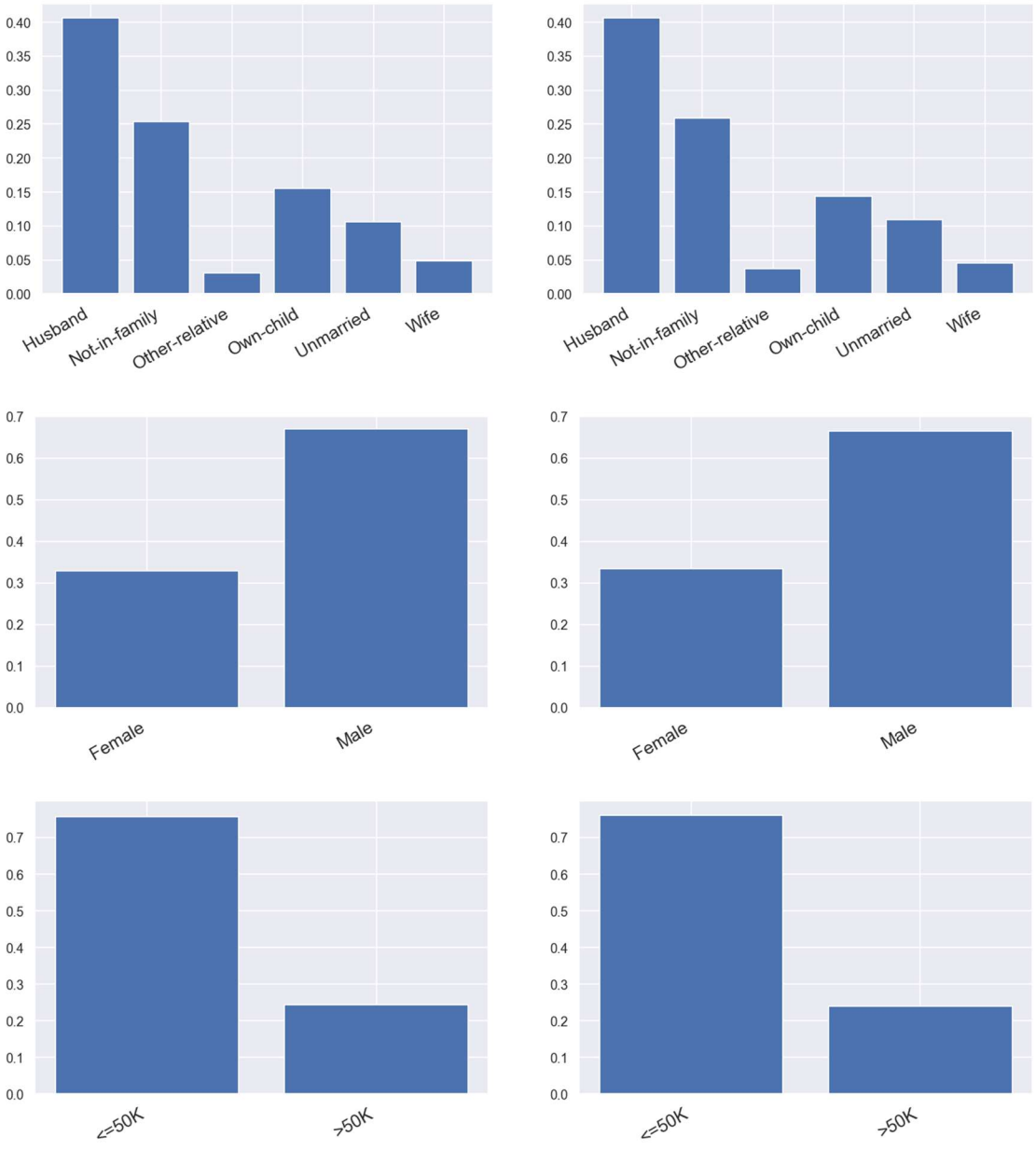

0.025
0.020
0.015
0.010
0.005
0.000
20
30
40
50
60
70
80
90


0.30
0.25
0.20
0.15
0.10
0.05
0.00
10th
11th
12th
1st-4th
5th-6th
7th-8th
9th
Assoc-acdm
Assoc-voc
Bachelors
Doctorate
HS-grad
Masters
Preschool
Prof-school
Some-college


0.30
0.25
0.20
0.15
0.10
0.05
0.00
10th
11th
12th
1st-4th
5th-6th
7th-8th
9th
Assoc-acdm
Assoc-voc
Bachelors
Doctorate
HS-grad
Masters
Preschool
Prof-school
Some-college

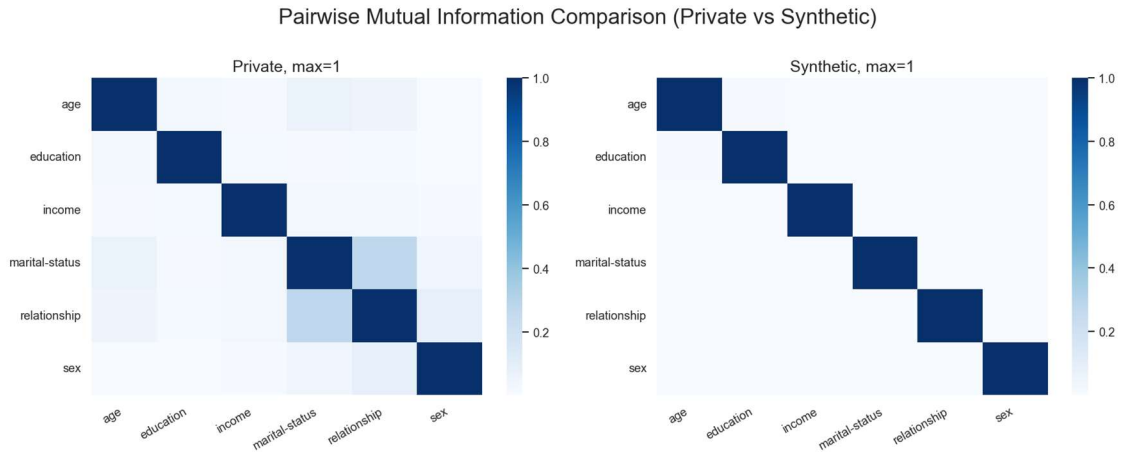

0.4
0.3
0.2
0.1
0.0
Divorced
Married-AF-spouse
Married-civ-spouse
Married-spouse-absent
Never-married
Separated
Widowed


0.4
0.3
0.2
0.1
0.0
Divorced
Married-AF-spouse
Married-civ-spouse
Married-spouse-absent
Never-married
Separated
Widowed



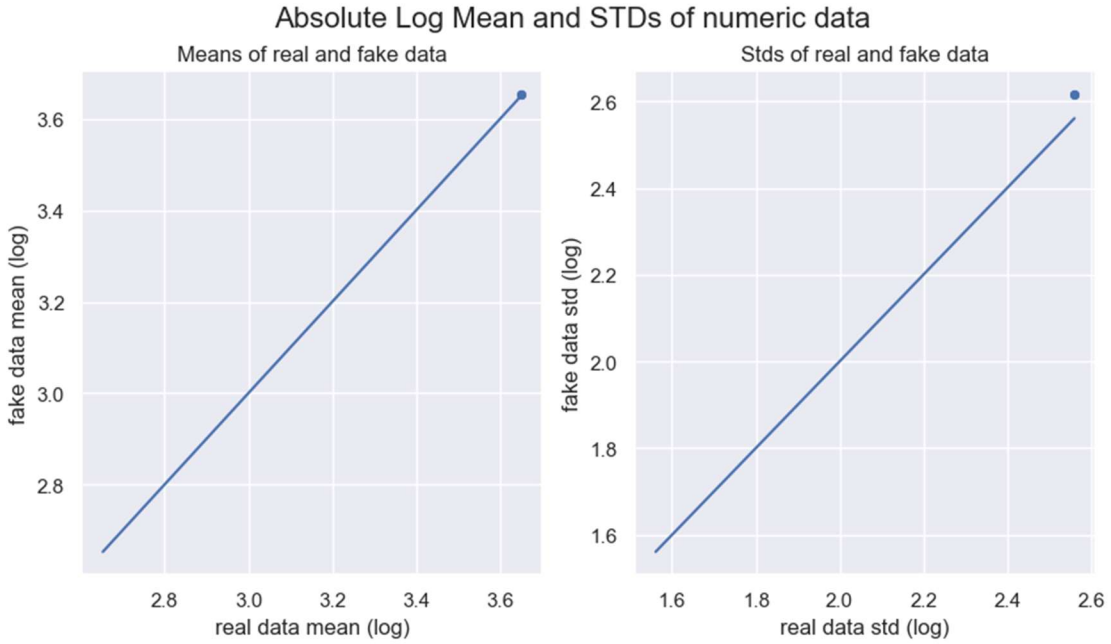
5.3 compare pairwise mutual information

```
inspector.mutual_information_heatmap()
```



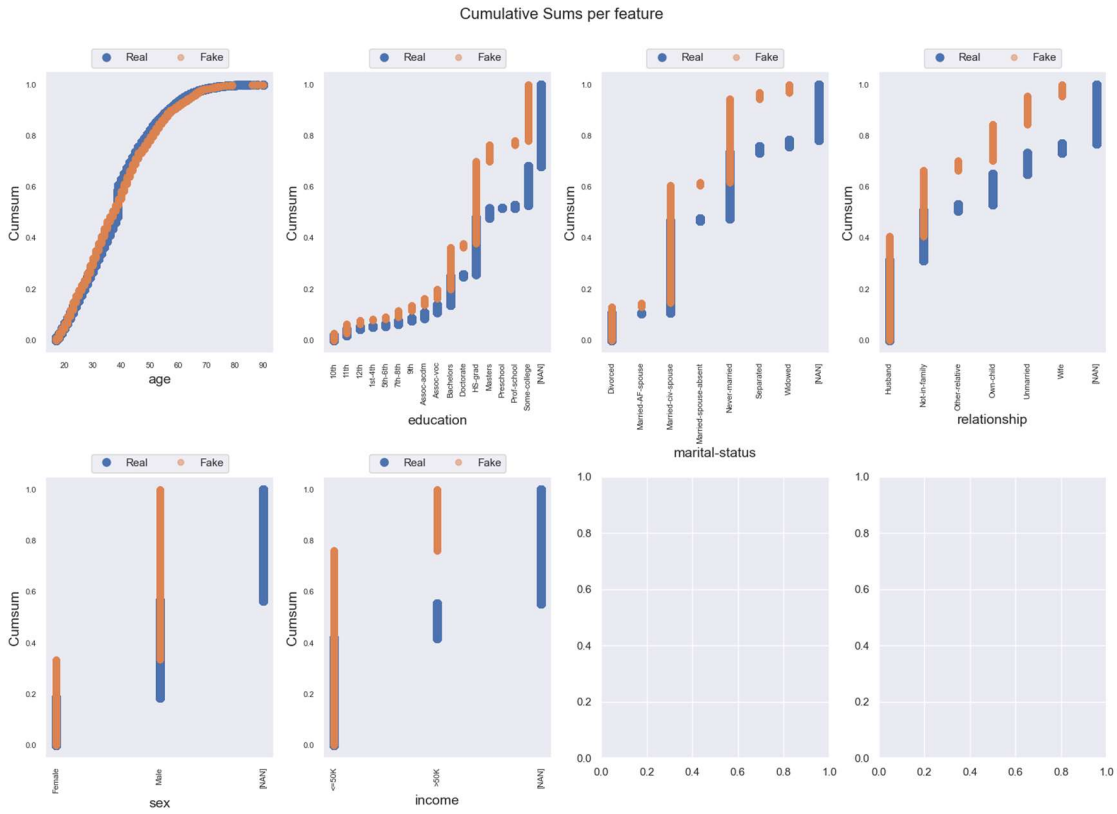

```
column_list = [ "education", "marital-status", "relationship", "sex",
"income"]
from table_evaluator import TableEvaluator
data =input_df
samples = synthetic_df
print(data.shape, samples.shape)
table_evaluator =  TableEvaluator(data, samples, cat_cols= column_list)

table_evaluator.visual_evaluation()
```

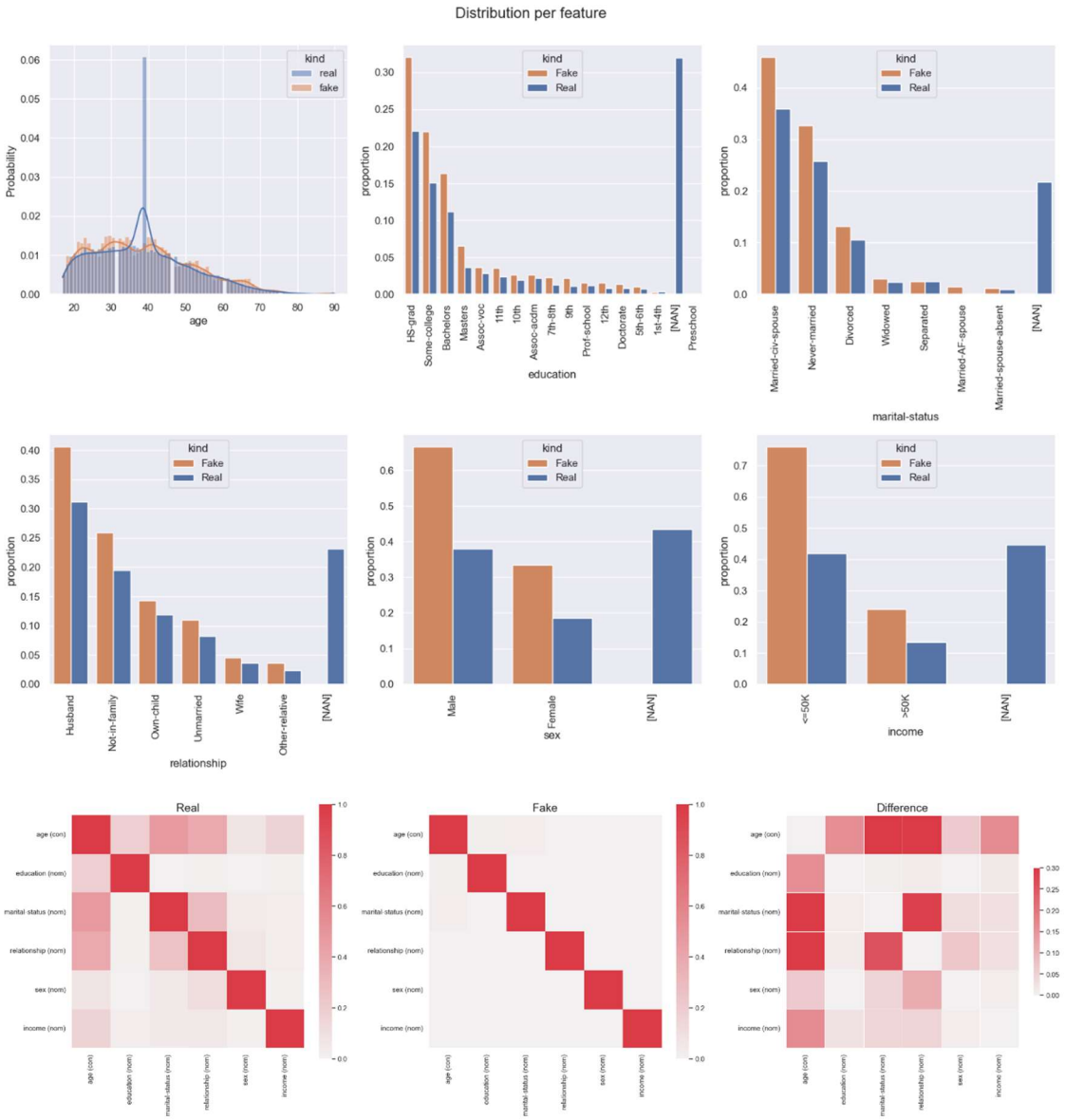
```
(32561, 6) (32561, 6)
```

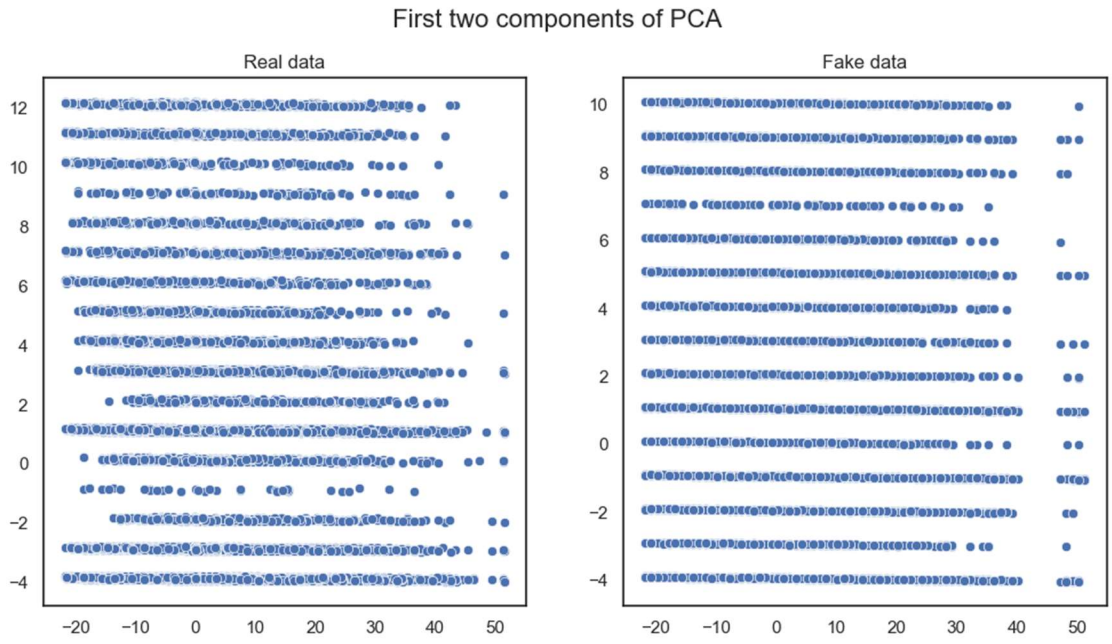

Absolute Log Mean and STDs of numeric data
Means of real and fake data
fake data mean (log)
real data mean (log)
2.8
3.0
3.2
3.4
3.6
Stds of real and fake data
fake data std (log)
real data std (log)
1.6
1.8
2.0
2.2
2.4
2.6

SVKM'S NMIMS Deemed to be UNIVERSITY

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING




Distribution per feature
kind
real
fake
Probability
age
kind
Fake
Real
proportion
education
HS-grad
Some-college
Bachelors
Masters
Assoc-voc
11th
10th
Assoc-acdm
7th-8th
9th
Prof-school
12th
Doctorate
5th-6th
1st-4th
[NAN]
Preschool
marital-status
Married-civ-spouse
Never-married
Divorced
Widowed
Separated
Married-AF-spouse
Married-spouse-absent
[NAN]
relationship
Husband
Not-in-family
Own-child
Unmarried
Wife
Other-relative
[NAN]
sex
Male
Female
[NAN]
income
<=50K
>50K
[NAN]
Real
Fake
Difference
age (con)
education (nom)
marital-status (nom)
relationship (nom)
sex (nom)
income (nom)



Outputs:

Please find the attached .csv files for the synthetic datasets generated using the above algorithms.

Conclusion:

In conclusion, the outcome of our study on the generation of synthetic data using three different algorithms, namely CTGAN, DataSynthesizer, and SMOTE-TOMEK, has provided valuable insights into their respective effectiveness. Our focus was on assessing the similarity between the synthetic data and the original datasets.

After a comprehensive analysis, it is evident that the SMOTE-TOMEK algorithm has demonstrated superior performance in terms of accuracy and data similarity. The first two PCA components of the synthetic data generated by SMOTE-TOMEK exhibited a high level of resemblance to each other. This indicates that

SMOTE-TOMEK effectively preserved the underlying structure and characteristics of the original data, making it a robust choice for synthetic data generation.

While all three algorithms have their merits and applications, this project's results highlight the distinct advantage of SMOTE-TOMEK when the goal is to maintain the integrity of the data's fundamental structure. These findings are not only significant for data privacy and augmentation but also provide a valuable resource for researchers, practitioners, and organizations seeking to leverage synthetic data for various purposes. As technology continues to evolve, the accurate generation of synthetic data becomes increasingly crucial, and the SMOTE-TOMEK algorithm emerges as a promising solution in this context.