Machine Learning - SS 2021
Exercise 3: Synthetic Data

Federico Ambrogi, e1449911@student.tuwien.ac.at
Adam Höfler, e11847620@student.tuwien.ac.at
Matteo Panzeri 12039996@student.tuwien.ac.at
TU Wien

July 26, 2021

Contents

Introduction

In this document we describe the results of the implementation of synthesizing algorithms to create synthetic replicas of three different data sets. We test them for their ability to create replicas of the original data set.

1 Data Sets Description

Here we briefly introduce our datasets, analyze the distributions of the input data and check for significant correlations between variables.

1.1 Income Data Set

The data set [1] provides us information about people, like: age, workclass, education, marital-status, occupation, capital-gain, ecc...
Our purpose is to determine if a person income is lower or greater than a given value specifically the threshold is 50000 Dollars per year. The distribution of a selection of relevant features, as well as a scatter plot to highlight pair-wise correlations are shown in Fig. 5, while a more general view on the correlations including all features are shown in Fig. 6.

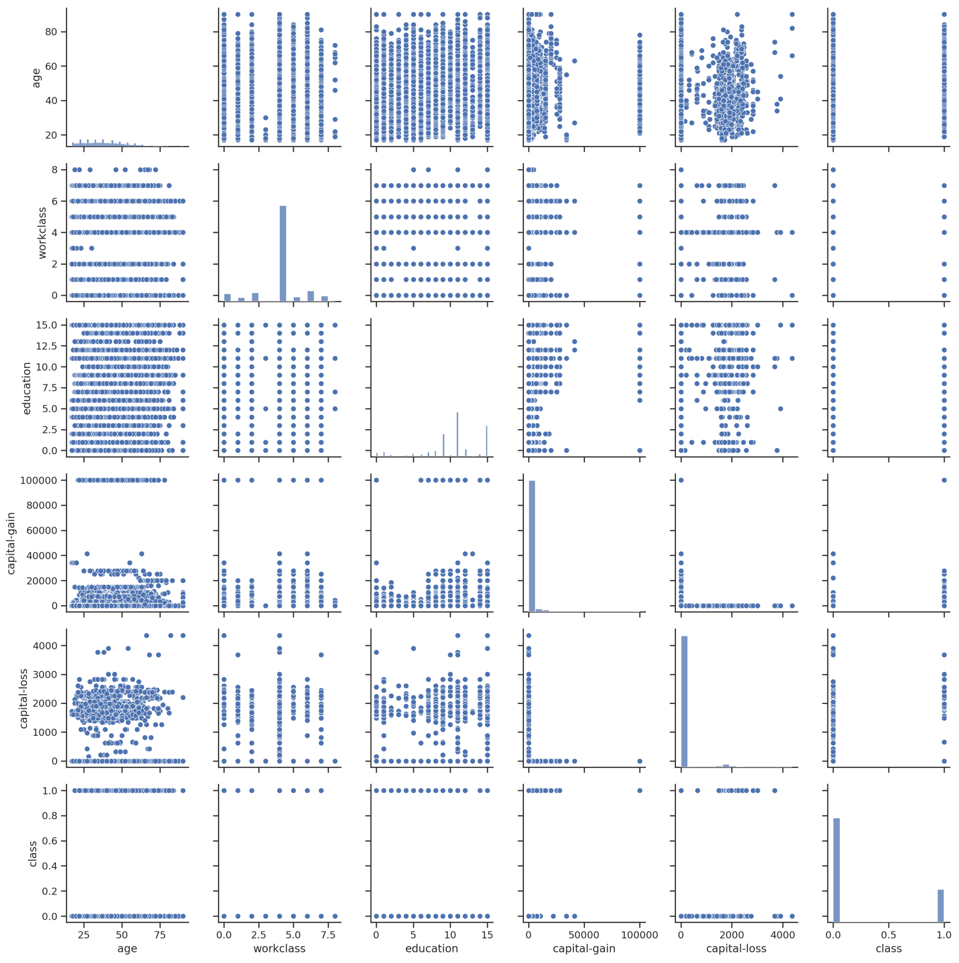

Figure 1: Distributions and pair-wise correlation for the features of the "income" data set.

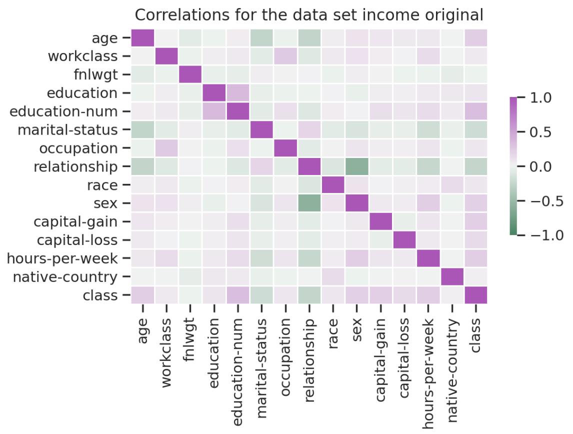

Figure 2: Correlation matrix for the "income" data set.

1.2 Titanic Data Set

The Titanic data set [2] contains informations about the Titanic's passengers. Given the observations we want to determine if a given person, represented by an observation, will survive on the Titanic.
Some features of the dataset are: Name, sex, age, Passenger class, ticket fare, where the passenger embarked, etc.

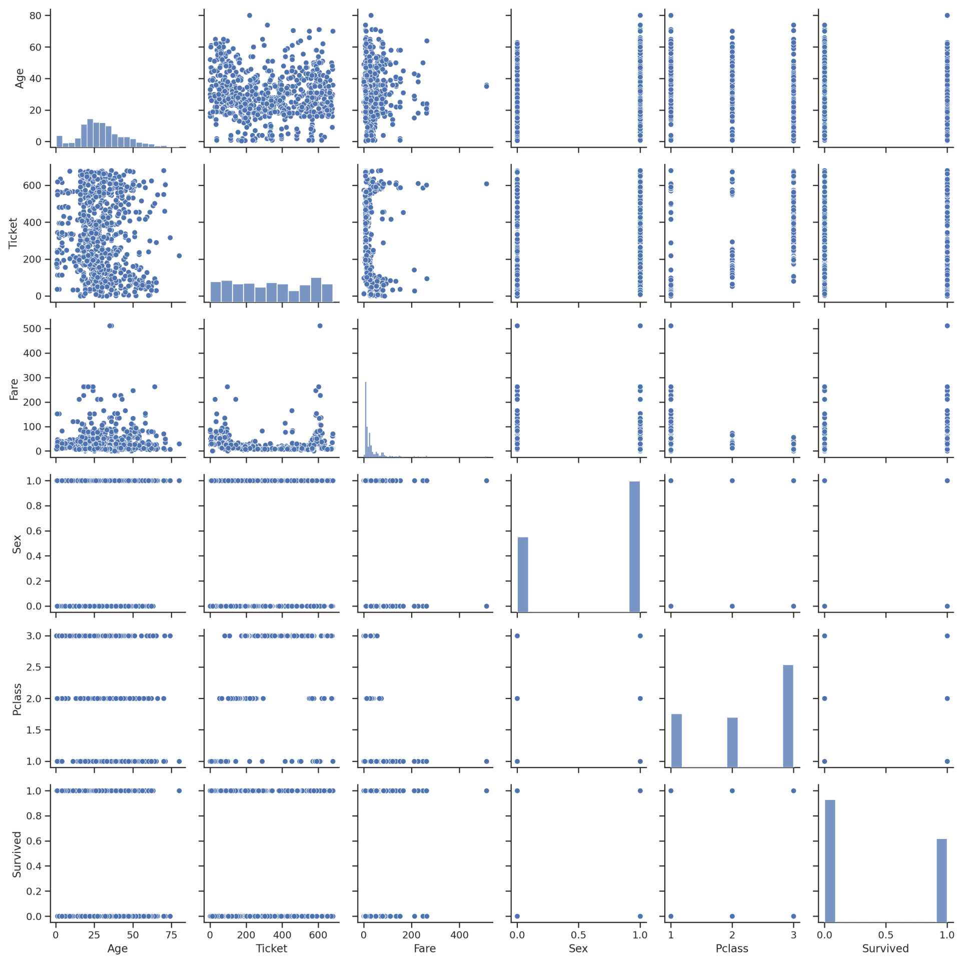

Figure 3: Distributions and pair-wise correlation for the features of the "social" data set.

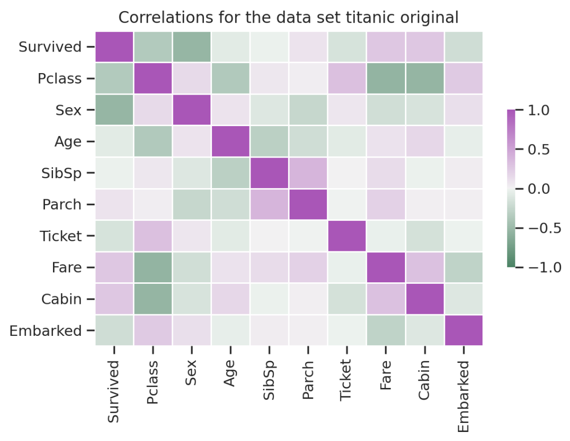

Figure 4: Correlation matrix for the "social" data set.

1.3 Social Data Set

The Social network ads data set [3] collects information about social network users in particular age,sex and income. Our objective is to understand if an user will buy the advised good.

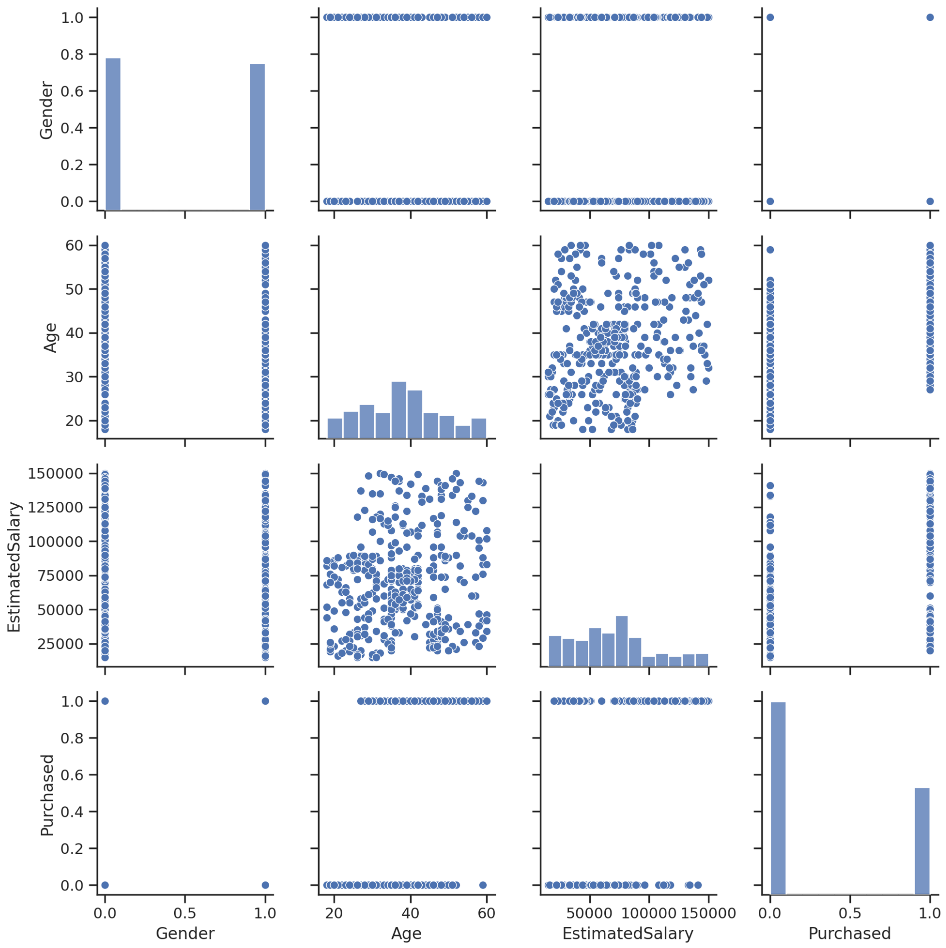

Figure 5: Distributions and pair-wise correlation for the features of the "Titanic" data set.

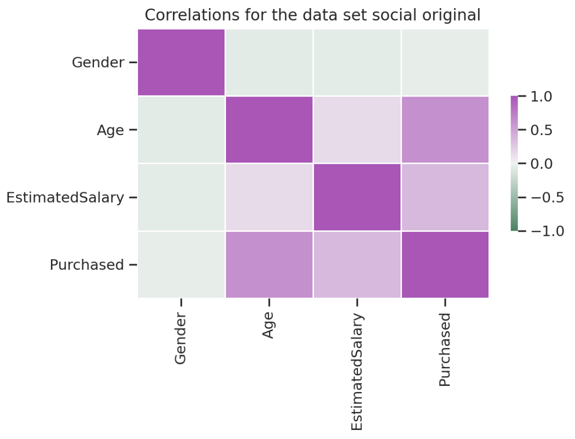

Figure 6: Correlation matrix for the "Titanic" data set.

2 Workflow Overview and Code Structure

Here we outline the structure of the code. The Project is build using the *python3.8* language and various standard libraries such as *pandas* and *numpy* (for basic data import and analysis), *sklearn* (sci-kit learn for model training and evaluation), (for the creation of the synthetic data sets), *matplotlib* and *seaborn* for data visualization.

The project includes five different scripts that perform different tasks:

- *clean_dataset.py*: module for the loading and cleaning of the input data sets. It reads the input data sets ad *pandas* data frames, which are the main format handled by the library for model training. Among the cleaning of the data, we find the outlier removals and data rescaling;
- *generate_data.py*: module for the creating of synthetic data, based on the *SDV* library;
- *classifier.py*: main module of the project, contains the main functionalities to perform all the steps from data loading, cleaning, generation of synthetic data, model training and comparison of results;
- *utils.py*: module containing utilities e.g. for data plotting;

In addition, a directory named ”input_data” is used to contain the input data sets as well as their respective cleaned version. To run the entire project it is sufficient to first call the *clean_dataset.py* module e.g.

```
python3.8 clean_dataset.py
```

This step will produce the cleaned data set inside the same ”input_data” directory as well as some control plots to show the distributions and correlation of the data.

In the second step it is required to run with

```
python3.8 classifier.py
```

that follows the following procedure:

1. splitting of the original cleaned dataset into *train* and *test* sets, using a ratio of $70\% - 30\%$ respectively;
2. training and evaluation of the original dataset using the chosen classifier (random forest);
3. creation of the synthetic data sets, using the training set from the previous splitting of the original dataset, for the three models GaussianCopula,CTGAN and Copula GAN (see 4);
4. evaluate the models trained with the original and the synthetic dataset with the test set, obtained from the original model.

3 Data Exploration and Pre-processing

In this section we describe the necessary preliminary steps to import and prepare the data to make them suitable for the learning algorithms.

The steps are handled by the script *data_preparation.py* which includes dedicated function to pre-process each data set.

3.1 Data Overview

Here we provide an overview of the original data.

Normalization Since we will be working with a random forest classifier, normalization is not needed for the data. Same goes for the SVD methods that should only fit distributions to the data and handle everything on their own.

Labelling In all datasets were present not numeric features. In order to synthetize them was necessary to change the strings into numeric values. In order to do so we used the cat.code method in this way for every not numeric features all the possible values where labelled using a different value. Using the labelling we had the possibility to use the three models.

4 Generation of Synthetic Data

In order to generate the synthetic data we relied on three different generation models. The models are defined in the *SDV* library. The three models aim to generate a synthetic data set whose properties are the same as in the original dataset. The three models we used are: GaussianCopula model, CTGAN model, CopulaGAN model. We now describe the idea behind these three models as reported on the SDV library user guide.

GaussianCopula The Gaussian copula model [4] is based on copulas which are distributions on $[0;1]^d$ which is constructed from a multivariate normal distribution over $\mathbb{R}^d$ by using the probability integral transform. Intuitively, a copula is a mathematical function that allows us to describe the joint distribution of multiple random variables by analyzing the dependencies between their marginal distributions.

CTGAN model The sdv.tabular.CTGAN model [5] is based on the GAN-based Deep Learning data synthesizer which was presented at the NeurIPS 2020 conference by the paper titled ”Modeling Tabular data using Conditional GAN”.

Copula GAN The sdv.tabular.CopulaGAN model [6] is a variation of the CTGAN Model which takes advantage of the CDF based transformation that the GaussianCopulas apply to make the underlying CTGAN model task of learning the data easier.

5 Model Implementation and Evaluation

We follow the schematic in Fig. 7. Thus, splitting the original data set in a train- and test-set (70% train data and 30% test data), synthesizing new data according to the chosen syntheziser methods and traning a classifier on the new and original data set to compare them in a classification task, tested on the test-set.

5.1 Classification

The classification is done with a random forest classifier in a hold-out-validation fashion (using the previously created test-set). Since classification was already done in Exercise 1, we wont go in any further detail on the implementation.

To train our model, we used the *schi-kit learn* implementation of a random forest algorithm *RandomForestClassifier()* with default parameters.

Metrics We use accuracy, precision and, as a combination, the f1-score as a metric for the classification results. Since our task is the creation of data that is close to the original one with respect to the underlying structure, it is not majorly important to achieve good classification results, but to achieve similar results for the original and synthesized data set. Thus, we were not optimizing the classification.

Additionally as an optical metric, we use the confusion matrix to see the results of the classification. In a confusion matrix it is easy to see how well the classification performed by looking at the entries on the main diagonal - off-diagonal entries represent misclassifications. Again, the performance is not important but rather the similarity between the data sets.

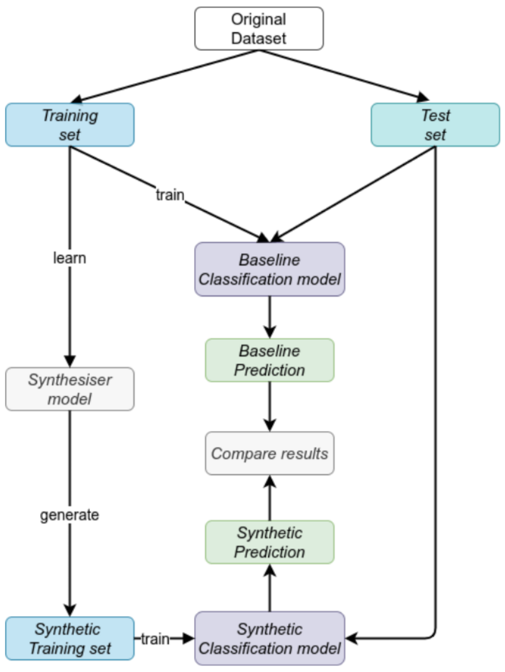

Figure 7: Outline of the used workflow.

5.2 Confusion Matrices

As outlined in the previous section, we classified both the original and synthetic data set and create the corresponding confusion matrices. Each entry i, j corresponds to the number of observations in group i, but predicted to be in group j. We normalize the entries according to the sum of each row. Since it is a binary classification, the matrix reduces to the number of true negatives (TN), false positives (FP), false negatives (FN) and true positives (TP). We then analyzed the performance of the implementation using precision (P) and recall (R), defined as :

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \qquad A = \frac{TP + TN}{all} \tag{1}$$

In this section we present the results of the confusion matrices, obtained after training our random forest classifier on the original data set and on the three different synthetic data sets. Results for the titanic, income and social data sets are shown respectively in Figures 8, 9 and 10.

5.3 Scores

Finally, in Figures 11 , 12 and 13 we show the values for the precision, accuracy and recall obtained after training the random classifier on the 4 different input data sets (original and three synthetic ones).

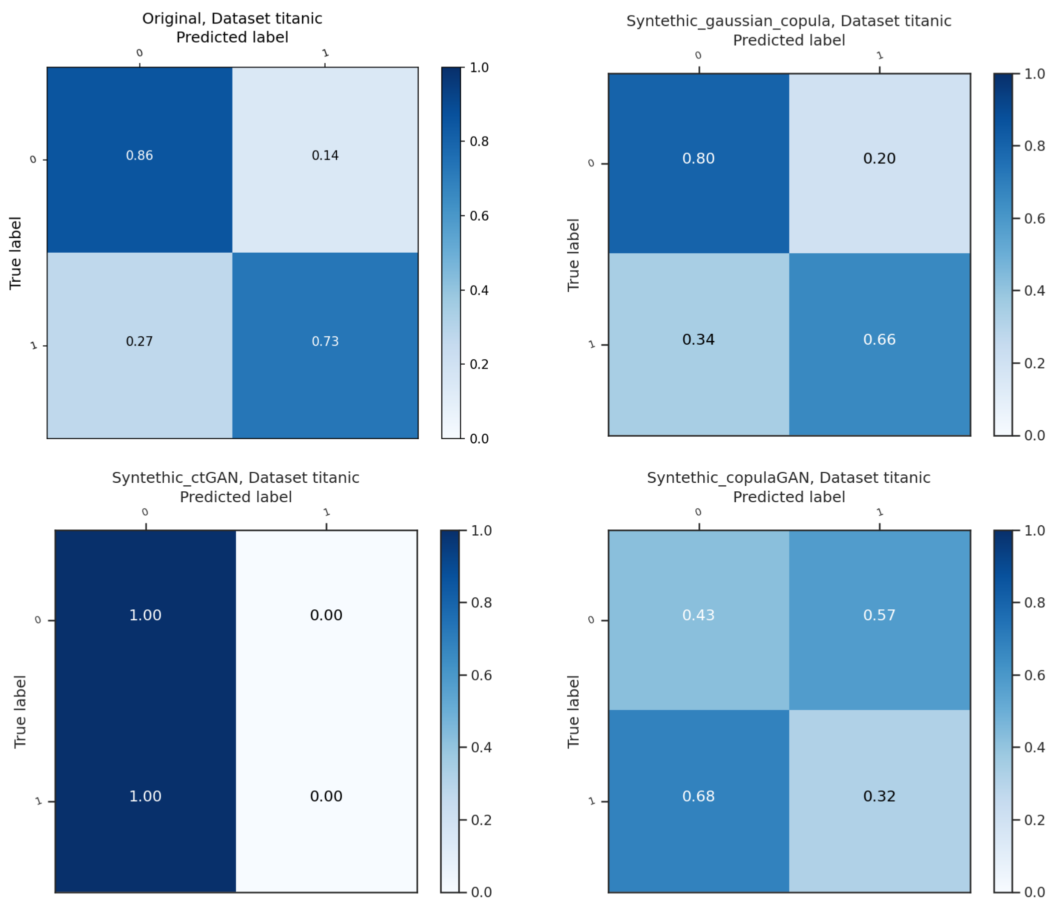

Figure 8: Confusion matrices for the ”Titanic” data set: original dataset (top-left panel), Gaussian Copula synthetic data set (top-right panel), GAN synthetic data set (bottom-left panel) and GAN Copula synthetic data set (bottom-right panel).

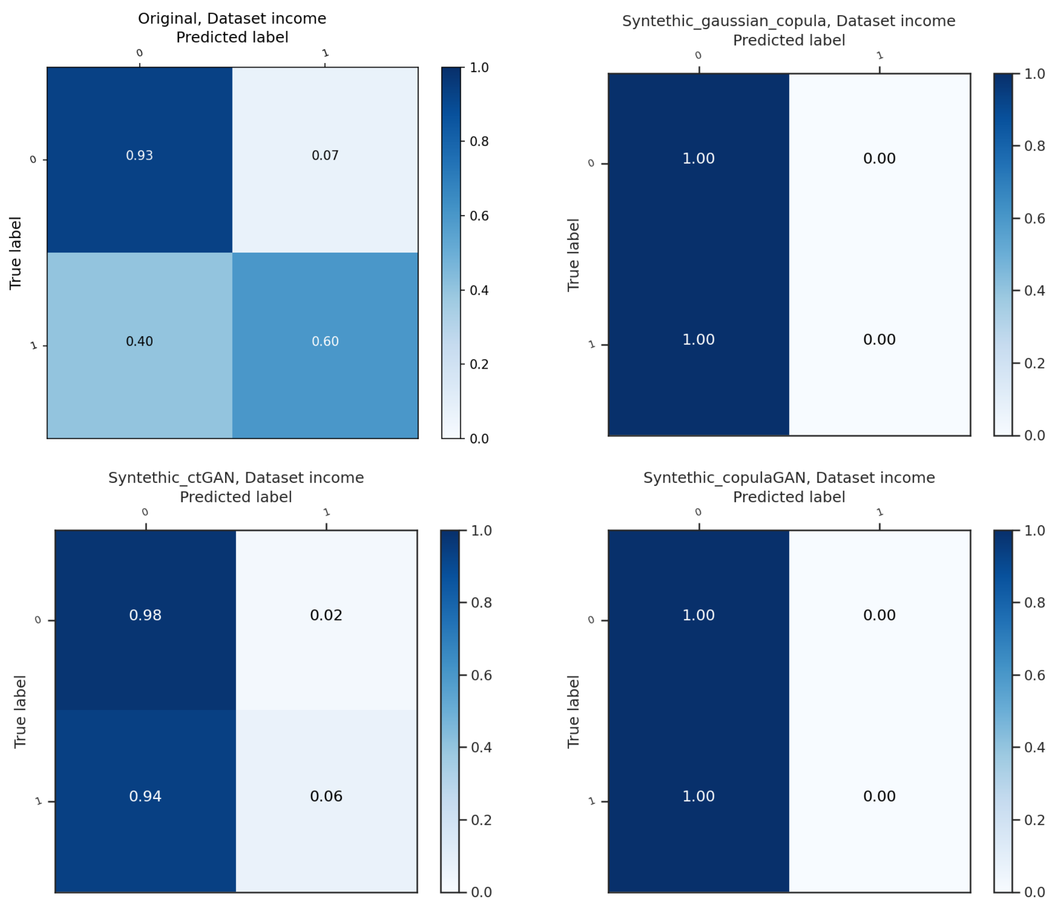

Figure 9: Confusion matrices for the "Income" data set: original dataset (top-left panel), Gaussian Copula synthetic data set (top-right panel), GAN synthetic data set (bottom-left panel) and GAN Copula synthetic data set (bottom-right panel).

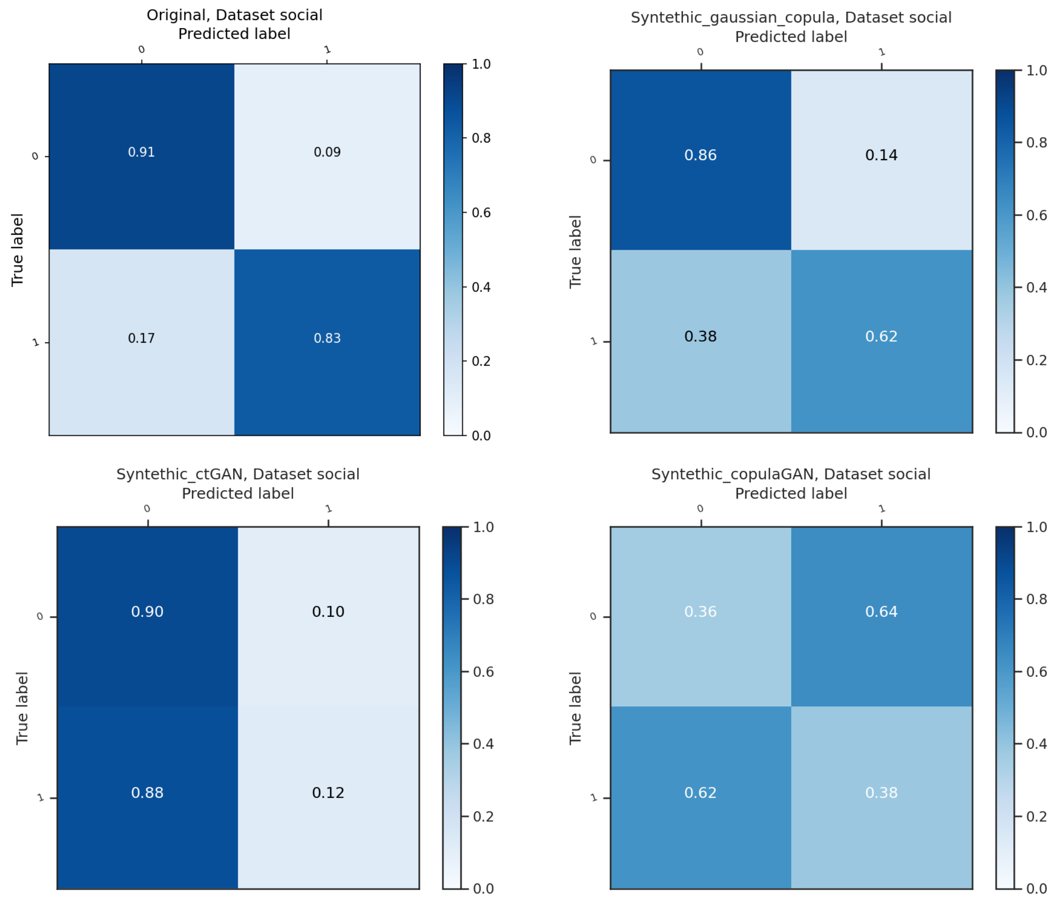

Figure 10: Confusion matrices for the "Social" data set: original dataset (top-left panel), Gaussian Copula synthetic data set (top-right panel), GAN synthetic data set (bottom-left panel) and GAN Copula synthetic data set (bottom-right panel).

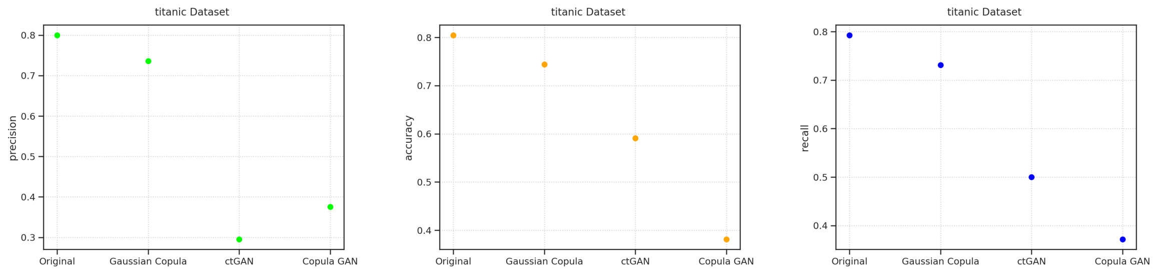

Figure 11: Precision, accuracy and recall scores for the rand forest classifier, trained on the original data set and on the three synthetic data sets, for the "Titanic" data set.

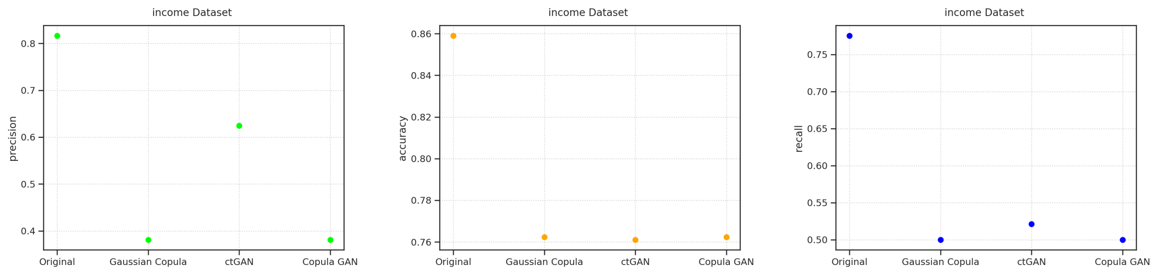

Figure 12: Precision, accuracy and recall scores for the rand forest classifier, trained on the original data set and on the three synthetic data sets, for the "Titanic" data set.

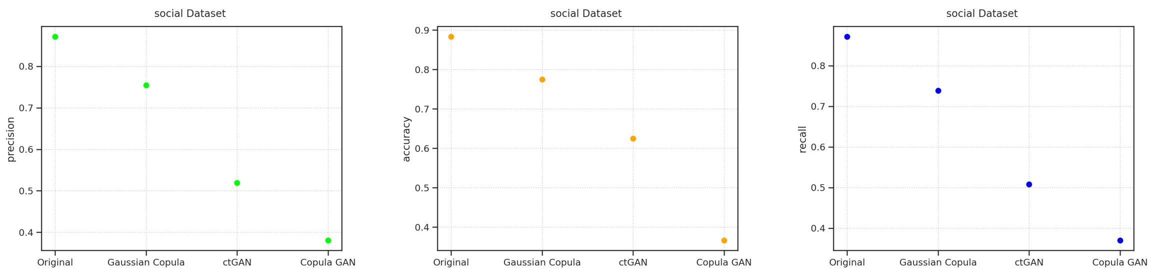

Figure 13: Precision, accuracy and recall scores for the rand forest classifier, trained on the original data set and on the three synthetic data sets, for the "Social" data set.

6 Conclusion

As can be seen in the figures above, the outcome of the data synthesizing depends on the underlying data set and the used SDV method. On the "adult income" dataset all methods produce a data set for which the best strategy of the Random Forest (RF) classifier is to constantly guess the same class. This makes sense when looking at the produced data - mainly data points of one class are produced. We did not find out what could be the cause for this and could be further investigated. But since the documentation on the SDV methods is rather sparse and it is not point of this assignment we will skip this.

For the other two data sets the *GaussianCopula* method was performing the best out of the SDV methods and actually gives quite reasonable results. *ctGAN* gives a low performance on all data sets and the RF classifier returns a trivial strategy (same reasons as before). The result for *copulaGAN* is rather surprising. Having lots of entries off the main diagonal, the synthesizer seems to often swap (binary labels) the labels a data point would normally have - he does this consistently with both classes.

Looking at the metrics we can see, again, that the *GaussianCopula* is performing the best out of our synthesizers. But looking at those metrics one could assume that *ctGAN* performs halfway between *GaussianCopula* and *copulaGAN*. This is of course a deception since both are performing bad, only that *ctGAN* makes correct guesses for one of the classes.

We cannot see what in the "adult income" data set causes it to perform so bad. It has a similar amount of features to the "titanic" data set and has a mixture of categorical and non-categorical features. It is a bit heavy on the categorical features though - testing it on a subset of the features could be a way to test for improvement. It is also by far the biggest data set having 32.560 entries (compared to 891 in the "titanic" and 400 in the "social media ad" data sets). But we don't see how more data could negatively affect the results of the synthesizer methods.

To wrap everything up, we think that the lack of a deep understanding on the production of the synthetic data, was limiting our ability to easily improve the results. We would've had the best shot by simply trying out a lot of things to hope and find an underlying pattern in the response of the model.

References

[1] Ronny Kohavi and Barry Becker. Uci - adult data set. https://archive.ics.uci.edu/ml/datasets/Adult. Accessed: 11.07.2021.

[2] Kaggle - titanic, machine learning from disaster. https://www.kaggle.com/c/titanic/data. Accessed: 11.07.2021.

[3] Kaggle - social network ads. https://www.kaggle.com/rakeshrau/social-network-ads. Accessed: 11.07.2021.

[4] Sdv. https://sdv.dev/SDV/user_guides/single_table/gaussian_copula.html#what-is-gaussiancopula. Accessed: 11.07.2021.

[5] Sdv. https://sdv.dev/SDV/user_guides/single_table/ctgan.html#what-is-ctgan. Accessed: 11.07.2021.

[6] Sdv. https://sdv.dev/SDV/user_guides/single_table/copulagan.html. Accessed: 11.07.2021.