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 25, 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 distribution of the 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 are shown in Fig. 6.
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.

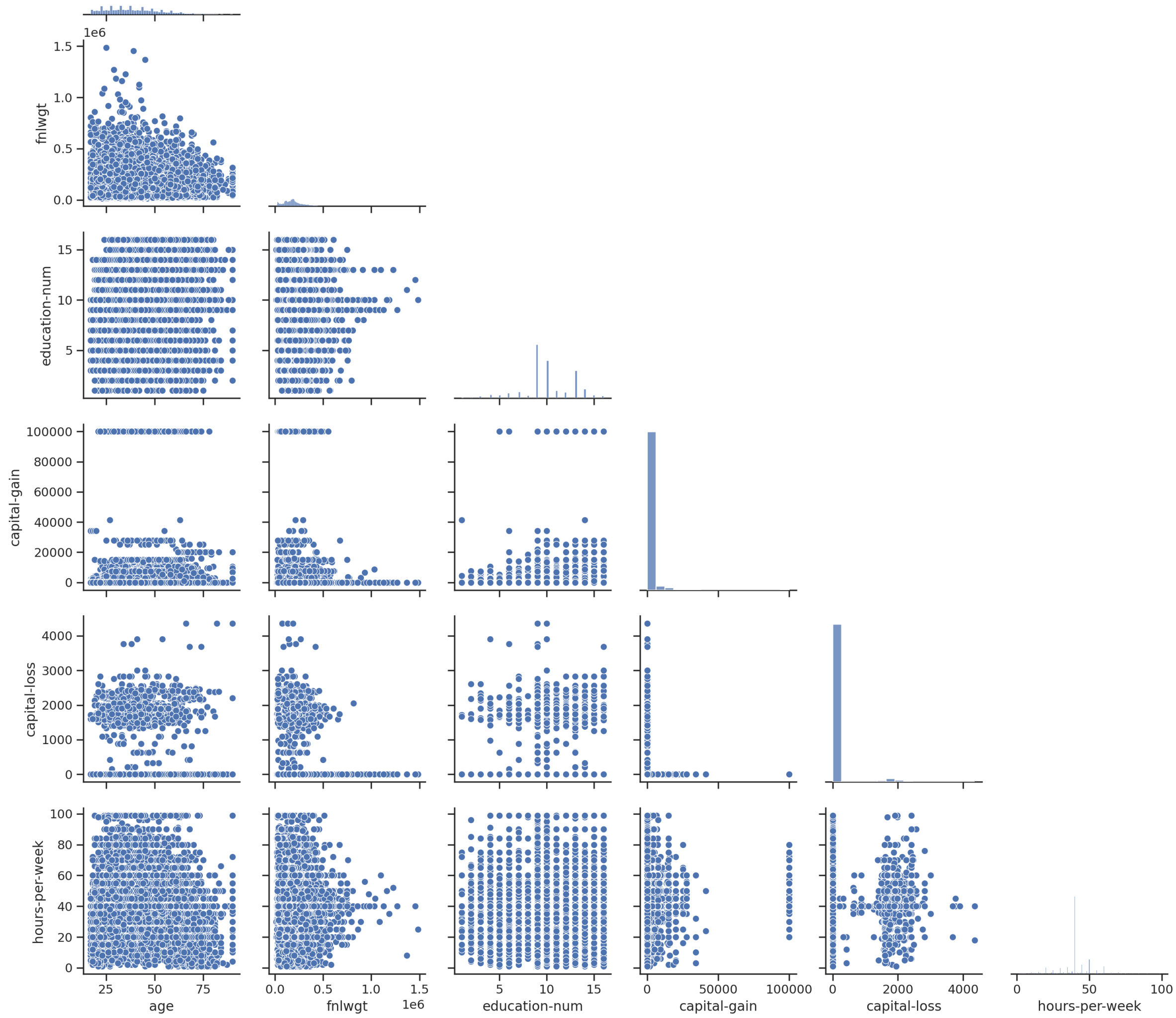

Figure 1: Distributions and pair-wise correlation for the features of the ”income” data set.

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.

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.

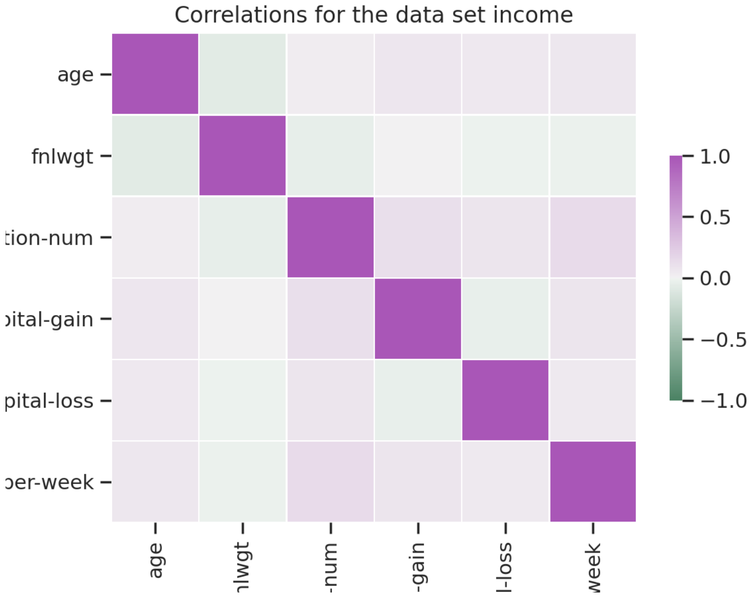

Figure 2: Correlation matrix for the "income" data set.

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;

To run the entire project it is sufficient to call the *classifier.py* module e.g.

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

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.

Missing Value The

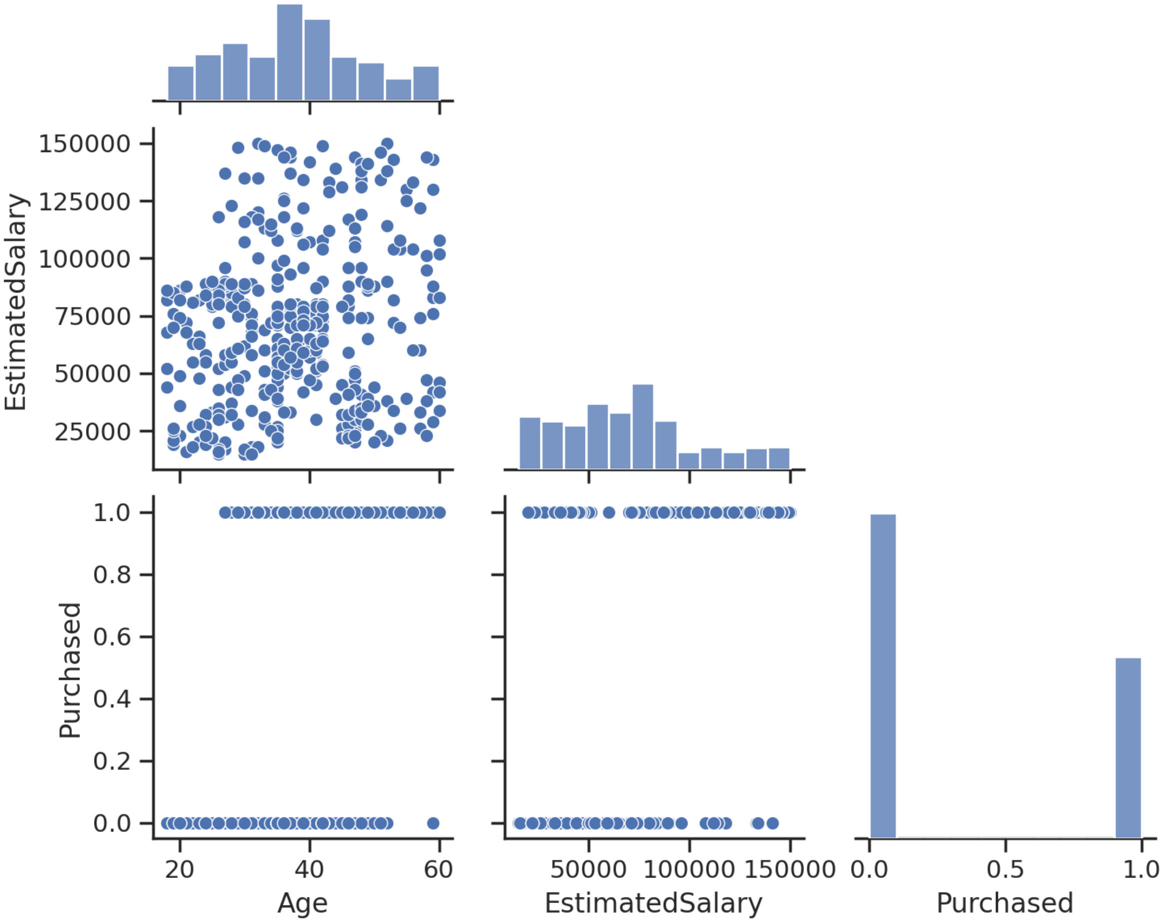

Figure 3: Distributions and pair-wise correlation for the features of the ”social” data set.

Normalization The

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.

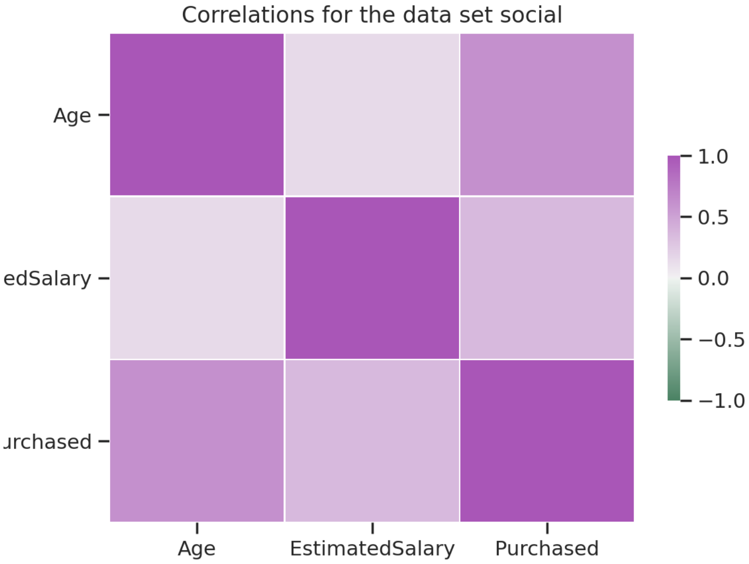

Figure 4: Correlation matrix for the "social" data set.

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

5.1 Holdout and Cross Validation

Here we describe briefly the techniques of "holdout" and "cross validation" that are used to evaluate a model. The **holdout** method is essentially based on the splitting on the input data set into two subset, one used for training the model, and one used for testing the model, for example in $80\% - 20\%$ proportion, although there is not fix recipe for this split. Once the model is trained, the evaluation can be performed on the test data set, and this is therefore possible to check if the prediction of the model match the data. One big issue of this model is that the training strongly depend on the splitting of the initial data set, for example if the characteristic of the training data set are not representative of the whole data set.

A more powerful method is the **cross-validation** or "k-fold cross validation". Form the full dataset, a test is held out for final evaluation, but the validation set is no longer needed. For this, the training set is split into k sets, so that the model is trained using $k - 1$ folds as training data, and the remaining one is used for the validation as a test set (to compute the interesting metric of the model). Once we obtain such k number of metrics, the final result is the average of these parameters, obtained for each iteration of the cross-validation on each distinct fold.

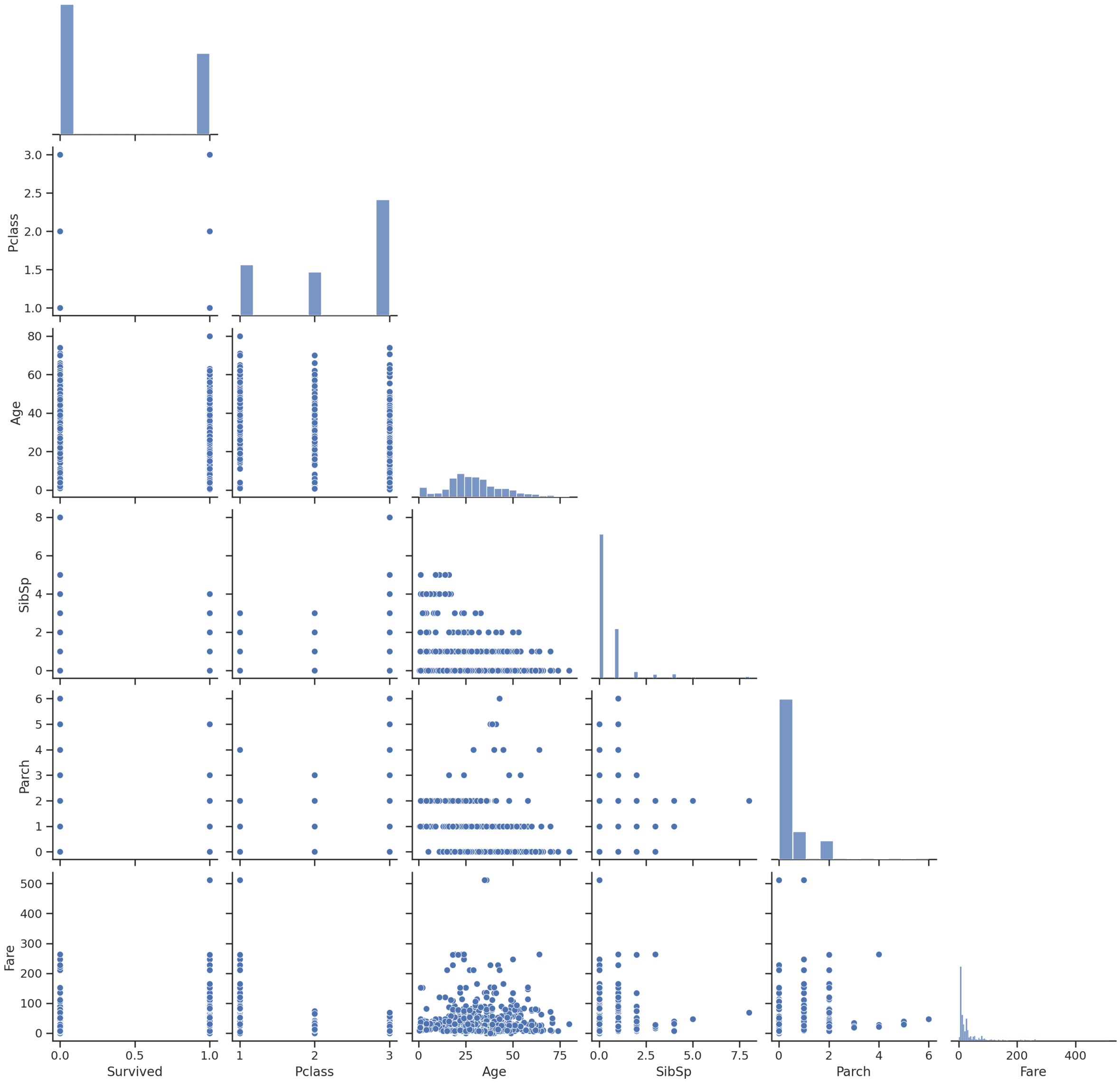

Figure 5: Distributions and pair-wise correlation for the features of the ”Titanic” data set.

6 Performance Tests

Confusion Matrix For each classifier, we produce a confusion matrix where each entry i, j corresponds to the number of observations in group i, but predicted to be in group j. We chose to normalize the entries according to the sum of each row. In case of binary classification, the matrix reduces to the number of true negatives (TN), false positives (FP), false negatives (FN) and true positives (TP).

Examples can be see in Fig. **??**.

We remind here the definition of the metric parameters we will used to quantify the performance of our classifiers i.e. precision (P), recall (R) , accuracy (A) and specificity S:

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

It is straightforward to calculate these parameters for binary classification tasks out of the confusion matrix. In case of multiple labels, we need to calculate these parameter for each class, given that:

1. TPs are the values in the diagonal;

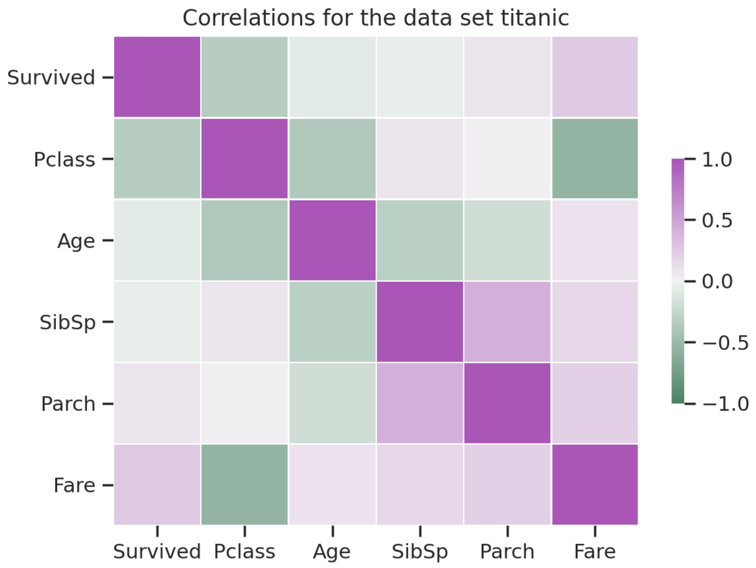

Figure 6: Correlation matrix for the "Titanic" data set.

2. FNs for a certain class are the sum of values in the corresponding row excluding the TP;
3. FPs for a certain class are the sum of values in the corresponding column excluding the TP;
4. TNs for a certain class are the sum of all rows and columns, excluding the class's column and row.

Another convenient metric, particularly because it is calculated directly from the proper *scikit-learn* function, is called $f1 - score$, which is defined as the harmonic mean of the precision and recall:

$$f1 - score = 2 \times \frac{P\dot{R}}{P + R} \tag{2}$$

Describe micro averaging and macro averaging, will we report both ?

7 Conclusion

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.