Generative modeling of mixed tabular data with the R package `arf`

Jan Kapar[1,2], David S. Watson, Kristin Blesch, and Marvin N. Wright

[1]Leibniz Institute for Prevention Research & Epidemiology – BIPS

[2]Faculty of Mathematics and Computer Science, University of Bremen

July 10, 2024

UseR! 2024

The hype about generative modeling

Generative modeling of mixed tabular data with `arf`

Motivation

The hype about generative modeling

Generative AI is revolutionizing content creation by producing human-like text, images, and music. Its ability to generate high-quality, original content quickly is driving widespread interest and adoption.

The hype about generative modeling

Generative AI is revolutionizing content creation by producing human-like text, images, and music. Its ability to generate high-quality, original content quickly is driving widespread interest and adoption.

Generative modeling of mixed tabular data with `arf`

Motivation

The hype about generative modeling

The hype about generative modeling

- Text, speech, image and video synthesis
- Deep-learning-based methods
- Implementations almost exclusively in Python

The hype about generative modeling

- Text, speech, image and video synthesis
- Deep-learning-based methods
- Implementations almost exclusively in Python

What about tabular data?

- Less research
- Existing deep learning solutions often fail

Generative modeling of mixed tabular data with `arf`

The hype about generative modeling

- Text, speech, image and video synthesis
- Deep-learning-based methods
- Implementations almost exclusively in Python

What about tabular data?

- Less research
- Existing deep learning solutions often fail

$\Longrightarrow$ `arf` package for tabular data synthesis based on `ranger`

Generative modeling

Definition

Given a data set consisting of vectors $\boldsymbol{x} \in \mathbb{R}^d$:

Generative modeling

Definition

Given a data set consisting of vectors $\boldsymbol{x} \in \mathbb{R}^d$:

- Generative modeling = modeling the joint density $p(\boldsymbol{x})$

Generative modeling

Definition

Given a data set consisting of vectors $\boldsymbol{x} \in \mathbb{R}^d$:

- Generative modeling = modeling the joint density $p(\boldsymbol{x})$

  (vs. discriminative modeling = directly modeling the conditional density $p(y|X = \boldsymbol{x})$ for some dependent variable $y \in \mathbb{R}$)

Generative modeling

Definition

Given a data set consisting of vectors $\boldsymbol{x} \in \mathbb{R}^d$:

- Generative modeling = modeling the joint density $p(\boldsymbol{x})$

  (vs. discriminative modeling = directly modeling the conditional density $p(y|X = \boldsymbol{x})$ for some dependent variable $y \in \mathbb{R}$)

- Not always explicit density estimation

Generative modeling

Use cases for synthetic data

- Direct usage:
  - Chatbots
  - Translation
  - Transcription
  - Image generation
  - ...

Generative modeling

Use cases for synthetic data

- Direct usage:
  - Chatbots
  - Translation
  - Transcription
  - Image generation
  - ...

- Indirect usage:
  - Data augmentation
  - Data balancing
  - Missing data imputation
  - Privacy preservation
  - ...

The package `arf`

Overview

Implementation of adversarial random forests[1]:

[1] Watson et al. [2023]

The package `arf`

Overview

Implementation of adversarial random forests[1]:

- Fast
  - Built on `data.table`
  - Optional parallelization via `future`, `doParallel`
  - Fast random forest implementation `ranger`

[1] Watson et al. [2023]

The package `arf`

Overview

Implementation of adversarial random forests[1]:

- Fast
  - Built on `data.table`
  - Optional parallelization via `future`, `doParallel`
  - Fast random forest implementation `ranger`
- Easy-to-use

[1] Watson et al. [2023]

The package `arf`

Overview

Implementation of adversarial random forests[1]:

- Fast
  - Built on `data.table`
  - Optional parallelization via `future`, `doParallel`
  - Fast random forest implementation `ranger`
- Easy-to-use
- No tuning required

[1] Watson et al. [2023]

The package `arf`

Overview

Implementation of adversarial random forests[1]:

- Fast
  - Built on `data.table`
  - Optional parallelization via `future`, `doParallel`
  - Fast random forest implementation `ranger`
- Easy-to-use
- No tuning required
- Able to handle missing values (`ranger` branch 'missing_values')

[1] Watson et al. [2023]

The package `arf`

Overview

Implementation of adversarial random forests[1]:

- Fast
  - Built on `data.table`
  - Optional parallelization via `future`, `doParallel`
  - Fast random forest implementation `ranger`
- Easy-to-use
- No tuning required
- Able to handle missing values (`ranger` branch 'missing_values')
- Comparable or superior performance on (mixed) tabular data

[1] Watson et al. [2023]

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

- Iterative approach: Train random forest classifiers to distinguish real data from naive synthetic data drawn from marginals of real data

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

- Iterative approach: Train random forest classifiers to distinguish real data from naive synthetic data drawn from marginals of real data
- Partitioning of data manifold into areas where **independence w.r.t. the variables** can be assumed locally

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

- Iterative approach: Train random forest classifiers to distinguish real data from naive synthetic data drawn from marginals of real data
- Partitioning of data manifold into areas where **independence w.r.t. the variables** can be assumed locally

Step 2: Estimate mixture distribution (`forde()`)

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

- Iterative approach: Train random forest classifiers to distinguish real data from naive synthetic data drawn from marginals of real data
- Partitioning of data manifold into areas where **independence w.r.t. the variables** can be assumed locally

Step 2: Estimate mixture distribution (`forde()`)

- Estimate univariate densities per variable within leaves

The package `arf`

Algorithm and functions

Step 1: Grow adversarial random forests (`adversarial_rf()`)

- Iterative approach: Train random forest classifiers to distinguish real data from naive synthetic data drawn from marginals of real data
- Partitioning of data manifold into areas where **independence w.r.t. the variables** can be assumed locally

Step 2: Estimate mixture distribution (`forde()`)

- Estimate univariate densities per variable within leaves
- Form mixture distribution over leaves weighted by share of real data

The package `arf`

Algorithm and functions

Step 3: Use mixture distribution for

The package `arf`

Algorithm and functions

Step 3: Use mixture distribution for

- (Conditional) sampling (`forge()`)

The package `arf`

Algorithm and functions

Step 3: Use mixture distribution for

- (Conditional) sampling (`forge()`)
- (Conditional) likelihood computation (`lik()`)

The package `arf`

Algorithm and functions

Step 3: Use mixture distribution for

- (Conditional) sampling (`forge()`)
- (Conditional) likelihood computation (`lik()`)
- (Conditional) expectation computation (`expct()`)

The package `arf`

Machine learning utility evaluation

Dataset	Characteristics	Model	Accuracy (sd)	F1 (sd)	Time in sec
`adult`	classes = 2	*Real*	*0.828 (0.006)*	*0.884 (0.004)*	
	n = 32,561	ARF	**0.819** (0.006)	**0.877** (0.005)	**2.9**
	d = 14	CTGAN	0.786 (0.020)	0.853 (0.019)	263.3
		CTABGAN+	0.808 (0.008)	0.869 (0.006)	561.6
		TVAE	0.804 (0.007)	0.865 (0.006)	115.1
`census`	classes = 2	*Real*	*0.922 (0.002)*	*0.957 (0.001)*	
	n = 298,006	ARF	0.903 (0.019)	0.946 (0.012)	**53.2**
	d = 40	CTGAN	0.916 (0.015)	0.954 (0.009)	4287.8
		CTABGAN+	0.912 (0.026)	0.952 (0.016)	10182.1
		TVAE	**0.928** (0.007)	**0.961** (0.004)	1814.9
`covertype`	classes = 7	*Real*	*0.895 (0.000)*	*0.838 (0.000)*	
	n = 581,012	ARF	**0.707** (0.006)	**0.549** (0.006)	**103.5**
	d = 54	CTGAN	0.633 (0.009)	0.400 (0.009)	13387.2
		CTABGAN+	NA	NA	$>$24h
		TVAE	0.698 (0.013)	0.459 (0.013)	4882.0

The package `arf`

Machine learning utility evaluation

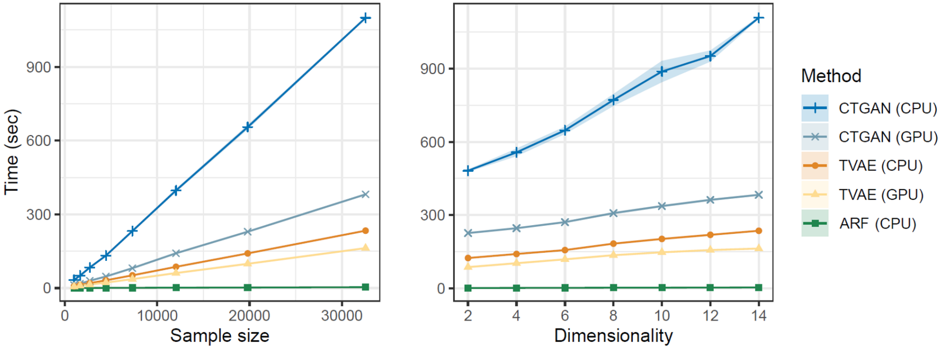
Dataset	Characteristics	Model	Accuracy (sd)	F1 (sd)	Time in sec
`credit`	classes = 2	*Real*	*0.997 (0.001)*	*0.607 (0.029)*	
	n = 284,807	ARF	0.995 (0.001)	**0.527** (0.036)	**32.2**
	d = 30	CTGAN	0.881 (0.099)	0.047 (0.031)	4898.0
		CTABGAN+	**0.998** (0.000)	0.000 (0.000)	7497.3
		TVAE	**0.998** (0.000)	0.000 (0.000)	3847.6
`intrusion`	classes = 5	*Real*	*0.998 (0.001)*	*0.833 (0.001)*	
	n = 494,021	ARF	**0.993** (0.001)	**0.656** (0.001)	**68.2**
	d = 40	CTGAN	0.944 (0.088)	0.645 (0.088)	8749.3
		CTABGAN+	NA	NA	$>$24h
		TVAE	0.990 (0.002)	0.598 (0.002)	4306.0

Omitted `IT-GAN` and `RCC-GAN`.

Watson et al. [2023]

The package `arf`

Runtime



Watson et al. [2023]

The package `arf`

Simple examples: Data synthesis

Generate synthetic samples from the `palmerpenguins` dataset:

The package `arf`

Simple examples: Data synthesis

Generate synthetic samples from the `palmerpenguins` dataset:

```
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```
params <- forde(arf, penguins)
forge(params, 5)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.4          17.1               197        3380
## 2 Adelie    Biscoe           42            18.8               190        4152
## 3 Chinstrap Dream            51.3          19.2               193        3407
## 4 Adelie    Dream            36.5          17.2               191        3779
## 5 Adelie    Biscoe           40.5          19.2               193        3919
## # i 2 more variables: sex <fct>, year <int>
```

The package arf

Simple examples: Data synthesis

Generate synthetic samples from the `palmerpenguins` dataset:

```
library(arf)
arf <- adversarial_rf(penguins, num_trees = 10, min_node_size = 2)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```
params <- forde(arf, penguins)
forge(params, 5)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.4          17.1               197        3380
## 2 Adelie    Biscoe           42            18.8               190        4152
## 3 Chinstrap Dream            51.3          19.2               193        3407
## 4 Adelie    Dream            36.5          17.2               191        3779
## 5 Adelie    Biscoe           40.5          19.2               193        3919
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: Data synthesis

Generate synthetic samples from the `palmerpenguins` dataset:

```
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```
params <- forde(arf, penguins, family = 'truncnorm', finite_bounds = 'no')
forge(params, 5)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.4          17.1               197        3380
## 2 Adelie    Biscoe           42            18.8               190        4152
## 3 Chinstrap Dream            51.3          19.2               193        3407
## 4 Adelie    Dream            36.5          17.2               191        3779
## 5 Adelie    Biscoe           40.5          19.2               193        3919
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: Data synthesis

Generate samples under some desired conditions:

The package `arf`

Simple examples: Data synthesis

Generate samples under some desired conditions:

```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
forge(params, 5, evidence = data.frame(island = 'Torgersen'))
```

```
## # A tibble: 5 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           45.8          20.8               192        4612
## 2 Adelie  Torgersen           45.6          19.3               196        3484
## 3 Adelie  Torgersen           36            20.9               190        3746
## 4 Adelie  Torgersen           36.6          20.8               191        3563
## 5 Adelie  Torgersen           40.6          17.7               180        3162
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: Data synthesis

Generate samples under some desired conditions:

```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
forge(params, 5, evidence = data.frame(island = 'Torgersen', bill_length_mm = '>40'))
```

```
## # A tibble: 5 x 8
##   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
## 1 Adelie  Torgersen           41.7          19.8               208        3647
## 2 Adelie  Torgersen           41.1          21.3               200        4535
## 3 Adelie  Torgersen           45.4          18.8               204        4852
## 4 Adelie  Torgersen           40.2          19.3               197        3661
## 5 Adelie  Torgersen           43.6          20.9               194        4268
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: (Conditional) likelihoods

The package `arf`

Simple examples: (Conditional) likelihoods


```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
lik(params, penguins[1:3,])
```

```
## [1] -3.655597 -4.023504 -4.154264
```

The package `arf`

Simple examples: (Conditional) likelihoods


```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
lik(params, penguins[1:3,], log = F)
```

```
## [1] 0.02584605 0.01789017 0.01569734
```

The package `arf`

Simple examples: (Conditional) likelihoods


```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
lik(params, penguins[1:3,], evidence = data.frame(sex = 'female'))
```

```
## [1] -4.179873 -3.654580 -4.259066
```

The package `arf`

Simple examples: (Conditional) expectations

The package `arf`

Simple examples: (Conditional) expectations


```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
expct(params)
```

```
## # A tibble: 1 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>           <dbl>         <dbl>             <dbl>       <dbl>
## 1 Adelie  Biscoe           43.9          17.2              201.       4200.
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: (Conditional) expectations


```r
library(arf)
arf <- adversarial_rf(penguins)
```

```
## Iteration: 0, Accuracy: 79.88%
## Iteration: 1, Accuracy: 44.95%
```

```r
params <- forde(arf, penguins)
expct(params, query = "flipper_length_mm", evidence = data.frame(body_mass_g = 3300))
```

```
## # A tibble: 1 x 1
##   flipper_length_mm
##               <dbl>
## 1              189.
```

The package `arf`

Simple examples: Shortcut functions

The package arf

Simple examples: Shortcut functions


```r
library(arf)
rarf(penguins, 3)
```

```
## # A tibble: 3 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.1          17.6               200        3408
## 2 Adelie    Biscoe           41.4          19                 183        3749
## 3 Chinstrap Dream            50.7          19.4               190        3484
## # i 2 more variables: sex <fct>, year <int>
```

```r
darf(penguins, query = penguins[1:3,])
```

```
## [1] -5.490055 -5.394553 -5.905513
```

```r
earf(penguins)
```

```
## # A tibble: 1 x 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>   <fct>           <dbl>         <dbl>             <dbl>       <dbl>
## 1 Adelie  Biscoe           43.9          17.2              201.       4201.
```

The package `arf`

Simple examples: Parallelization

The package `arf`

Simple examples: Parallelization


```r
library(arf)
doFuture::registerDoFuture()
future::plan("multisession", workers = 4)
rarf(penguins, 5)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Chinstrap Dream            52.1          17.9               185        3409
## 2 Gentoo    Biscoe           44.3          13.9               210        5479
## 3 Chinstrap Dream            51.5          19.8               198        4063
## 4 Gentoo    Biscoe           50.7          14.9               219        6256
## 5 Gentoo    Biscoe           49.3          16.1               224        5951
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: Parallelization


```r
library(arf)
doParallel::registerDoParallel(cores = 4)

rarf(penguins, 5)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.4          17.1               197        3380
## 2 Adelie    Biscoe           42            18.8               190        4152
## 3 Chinstrap Dream            51.3          19.2               193        3407
## 4 Adelie    Dream            36.5          17.2               191        3779
## 5 Adelie    Biscoe           40.5          19.2               193        3919
## # i 2 more variables: sex <fct>, year <int>
```

The package `arf`

Simple examples: Parallelization


```r
library(arf)
doParallel::registerDoParallel(cores = 4)

rarf(penguins, 5, parallel = F)
```

```
## # A tibble: 5 x 8
##   species   island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>     <fct>           <dbl>         <dbl>             <int>       <int>
## 1 Adelie    Dream            41.4          17.1               197        3380
## 2 Adelie    Biscoe           42            18.8               190        4152
## 3 Chinstrap Dream            51.3          19.2               193        3407
## 4 Adelie    Dream            36.5          17.2               191        3779
## 5 Adelie    Biscoe           40.5          19.2               193        3919
## # i 2 more variables: sex <fct>, year <int>
```

Application

Statistical utility: Setup

Replication of published German national cohort health studies with `arf`-generated synthetic data.

Application

Statistical utility: Setup

Replication of published German national cohort health studies with `arf`-generated synthetic data.

Procedure:

- Learn 5 `arf` models

Application

Statistical utility: Setup

Replication of published German national cohort health studies with `arf`-generated synthetic data.

Procedure:

- Learn 5 `arf` models
- Sample 5 data sets from each

$\Rightarrow$ 25 synthetic datasets

Application

Statistical utility: Setup

Replication of published German national cohort health studies with `arf`-generated synthetic data.

Procedure:

- Learn 5 `arf` models
- Sample 5 data sets from each

$\Rightarrow$ 25 synthetic datasets

- Perform study analysis on every synthetic dataset
- Calculate median and 95% confidence intervals

Application

Statistical utility: Setup

Replication of published German national cohort health studies with `arf`-generated synthetic data.

Procedure:

- Learn 5 `arf` models
- Sample 5 data sets from each

$\Rightarrow$ 25 synthetic datasets

- Perform study analysis on every synthetic dataset
- Calculate median and 95% confidence intervals
- Compare with original data results

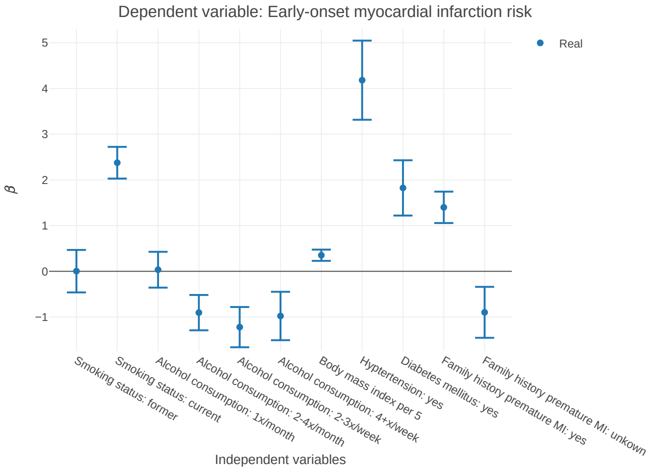
Application

Wienbergen et al. [2022] (n = 1,713, d = 12) - Myocardial infarction

Application

Wienbergen et al. [2022] (n = 1,713, d = 12) - Myocardial infarction

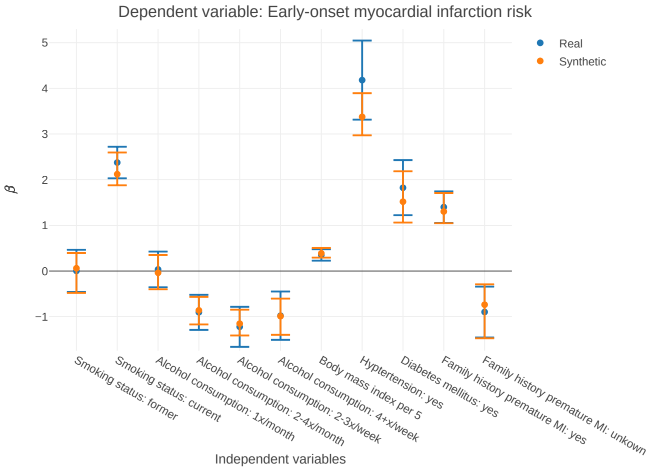
Logistic regression:



Application

Wienbergen et al. [2022] (n = 1,713, d = 12) - Myocardial infarction

Logistic regression:



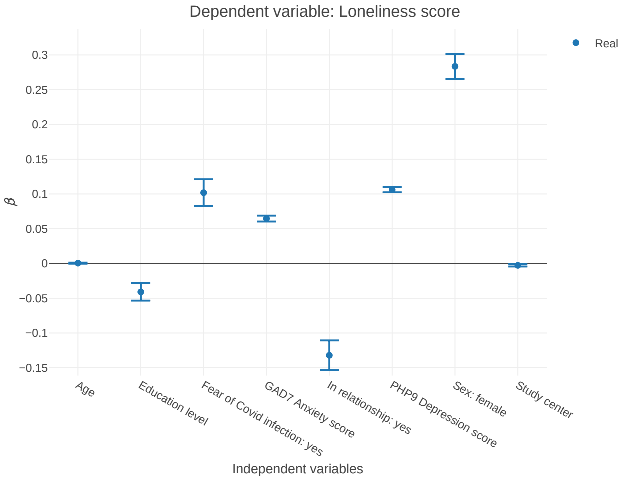
Application

Berger et al. [2021] (n = 82,205, d = 47) - Loneliness during COVID

Application

Berger et al. [2021] (n = 82,205, d = 47) - Loneliness during COVID

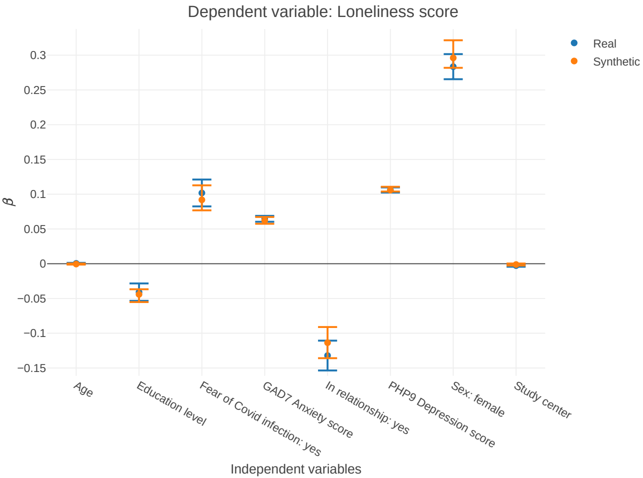
Linear regression:



Application

Berger et al. [2021] (n = 82,205, d = 47) - Loneliness during COVID

Linear regression:



Conclusion & Outlook

The package `arf`

- Little or no tuning required
- Fast computation
- Competitive performance on tabular data

Conclusion & Outlook

The package `arf`

- Little or no tuning required
- Fast computation
- Competitive performance on tabular data

Outlook

- (Conditional) variances, covariances, CDFs
- Missing data imputation
- Explainable AI
- Counterfactuals generation
- Privacy guarantees: Differential privacy

Download `arf`

https://cran.r-project.org/package=arf

https://github.com/bips-hb/arf

References

K. Berger, S. Riedel-Heller, A. Pabst, M. Rietschel, and D. Richter. Loneliness during the first wave of the SARS-CoV-2 pandemic—results of the German National Cohort (NAKO). Bundesgesundheitsblatt-Gesundheitsforschung-Gesundheitsschutz, 64:1157–1164, 2021.

D. S. Watson, K. Blesch, J. Kapar, and M. N. Wright. Adversarial random forests for density estimation and generative modeling. In International Conference on Artificial Intelligence and Statistics, pages 5357–5375. PMLR, 2023.

H. Wienbergen, D. Boakye, K. Günther, J. Schmucker, L. A. Mata Marín, H. Kerniss, R. Nagrani, L. Struß, S. Rühle, T. Retzlaff, et al. Lifestyle and metabolic risk factors in patients with early-onset myocardial infarction: a case-control study. European Journal of Preventive Cardiology, 29(16):2076–2087, 2022.

L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling tabular data using conditional GAN. In Advances in Neural Information Processing Systems, volume 32, 2019.

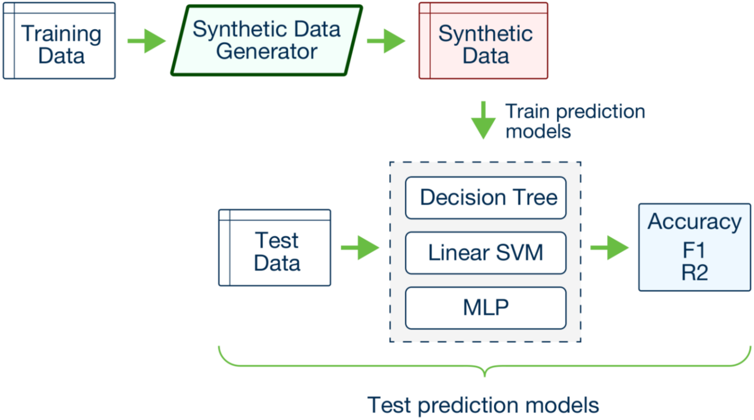
Thank you for your attention

www.leibniz-bips.de/en

Contact
Jan Kapar
Leibniz Institute for Prevention Research
and Epidemiology – BIPS
Achterstraße 30
D-28359 Bremen
kapar@leibniz-bips.de

The package `arf`

Machine learning utility evaluation (Setup)



Xu et al. [2019]