Autoreplicative Random Forests for missing value imputation

Ekaterina Antonenko
Kortrijk, Belgium, February 13, 2023

Laboratoire d’informatique, École Polytechnique, IP Paris
DigitalentLab, MIA, Moteurs d’Intelligence Artificielle
Missing data

Why is data missing?

• Errors in sensors
• Human factor (reluctance to answer particular questions)
• Combining different studies
• ...

Why to impute the missing data?

• Most off-the-shelf statistical and machine learning methods cannot handle missing values
• Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset

Types of missingness

• Missing Completely at Random (MCAR): entirely independently of feature values
• Missing at Random (MAR): depends only on the observed feature values
• Missing Not at Random (MNAR): depends on both the observed and the unobserved feature values
Missing data

Why is data missing?

• Errors in sensors
• Human factor (reluctance to answer particular questions)
• Combining different studies
• ...

Why to impute the missing data?

• Most off-the-shelf statistical and machine learning methods cannot handle missing values
• Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
• Missing data itself might be of interest
Missing data

Why is data missing?

- Errors in sensors
- Human factor (reluctance to answer particular questions)
- Combining different studies
- ...

Why to impute the missing data?

- Most off-the-shelf statistical and machine learning methods cannot handle missing values
- Considering only instances with complete information can lead to a loss of necessary information and can yield a very poor or even empty dataset
- Missing data itself might be of interest

Types of missingness

- Missing Completely at Random (MCAR): entirely independently of feature values
- Missing at Random (MAR): depends only on the observed feature values
- Missing Not at Random (MNAR): depends on both the observed and the unobserved feature values
Imputation methods

- Multiple Imputation by Chained Equations (MICE)
  - first, imputes randomly
  - then iteratively models each feature by all other features
- Autoencoders
  - are neural networks with an output equal to the input
  - model hidden structure
  - are able to “denoise” data
  - require complete data for training
- PCA transformation
  - is essentially similar to Autoencoders with one hidden layer and linear activation function
Imputation methods

• Multiple Imputation by Chained Equations (MICE)
  • first, imputes randomly
  • then iteratively models each feature by all other features

• Autoencoders
  • are neural networks with an output equal to the input
  • model hidden structure
  • are able to “denoise” data
  • require complete data for training

• PCA transformation
  • is essentially similar to Autoencoders with one hidden layer and linear activation function

Our contribution:

• framework unifying the methods above
• code implementation of the framework
• new methodology: Autoreplicative Random Forests (ARF)
Procedural and Iterative Imputation
### Framework

**Iterative:** first, impute randomly; then update iteratively

**Procedural:** train on complete instances; predict for incomplete instances

<table>
<thead>
<tr>
<th></th>
<th>Iterative</th>
<th>Procedural</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-target</strong></td>
<td>MICE</td>
<td></td>
</tr>
<tr>
<td><strong>Multi-target</strong></td>
<td></td>
<td>Autoencoders PCA</td>
</tr>
</tbody>
</table>
**Framework**

**Iterative:** first, impute randomly; then update iteratively

**Procedural:** train on complete instances; predict for incomplete instances

<table>
<thead>
<tr>
<th></th>
<th>Iterative</th>
<th>Procedural</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-target</strong></td>
<td>MICE</td>
<td>Autoencoders PCA</td>
</tr>
<tr>
<td><strong>Multi-target</strong></td>
<td>Autoencoders PCA</td>
<td>Autoencoders PCA</td>
</tr>
</tbody>
</table>
### Framework

**Iterative:** first, impute randomly; then update iteratively

**Procedural:** train on complete instances; predict for incomplete instances

<table>
<thead>
<tr>
<th></th>
<th>Iterative</th>
<th>Procedural</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-target</strong></td>
<td>MICE</td>
<td></td>
</tr>
<tr>
<td><strong>Multi-target</strong></td>
<td>Autoencoders</td>
<td>Autoencoders</td>
</tr>
<tr>
<td></td>
<td>PCA</td>
<td>PCA</td>
</tr>
<tr>
<td></td>
<td>ARF</td>
<td>ARF</td>
</tr>
</tbody>
</table>
One procedure:

- Train a [multi-target] model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

1: procedure PROCEDURAL IMPUTATION($X_{na}$)
2: $X_{train} \leftarrow X_{complete}$ \hfill $\triangleright$ Select complete cases for training
3: $\tilde{X}_{train} \leftarrow X_{train}$ corrupted with missing values \hfill $\triangleright$ Uniformly distributed, \% of m.v. calculated from $X_{missing}$
4: $X_{test} \leftarrow X_{missing}$
5: Fit model on ($\tilde{X}_{train}, X_{train}$)
6: $X_{pred} \leftarrow$ replace m.v. with predictions of fitted model on $X_{test}$
Procedural models

One procedure:

- Train a [multi-target] model on complete instances
- Use the fitted model to predict on instances containing missing values
- Correct observed values if changed

1: procedure PROCEDURAL IMPUTATION($X_{na}$)
2: $X_{train} \leftarrow X_{complete}$ ▶ Select complete cases for training
3: $\tilde{X}_{train} \leftarrow X_{train}$ corrupted with missing values ▶ Uniformly distributed, % of m.v. calculated from $X_{missing}$
4: $X_{test} \leftarrow X_{missing}$
5: Fit model on ($\tilde{X}_{train}$, $X_{train}$)
6: $X_{pred} \leftarrow$ replace m.v. with predictions of fitted model on $X_{test}$

NB: needs enough complete data to train a reliable model
Iterative model

First, impute randomly

Then, iteratively until convergence is reached:

- Train a [multi-target] model on previous imputation
- Use the fitted model to predict on all instances
- Correct observed values if changed

---

1: **procedure** Iterative Imputation($X_{na}$, $\alpha$)  
2: $X_{imp}^0 \leftarrow$ random imputation of m.v. in $X_{na}$  
3: **while** $\Delta_{imp} > \alpha$ **do**  
4: Fit model on ($X_{na}$, $X_{imp}^{n-1}$)  
5: $X_{imp}^n \leftarrow$ replace m.v. with predictions of fitted model on $X_{na}$  
6: $\Delta_{imp} \leftarrow$ distance [accuracy] between $X_{imp}^n$ and $X_{imp}^{n-1}$
Autoencoders $\Rightarrow$ Autoreplicative Random Forests

- Autoencoder Denoising Autoencoder

Why to use multi-label methods?
- fewer parameters than neural networks (good for low-sampled data)
- no need for hidden layers

Which methods?
- Decision Trees, Random Forests
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.
Autoencoders $\implies$ Autoreplicative Random Forests

Why to use multi-label methods?
- fewer parameters than neural networks (good for low-sampled data)
- no need for hidden layers

Which methods?
- Decision Trees, Random Forests, Extra Trees
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.
Autoencoders $\iff$ Autoreplicative Random Forests

Why to use multi-label methods?

- fewer parameters than neural networks (good for low-sampled data)
- no need for hidden layers
Autoencoders \[\Rightarrow\] Autoreplicative Random Forests

Why to use multi-label methods?

- fewer parameters than neural networks (good for low-sampled data)
- no need for hidden layers

Which methods?

- Decision Trees, Random Forests, Extra Trees
- Classifier Chains, Multilabel k Nearest Neighbours, Random k-Labelsets, Conditional Dependency Networks, etc.
Results: iterative ARF do converge

Imputation via Iterative Random Forests converges after several iterations

- **Mushrooms**
  - Accuracy vs. Iterations
  - Lines for different parameters: 0.01, 0.05, 0.1, 0.2, 0.3

- **Votes**
  - Accuracy vs. Iterations
  - Lines for different parameters: 0.01, 0.05, 0.1, 0.2, 0.3

- **Eucalyptus**
  - Accuracy vs. Iterations
  - Lines for different parameters: 0.01, 0.05, 0.1, 0.2, 0.3
### Accuracy of imputation

<table>
<thead>
<tr>
<th>MVR</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mushroom [8,124 x 22]</td>
<td>Soybean [307 x 35]</td>
<td>Tumor [339 x 17]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complete c.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MICE</td>
<td>80.1%</td>
<td>32.3%</td>
<td>10.1%</td>
<td>0.7%</td>
<td>0.04%</td>
<td>69.7%</td>
<td>13.7%</td>
<td>1.0%</td>
<td>0%</td>
<td>0%</td>
<td>83.8%</td>
<td>38.9%</td>
<td>15.0%</td>
<td>1.2%</td>
<td>0.3%</td>
</tr>
<tr>
<td>itARF</td>
<td>0.658</td>
<td>0.715</td>
<td>0.741</td>
<td><strong>0.769</strong></td>
<td><strong>0.777</strong></td>
<td>0.884</td>
<td>0.884</td>
<td>0.879</td>
<td>0.867</td>
<td>0.850</td>
<td><strong>0.761</strong></td>
<td><strong>0.768</strong></td>
<td><strong>0.748</strong></td>
<td><strong>0.754</strong></td>
<td><strong>0.735</strong></td>
</tr>
<tr>
<td>pARF</td>
<td>0.730</td>
<td>0.740</td>
<td>0.747</td>
<td>0.734</td>
<td>0.707</td>
<td>0.824</td>
<td>0.850</td>
<td>0.832</td>
<td>0.815</td>
<td>0.789</td>
<td>0.652</td>
<td>0.672</td>
<td>0.645</td>
<td>0.660</td>
<td>0.620</td>
</tr>
<tr>
<td>itAE</td>
<td><strong>0.748</strong></td>
<td><strong>0.774</strong></td>
<td><strong>0.761</strong></td>
<td>0.671</td>
<td>0.478</td>
<td>0.804</td>
<td>0.779</td>
<td>0.600</td>
<td>–</td>
<td>–</td>
<td>0.639</td>
<td>0.696</td>
<td>0.650</td>
<td>0.694</td>
<td>0.635</td>
</tr>
<tr>
<td>pAE</td>
<td>0.608</td>
<td>0.618</td>
<td>0.604</td>
<td>0.584</td>
<td>0.569</td>
<td>0.653</td>
<td>0.607</td>
<td>0.608</td>
<td>0.584</td>
<td>0.590</td>
<td><strong>0.721</strong></td>
<td><strong>0.732</strong></td>
<td>0.692</td>
<td>0.711</td>
<td>0.710</td>
</tr>
<tr>
<td>itPCA</td>
<td>0.580</td>
<td>0.494</td>
<td>0.491</td>
<td>0.538</td>
<td>0.428</td>
<td>0.653</td>
<td>0.622</td>
<td>0.594</td>
<td>–</td>
<td>–</td>
<td><strong>0.721</strong></td>
<td><strong>0.718</strong></td>
<td>0.692</td>
<td>0.690</td>
<td>0.497</td>
</tr>
<tr>
<td>pPCA</td>
<td>0.604</td>
<td>0.627</td>
<td>0.622</td>
<td>0.623</td>
<td>0.618</td>
<td>0.667</td>
<td>0.692</td>
<td>0.671</td>
<td>0.646</td>
<td>0.603</td>
<td><strong>0.721</strong></td>
<td>0.740</td>
<td><strong>0.692</strong></td>
<td><strong>0.711</strong></td>
<td>0.710</td>
</tr>
</tbody>
</table>

|     | Votes [435 x 16] |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|     | Complete cases |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| MICE | 85.3% | 42.2% | 18.5% | 1.3% | 0.4% | 81.8% | 40.5% | 14.9% | 2.7% | 0% | 11.8% | 0% | 0% | 0% | 0% |
| itARF | 0.768 | 0.795 | 0.771 | **0.768** | **0.782** | 0.750 | 0.679 | 0.665 | **0.648** | **0.651** | – | – | – | – | – |
| pARF | 0.719 | 0.726 | 0.728 | 0.723 | 0.718 | 0.714 | 0.639 | 0.638 | 0.628 | 0.600 | 0.684 | 0.677 | **0.676** | **0.667** | **0.661** |
| itAE | 0.671 | 0.756 | 0.522 | 0.495 | 0.636 | 0.647 | 0.604 | 0.608 | – | – | 0.633 | – | – | – | – |
| pAE | 0.697 | 0.563 | 0.602 | 0.578 | 0.570 | 0.700 | 0.474 | 0.485 | 0.448 | 0.487 | 0.626 | 0.617 | 0.616 | 0.604 | 0.596 |
| itPCA | 0.638 | 0.546 | 0.600 | 0.524 | 0.488 | 0.679 | 0.514 | 0.563 | 0.611 | – | 0.313 | – | – | – | – |
| pPCA | 0.665 | 0.583 | 0.567 | 0.572 | 0.570 | 0.686 | 0.513 | 0.477 | 0.468 | 0.484 | 0.653 | 0.645 | 0.645 | 0.634 | 0.627 |

- Procedural ARFs: may be powerful when enough complete instances
- MICE: as powerful as computationally expensive
- Iterative ARFs: still powerful + significantly quicker
Time complexity

In theory:

- $p = \text{number of features}$
- $n_{iter} = \text{number of iterations}$

<table>
<thead>
<tr>
<th></th>
<th>itARF</th>
<th>pARF</th>
<th>MICE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(n_{iter} \cdot p)$</td>
<td>$O(p)$</td>
<td>$O(n_{iter} \cdot p^2)$</td>
</tr>
</tbody>
</table>
Time complexity

**In theory:**

- $p =$ number of features
- $n_{iter} =$ number of iterations

<table>
<thead>
<tr>
<th></th>
<th>itARF</th>
<th>pARF</th>
<th>MICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n_{iter} \cdot p)$</td>
<td>$O(p)$</td>
<td>$O(n_{iter} \cdot p^2)$</td>
<td></td>
</tr>
</tbody>
</table>

**In practice:**

![Graph showing time complexity vs number of features](chart.png)
Probabilistic Autoreplicative Random Forests

Extension of iterative Autoreplicative Random Forests:

- First, randomly impute with probabilities
- In each iteration:
  - $M$ imputations are sampled from the previous distribution
  - $M$ trees of a Random Forest are trained on different imputations
  - The Random Forest produces one probabilistic imputation

```plaintext
1: procedure Probabilistic Iterative Imputation($X_{na}$, $\alpha$)
2: \[ \mathcal{H}^0 \leftarrow \{ h_1^0, h_2^0, \ldots, h_M^0 \} \] ▷ Random Forest of $M$ trees
3: \[ p_{imp}^0 \leftarrow \text{random imputation with probabilities from } \{U_{[0,1]}\} \]
4: while $\Delta_{imp} > \alpha$ do
5: \[ \mathcal{H}^n \leftarrow \{ h_1^n, h_2^n, \ldots, h_M^n \} \] ▷ Random Forest of $M$ trees
6: for $h_m^n \in \mathcal{H}^n$ do
7: \[ X_{imp}^{n,m} \sim p_{imp}^{n-1} \] ▷ Impute by sampling from distributions
8: Fit a tree $h_m^n$ on $(X_{na}, X_{imp}^{n,m})$
9: \[ p_{imp}^n \leftarrow \text{probabilities provided by fitted } \mathcal{H}^n \]
10: $\Delta_{imp} \leftarrow \text{distance between } p_{imp}^n \text{ and } p_{imp}^{n-1}$
```
Chains of Autoreplicative Random Forests
Usecase: Single Nucleotide Polymorphisms (SNP)

- Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
- High-dimensional \((10^5 - 10^6)\) and low-sampled \((10^2 - 10^3)\)
- Ordering is important
- Missing values occur due to external mechanisms \(\Rightarrow\) MCAR
Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)

High-dimensional ($10^5 - 10^6$) and low-sampled ($10^2 - 10^3$)

Ordering is important

Missing values occur due to external mechanisms $\implies$ MCAR

Methods:

- reference-based (state-of-the-art for human data)
- reference-free (when reference panels are not available).
• Categorical: 0 (dominant-dominant), 1 (dominant-mutant), 2 (mutant-mutant)
• High-dimensional \( (10^5 - 10^6) \) and low-sampled \( (10^2 - 10^3) \)
• Ordering is important
• Missing values occur due to external mechanisms $\Rightarrow$ MCAR

Methods:

• reference-based (state-of-the-art for human data)
• reference-free (when reference panels are not available).

Other possible examples: gene expression arrays, classification problems in astronomy, tool development for finance data, and weather prediction.
Chains of Autoreplicative Random Forests

- **Procedural approach:**
  one window of size $\Delta = \text{complete instances} + \text{instances with missing values}$

- **Chain of windows:**
  on each step, stacking $\nu$ windows with already imputed values as additional features

- **Ensemble of chains:**
  one forward chain, one backward chain, several random chains
Gridsearch for parameters $\Delta$ and $\nu$

<table>
<thead>
<tr>
<th></th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>1%</td>
<td>0.15</td>
<td>0.08</td>
<td>0.10</td>
<td>0.08</td>
<td>0.05</td>
</tr>
<tr>
<td>5%</td>
<td>0.06</td>
<td>0.03</td>
<td>0.05</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>10%</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>20%</td>
<td>0.00</td>
<td>0.01</td>
<td>0.02</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>30%</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Lighter color / higher accuracy

$\Delta$: bigger fraction of missing values $\rightarrow$ smaller size of window $\Rightarrow$ can be estimated theoretically, no need for search

$\nu$: may depend on problem
<table>
<thead>
<tr>
<th></th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.01</th>
<th>0.05</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChARF</td>
<td>0.952</td>
<td>0.935</td>
<td>0.916</td>
<td>0.882</td>
<td>0.845</td>
<td>0.970</td>
<td>0.950</td>
<td>0.926</td>
<td>0.866</td>
<td>0.810</td>
</tr>
<tr>
<td>kNN (5/10)</td>
<td>0.803</td>
<td>0.802</td>
<td>0.801</td>
<td>0.798</td>
<td>0.794</td>
<td>0.851</td>
<td>0.849</td>
<td>0.847</td>
<td>0.843</td>
<td><strong>0.839</strong></td>
</tr>
<tr>
<td>mode</td>
<td>0.727</td>
<td>0.727</td>
<td>0.726</td>
<td>0.727</td>
<td>0.726</td>
<td>0.725</td>
<td>0.732</td>
<td>0.731</td>
<td>0.730</td>
<td>0.729</td>
</tr>
<tr>
<td>SVD (50/500)</td>
<td>0.647</td>
<td>0.648</td>
<td>0.645</td>
<td>0.643</td>
<td>0.636</td>
<td>0.788</td>
<td>0.788</td>
<td>0.788</td>
<td>0.785</td>
<td>0.780</td>
</tr>
<tr>
<td>MICE</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.835</td>
<td>0.824</td>
<td>0.818</td>
<td>0.805</td>
<td>0.792</td>
<td>0.897</td>
<td>0.886</td>
<td>0.878</td>
<td>0.866</td>
<td>0.854</td>
</tr>
<tr>
<td>ChARF</td>
<td>0.765</td>
<td>0.763</td>
<td>0.765</td>
<td>0.765</td>
<td>0.764</td>
<td>0.867</td>
<td>0.866</td>
<td>0.866</td>
<td>0.865</td>
<td><strong>0.864</strong></td>
</tr>
<tr>
<td>kNN (50/10)</td>
<td>0.761</td>
<td>0.760</td>
<td>0.762</td>
<td>0.761</td>
<td>0.761</td>
<td>0.807</td>
<td>0.804</td>
<td>0.805</td>
<td>0.805</td>
<td>0.804</td>
</tr>
<tr>
<td>mode</td>
<td>0.740</td>
<td>0.737</td>
<td>0.737</td>
<td>0.735</td>
<td>0.734</td>
<td>0.693</td>
<td>0.694</td>
<td>0.696</td>
<td>0.692</td>
<td>0.690</td>
</tr>
<tr>
<td>SVD (50/100)</td>
<td>0.740</td>
<td>0.737</td>
<td>0.737</td>
<td>0.735</td>
<td>0.734</td>
<td>0.693</td>
<td>0.694</td>
<td>0.696</td>
<td>0.692</td>
<td>0.690</td>
</tr>
<tr>
<td>MICE</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.757</td>
<td>0.741</td>
<td>0.724</td>
<td>0.689</td>
<td>0.664</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.821</td>
<td>0.808</td>
<td>0.795</td>
<td>0.777</td>
<td>0.762</td>
<td><strong>0.799</strong></td>
<td>0.781</td>
<td>0.761</td>
<td>0.731</td>
<td>0.717</td>
</tr>
<tr>
<td>ChARF</td>
<td><strong>0.823</strong></td>
<td>0.819</td>
<td>0.818</td>
<td>0.815</td>
<td>0.811</td>
<td><strong>0.799</strong></td>
<td>0.781</td>
<td>0.761</td>
<td>0.731</td>
<td>0.717</td>
</tr>
<tr>
<td>kNN (10/10)</td>
<td>0.729</td>
<td>0.727</td>
<td>0.729</td>
<td>0.729</td>
<td>0.727</td>
<td>0.691</td>
<td>0.693</td>
<td>0.692</td>
<td>0.692</td>
<td>0.691</td>
</tr>
<tr>
<td>mode</td>
<td>0.622</td>
<td>0.618</td>
<td>0.609</td>
<td>0.600</td>
<td>0.594</td>
<td>0.456</td>
<td>0.453</td>
<td>0.450</td>
<td>0.449</td>
<td>0.450</td>
</tr>
<tr>
<td>SVD (200/50)</td>
<td>0.641</td>
<td>0.635</td>
<td>0.621</td>
<td>0.585</td>
<td>0.545</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>MICE</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

- MICE: run with 10 neighbors for each feature, still worked only for smaller data
- Autoencoders: not taken into comparison (no complete data for training)
- Well-known methods for SNP imputation: $k$ Nearest Neighbors, Single Value Decomposition
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

- We show how probabilistic training can be easily added to the model

- ARF vs MICE: high quality and much faster

- ARF vs Autoencoders:
  - no need for one-hot encoding ⇒ less features.
  - lower time complexity ⇒ works for high-dimensional datasets
  - no need for complete data
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

- We show how probabilistic training can be easily added to the model
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection

- We show how probabilistic training can be easily added to the model

- ARF vs MICE: high quality and much faster
Conclusions

- Unusual and effective usage of multi-label methods, e.g. Random Forests:
  - autoreplication
  - missing value imputation
  - denoising
  - outlier detection
- We show how probabilistic training can be easily added to the model
- ARF vs MICE: high quality and much faster
- ARF vs Autoencoders:
  - no need for one-hot encoding $\Rightarrow$ less features.
  - lower time complexity $\Rightarrow$ works for high-dimensional datasets
  - no need for complete data
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
Questions of interest

• Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
• For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
• Studies for MAR and MNAR scenarios
Questions of interest

- Multi-target Random Forests still optimize decomposable metrics (entropy, gini):
  can we model the labels indeed jointly?
- For very wide datasets (> 10,000 features) multi-target methods are very memory expensive
- Studies for MAR and MNAR scenarios
- Regression (e.g. gene expression)
Thank you!