datacleanbot Documentation

Release 0.1

Ji Zhang

Aug 30, 2020

Contents

Welcome to the documentation of the `datacleanbot` Python API. `datacleanbot` offers automated, data-driven support to help users clean data effectively and smoothly. Given a random raw dataset representing a machine learning problem, the Python tool is capable of automatically identifying the potential issues and reporting the results and recommendations to the end-user in an effective way. `datacleanbot` is designed with a strong connection to OpenML which is a platform where people can easily share data, experiments and machine learning models. Users can easily acquire datasets from OpenML with the dataset ID and clean them with `datacleanbot`.

CHAPTER 1

User's Guide

1.1 Usage

1.1.1 Acquire Data

The first step is to acquire data from OpenML.

```
import openml as oml
import datacleanbot.dataclean as dc
import numpy as np

data = oml.datasets.get_dataset(id) # id: openml dataset id
X, y, categorical_indicator, features = data.get_data(target=data.default_target_
↪attribute, dataset_format='array')
Xy = np.concatenate((X,y.reshape((y.shape[0],1))), axis=1)
```

1.1.2 Show Impotant Features

`datacleanbot` computes the most important features of the given dataset using random forest and present the 15 most useful features to the user.

```
dc.show_important_features(X, y, data.name, features)
```

1.1.3 Unify Column Names

Inconsistent capitalization of column names can be detected and reported to the user. Users can decide whether to unify them or not. The capitalization can be unified to either upper case or lower case.

```
dc.unify_name_consistency(features)
```

1.1.4 Show Statistical Inforamtion

`datacleanbot` can present the statistical information to help users gain a better understanding of the data distribution.

```
dc.show_statistical_info(Xy)
```

1.1.5 Discover Data Types

`datacleanbot` can discover feature data types. Basic data types discovered are 'datetime', 'float', 'integer', 'bool' and 'string'. `datacleanbot` also can discover statistical data types (real, positive real, categorical and count) using Bayesian Model abda.

```
dc.discover_types(Xy)
```

1.1.6 Clean Duplicated Rows

`datacleanbot` detects the duplicated records and reports them to users.

```
dc.clean_duplicated_rows(Xy)
```

1.1.7 Handle Missing Values

`datacleanbot` identifies characters 'n/a', 'na', '–' and '?' as missing values. Users can add extra characters to be considered as missing. After the missing values being detected, `datacleanbot` will present the missing values in effective visualizations to help users identify the missing mechanism. Afterwards, `datacleanbot` recommends the appropriate approach to clean missing values according the missing mechanism.

```
features, Xy = dc.handle_missing(features, Xy)
```

1.1.8 Outlier Detection

A meta-learner is trained beforehand to recommend the outlier detection algorithm according to the meta features og the given dataset. Users can apply the recommended algorithm or any other available algorithm to detect outliers. After the detection, outliers will be present to users in effective visualizations and users can choose to drop them or not.

```
Xy = dc.handle_outlier(features, Xy)
```

1.2 Example

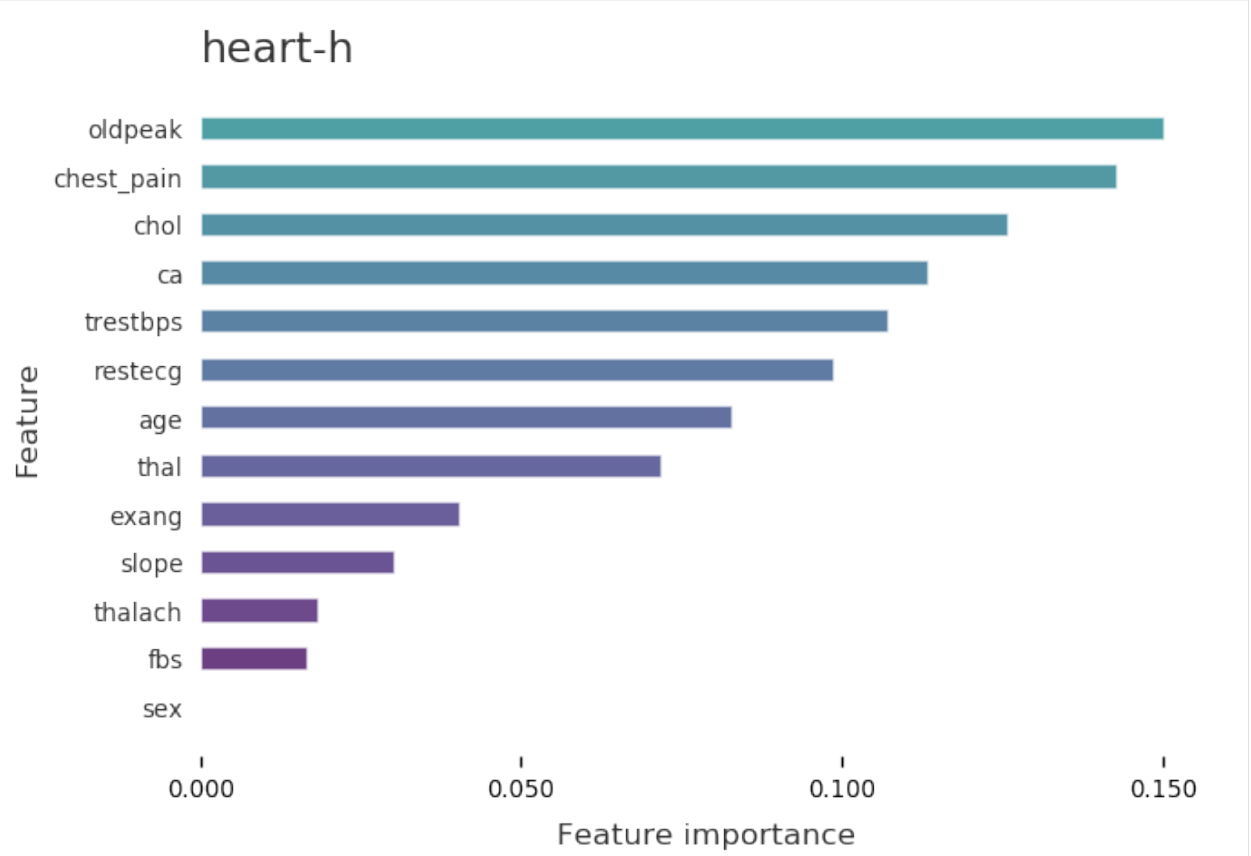
1.2.1 Example_autoclean

[4]:
```
import datacleanbot.dataclean as dc
import openml as oml
import numpy as np
```

[5]:
```
# acquire data
data = oml.datasets.get_dataset(51)
X, y, categorical_indicator, features = data.get_data(target=data.default_target_
↪attribute, dataset_format='array')
Xy = np.concatenate((X,y.reshape((y.shape[0],1))), axis=1)
```

[6]:
```
# input openml dataset id
Xy = dc.autoclean(Xy, data.name, features)
```

```
<IPython.core.display.HTML object>
```




```
<IPython.core.display.HTML object>
```

```
                0    1           2           3           4           5   \
count  294.000000  3.0  294.000000  271.000000  293.000000  286.000000
mean    47.826531  0.0    1.867347  250.848708    0.303754    0.930070
std      7.811812  0.0    0.956077   67.657711    0.460665    0.255476
min     28.000000  0.0    0.000000   85.000000    0.000000    0.000000
25%     42.000000  0.0    1.000000  209.000000    0.000000    1.000000
50%     49.000000  0.0    2.000000  243.000000    0.000000    1.000000
75%     54.000000  0.0    3.000000  282.500000    1.000000    1.000000
max     66.000000  0.0    3.000000  603.000000    1.000000    1.000000

                6           7           8           9          10          11  \
count  294.000000  293.000000  294.000000  104.000000  28.000000  293.000000
mean     0.586054    1.156997    0.724490    1.105769   1.035714  139.129693
std      0.908648    0.417011    0.447533    0.338995   0.881167   23.589749
min      0.000000    0.000000    0.000000    0.000000   0.000000   82.000000
```

(continues on next page)

(continued from previous page)

```
25%      0.000000    1.000000    0.000000    1.000000   0.000000  122.000000
50%      0.000000    1.000000    1.000000    1.000000   1.000000  140.000000
75%      1.000000    1.000000    1.000000    1.000000   2.000000  155.000000
max      5.000000    2.000000    1.000000    2.000000   2.000000  190.000000

                12          13
count  293.000000  294.000000
mean   132.583618    0.360544
std     17.626568    0.480977
min     92.000000    0.000000
25%    120.000000    0.000000
50%    130.000000    0.000000
75%    140.000000    1.000000
max    200.000000    1.000000
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
['int64', 'int64', 'int64', 'int64', 'int64', 'int64', 'int64', 'int64', 'bool',␣
↪'float64', 'float64', 'int64', 'int64', 'bool']
```

```
<IPython.core.display.HTML object>
```

```
['Type.POSITIVE', 'Type.CATEGORICAL', 'Type.CATEGORICAL', 'Type.POSITIVE', 'Type.
↪COUNT', 'Type.CATEGORICAL', 'Type.POSITIVE', 'Type.COUNT', 'Type.COUNT', 'Type.
↪CATEGORICAL', 'Type.CATEGORICAL', 'Type.POSITIVE', 'Type.POSITIVE', 'Type.
↪CATEGORICAL']
```

```
<IPython.core.display.HTML object>
```

```
Identifying Duplicated Rows ...
```

```
<IPython.core.display.HTML object>
```

```
       0   1    2   3    4    5    6    7    8   9   10     11     12   13
101  49.0 NaN  3.0 NaN  0.0  1.0  0.0  1.0  0.0 NaN NaN  160.0  110.0  0.0
102  49.0 NaN  3.0 NaN  0.0  1.0  0.0  1.0  0.0 NaN NaN  160.0  110.0  0.0

Do you want to drop the duplicated rows? [y/n]y

Duplicated rows are dropped.
```

```
<IPython.core.display.HTML object>
```

```

Column names
============
['age', 'sex', 'chest_pain', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach', 'exang',
↪ 'oldpeak', 'slope', 'ca', 'thal']

Column names are consistent
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
The default setting of missing characters is ['n/a', 'na', '-', '?']
Do you want to add extra character? [y/n]n
```

```
<IPython.core.display.HTML object>


Number of missing in each feature
0       0
1     290
2       0
3      22
4       1
5       8
6       0
7       1
8       0
9     189
10    265
11      1
12      1
13      0
dtype: int64

Records containing missing values:
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	28.0	NaN	3.0	132.0	0.0	1.0	0.0	0.0	1.0	NaN	NaN	185.0	130.0	0.0
1	29.0	NaN	3.0	243.0	0.0	1.0	0.0	1.0	1.0	NaN	NaN	160.0	120.0	0.0
2	29.0	NaN	3.0	NaN	0.0	1.0	0.0	1.0	1.0	NaN	NaN	170.0	140.0	0.0
3	30.0	NaN	0.0	237.0	0.0	1.0	0.0	2.0	0.0	NaN	0.0	170.0	170.0	0.0
4	31.0	NaN	3.0	219.0	0.0	1.0	0.0	2.0	0.0	NaN	NaN	150.0	100.0	0.0

```
Missing correlation between features containing missing values and other features
```

	1	3	4	5	7	9	10	11	12
0	-0.054393	0.019737	0.001330	-0.014961	0.053771	-0.234171	0.001532	0.001330	0.001330
1	1.000000	-0.099671	0.005952	0.017041	0.005952	-0.004595	0.082259	0.005952	0.005952
2	0.020988	0.067940	0.069733	0.023981	-0.114337	0.342524	0.075190	0.069733	0.069733
3	-0.099671	1.000000	-0.016674	-0.047736	-0.016674	0.076025	0.048563	-0.016674	-0.016674
4	0.005952	-0.016674	1.000000	-0.009805	-0.003425	-0.078890	0.019022	1.000000	1.000000
5	0.017041	-0.047736	-0.009805	1.000000	-0.009805	-0.007021	-0.088011	-0.009805	-0.009805
6	0.009863	-0.077552	0.091000	-0.039312	-0.037900	-0.841642	0.005952	0.091000	0.091000
7	0.005952	-0.016674	-0.003425	-0.009805	1.000000	0.043410	0.019022	-0.003425	-0.003425
8	-0.062333	0.000198	-0.095489	-0.085351	0.035864	-0.038358	-0.016808	-0.095489	-0.095489
9	-0.004595	0.076025	-0.078890	-0.007021	0.043410	1.000000	0.025752	-0.078890	-0.078890
10	0.082259	0.048563	0.019022	-0.088011	0.019022	0.025752	1.000000	0.019022	0.019022
11	0.005952	-0.016674	1.000000	-0.009805	-0.003425	-0.078890	0.019022	1.000000	1.000000
12	0.005952	-0.016674	1.000000	-0.009805	-0.003425	-0.078890	0.019022	1.000000	1.000000

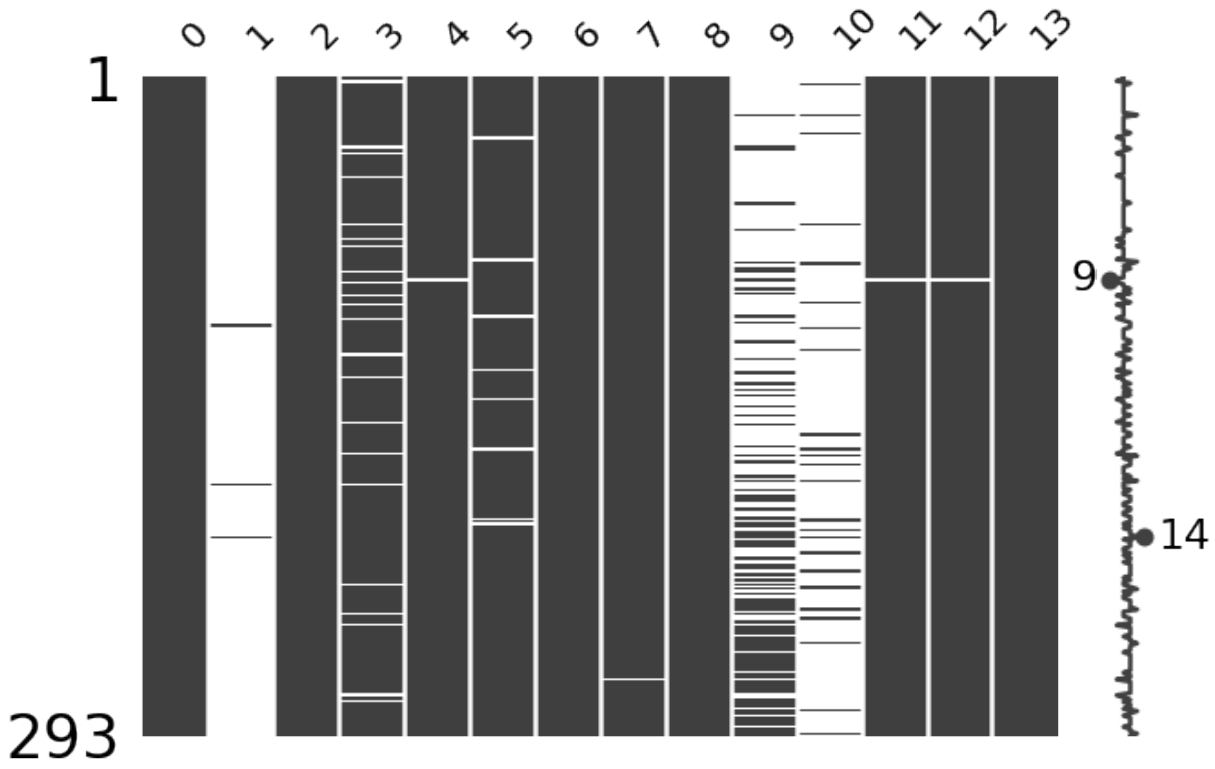
(continues on next page)

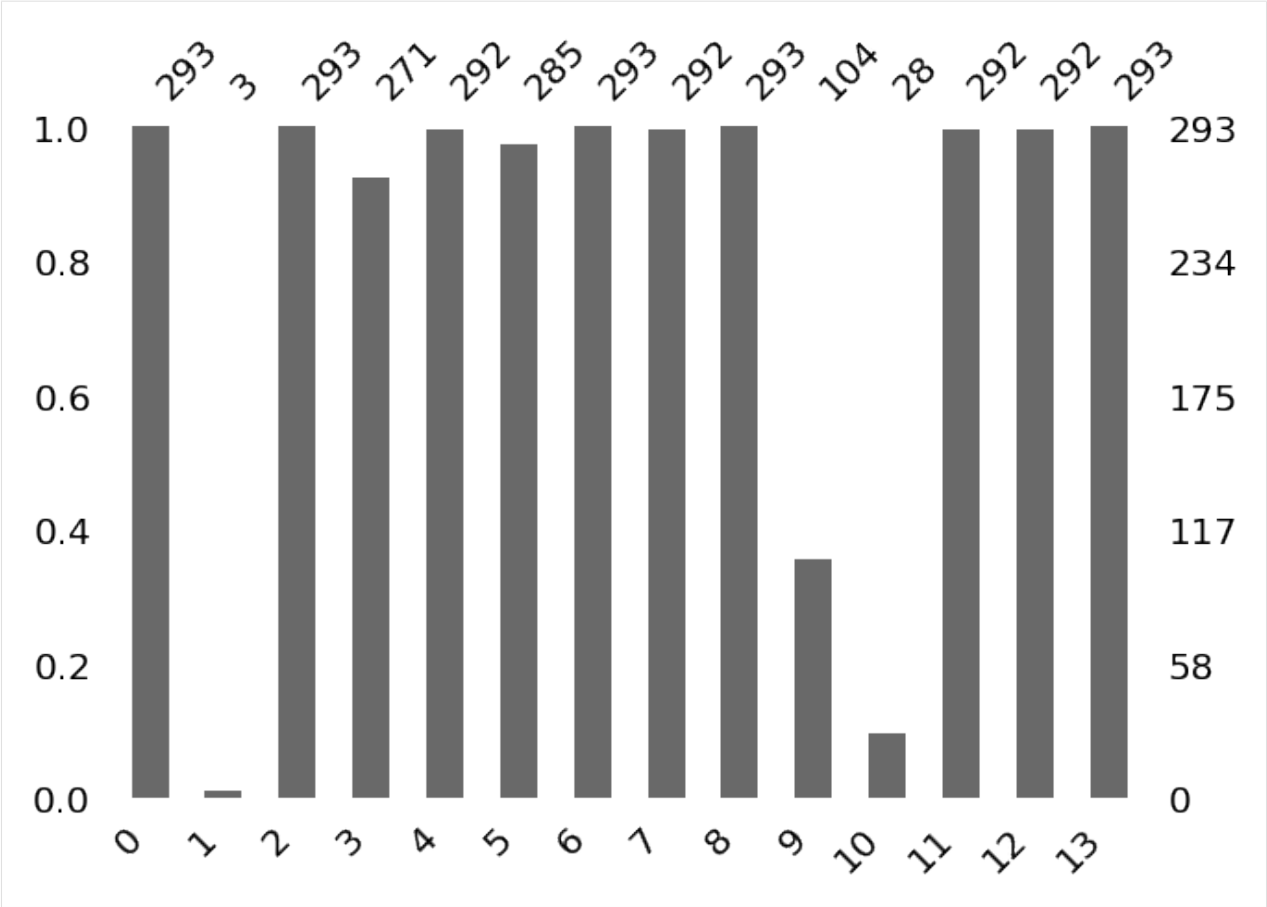
(continued from previous page)

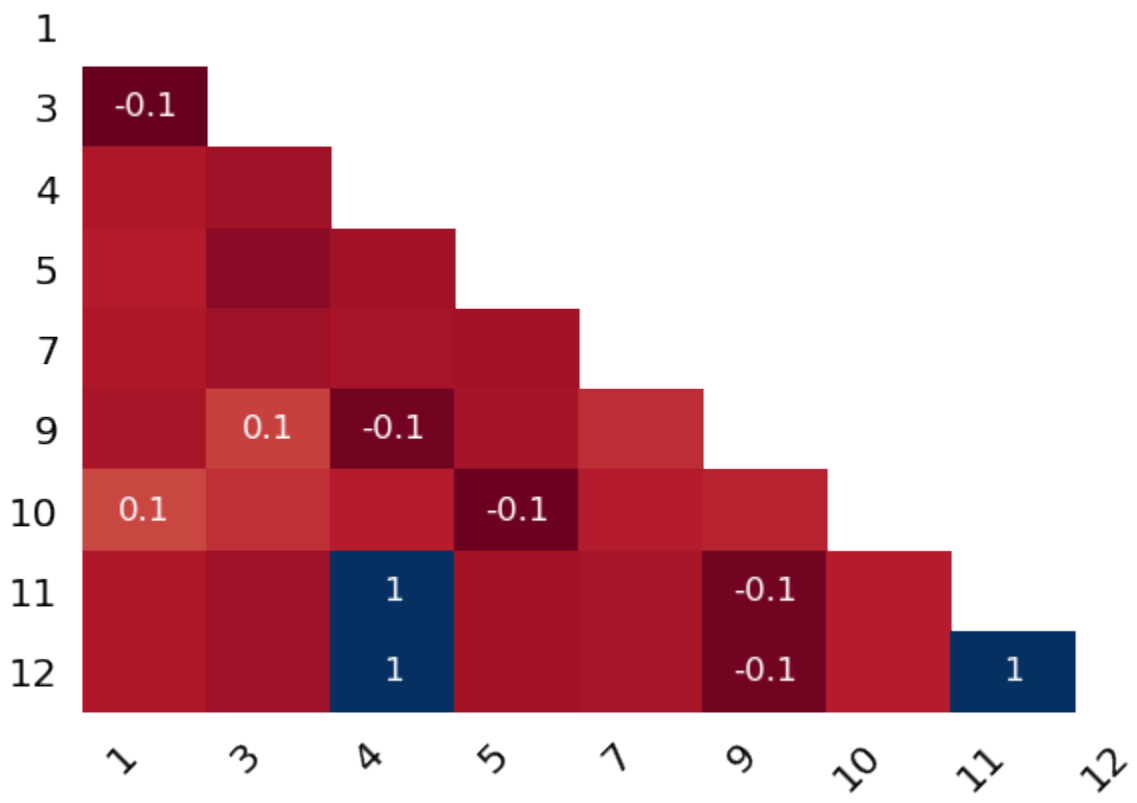
```
11  1.000000  1.000000
12  1.000000  1.000000
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```




293
3
293
271
292
285
293
292
293
104
28
292
292
293
1.0
0.8
0.6
0.4
0.2
0.0
293
234
175
117
58
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13




```
<IPython.core.display.HTML object>
```

```
Feature [1, 10] has extreme large proportion of missing data
Do you want to delete the above features? [y/n]y

Choose the missing mechanism [a/b/c/d]:
a.MCAR b.MAR c.MNAR d.Skip
b
Imputation score of knn is 0.7567397233586597
Imputation score of matrix factorization is 0.7567397233586597
Imputation score of multiple imputation is 0.8122681667640756
Imputation method with the highest socre is multiple imputation
```

```
<IPython.core.display.HTML object>
```

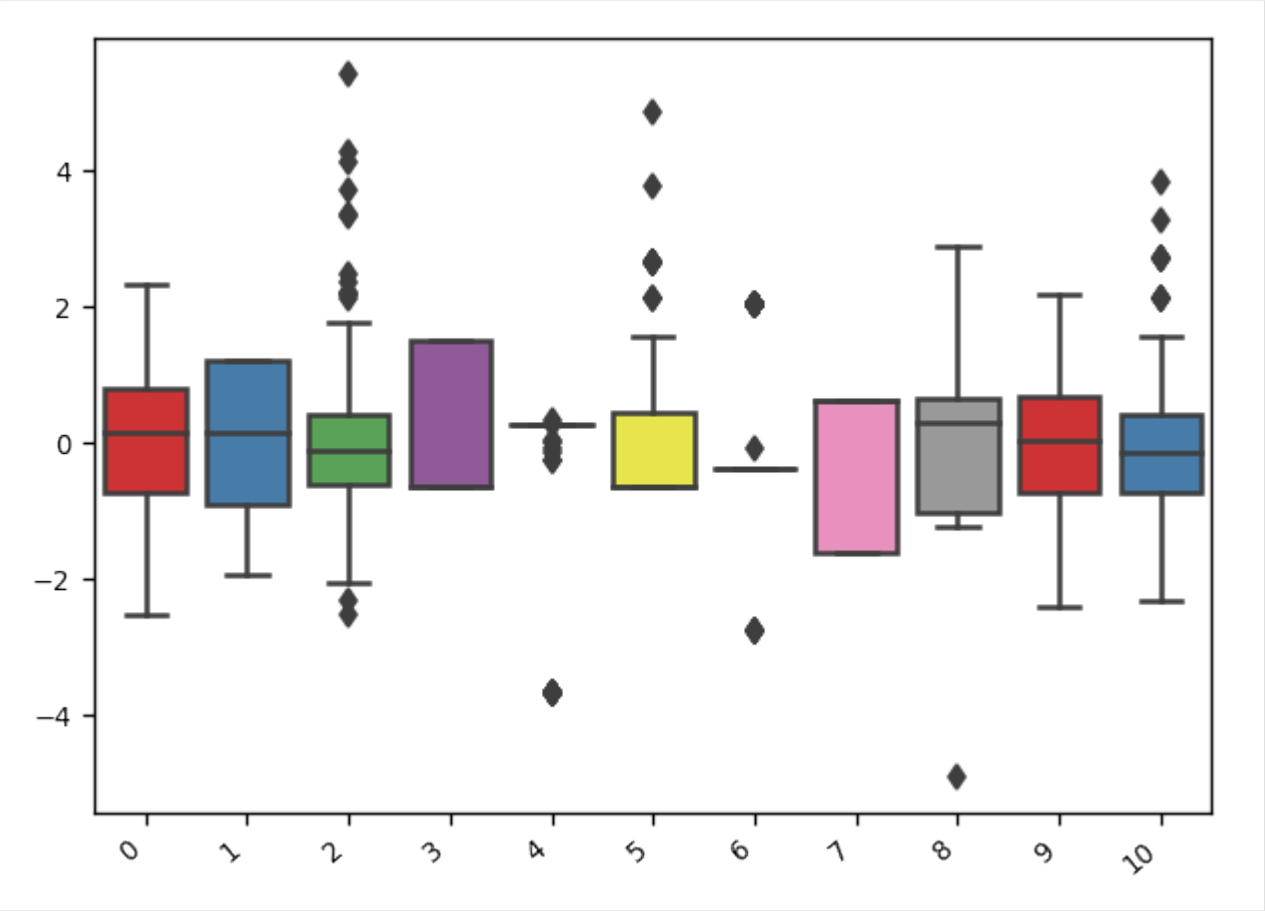
```
The recommended approach is multiple imputation
Do you want to apply the recommended approach? [y/n]y

Applying multiple imputation ...
Missing values cleaned!
```

```
<IPython.core.display.HTML object>
```

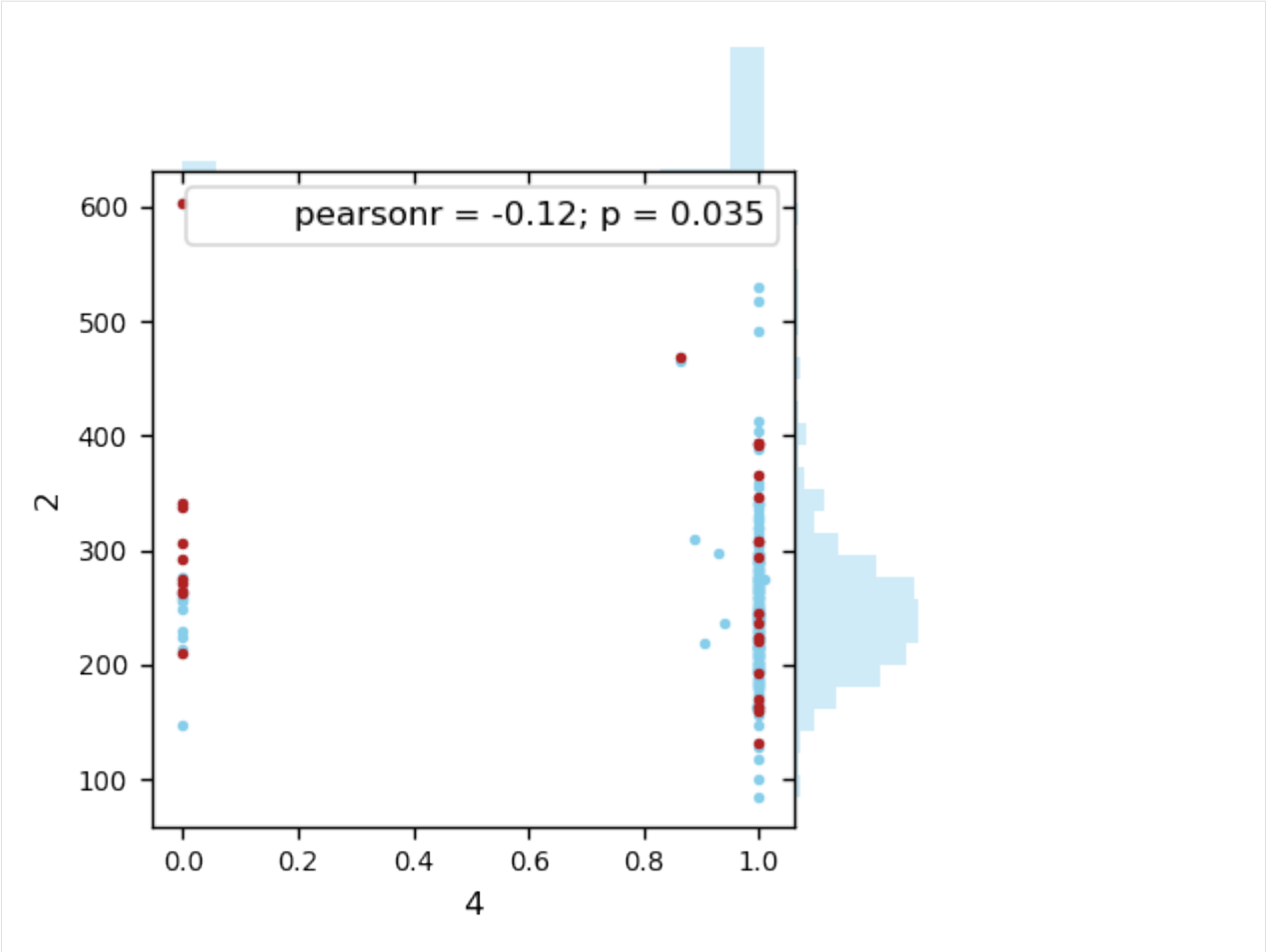
```
<IPython.core.display.HTML object>
```

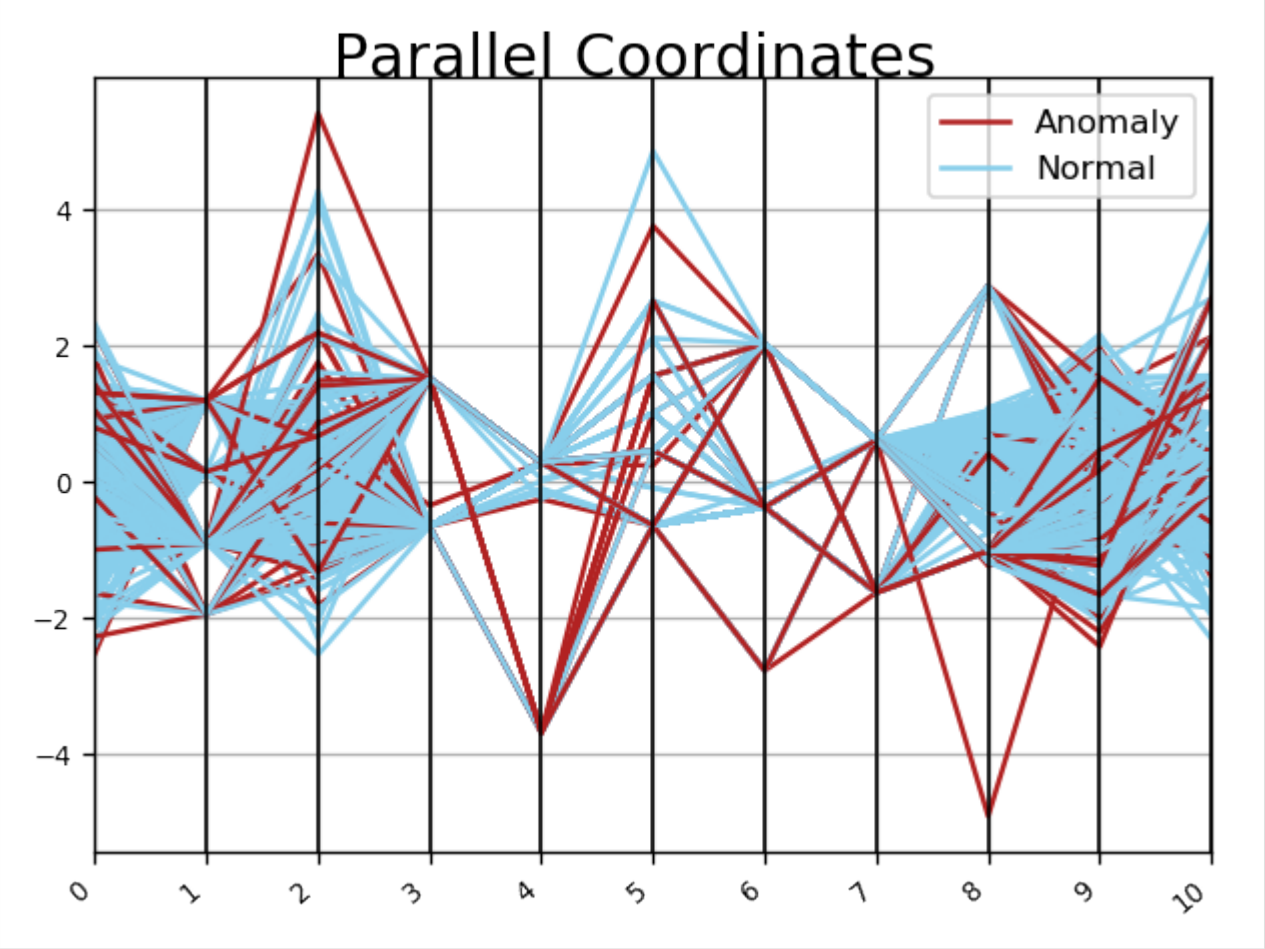
```
The recommended approach is isolation forest.
Do you want to apply the recommended outlier detection approach? [y/n]y
```

```
<IPython.core.display.HTML object>
```




```
<pandas.io.formats.style.Styler at 0x6fd14f5be0>
```


pearsonr = -0.12; p = 0.035
600
500
400
300
200
100
2
0.0
0.2
0.4
0.6
0.8
1.0
4




```
<IPython.core.display.HTML object>
```

```
Do you want to drop outliers? [y/n]y
Outliers are dropped.
```

[ ]:

1.2.2 Example_tasks

[2]:
```
# import datacleanbot and openml
import datacleanbot.dataclean as dc
import openml as oml
import numpy as np
```

Preparation: Acquire Data

The first step is to acquire data from OpneML. The dataset ID can be found in the address.

[3]:
```
# acquire dataset with dataset ID 4
data = oml.datasets.get_dataset(4)
```

(continues on next page)

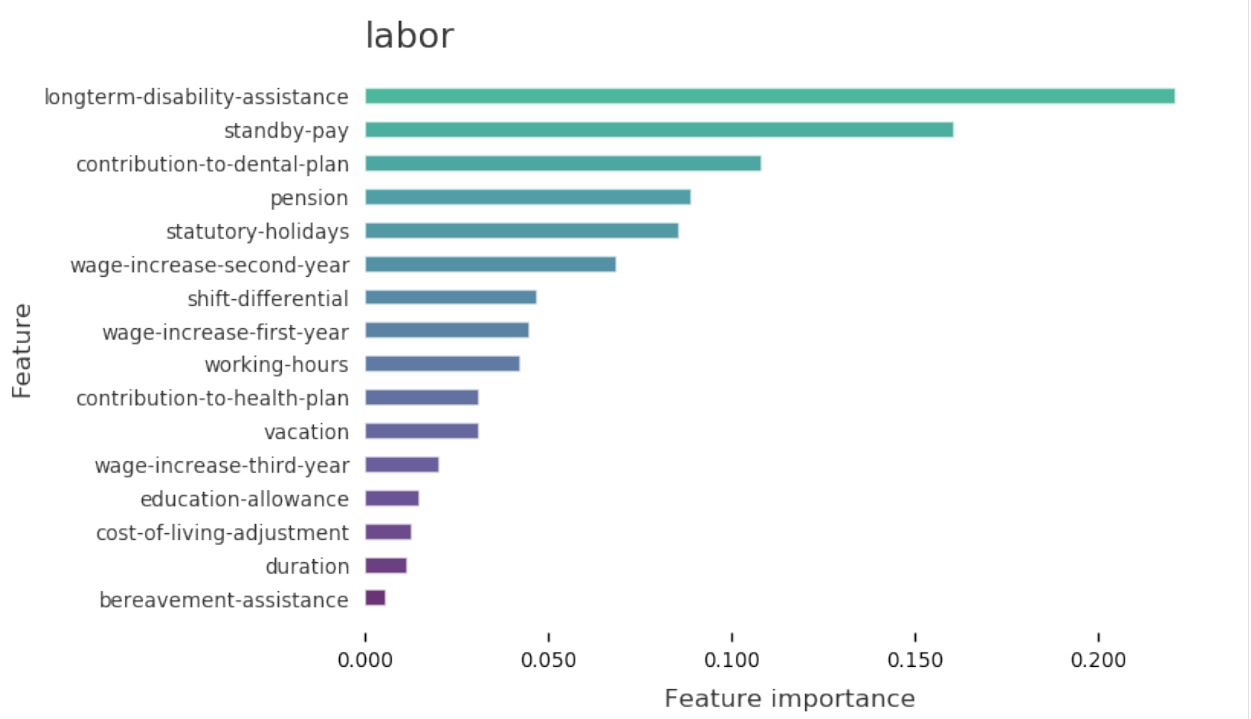
(continued from previous page)

```
X, y, categorical_indicator, features = data.get_data(target=data.default_target_
↪attribute, dataset_format='array')
Xy = np.concatenate((X,y.reshape((y.shape[0],1))), axis=1)
```

Task 1: Show Important Features

[4]:
```
dc.show_important_features(X, y, data.name, features)
```

```
<IPython.core.display.HTML object>
```



Task 2: Unify Column Names

[5]:
```
features = dc.unify_name_consistency(features)
```

```
<IPython.core.display.HTML object>
```

```
Column names
============
['duration', 'wage-increase-first-year', 'wage-increase-second-year',␣
↪'wage-increase-third-year', 'cost-of-living-adjustment', 'working-hours', 'pension',
↪ 'standby-pay', 'shift-differential', 'education-allowance', 'statutory-holidays',␣
↪'vacation', 'longterm-disability-assistance', 'contribution-to-dental-plan',␣
↪'bereavement-assistance', 'contribution-to-health-plan']

Column names are consistent
```

Task 3: Show Statistical Information

[6]:
```
dc.show_statistical_info(Xy)
```

```
<IPython.core.display.HTML object>
```

	0	1	2	3	4	5	\
count	30.000000	37.000000	37.000000	37.000000	56.000000	22.000000	
mean	0.100000	1.108108	1.324324	0.594595	2.160714	0.545455	
std	0.305129	0.774015	0.818333	0.797895	0.707795	0.509647	
min	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	
25%	0.000000	1.000000	1.000000	0.000000	2.000000	0.000000	
50%	0.000000	1.000000	2.000000	0.000000	2.000000	1.000000	
75%	0.000000	2.000000	2.000000	1.000000	3.000000	1.000000	
max	1.000000	2.000000	2.000000	2.000000	3.000000	1.000000	

	6	7	8	9	10	11	\
count	28.000000	27.000000	31.000000	9.000000	53.000000	51.000000	
mean	0.285714	1.037037	4.870968	7.444444	11.094340	0.960784	
std	0.460044	0.939782	4.544168	5.027701	1.259795	0.823669	
min	0.000000	0.000000	0.000000	2.000000	9.000000	0.000000	
25%	0.000000	0.000000	3.000000	2.000000	10.000000	0.000000	
50%	0.000000	1.000000	4.000000	8.000000	11.000000	1.000000	
75%	1.000000	2.000000	5.000000	12.000000	12.000000	2.000000	
max	1.000000	2.000000	25.000000	14.000000	15.000000	2.000000	

	12	13	14	15	16
count	56.000000	46.000000	15.000000	51.000000	57.000000
mean	3.803571	3.971739	3.913333	38.039216	0.649123
std	1.370596	1.164028	1.304315	2.505680	0.481487
min	2.000000	2.000000	2.000000	27.000000	0.000000
25%	2.500000	3.000000	2.400000	37.000000	0.000000
50%	4.000000	4.000000	4.600000	38.000000	1.000000
75%	4.500000	4.500000	5.000000	40.000000	1.000000
max	7.000000	7.000000	5.100000	40.000000	1.000000

Task 4: Discover Data Types

[7]:
```
# input can be Xy or X
dc.discover_types(Xy)
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
['float64', 'int64', 'float64', 'int64', 'int64', 'int64', 'float64', 'int64',␣
↪'int64', 'float64', 'int64', 'int64', 'float64', 'float64', 'float64', 'int64',␣
↪'bool']
```

```
<IPython.core.display.HTML object>
```

```
['Type.POSITIVE', 'Type.REAL', 'Type.REAL', 'Type.POSITIVE', 'Type.POSITIVE', 'Type.
↪POSITIVE', 'Type.POSITIVE', 'Type.POSITIVE', 'Type.POSITIVE', 'Type.POSITIVE',␣
↪'Type.POSITIVE', 'Type.REAL', 'Type.POSITIVE', 'Type.POSITIVE', 'Type.REAL', 'Type.
↪POSITIVE', 'Type.COUNT']
```

Task 5: Clean Duplicated Rows

[8]:
```
Xy = dc.clean_duplicated_rows(Xy)
```

```
<IPython.core.display.HTML object>
```

```
Identifying Duplicated Rows ...
```

```
<IPython.core.display.HTML object>
```

Task 6: Handle Missing Values

[9]:
```
features, Xy = dc.handle_missing(features, Xy)
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```

```
The default setting of missing characters is ['n/a', 'na', '-', '?']
Do you want to add extra character? [y/n]n
```

```
<IPython.core.display.HTML object>
```

```
Number of missing in each feature
0     27
1     20
2     20
3     20
4      1
5     35
6     29
7     30
8     26
9     48
10     4
11     6
12     1
13    11
14    42
15     6
16     0
dtype: int64

Records containing missing values:
    0    1    2    3    4    5    6    7    8    9     10   11   12   13   14  \
0  0.0  NaN  NaN  NaN  1.0  NaN  NaN  NaN  2.0  NaN  11.0  1.0  5.0  NaN  NaN
1  NaN  2.0  2.0  NaN  2.0  0.0  NaN  1.0  NaN  NaN  11.0  0.0  4.5  5.8  NaN
2  0.0  1.0  1.0  NaN  NaN  NaN  0.0  2.0  5.0  NaN  11.0  2.0  NaN  NaN  NaN
3  0.0  NaN  NaN  2.0  3.0  0.0  NaN  NaN  NaN  NaN   NaN  NaN  3.7  4.0  5.0
4  0.0  1.0  1.0  NaN  3.0  NaN  NaN  NaN  NaN  NaN  12.0  1.0  4.5  4.5  5.0

     15   16
0  40.0  1.0
1  35.0  1.0
```

(continues on next page)

(continued from previous page)

```
2  38.0  1.0
3   NaN  1.0
4  40.0  1.0


Missing correlation between features containing missing values and other features
          0         1         2         3         4         5         6   \
0   1.000000  0.112373  0.259620  0.185996 -0.126773  0.030389  0.018496
1   0.112373  1.000000  0.460811  0.152703 -0.098247  0.280850  0.575362
2   0.259620  0.460811  1.000000  0.229730 -0.098247  0.129827  0.428296
3   0.185996  0.152703  0.229730  1.000000  0.181757  0.280850  0.354763
4  -0.126773 -0.098247 -0.098247  0.181757  1.000000  0.105946 -0.135996
5   0.030389  0.280850  0.129827  0.280850  0.105946  1.000000  0.374342
6   0.018496  0.575362  0.428296  0.354763 -0.135996  0.374342  1.000000
7   0.196296  0.182121  0.182121  0.255745 -0.140859  0.113961  0.332923
8  -0.304456  0.064742  0.064742 -0.082870 -0.122380  0.002539  0.195304
9  -0.167360 -0.084895  0.015918  0.116731  0.057864  0.150845  0.248198
10  0.014479  0.085841  0.229751 -0.201979 -0.036711 -0.064352 -0.004820
11  0.132569 -0.012609  0.346743 -0.012609 -0.045835  0.037082 -0.006018
12 -0.126773 -0.098247 -0.098247  0.181757  1.000000  0.105946 -0.135996
13  0.070290  0.013074  0.106224  0.013074  0.273268 -0.160206 -0.141967
14  0.327569 -0.061512  0.105450  0.021969  0.079860 -0.146448 -0.268444
15 -0.096414 -0.132393  0.107175 -0.132393 -0.045835 -0.197772 -0.234718

          7         8         9         10        11        12        13  \
0   0.196296 -0.304456 -0.167360  0.014479  0.132569 -0.126773  0.070290
1   0.182121  0.064742 -0.084895  0.085841 -0.012609 -0.098247  0.013074
2   0.182121  0.064742  0.015918  0.229751  0.346743 -0.098247  0.106224
3   0.255745 -0.082870  0.116731 -0.201979 -0.012609  0.181757  0.013074
4  -0.140859 -0.122380  0.057864 -0.036711 -0.045835  1.000000  0.273268
5   0.113961  0.002539  0.150845 -0.064352  0.037082  0.105946 -0.160206
6   0.332923  0.195304  0.248198 -0.004820 -0.006018 -0.135996 -0.141967
7   1.000000 -0.259902  0.071001  0.123072  0.210905 -0.140859 -0.159324
8  -0.259902  1.000000  0.396558  0.299976  0.259754 -0.122380 -0.269331
9   0.071001  0.396558  1.000000  0.118958  0.148522  0.057864 -0.275913
10  0.123072  0.299976  0.118958  1.000000  0.800943 -0.036711 -0.134341
11  0.210905  0.259754  0.148522  0.800943  1.000000 -0.045835 -0.167729
12 -0.140859 -0.122380  0.057864 -0.036711 -0.045835  1.000000  0.273268
13 -0.159324 -0.269331 -0.275913 -0.134341 -0.167729  0.273268  1.000000
14 -0.167984 -0.172611 -0.149514 -0.147760 -0.054661  0.079860  0.292239
15 -0.018078  0.374528  0.148522  0.353357  0.441176 -0.045835 -0.022872

          14        15
0   0.327569 -0.096414
1  -0.061512 -0.132393
2   0.105450  0.107175
3   0.021969 -0.132393
4   0.079860 -0.045835
5  -0.146448 -0.197772
6  -0.268444 -0.234718
7  -0.167984 -0.018078
8  -0.172611  0.374528
9  -0.149514  0.148522
10 -0.147760  0.353357
11 -0.054661  0.441176
12  0.079860 -0.045835
13  0.292239 -0.022872
```

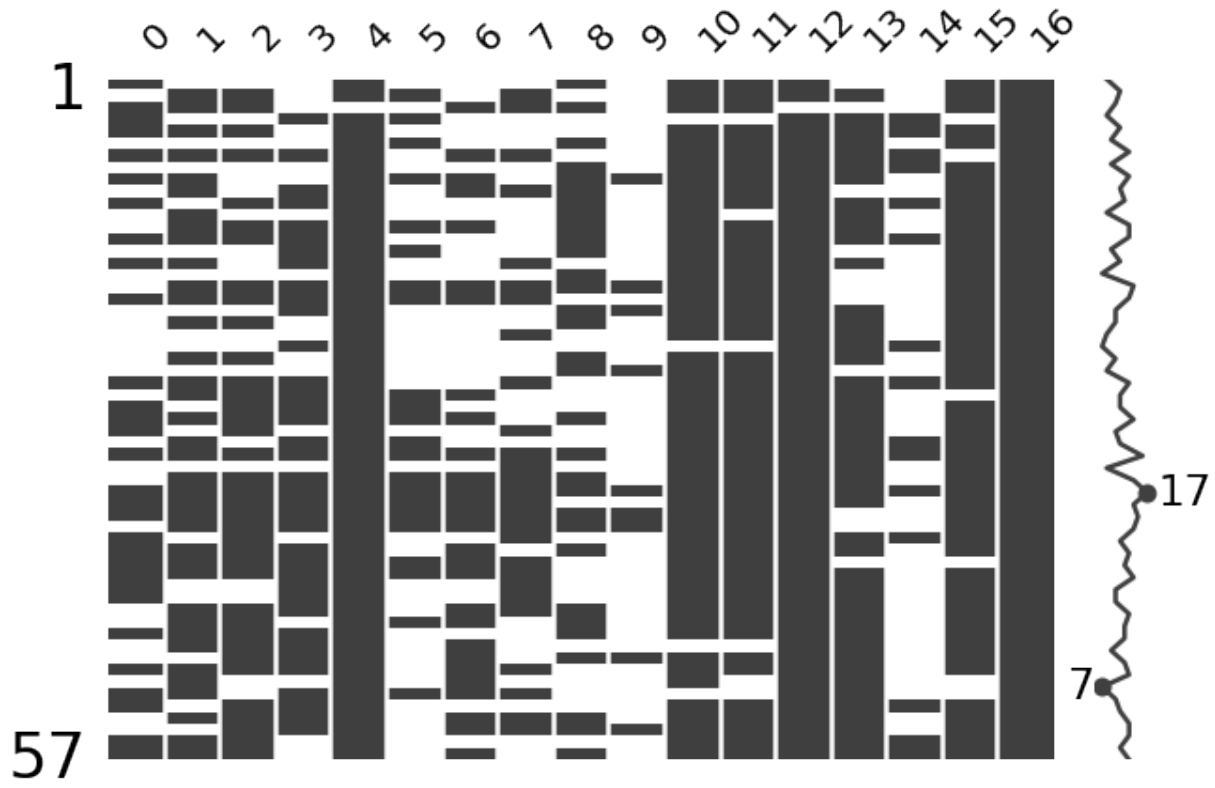
(continues on next page)

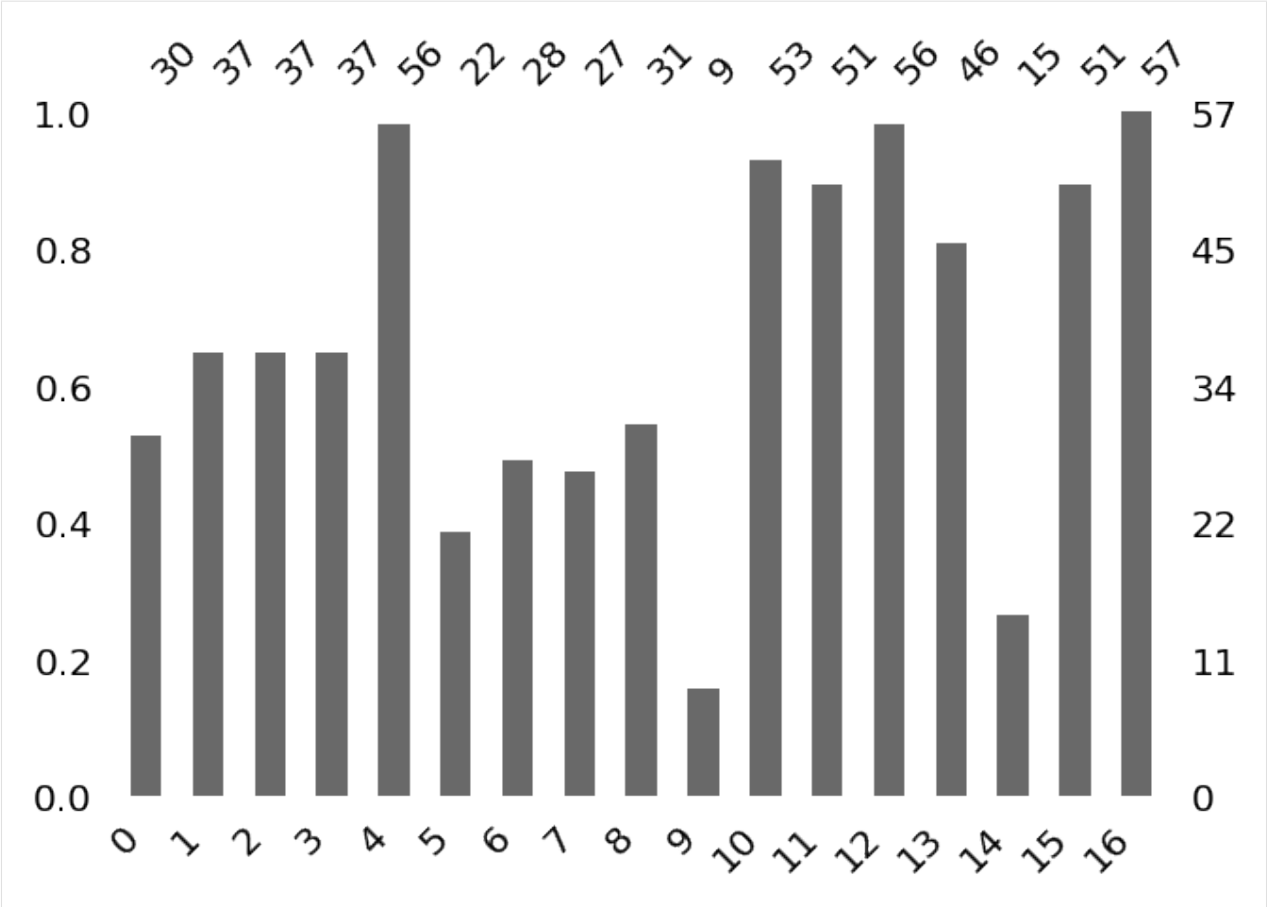
(continued from previous page)

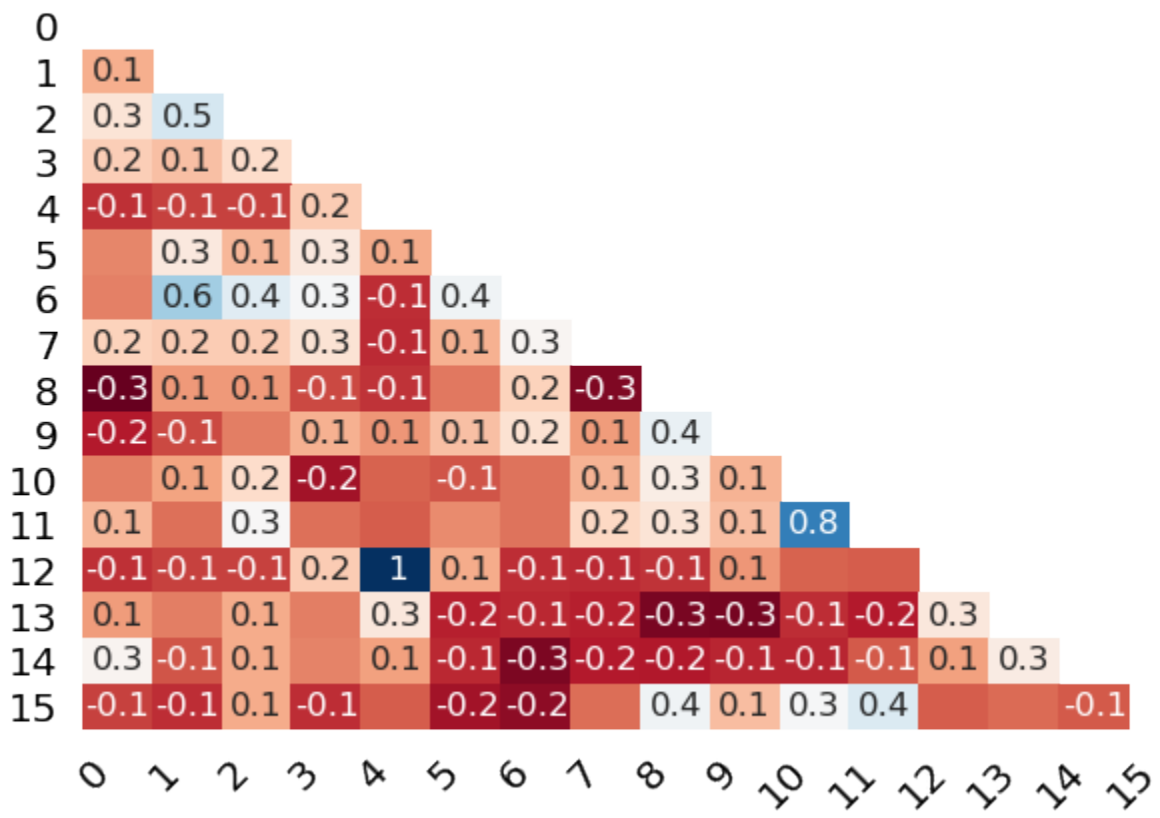
```
14  1.000000 -0.054661
15 -0.054661  1.000000
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.HTML object>
```




30 37 37 37 56 22 28 27 31 9 53 51 56 46 15 51 57
1.0
0.8
0.6
0.4
0.2
0.0
57
45
34
22
11
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16




```
<IPython.core.display.HTML object>
```

```
Choose the missing mechanism [a/b/c/d]:
a.MCAR b.MAR c.MNAR d.Skip
a
Missing percentage is 0.9824561403508771
Imputation score of mean is 0.8515151515151516
Imputation score of mode is 0.8674242424242424
Imputation score of knn is 0.9299242424242424
Imputation score of matrix factorization is 0.9299242424242424
Imputation score of multiple imputation is 0.9291666666666667
Imputation method with the highest socre is knn
```

```
<IPython.core.display.HTML object>
```

```
The recommended approach is knn
Do you want to apply the recommended approach? [y/n]n


Choose the approach you want to apply [a/b/c/d/e/skip]:
a.Mean b.Mode c.K Nearest Neighbor d.Matrix Factorization e. Multiple Imputation
a

Applying mean imputation ...
Missing values cleaned!
```

Task 7: Handle Outliers

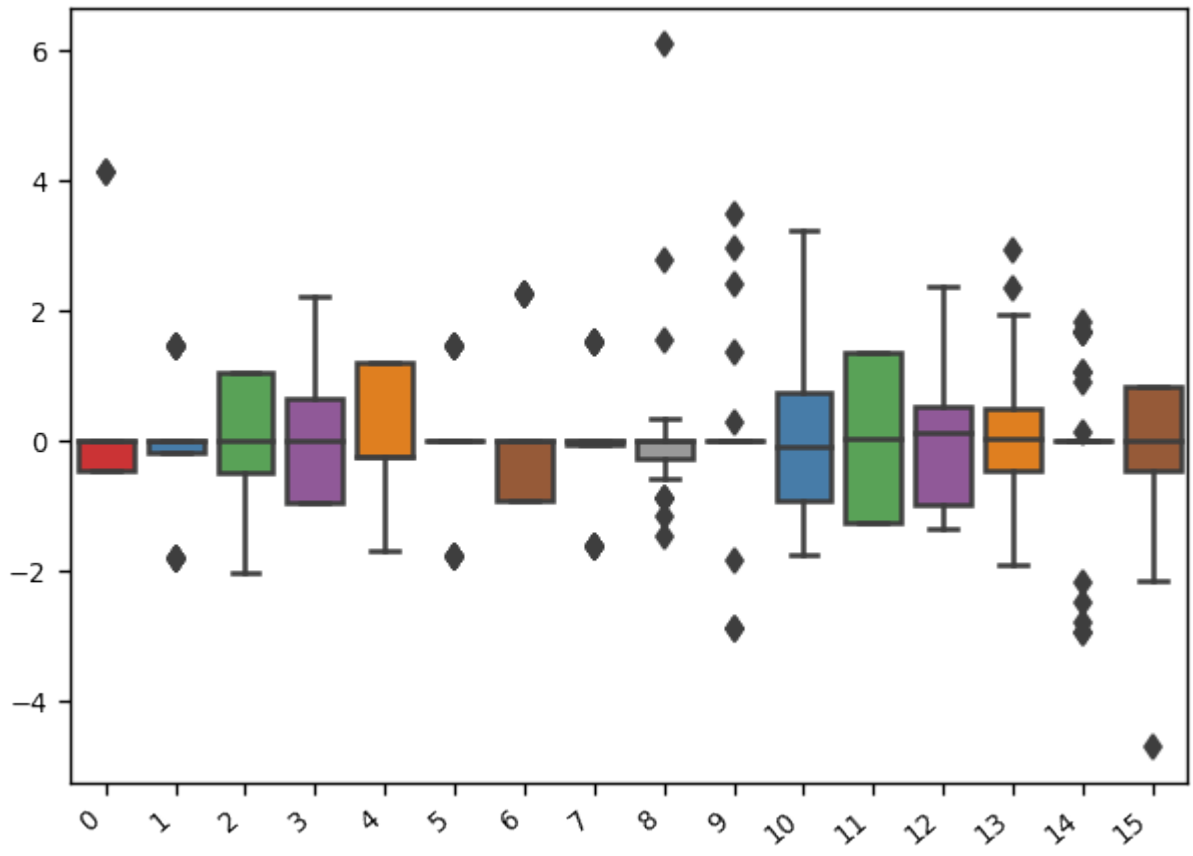
[10]:
```
Xy = dc.handle_outlier(features,Xy)
```

```
<IPython.core.display.HTML object>
```

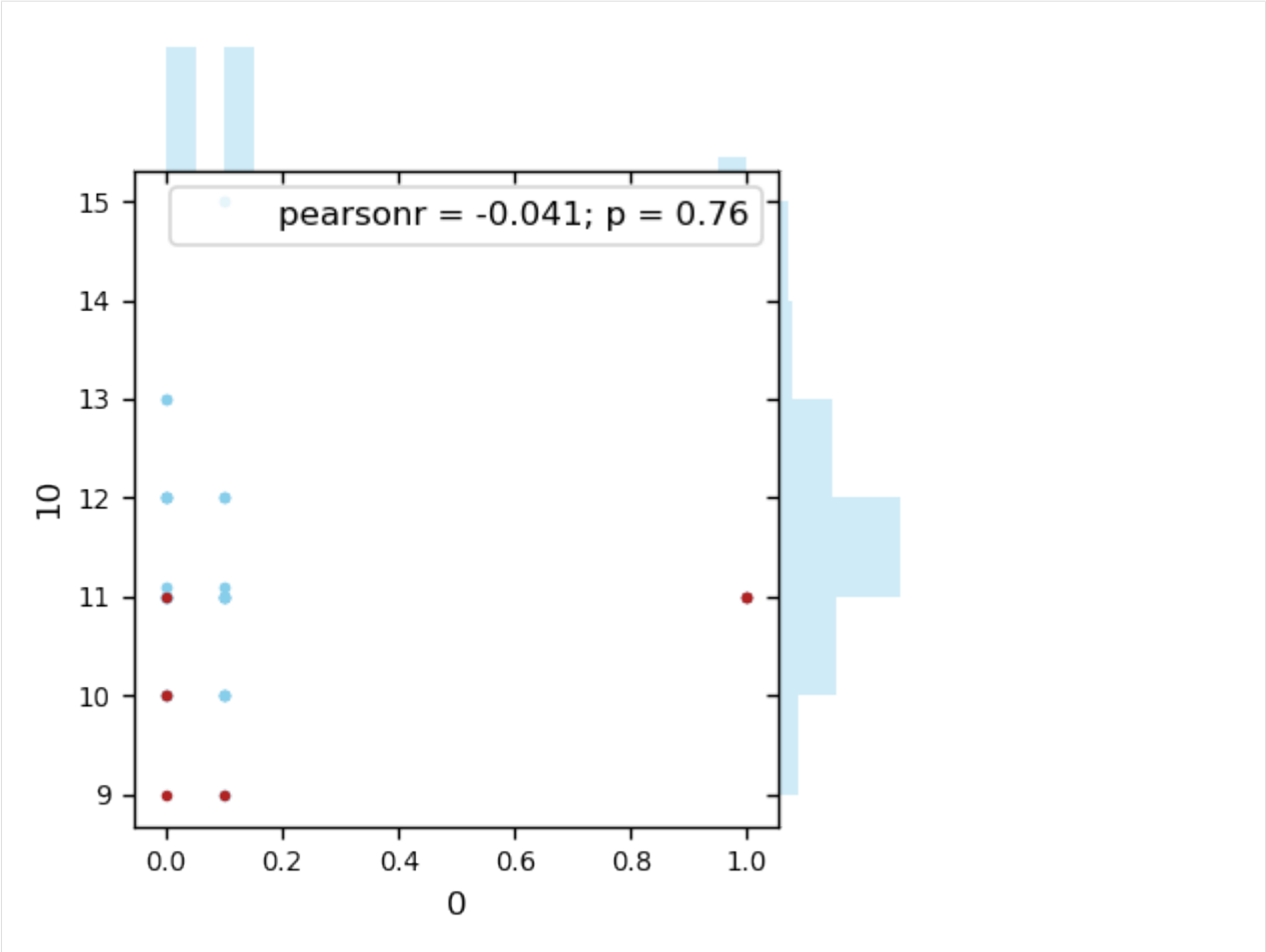
```
<IPython.core.display.HTML object>
```

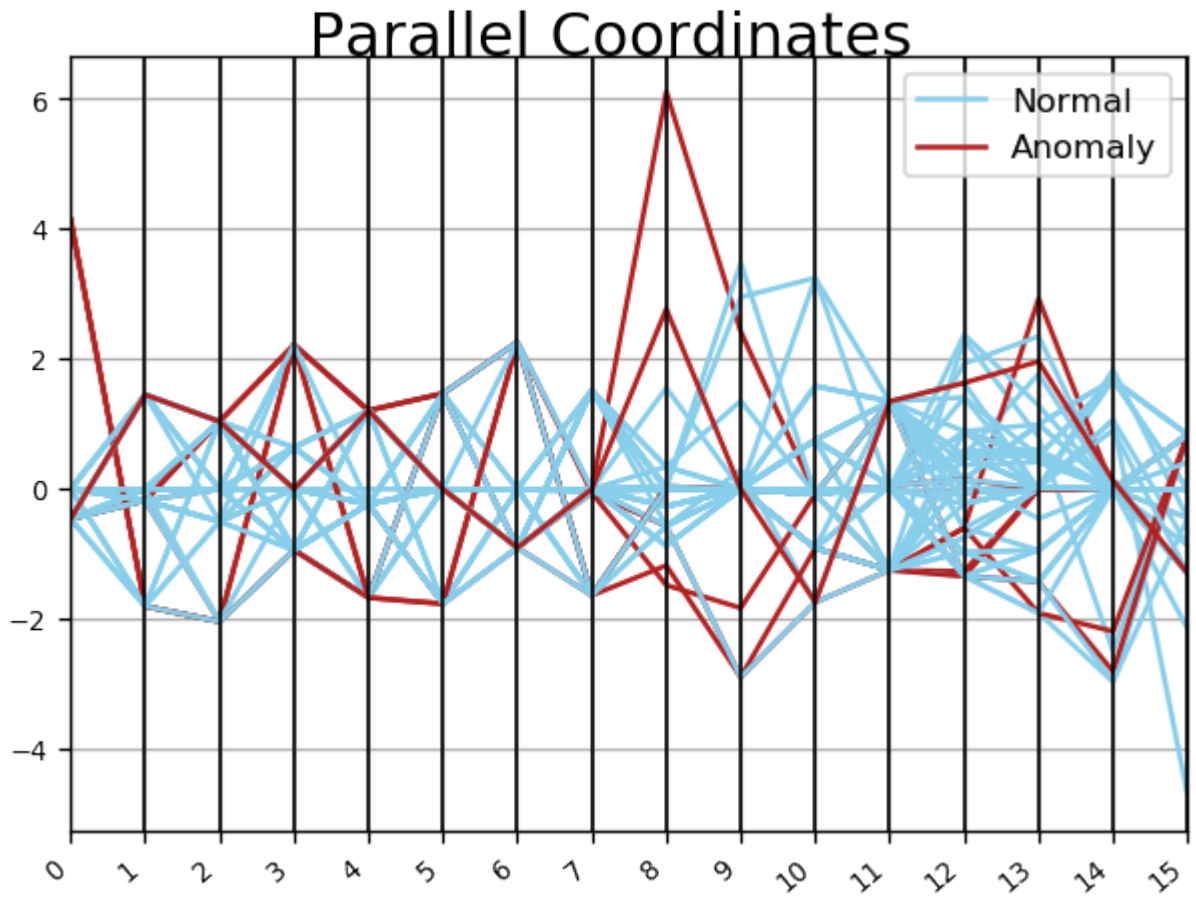
```
The recommended approach is isolation forest.
Do you want to apply the recommended outlier detection approach? [y/n]y
```

```
<IPython.core.display.HTML object>
```




```
<pandas.io.formats.style.Styler at 0x210cddb4e0>
```


pearsonr = -0.041; p = 0.76
15
14
13
12
11
10
9
10
0.0
0.2
0.4
0.6
0.8
1.0
0




```
<IPython.core.display.HTML object>
```

```
Do you want to drop outliers? [y/n]n
Outliers are kept.
```

[ ]:

CHAPTER 2

API Reference

2.1 API

A data cleaning Python tool.

dataclean.**autoclean**(*Xy*, *dataset_name*, *features*)

Auto-cleans data.

The following aspects are automatically cleaned: show important features; show statistical information; discover the data type for each feature; identify the duplicated rowsl; unify the inconsistent column names; handle missing values; handle outliers.

Parameters **Xy** : array-like

Complete data.

dataset_name : string

features : list

List of feature names.

Returns **Xy_cleaned** : array-like

Cleaned data.

dataclean.**build_forest**(*X*, *y*)

Build random forest model from the dataset and compute important features

Parameters **X** : array-like, shape (n_samples, n_features)

Training vectors, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples,)

Target values (class labels in classification, real numbers in regression).

Returns **importances** : array, shape = [n_features]

The feature importances (the higher, the more important the feature).

indices : array, shape = [n_features]

Reverse the importances.

dataclean.**clean_duplicated_rows**(*Xy*)

Clean duplicated rows.

Parameters **Xy** : array-like

Complete numpy array (target required) of the dataset.

Returns **Xy** : array-like

Original data.

Xy_no_duplicate : array-like

Cleaned data without duplicated rows if user wants to drop the duplicated rows.

dataclean.**clean_missing**(*df*, *features*)

Clean missing values in the dataset.

Parameters **df** : DataFrame

features : List

List of feature names.

Returns **features_new** : List

List of feature names after cleaning.

Xy_filled : array-like

Numpy array where missing values have been cleaned.

dataclean.**compute_clustering_metafeatures**(*X*)

Computes clustering meta features.

The following 3 clustering meta features are adopted: Silhouette Coefficient; Calinski_Harabasz Index; Davies_Bouldin Index.

dataclean.**compute_imputation_score**(*Xy*)

Computes score of the imputation by applying simple classifiers.

The following simple learners are evaluated: Naive Bayes Learner; Linear Discriminant Learner; One Nearest Neighbor Learner; Decision Node Learner.

Parameters **Xy** : array-like

Complete numpy array of the dataset. The training array X has to be imputed already, and the target y is required here and not optional in order to predict the performance of the imputation method.

Returns **imputation_score** : float

Predicted score of the imputation method.

dataclean.**compute_metafeatures**(*X*, *y*)

Computes landmarking meta features.

The following landmarking features are computed: Naive Bayes Learner; Linear Discriminant Learner; One Nearest Neighbor Learner; Decision Node Learner; Randomly Chosen Node Learner.

dataclean.**deal_mar**(*df*)

Deal with missing data with missing at random pattern.

dataclean.**deal_mcar**(*df*)
Deal with missing data with missing completely at random pattern.

dataclean.**deal_mnar**(*df*)
Deal with missing data with missing at random pattern.

dataclean.**discover_type_heuristic**(*data*)
Infer data types for each feature using simple logic

Parameters **data** : numpy array or dataframe

Numeric data needs to be 64 bit.

Returns **result** : list

List of data types.

dataclean.**discover_types**(*Xy*)
Discover types for numpy array.

Both simple logic rules and Bayesian methods are applied. Bayesian methods can only be applied if Xy are numeric.

Parameters **Xy** : numpy array or DataFrame

Xy can only be numeric in order to run the Bayesian model.

dataclean.**drop_duplicated_rows**(*dataframe*)
Drop duplicatd rows.

dataclean.**drop_outliers**(*df*, *df_outliers*)
Drops the detected outliers.

dataclean.**handle_missing**(*features*, *Xy*)
Handle missing values.

Recommend the approprate approach to the user given the missing mechanism of the dataset. The user can choose to adopt the recommended approach or take another available approach.

For MCAR, the following methods are evaluated: 'list deletion', 'mean', 'mode', 'k nearest neighbors', 'matrix factorization', 'multiple imputation'.

For MAR, the following methods are evaluated: 'k nearest neighbors', 'matrix factorization', 'multiple imputation'.

For MNAR, 'multiple imputation' is adopted.

Parameters **features** : list

List of feature names.

Xy : array-like

Complete numpy array (target required and not optional).

Returns **features_new** : List

List of feature names after cleaning.

Xy_filled : array-like

Numpy array where missing values have been cleaned.

dataclean.**handle_outlier**(*features*, *Xy*)
Cleans the outliers.

Recommends the algorithm to the user to detect the outliers and presents the outliers to the user in effective visualizations. The user can decides whether or not to keep the outliers.

Parameters **features** : list

List of feature names.

Xy : array-like

Numpy array. Both training vectors and target are required.

Returns **Xy_no_outliers** : array-like

Cleaned data where outliers are dropped.

Xy : array-like

Original data where outliers are not found or kept.

dataclean.**highlight_outlier**(*data*)
Highlight the maximum in a Series yellow.

dataclean.**identify_missing**(*df=None*)
Detect missing values.

Identify the common missing characters such as 'n/a', 'na', '–' and '?' as missing. User can also customize the characters to be identified as missing.

Parameters **df** : DataFrame

Raw data formatted in DataFrame.

Returns **flag** : bool

Indicates whether missing values are detected. If true, missing values are detected. Otherwise not.

dataclean.**identify_missing_mechanism**(*df=None*)
Tries to guess the missing mechanism of the dataset.

Missing mechanism is not really testable. There may be reasons to suspect that the dataset belongs to one missing mechanism based on the missing correlation between features, but the result is not definite. Relevant information are provided to help the user make the decision. Three missng mechanisms to be guessed: MCAR: Missing completely at ramdom MAR: Missing at random MNAR: Missing not at random (not available here, normally involes field expert)

Parameters **df** : DataFrame

Raw data formatted in DataFrame.

dataclean.**identify_outliers**(*df*, *algorithm=0*, *detailed=False*)
Identifies outliers in multi dimension.

Dataset has to be parsed as numeric beforehand.

dataclean.**infer_feature_type**(*feature*)
Infer data types for the given feature using simple logic.

Possible data types to infer: boolean, date, float, integer, string Feature that is not either a boolean, a date, a float or an integer, is classified as a string.

Parameters **feature** : array-like

A feature/attribute vector.

Returns **data_type** : string

The data type of the given feature/attribute.

dataclean.**missing_preprocess**(*features*, *df=None*)
Drops the redundant information.

Redundant information is dropped before imputation. Detects and drops empty rows. Detects features and instances with extreme large proportion of missing data and reports to the user.

Parameters **features** : list

List of feature names.

df : DataFrame

Returns **df** : DataFrame

New DataFrame where redundant information may have been deleted.

features_new: list

List of feature names after preprocessing.

dataclean.**plot_feature_importances**(*dataset_name*, *features*, *importances*, *indices*)
Plot the 15 most important features.

dataclean.**predict_best_anomaly_algorithm**(*X*, *y*)
Predicts best anomaly detection algorithm.

Recommends the best anomaly detection algorithm to the user given the characteristics of the dataset. The following algorithms are considered: 0: isolation forest; 1: local outlier factor; 2: one class support vector machine.

dataclean.**show_important_features**(*X*, *y*, *data_name*, *features*)
Show the most important features of the given dataset.

Computes the most important features of the given dataset using random forest, and present the 15 most useful features to the user with a bar chart.

Parameters **X** : array-like, shape (n_samples, n_features)

Training vectors, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples,)

Target values (class labels in classification, real numbers in regression).

data_name : string

Dataset name.

features : list

List of feature names.

dataclean.**show_statistical_info**(*Xy*)
Show statistical information of the given dataset

Parameters **Xy** : array-like

dataclean.**train_metalearner**()
Train metalearner

dataclean.**unify_name_consistency**(*names*)
Unify inconsistent column names.

Parameters **names** : list

List of original column names.

Returns **names** : list

Unified column names.

dataclean.**visualize_missing**(*df=None*)
Visualize missing values.

The missingness of the dataset is visualized in bar chart, matrix and heatmap.

dataclean.**visualize_outliers_parallel_coordinates**(*df_scaled*, *df_pred*)
Visualizes high-dimensional outliers with a parallel coordinates plot.

dataclean.**visualize_outliers_scatter**(*df*, *df_pred*)
Visualizes high-dimensional outliers with a scatter plot.

Selects out the two features most likely to have outliers and shows them in a scatter plot.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index