Missing Data Handeling

Author: Eri G Osta

Date: April 5, 2023

MissingDataHandler Class

The `MissingDataHandler` class is a data analysis tool that can handle missing data in a given dataset. It has methods to perform mean substitution, simple regression, and multiple imputation to impute the missing values. It also has methods to calculate statistics, compare statistics between the original and imputed data, and produce a correlation matrix with missing data information.

The class can be initialized with a filename of a CSV file that contains the dataset. The CSV file should have columns of numerical data, and missing values should be represented as NaNs. Once initialized, the user can call various methods of the class to perform the desired analysis.

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.impute import SimpleImputer
from sklearn.impute import IterativeImputer
import pandas as pd
from scipy.stats import pearsonr

class MissingDataHandler:
    """
    Attributes
    ----------
    df : pandas.DataFrame
        The original dataset.
    mean_imputed_df : pandas.DataFrame
        The dataset after mean substitution.
    simple_imputed_df : pandas.DataFrame
        The dataset after simple regression imputation.
    multi_imputed_df : pandas.DataFrame
        The dataset after multiple imputation.

    Methods
    -------
    calculate_statistics()
        Calculate mean and standard deviation of each column in the
dataset.
    mean_substitution()
        Perform mean substitution on the dataset.
    simple_regression()
```

```
        Perform simple regression imputation on the dataset.
    multiple_imputation()
        Perform multiple imputation on the dataset.
    compare_statistics(imputed_df, original_df)
        Compare mean and standard deviation of each column in the
original dataset and the imputed dataset.
    descriptive_statistics(df)
        Print descriptive statistics for the dataset.
    correlation_matrix(df)
        Calculate the correlation matrix with Pearson correlation
coefficients with their p-values and missing data.
    """

    def __init__(self, filename):
        """
        Parameters
        ----------
        filename : str
            The name of the CSV file containing the dataset.
        """
        self.df = pd.read_csv(filename)
        self.mean_imputed_df = None
        self.simple_imputed_df = None
        self.multi_imputed_df = None

    def calculate_statistics(self):
        """Calculate mean and standard deviation of each column in the
dataset."""
        stats = pd.DataFrame({'mean': self.df.mean(), 'std':
self.df.std()})
        print(stats)

    def mean_substitution(self):
        """Perform mean substitution on the dataset."""
        imputer = SimpleImputer(missing_values=np.nan,
strategy='mean')
        mean_imputed_df = pd.DataFrame(imputer.fit_transform(self.df),
columns=self.df.columns)
        self.mean_imputed_df = mean_imputed_df
        return mean_imputed_df

    def simple_regression(self):
        """Perform simple regression imputation on the dataset."""
        reg = LinearRegression()
        df = self.df.copy()
        df.dropna(subset=['v3_miss'], inplace=True)  # drop rows with
missing values in v3_miss
        y = df['v3_miss']
        X = df[['v1_miss', 'v2_miss', 'v4_miss', 'v5_miss']]
        imputer = SimpleImputer(missing_values=np.nan,
```

```
strategy='mean')
        X_imputed = pd.DataFrame(imputer.fit_transform(X),
columns=X.columns)
        reg.fit(X_imputed, y)
        X_test = df[['v1_miss', 'v2_miss', 'v4_miss', 'v5_miss']]
        X_test_imputed = pd.DataFrame(imputer.transform(X_test),
columns=X_test.columns)
        y_pred = pd.Series(reg.predict(X_test_imputed),
index=df.index)
        simple_imputed_df = df.copy()
        simple_imputed_df['v3_miss'] = np.where(df['v3_miss'].isna(),
y_pred, df['v3_miss'])
        self.simple_imputed_df = simple_imputed_df
        return simple_imputed_df

    def multiple_imputation(self):
        """Perform multiple imputation on the dataset."""
        imputer = IterativeImputer()
        multi_imputed_df =
pd.DataFrame(imputer.fit_transform(self.df), columns=self.df.columns)
        self.multi_imputed_df = multi_imputed_df
        return multi_imputed_df

    def compare_statistics(self, imputed_df, original_df):
        orig_stats = pd.DataFrame({'mean': original_df.mean(), 'std':
original_df.std()}, index=original_df.columns)
        new_stats = pd.DataFrame({'mean': imputed_df.mean(), 'std':
imputed_df.std()}, index=imputed_df.columns)
        print('Original:\n', orig_stats)
        print('New:\n', new_stats)

    def descriptive_statistics(self, df):
        print('Descriptive Statistics:')
        print(df.describe())

    def correlation_matrix(self, df):
        """Prints a correlation matrix with Pearson correlation
coefficients and corresponding p-values and missing data.

        Args:
            df (pandas.DataFrame): The DataFrame to calculate the
correlation matrix from.

        Returns:
            pandas.DataFrame: The correlation matrix with Pearson
correlation coefficients and corresponding p-values and missing data.
        """
        corr_matrix = pd.DataFrame(index=df.columns,
columns=df.columns, dtype=np.float64)
        p_values = pd.DataFrame(index=df.columns, columns=df.columns,
```

```
dtype=np.float64)
        missing_values = pd.DataFrame(index=df.columns,
columns=df.columns, dtype=np.int64)
        for i, col_i in enumerate(df.columns):
            for j, col_j in enumerate(df.columns):
                data_i = df[col_i].dropna()
                data_j = df[col_j].dropna()
                intersection = data_i.index.intersection(data_j.index)
                n_missing = len(df) - df[[col_i,
col_j]].notna().all(axis=1).sum()
                if len(intersection) > 1:
                    corr, p = pearsonr(data_i[intersection],
data_j[intersection])
                else:
                    corr, p = np.nan, np.nan
                corr_matrix.iloc[i, j] = corr
                p_values.iloc[i, j] = p
                missing_values.iloc[i, j] = n_missing
        corr_matrix = corr_matrix.round(2)
        p_values = p_values.round(3)
        missing_values = missing_values.astype(str).replace('\.0$',
'', regex=True)
        result = corr_matrix.astype(str) + ' (p-value: ' +
p_values.astype(str) + ' | missing: ' + 'missing: ' +
missing_values.astype(str) + ')'
        return result
```

Initialize `MissingDataHandler` and add path to data source.

```
mdh = MissingDataHandler('data.csv')
```

Calculate mean and standard deviantion for all columns

```
mdh.calculate_statistics()
```

```
             mean       std
v1_miss  3.128788  1.213193
v2_miss  3.366412  1.144929
v3_miss  1.976562  1.090148
v4_miss  2.201550  1.134542
v5_miss  2.178862  1.293315
```

###. Replace missing data with the mean value for its corresponding column

```
mean_sub_df = mdh.mean_substitution()
# print all rows
with pd.option_context('display.max_rows', None,
'display.max_columns', None):
    print(mean_sub_df)
```

```
     v1_miss  v2_miss   v3_miss  v4_miss   v5_miss
0        4.0      3.0  1.000000  3.00000  1.000000
1        2.0      4.0  1.000000  2.20155  3.000000
```

```
2        3.0      3.0  3.000000  3.00000  2.178862
3        2.0      4.0  1.000000  2.00000  2.000000
4        2.0      4.0  2.000000  5.00000  5.000000
..       ...      ...       ...      ...       ...
144      4.0      4.0  4.000000  3.00000  2.000000
145      2.0      4.0  2.000000  1.00000  2.178862
146      4.0      2.0  3.000000  4.00000  1.000000
147      3.0      3.0  3.000000  3.00000  3.000000
148      3.0      3.0  1.976562  3.00000  4.000000

[149 rows x 5 columns]
      v1_miss   v2_miss   v3_miss  v4_miss   v5_miss
0    4.000000  3.000000  1.000000  3.00000  1.000000
1    2.000000  4.000000  1.000000  2.20155  3.000000
2    3.000000  3.000000  3.000000  3.00000  2.178862
3    2.000000  4.000000  1.000000  2.00000  2.000000
4    2.000000  4.000000  2.000000  5.00000  5.000000
5    4.000000  3.366412  1.000000  2.00000  1.000000
6    2.000000  4.000000  3.000000  1.00000  3.000000
7    2.000000  5.000000  3.000000  3.00000  1.000000
8    1.000000  5.000000  2.000000  3.00000  2.000000
9    3.000000  3.000000  1.000000  1.00000  3.000000
10   4.000000  1.000000  1.000000  1.00000  1.000000
11   5.000000  4.000000  1.000000  2.00000  1.000000
12   2.000000  4.000000  3.000000  4.00000  1.000000
13   4.000000  1.000000  1.000000  1.00000  1.000000
14   2.000000  3.000000  1.000000  1.00000  1.000000
15   3.128788  3.366412  1.976562  2.20155  2.178862
16   4.000000  4.000000  4.000000  5.00000  4.000000
17   3.000000  3.000000  2.000000  1.00000  1.000000
18   3.000000  2.000000  2.000000  3.00000  5.000000
19   1.000000  3.000000  1.000000  1.00000  1.000000
20   3.128788  4.000000  1.000000  3.00000  1.000000
21   3.000000  4.000000  2.000000  2.20155  4.000000
22   4.000000  3.000000  1.000000  3.00000  3.000000
23   5.000000  4.000000  1.000000  3.00000  2.000000
24   4.000000  4.000000  5.000000  4.00000  2.178862
25   4.000000  5.000000  1.000000  1.00000  1.000000
26   1.000000  2.000000  3.000000  2.00000  3.000000
27   3.000000  3.000000  1.000000  3.00000  1.000000
28   3.000000  3.366412  4.000000  3.00000  3.000000
29   1.000000  5.000000  3.000000  1.00000  1.000000
30   3.000000  3.000000  1.000000  1.00000  2.178862
31   3.000000  3.366412  1.976562  2.00000  2.000000
32   3.128788  3.366412  1.976562  2.20155  2.178862
33   2.000000  1.000000  1.000000  1.00000  1.000000
34   3.128788  3.366412  1.000000  3.00000  1.000000
35   4.000000  5.000000  3.000000  3.00000  2.000000
36   3.128788  3.366412  1.000000  2.20155  1.000000
37   4.000000  4.000000  1.000000  1.00000  1.000000
```

38	3.128788	3.000000	1.976562	3.00000	3.000000
39	3.000000	2.000000	2.000000	3.00000	4.000000
40	5.000000	2.000000	1.000000	1.00000	1.000000
41	1.000000	2.000000	2.000000	2.00000	2.178862
42	3.000000	3.000000	1.000000	1.00000	2.178862
43	4.000000	4.000000	4.000000	1.00000	2.000000
44	4.000000	4.000000	3.000000	2.00000	2.000000
45	5.000000	5.000000	3.000000	2.00000	3.000000
46	4.000000	4.000000	3.000000	1.00000	3.000000
47	5.000000	2.000000	2.000000	1.00000	1.000000
48	3.128788	2.000000	1.000000	2.20155	2.178862
49	3.000000	4.000000	1.000000	1.00000	1.000000
50	3.128788	1.000000	1.000000	2.20155	1.000000
51	3.000000	5.000000	3.000000	2.20155	1.000000
52	2.000000	2.000000	2.000000	2.00000	1.000000
53	4.000000	2.000000	2.000000	4.00000	1.000000
54	3.128788	3.366412	1.976562	2.20155	2.178862
55	3.128788	3.366412	1.976562	2.20155	2.178862
56	4.000000	3.366412	1.000000	1.00000	2.000000
57	3.128788	2.000000	3.000000	4.00000	3.000000
58	2.000000	3.000000	1.000000	1.00000	1.000000
59	4.000000	4.000000	1.976562	3.00000	2.178862
60	3.128788	4.000000	3.000000	3.00000	3.000000
61	2.000000	3.000000	1.000000	2.00000	5.000000
62	1.000000	1.000000	1.000000	3.00000	1.000000
63	5.000000	2.000000	1.000000	1.00000	2.178862
64	3.000000	3.000000	3.000000	3.00000	3.000000
65	4.000000	4.000000	1.000000	1.00000	2.178862
66	4.000000	4.000000	2.000000	2.00000	2.000000
67	4.000000	3.000000	3.000000	2.00000	2.000000
68	5.000000	5.000000	2.000000	3.00000	3.000000
69	5.000000	3.366412	1.000000	3.00000	1.000000
70	3.000000	3.000000	3.000000	2.20155	3.000000
71	3.128788	3.000000	4.000000	3.00000	3.000000
72	2.000000	4.000000	3.000000	1.00000	3.000000
73	3.000000	3.000000	3.000000	2.20155	2.178862
74	3.000000	4.000000	1.000000	1.00000	4.000000
75	4.000000	5.000000	1.000000	1.00000	3.000000
76	1.000000	3.000000	1.000000	1.00000	1.000000
77	4.000000	4.000000	2.000000	2.20155	4.000000
78	2.000000	4.000000	3.000000	2.00000	1.000000
79	5.000000	1.000000	1.976562	1.00000	5.000000
80	4.000000	4.000000	1.976562	1.00000	4.000000
81	4.000000	4.000000	1.000000	1.00000	1.000000
82	3.000000	3.000000	1.000000	3.00000	2.000000
83	2.000000	5.000000	1.000000	1.00000	1.000000
84	4.000000	3.000000	1.976562	3.00000	1.000000
85	4.000000	3.000000	1.000000	1.00000	2.000000
86	1.000000	4.000000	1.000000	1.00000	1.000000
87	2.000000	4.000000	1.976562	2.00000	2.000000

88	4.000000	4.000000	2.000000	3.00000	4.000000
89	3.000000	3.000000	1.976562	2.00000	2.178862
90	3.000000	5.000000	1.000000	1.00000	1.000000
91	2.000000	4.000000	1.000000	1.00000	1.000000
92	5.000000	1.000000	1.976562	1.00000	5.000000
93	1.000000	4.000000	1.000000	2.00000	1.000000
94	3.000000	2.000000	2.000000	3.00000	1.000000
95	3.000000	3.000000	3.000000	3.00000	3.000000
96	3.000000	4.000000	2.000000	2.00000	2.000000
97	4.000000	2.000000	2.000000	2.00000	1.000000
98	3.000000	3.366412	3.000000	2.00000	2.178862
99	1.000000	3.000000	1.000000	3.00000	1.000000
100	1.000000	1.000000	1.000000	1.00000	2.000000
101	2.000000	3.000000	1.000000	3.00000	1.000000
102	5.000000	4.000000	4.000000	3.00000	3.000000
103	3.000000	4.000000	3.000000	1.00000	2.178862
104	4.000000	4.000000	1.000000	4.00000	1.000000
105	3.000000	3.366412	3.000000	2.20155	2.178862
106	1.000000	2.000000	1.976562	1.00000	1.000000
107	4.000000	3.000000	4.000000	3.00000	2.178862
108	4.000000	3.366412	1.000000	1.00000	2.178862
109	3.000000	3.000000	1.000000	3.00000	4.000000
110	2.000000	3.366412	3.000000	2.00000	3.000000
111	1.000000	5.000000	1.000000	1.00000	1.000000
112	4.000000	3.366412	2.000000	2.00000	2.178862
113	4.000000	5.000000	2.000000	1.00000	4.000000
114	4.000000	5.000000	1.976562	3.00000	2.178862
115	3.128788	5.000000	2.000000	2.20155	2.178862
116	3.000000	3.000000	3.000000	3.00000	2.000000
117	1.000000	5.000000	1.000000	2.20155	1.000000
118	3.128788	4.000000	1.000000	3.00000	2.000000
119	3.000000	4.000000	5.000000	5.00000	5.000000
120	5.000000	5.000000	1.000000	5.00000	4.000000
121	2.000000	4.000000	3.000000	1.00000	5.000000
122	1.000000	1.000000	1.000000	1.00000	4.000000
123	5.000000	4.000000	3.000000	3.00000	2.000000
124	3.000000	3.000000	3.000000	2.00000	3.000000
125	2.000000	3.000000	2.000000	2.20155	4.000000
126	5.000000	5.000000	1.000000	5.00000	2.178862
127	4.000000	4.000000	1.000000	2.20155	1.000000
128	2.000000	4.000000	3.000000	3.00000	2.000000
129	5.000000	3.000000	1.000000	1.00000	2.000000
130	4.000000	3.000000	2.000000	2.00000	2.000000
131	3.000000	4.000000	1.976562	3.00000	2.178862
132	2.000000	2.000000	1.976562	2.20155	2.000000
133	3.000000	3.000000	3.000000	3.00000	1.000000
134	5.000000	5.000000	1.976562	2.20155	4.000000
135	4.000000	5.000000	2.000000	4.00000	1.000000
136	3.000000	3.366412	1.000000	1.00000	1.000000
137	2.000000	3.000000	5.000000	3.00000	5.000000

```
138  5.000000  3.366412  1.976562  1.00000  2.000000
139  3.128788  1.000000  2.000000  1.00000  1.000000
140  3.000000  5.000000  1.976562  3.00000  1.000000
141  4.000000  3.000000  4.000000  4.00000  4.000000
142  3.128788  3.000000  1.000000  3.00000  1.000000
143  1.000000  1.000000  1.000000  1.00000  1.000000
144  4.000000  4.000000  4.000000  3.00000  2.000000
145  2.000000  4.000000  2.000000  1.00000  2.178862
146  4.000000  2.000000  3.000000  4.00000  1.000000
147  3.000000  3.000000  3.000000  3.00000  3.000000
148  3.000000  3.000000  1.976562  3.00000  4.000000
```

Compare results to original values

```
original_df = pd.read_csv('data.csv')
mdh.compare_statistics(imputed_df=mean_sub_df,
original_df=original_df)
```

```
Original:
              mean       std
v1_miss  3.128788  1.213193
v2_miss  3.366412  1.144929
v3_miss  1.976562  1.090148
v4_miss  2.201550  1.134542
v5_miss  2.178862  1.293315
New:
              mean       std
v1_miss  3.128788  1.141391
v2_miss  3.366412  1.073049
v3_miss  1.976562  1.009849
v4_miss  2.201550  1.055102
v5_miss  2.178862  1.174231
```

Perform missing data imputation for column `v3_miss` using single regression

```
simple_imput_df = mdh.simple_regression()
# print all rows for only v3_miss
with pd.option_context('display.max_rows', None,
'display.max_columns', None):
    print(simple_imput_df['v3_miss'])
```

```
     v1_miss  v2_miss  v3_miss  v4_miss  v5_miss
0        4.0      3.0      1.0      3.0      1.0
1        2.0      4.0      1.0      NaN      3.0
2        3.0      3.0      3.0      3.0      NaN
3        2.0      4.0      1.0      2.0      2.0
4        2.0      4.0      2.0      5.0      5.0
..       ...      ...      ...      ...      ...
143      1.0      1.0      1.0      1.0      1.0
144      4.0      4.0      4.0      3.0      2.0
145      2.0      4.0      2.0      1.0      NaN
146      4.0      2.0      3.0      4.0      1.0
147      3.0      3.0      3.0      3.0      3.0
```

```
[128 rows x 5 columns]
0      1.0
1      1.0
2      3.0
3      1.0
4      2.0
5      1.0
6      3.0
7      3.0
8      2.0
9      1.0
10     1.0
11     1.0
12     3.0
13     1.0
14     1.0
16     4.0
17     2.0
18     2.0
19     1.0
20     1.0
21     2.0
22     1.0
23     1.0
24     5.0
25     1.0
26     3.0
27     1.0
28     4.0
29     3.0
30     1.0
33     1.0
34     1.0
35     3.0
36     1.0
37     1.0
39     2.0
40     1.0
41     2.0
42     1.0
43     4.0
44     3.0
45     3.0
46     3.0
47     2.0
48     1.0
49     1.0
50     1.0
51     3.0
```

52	2.0
53	2.0
56	1.0
57	3.0
58	1.0
60	3.0
61	1.0
62	1.0
63	1.0
64	3.0
65	1.0
66	2.0
67	3.0
68	2.0
69	1.0
70	3.0
71	4.0
72	3.0
73	3.0
74	1.0
75	1.0
76	1.0
77	2.0
78	3.0
81	1.0
82	1.0
83	1.0
85	1.0
86	1.0
88	2.0
90	1.0
91	1.0
93	1.0
94	2.0
95	3.0
96	2.0
97	2.0
98	3.0
99	1.0
100	1.0
101	1.0
102	4.0
103	3.0
104	1.0
105	3.0
107	4.0
108	1.0
109	1.0
110	3.0
111	1.0

```
112    2.0
113    2.0
115    2.0
116    3.0
117    1.0
118    1.0
119    5.0
120    1.0
121    3.0
122    1.0
123    3.0
124    3.0
125    2.0
126    1.0
127    1.0
128    3.0
129    1.0
130    2.0
133    3.0
135    2.0
136    1.0
137    5.0
139    2.0
141    4.0
142    1.0
143    1.0
144    4.0
145    2.0
146    3.0
147    3.0
Name: v3_miss, dtype: float64
```

Compare results to the original values

```
mdh.compare_statistics(imputed_df=simple_imput_df,
original_df=original_df)
```

```
Original:
              mean       std
v1_miss  3.128788  1.213193
v2_miss  3.366412  1.144929
v3_miss  1.976562  1.090148
v4_miss  2.201550  1.134542
v5_miss  2.178862  1.293315
New:
              mean       std
v1_miss  3.077586  1.209749
v2_miss  3.379310  1.124086
v3_miss  1.976562  1.090148
v4_miss  2.210526  1.163407
v5_miss  2.109091  1.258698
```

Perform missing data imputationwith multivariate regression

```
multi_input_df = mdh.multiple_imputation()
# print all rows
with pd.option_context('display.max_rows', None,
'display.max_columns', None):
    print(multi_imput_df)

     v1_miss  v2_miss   v3_miss   v4_miss   v5_miss
0        4.0      3.0  1.000000  3.000000  1.000000
1        2.0      4.0  1.000000  1.947759  3.000000
2        3.0      3.0  3.000000  3.000000  2.703470
3        2.0      4.0  1.000000  2.000000  2.000000
4        2.0      4.0  2.000000  5.000000  5.000000
..       ...      ...       ...       ...       ...
144      4.0      4.0  4.000000  3.000000  2.000000
145      2.0      4.0  2.000000  1.000000  1.964089
146      4.0      2.0  3.000000  4.000000  1.000000
147      3.0      3.0  3.000000  3.000000  3.000000
148      3.0      3.0  2.728263  3.000000  4.000000

[149 rows x 5 columns]
      v1_miss   v2_miss   v3_miss   v4_miss   v5_miss
0    4.000000  3.000000  1.000000  3.000000  1.000000
1    2.000000  4.000000  1.000000  1.947759  3.000000
2    3.000000  3.000000  3.000000  3.000000  2.703470
3    2.000000  4.000000  1.000000  2.000000  2.000000
4    2.000000  4.000000  2.000000  5.000000  5.000000
5    4.000000  3.355189  1.000000  2.000000  1.000000
6    2.000000  4.000000  3.000000  1.000000  3.000000
7    2.000000  5.000000  3.000000  3.000000  1.000000
8    1.000000  5.000000  2.000000  3.000000  2.000000
9    3.000000  3.000000  1.000000  1.000000  3.000000
10   4.000000  1.000000  1.000000  1.000000  1.000000
11   5.000000  4.000000  1.000000  2.000000  1.000000
12   2.000000  4.000000  3.000000  4.000000  1.000000
13   4.000000  1.000000  1.000000  1.000000  1.000000
14   2.000000  3.000000  1.000000  1.000000  1.000000
15   3.127124  3.365838  1.993718  2.200919  2.195495
16   4.000000  4.000000  4.000000  5.000000  4.000000
17   3.000000  3.000000  2.000000  1.000000  1.000000
18   3.000000  2.000000  2.000000  3.000000  5.000000
19   1.000000  3.000000  1.000000  1.000000  1.000000
20   3.128590  4.000000  1.000000  3.000000  1.000000
21   3.000000  4.000000  2.000000  2.404680  4.000000
22   4.000000  3.000000  1.000000  3.000000  3.000000
23   5.000000  4.000000  1.000000  3.000000  2.000000
24   4.000000  4.000000  5.000000  4.000000  3.692981
25   4.000000  5.000000  1.000000  1.000000  1.000000
26   1.000000  2.000000  3.000000  2.000000  3.000000
27   3.000000  3.000000  1.000000  3.000000  1.000000
```

28	3.000000	3.497654	4.000000	3.000000	3.000000
29	1.000000	5.000000	3.000000	1.000000	1.000000
30	3.000000	3.000000	1.000000	1.000000	1.665844
31	3.000000	3.342463	1.881716	2.000000	2.000000
32	3.127124	3.365838	1.993718	2.200919	2.195495
33	2.000000	1.000000	1.000000	1.000000	1.000000
34	3.098503	3.382272	1.000000	3.000000	1.000000
35	4.000000	5.000000	3.000000	3.000000	2.000000
36	3.016429	3.297023	1.000000	1.807514	1.000000
37	4.000000	4.000000	1.000000	1.000000	1.000000
38	3.211778	3.000000	2.420431	3.000000	3.000000
39	3.000000	2.000000	2.000000	3.000000	4.000000
40	5.000000	2.000000	1.000000	1.000000	1.000000
41	1.000000	2.000000	2.000000	2.000000	2.055307
42	3.000000	3.000000	1.000000	1.000000	1.665844
43	4.000000	4.000000	4.000000	1.000000	2.000000
44	4.000000	4.000000	3.000000	2.000000	2.000000
45	5.000000	5.000000	3.000000	2.000000	3.000000
46	4.000000	4.000000	3.000000	1.000000	3.000000
47	5.000000	2.000000	2.000000	1.000000	1.000000
48	2.988084	2.000000	1.000000	1.721940	1.773788
49	3.000000	4.000000	1.000000	1.000000	1.000000
50	2.887081	1.000000	1.000000	1.539406	1.000000
51	3.000000	5.000000	3.000000	2.559285	1.000000
52	2.000000	2.000000	2.000000	2.000000	1.000000
53	4.000000	2.000000	2.000000	4.000000	1.000000
54	3.127124	3.365838	1.993718	2.200919	2.195495
55	3.127124	3.365838	1.993718	2.200919	2.195495
56	4.000000	3.281509	1.000000	1.000000	2.000000
57	3.239658	2.000000	3.000000	4.000000	3.000000
58	2.000000	3.000000	1.000000	1.000000	1.000000
59	4.000000	4.000000	2.311668	3.000000	2.452279
60	3.271685	4.000000	3.000000	3.000000	3.000000
61	2.000000	3.000000	1.000000	2.000000	5.000000
62	1.000000	1.000000	1.000000	3.000000	1.000000
63	5.000000	2.000000	1.000000	1.000000	1.881401
64	3.000000	3.000000	3.000000	3.000000	3.000000
65	4.000000	4.000000	1.000000	1.000000	1.708967
66	4.000000	4.000000	2.000000	2.000000	2.000000
67	4.000000	3.000000	3.000000	2.000000	2.000000
68	5.000000	5.000000	2.000000	3.000000	3.000000
69	5.000000	3.469282	1.000000	3.000000	1.000000
70	3.000000	3.000000	3.000000	2.495671	3.000000
71	3.242356	3.000000	4.000000	3.000000	3.000000
72	2.000000	4.000000	3.000000	1.000000	3.000000
73	3.000000	3.000000	3.000000	2.468229	2.654952
74	3.000000	4.000000	1.000000	1.000000	4.000000
75	4.000000	5.000000	1.000000	1.000000	3.000000
76	1.000000	3.000000	1.000000	1.000000	1.000000
77	4.000000	4.000000	2.000000	2.500203	4.000000

78	2.000000	4.000000	3.000000	2.000000	1.000000
79	5.000000	1.000000	2.392139	1.000000	5.000000
80	4.000000	4.000000	2.278280	1.000000	4.000000
81	4.000000	4.000000	1.000000	1.000000	1.000000
82	3.000000	3.000000	1.000000	3.000000	2.000000
83	2.000000	5.000000	1.000000	1.000000	1.000000
84	4.000000	3.000000	1.803179	3.000000	1.000000
85	4.000000	3.000000	1.000000	1.000000	2.000000
86	1.000000	4.000000	1.000000	1.000000	1.000000
87	2.000000	4.000000	1.927426	2.000000	2.000000
88	4.000000	4.000000	2.000000	3.000000	4.000000
89	3.000000	3.000000	1.904317	2.000000	2.143745
90	3.000000	5.000000	1.000000	1.000000	1.000000
91	2.000000	4.000000	1.000000	1.000000	1.000000
92	5.000000	1.000000	2.392139	1.000000	5.000000
93	1.000000	4.000000	1.000000	2.000000	1.000000
94	3.000000	2.000000	2.000000	3.000000	1.000000
95	3.000000	3.000000	3.000000	3.000000	3.000000
96	3.000000	4.000000	2.000000	2.000000	2.000000
97	4.000000	2.000000	2.000000	2.000000	1.000000
98	3.000000	3.387955	3.000000	2.000000	2.595510
99	1.000000	3.000000	1.000000	3.000000	1.000000
100	1.000000	1.000000	1.000000	1.000000	2.000000
101	2.000000	3.000000	1.000000	3.000000	1.000000
102	5.000000	4.000000	4.000000	3.000000	3.000000
103	3.000000	4.000000	3.000000	1.000000	2.477891
104	4.000000	4.000000	1.000000	4.000000	1.000000
105	3.000000	3.422867	3.000000	2.514195	2.640919
106	1.000000	2.000000	1.244799	1.000000	1.000000
107	4.000000	3.000000	4.000000	3.000000	3.217272
108	4.000000	3.282900	1.000000	1.000000	1.739877
109	3.000000	3.000000	1.000000	3.000000	4.000000
110	2.000000	3.340034	3.000000	2.000000	3.000000
111	1.000000	5.000000	1.000000	1.000000	1.000000
112	4.000000	3.392012	2.000000	2.000000	2.253986
113	4.000000	5.000000	2.000000	1.000000	4.000000
114	4.000000	5.000000	2.368870	3.000000	2.433634
115	3.216925	5.000000	2.000000	2.389823	2.152721
116	3.000000	3.000000	3.000000	3.000000	2.000000
117	1.000000	5.000000	1.000000	1.804509	1.000000
118	3.180779	4.000000	1.000000	3.000000	2.000000
119	3.000000	4.000000	5.000000	5.000000	5.000000
120	5.000000	5.000000	1.000000	5.000000	4.000000
121	2.000000	4.000000	3.000000	1.000000	5.000000
122	1.000000	1.000000	1.000000	1.000000	4.000000
123	5.000000	4.000000	3.000000	3.000000	2.000000
124	3.000000	3.000000	3.000000	2.000000	3.000000
125	2.000000	3.000000	2.000000	2.197817	4.000000
126	5.000000	5.000000	1.000000	5.000000	2.117040
127	4.000000	4.000000	1.000000	1.979738	1.000000

```
128  2.000000  4.000000  3.000000  3.000000  2.000000
129  5.000000  3.000000  1.000000  1.000000  2.000000
130  4.000000  3.000000  2.000000  2.000000  2.000000
131  3.000000  4.000000  2.286274  3.000000  2.355195
132  2.000000  2.000000  1.766435  1.861598  2.000000
133  3.000000  3.000000  3.000000  3.000000  1.000000
134  5.000000  5.000000  2.830334  2.940898  4.000000
135  4.000000  5.000000  2.000000  4.000000  1.000000
136  3.000000  3.241097  1.000000  1.000000  1.000000
137  2.000000  3.000000  5.000000  3.000000  5.000000
138  5.000000  3.354230  1.619488  1.000000  2.000000
139  2.871191  1.000000  2.000000  1.000000  1.000000
140  3.000000  5.000000  1.933331  3.000000  1.000000
141  4.000000  3.000000  4.000000  4.000000  4.000000
142  3.079903  3.000000  1.000000  3.000000  1.000000
143  1.000000  1.000000  1.000000  1.000000  1.000000
144  4.000000  4.000000  4.000000  3.000000  2.000000
145  2.000000  4.000000  2.000000  1.000000  1.964089
146  4.000000  2.000000  3.000000  4.000000  1.000000
147  3.000000  3.000000  3.000000  3.000000  3.000000
148  3.000000  3.000000  2.728263  3.000000  4.000000
```

Compare results to the original values

```
### compare stats
mdh.compare_statistics(imputed_df=multi_imput_df,
original_df=original_df)
```

```
Original:
              mean       std
v1_miss  3.128788  1.213193
v2_miss  3.366412  1.144929
v3_miss  1.976562  1.090148
v4_miss  2.201550  1.134542
v5_miss  2.178862  1.293315
New:
              mean       std
v1_miss  3.127124  1.142051
v2_miss  3.365838  1.073265
v3_miss  1.993718  1.019845
v4_miss  2.200919  1.062462
v5_miss  2.195495  1.190645
```

Correlation matrices

Original data

```
mdh.correlation_matrix(mdh.df)
```

```
                                                 v1_miss  \
v1_miss     1.0 (p-value: 0.0 | missing: missing: 17)
v2_miss  0.13 (p-value: 0.169 | missing: missing: 29)
v3_miss  0.07 (p-value: 0.441 | missing: missing: 33)
```

```
v4_miss  0.16 (p-value: 0.074 | missing: missing: 29)
v5_miss   0.14 (p-value: 0.15 | missing: missing: 37)

                                              v2_miss  \
v1_miss  0.13 (p-value: 0.169 | missing: missing: 29)
v2_miss     1.0 (p-value: 0.0 | missing: missing: 18)
v3_miss  0.14 (p-value: 0.141 | missing: missing: 33)
v4_miss  0.18 (p-value: 0.058 | missing: missing: 32)
v5_miss  0.02 (p-value: 0.838 | missing: missing: 36)

                                              v3_miss  \
v1_miss  0.07 (p-value: 0.441 | missing: missing: 33)
v2_miss  0.14 (p-value: 0.141 | missing: missing: 33)
v3_miss     1.0 (p-value: 0.0 | missing: missing: 21)
v4_miss    0.39 (p-value: 0.0 | missing: missing: 35)
v5_miss    0.43 (p-value: 0.0 | missing: missing: 39)

                                              v4_miss  \
v1_miss  0.16 (p-value: 0.074 | missing: missing: 29)
v2_miss  0.18 (p-value: 0.058 | missing: missing: 32)
v3_miss    0.39 (p-value: 0.0 | missing: missing: 35)
v4_miss     1.0 (p-value: 0.0 | missing: missing: 20)
v5_miss  0.22 (p-value: 0.023 | missing: missing: 38)

                                              v5_miss
v1_miss   0.14 (p-value: 0.15 | missing: missing: 37)
v2_miss  0.02 (p-value: 0.838 | missing: missing: 36)
v3_miss    0.43 (p-value: 0.0 | missing: missing: 39)
v4_miss  0.22 (p-value: 0.023 | missing: missing: 38)
v5_miss     1.0 (p-value: 0.0 | missing: missing: 26)
```

Mean imputed

```
mdh.correlation_matrix(mdh.mean_imputed_df)

                                             v1_miss  \
v1_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v2_miss  0.11 (p-value: 0.166 | missing: missing: 0)
v3_miss   0.06 (p-value: 0.44 | missing: missing: 0)
v4_miss  0.15 (p-value: 0.063 | missing: missing: 0)
v5_miss  0.13 (p-value: 0.127 | missing: missing: 0)

                                             v2_miss  \
v1_miss  0.11 (p-value: 0.166 | missing: missing: 0)
v2_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v3_miss  0.12 (p-value: 0.142 | missing: missing: 0)
v4_miss  0.16 (p-value: 0.052 | missing: missing: 0)
v5_miss  0.02 (p-value: 0.832 | missing: missing: 0)

                                             v3_miss  \
v1_miss   0.06 (p-value: 0.44 | missing: missing: 0)
```

```
v2_miss  0.12 (p-value: 0.142 | missing: missing: 0)
v3_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v4_miss    0.36 (p-value: 0.0 | missing: missing: 0)
v5_miss    0.36 (p-value: 0.0 | missing: missing: 0)

                                             v4_miss  \
v1_miss  0.15 (p-value: 0.063 | missing: missing: 0)
v2_miss  0.16 (p-value: 0.052 | missing: missing: 0)
v3_miss    0.36 (p-value: 0.0 | missing: missing: 0)
v4_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v5_miss  0.19 (p-value: 0.022 | missing: missing: 0)

                                             v5_miss
v1_miss  0.13 (p-value: 0.127 | missing: missing: 0)
v2_miss  0.02 (p-value: 0.832 | missing: missing: 0)
v3_miss    0.36 (p-value: 0.0 | missing: missing: 0)
v4_miss  0.19 (p-value: 0.022 | missing: missing: 0)
v5_miss     1.0 (p-value: 0.0 | missing: missing: 0)
```

Multivariate regression imputed

```
mdh.correlation_matrix(mdh.multi_imputed_df)
```

```
                                             v1_miss  \
v1_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v2_miss  0.12 (p-value: 0.137 | missing: missing: 0)
v3_miss   0.1 (p-value: 0.235 | missing: missing: 0)
v4_miss  0.17 (p-value: 0.035 | missing: missing: 0)
v5_miss  0.14 (p-value: 0.094 | missing: missing: 0)

                                             v2_miss  \
v1_miss  0.12 (p-value: 0.137 | missing: missing: 0)
v2_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v3_miss  0.13 (p-value: 0.107 | missing: missing: 0)
v4_miss  0.19 (p-value: 0.024 | missing: missing: 0)
v5_miss   0.03 (p-value: 0.72 | missing: missing: 0)

                                             v3_miss  \
v1_miss   0.1 (p-value: 0.235 | missing: missing: 0)
v2_miss  0.13 (p-value: 0.107 | missing: missing: 0)
v3_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v4_miss    0.39 (p-value: 0.0 | missing: missing: 0)
v5_miss    0.46 (p-value: 0.0 | missing: missing: 0)

                                             v4_miss  \
v1_miss  0.17 (p-value: 0.035 | missing: missing: 0)
v2_miss  0.19 (p-value: 0.024 | missing: missing: 0)
v3_miss    0.39 (p-value: 0.0 | missing: missing: 0)
v4_miss     1.0 (p-value: 0.0 | missing: missing: 0)
v5_miss  0.24 (p-value: 0.003 | missing: missing: 0)
```

	v5_miss
v1_miss	0.14 (p-value: 0.094 | missing: missing: 0)
v2_miss	0.03 (p-value: 0.72 | missing: missing: 0)
v3_miss	0.46 (p-value: 0.0 | missing: missing: 0)
v4_miss	0.24 (p-value: 0.003 | missing: missing: 0)
v5_miss	1.0 (p-value: 0.0 | missing: missing: 0)