```
!pip install feature-engine
```

```
Collecting feature-engine
  Downloading feature_engine-1.1.2-py2.py3-none-any.whl (180 kB)
ent already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.7/dist-
packages (from feature-engine) (1.1.5)
Requirement already satisfied: scikit-learn>=0.22.2 in
/usr/local/lib/python3.7/dist-packages (from feature-engine) (1.0.1)
Requirement already satisfied: scipy>=1.4.1 in
/usr/local/lib/python3.7/dist-packages (from feature-engine) (1.4.1)
Collecting statsmodels>=0.11.1
  Downloading statsmodels-0.13.1-cp37-cp37m-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (9.8 MB)
ent already satisfied: numpy>=1.18.2 in /usr/local/lib/python3.7/dist-
packages (from feature-engine) (1.19.5)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=1.0.3->feature-
engine) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in
/usr/local/lib/python3.7/dist-packages (from pandas>=1.0.3->feature-
engine) (2018.9)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3-
>pandas>=1.0.3->feature-engine) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.22.2-
>feature-engine) (3.0.0)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.22.2-
>feature-engine) (1.1.0)
Requirement already satisfied: patsy>=0.5.2 in
/usr/local/lib/python3.7/dist-packages (from statsmodels>=0.11.1-
>feature-engine) (0.5.2)
Installing collected packages: statsmodels, feature-engine
  Attempting uninstall: statsmodels
    Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
Successfully installed feature-engine-1.1.2 statsmodels-0.13.1
```

```
# Latihan 1
# import library pandas

import pandas as pd

# Import library scipy

import scipy

# Import library winsorize dari scipy
```

```
from scipy.stats.mstats import winsorize

# Import library trima dari scipy

from scipy.stats.mstats import trima

# Import library RandomSampleImputer dari feature engine imputation

from feature_engine.imputation import RandomSampleImputer

# import library StandardScaler dari sklearn

from sklearn.preprocessing import StandardScaler

# Latihan 2
# load dataset Iris_Unclean

df = pd.read_csv("Iris_unclean.csv")

# tampilkan dataset

df
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0              NaN           3.5            1.4           0.2
Iris-setosa
1              4.9        2000.0            1.4           0.2
Iris-setosa
2              4.7           3.2           -1.3           0.2
Iris-setosa
3              4.6           3.1            1.5           0.2
Iris-setosa
4              5.0           3.6            1.4           0.2
Iris-setosa
..             ...           ...            ...           ...
...
145            6.7           3.0            5.2           2.3  Iris-
virginica
146            6.3           2.5            5.0           1.9  Iris-
virginica
147            6.5           3.0            5.2           2.0  Iris-
virginica
148            6.2           3.4            5.4           2.3  Iris-
virginica
149            5.9           3.0            5.1           1.8  Iris-
virginica

[150 rows x 5 columns]
```

```
# hitung jumlah nilai null pada dataset

df.isnull().sum()
```

```
SepalLengthCm    2
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
# latihan 3
# load dataset Iris_Unclean

df = pd.read_csv("Iris_unclean.csv")

# ambil 10 data teratas SepalLengthCm, kemudian tampilkan
df = df.head(10)["SepalLengthCm"]
df
```

```
0    NaN
1    4.9
2    4.7
3    4.6
4    5.0
5    5.4
6    NaN
7    5.0
8    4.4
9    4.9
Name: SepalLengthCm, dtype: float64
```

```
# mengganti missing value dengan mean(), kemudian masukkan ke variabel

df = df.fillna(df.mean())

# tampilkan 10 data teratas SepalLengthCm setelah handle missing value
dengan imputasi mean

df
```

```
0    4.8625
1    4.9000
2    4.7000
3    4.6000
4    5.0000
5    5.4000
6    4.8625
7    5.0000
8    4.4000
```

```
9    4.9000
Name: SepalLengthCm, dtype: float64
```

```
# Latihan 4
df = pd.read_csv("Iris_unclean.csv")

# ambil 10 data teratas SepalLengthCm, kemudian tampilkan

df = df.head(10)["SepalLengthCm"]
df
```

```
0    NaN
1    4.9
2    4.7
3    4.6
4    5.0
5    5.4
6    NaN
7    5.0
8    4.4
9    4.9
Name: SepalLengthCm, dtype: float64
```

```
# melakukan imputasi nilai suka-suka (Arbitrary), masukkan ke dalam
variabel

df = df.fillna(99)

df
```

```
0    99.0
1     4.9
2     4.7
3     4.6
4     5.0
5     5.4
6    99.0
7     5.0
8     4.4
9     4.9
Name: SepalLengthCm, dtype: float64
```

```
# Latihan 5
# load dataset Iris_Unclean

data = pd.read_csv("Iris_unclean.csv")

# tampilkan 10 data teratas kolom SepalLengthCm

df = data.head(10)["SepalLengthCm"]
df
```

```
0    NaN
1    4.9
2    4.7
3    4.6
4    5.0
5    5.4
6    NaN
7    5.0
8    4.4
9    4.9
Name: SepalLengthCm, dtype: float64
```

```
# Import SimpleImputer dari sklearn.impute

from sklearn.impute import SimpleImputer

# Mengatasi missing value dengan frequent category / modus

imp = SimpleImputer(strategy='most_frequent')

# Tampilkan hasil imputasi "SepalLengthCm"

imp.fit_transform(data[['SepalLengthCm']])
```

```
array([[5. ],
       [4.9],
       [4.7],
       [4.6],
       [5. ],
       [5.4],
       [5. ],
       [5. ],
       [4.4],
       [4.9],
       [5.4],
       [4.8],
       [4.8],
       [4.3],
       [5.8],
       [5.7],
       [5.4],
       [5.1],
       [5.7],
       [5.1],
       [5.4],
       [5.1],
       [4.6],
       [5.1],
       [4.8],
       [5. ],
```

[5.],
[5.2],
[5.2],
[4.7],
[4.8],
[5.4],
[5.2],
[5.5],
[4.9],
[5.],
[5.5],
[4.9],
[4.4],
[5.1],
[5.],
[4.5],
[4.4],
[5.],
[5.1],
[4.8],
[5.1],
[4.6],
[5.3],
[5.],
[7.],
[6.4],
[6.9],
[5.5],
[6.5],
[5.7],
[6.3],
[4.9],
[6.6],
[5.2],
[5.],
[5.9],
[6.],
[6.1],
[5.6],
[6.7],
[5.6],
[5.8],
[6.2],
[5.6],
[5.9],
[6.1],
[6.3],
[6.1],
[6.4],
[6.6],

[6.8],
[6.7],
[6.],
[5.7],
[5.5],
[5.5],
[5.8],
[6.],
[5.4],
[6.],
[6.7],
[6.3],
[5.6],
[5.5],
[5.5],
[6.1],
[5.8],
[5.],
[5.6],
[5.7],
[5.7],
[6.2],
[5.1],
[5.7],
[6.3],
[5.8],
[7.1],
[6.3],
[6.5],
[7.6],
[4.9],
[7.3],
[6.7],
[7.2],
[6.5],
[6.4],
[6.8],
[5.7],
[5.8],
[6.4],
[6.5],
[7.7],
[7.7],
[6.],
[6.9],
[5.6],
[7.7],
[6.3],
[6.7],
[7.2],

```
        [6.2],
        [6.1],
        [6.4],
        [7.2],
        [7.4],
        [7.9],
        [6.4],
        [6.3],
        [6.1],
        [7.7],
        [6.3],
        [6.4],
        [6. ],
        [6.9],
        [6.7],
        [6.9],
        [5.8],
        [6.8],
        [6.7],
        [6.7],
        [6.3],
        [6.5],
        [6.2],
        [5.9]])
```

```
# Latihan 6
df = pd.read_csv("Iris_unclean.csv")

# tampilkan 10 data teratas SepalLengthCm

data = df.head(10)["SepalLengthCm"]
data
```

```
0    NaN
1    4.9
2    4.7
3    4.6
4    5.0
5    5.4
6    NaN
7    5.0
8    4.4
9    4.9
Name: SepalLengthCm, dtype: float64
```

```
# Membuat imputer random sample dengan random state = 5

imputer = RandomSampleImputer(random_state = 5)

# Cocokan imputer ke data
```

```
imputer.fit(df)

# Ubah data dengan imputer masukkan ke dalam variable

test_t = imputer.transform(df)

# Tampilkan data hasil imputasi data "SepalLengthCm"

test_t
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0              5.8           3.5            1.4           0.2
Iris-setosa
1              4.9        2000.0            1.4           0.2
Iris-setosa
2              4.7           3.2           -1.3           0.2
Iris-setosa
3              4.6           3.1            1.5           0.2
Iris-setosa
4              5.0           3.6            1.4           0.2
Iris-setosa
..             ...           ...            ...           ...
...
145            6.7           3.0            5.2           2.3  Iris-
virginica
146            6.3           2.5            5.0           1.9  Iris-
virginica
147            6.5           3.0            5.2           2.0  Iris-
virginica
148            6.2           3.4            5.4           2.3  Iris-
virginica
149            5.9           3.0            5.1           1.8  Iris-
virginica

[150 rows x 5 columns]
```

```
# Latihan 7
# Import library scipy

import numpy as np
from scipy.stats.mstats import winsorize
from scipy.stats.mstats import trima

# Load data Iris_AfterClean
data = pd.read_csv("Iris_AfterClean.csv")

# Ambil 10 data teratas "SepalLengthCm", kemudian masukkan ke dalam
variabel datan tampilkan
```

```
a = data.head(10)["SepalLengthCm"]
a
```

```
0    4.6
1    5.0
2    5.4
3    4.9
4    5.4
5    4.8
6    4.8
7    4.3
8    5.8
9    5.4
Name: SepalLengthCm, dtype: float64
```

```
# Winsorize data dengan batas nilai terendah 10% dan batas nilai
tinggi 20%

wins = winsorize(a, limits=[0.1, 0.2])

# Tampilkan hasil winsorize
print(wins)
```

```
[4.6 5.  5.4 4.9 5.4 4.8 4.8 4.6 5.4 5.4]
```

```
# Latihan 8
# Import library trima dari scopy

from scipy.stats.mstats import trima

# Load data Iris_AfterClean
data = pd.read_csv("Iris_AfterClean.csv")

# Ambil 10 data teratas "SepalLengthCm", kemudian masukkan ke dalam
variabel datan tampilkan
a = data.head(10)["SepalLengthCm"]
a
```

```
0    4.6
1    5.0
2    5.4
3    4.9
4    5.4
5    4.8
6    4.8
7    4.3
8    5.8
9    5.4
Name: SepalLengthCm, dtype: float64
```

```
# Trimming data dengan batas nilai terendah 2 dan batas nilai tinggi 5

trims = trima(a, limits=(2,5))

# Tampilkan hasil trimming

print(trims)
```

```
[4.6 5.0 -- 4.9 -- 4.8 4.8 4.3 -- --]
```

```
# Latihan 9
# Load data Iris_AfterClean

data = pd.read_csv("Iris_AfterClean.csv")

# Ambil 10 data teratas SepalLengthCm dan SepalWidthCm

data = data.head(10)[["SepalLengthCm", "SepalWidthCm"]]
data
```

	SepalLengthCm	SepalWidthCm
0	4.6	3.1
1	5.0	3.6
2	5.4	3.9
3	4.9	3.1
4	5.4	3.7
5	4.8	3.4
6	4.8	3.0
7	4.3	3.0
8	5.8	4.0
9	5.4	3.9

```
# Menghitung mean
means = data.mean(axis = 0)

# menghitung max - min
max_min = data.max(axis = 0) - data.min(axis = 0)

# menerapkan transformasi ke data
train_scaled = (data - means) / max_min

# Tampilkan hasil scalling

train_scaled
```

	SepalLengthCm	SepalWidthCm
0	-0.293333	-0.37
1	-0.026667	0.13
2	0.240000	0.43
3	-0.093333	-0.37
4	0.240000	0.23

```
5      -0.160000         -0.07
6      -0.160000         -0.47
7      -0.493333         -0.47
8       0.506667          0.53
9       0.240000          0.43
```

```
# Load data Iris_AfterClean

data = pd.read_csv("Iris_AfterClean.csv")

# Ambil 10 data teratas SepalLengthCm dan SepalWidthCm

data = data.head(10)[["SepalLengthCm", "SepalWidthCm"]]
data
```

```
   SepalLengthCm  SepalWidthCm
0            4.6           3.1
1            5.0           3.6
2            5.4           3.9
3            4.9           3.1
4            5.4           3.7
5            4.8           3.4
6            4.8           3.0
7            4.3           3.0
8            5.8           4.0
9            5.4           3.9
```

```
# import library StandardScaler dari sklearn
from sklearn.preprocessing import StandardScaler

# Buat objek scaler
scaler = StandardScaler()

# Sesuaikan scaler dengan data
scaler.fit(data)

# Mengubah data
train_scaled = scaler.transform(data)

# Tampilkan hasil

train_scaled
```

```
array([[-1.02464215, -0.97469723],
       [-0.09314929,  0.34246119],
       [ 0.83834358,  1.13275625],
       [-0.3260225 , -0.97469723],
       [ 0.83834358,  0.60589288],
       [-0.55889572, -0.18440218],
       [-0.55889572, -1.23812892],
       [-1.7232618 , -1.23812892],
```

```
[ 1.76983644,  1.39618793],
[ 0.83834358,  1.13275625]])
```