Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»


# «Методы машинного обучения в автоматизированных системах обработки информации и управления»

# Лабораторная работа №3

# «Обработка признаков (часть 2)»

**ИСПОЛНИТЕЛЬ:**

Демирев Н.К.
Группа ИУ5-21М

_____

"__"_____2023 г.

Москва 2023

[ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
%matplotlib inline
sns.set(style="ticks")
```

[ ]:
```
df = pd.read_csv('dataset.csv')
df.head()
```

[ ]:

| | Rank | City | State | Land area (sq mi) | (km2) | Water area (sq mi) | \ |
|---|---|---|---|---|---|---|---|
| 0 | 1 | Sitka | Alaska | 2,870.10 | 7,434 | 1,945.10 | |
| 1 | 2 | Juneau | Alaska | 2,704.00 | 7,003 | 550.7 | |
| 2 | 3 | Wrangell | Alaska | 2,556.00 | 6,620 | 920.6 | |
| 3 | 4 | Anchorage | Alaska | 1,706.80 | 4,421 | 239.9 | |
| 4 | 5 | Tribune | Kansas | 778.2 | 2,016 | 0 | |

| | (km2).1 | Total area (sq mi) | (km2).2 | Population (2020) |
|---|---|---|---|---|
| 0 | 5,038 | 4,815.10 | 12,471 | 8,458 |
| 1 | 1,426 | 3,254.70 | 8,430 | 32,255 |
| 2 | 2,384 | 3,476.60 | 9,004 | 2,127 |
| 3 | 621 | 1,946.70 | 5,042 | 2,91,247 |
| 4 | 0 | 778.2 | 2,016 | 1,182 |

[ ]:
```
def repair_df(df : pd.DataFrame):
    df['Population (2020)'] = df['Population (2020)'].apply(lambda x: int(''.
 ↪join(x.split(','))))
    df[' (km2).2'] = df[' (km2).2'].apply(lambda x: float(''.join(x.
 ↪split(','))))
```

```
    df[' (km2).1'] = df[' (km2).1'].apply(lambda x: float(''.join(x.
↪split(','))))
    df[' (km2)'] = df[' (km2)'].apply(lambda x: int(''.join(x.split(','))))
    df['Water area (sq mi)'] = df['Water area (sq mi)'].apply(lambda x:␣
↪float(''.join(x.split(','))))
    df['Land area (sq mi)'] = df['Land area (sq mi)'].apply(lambda x: float(''.
↪join(x.split(','))))
    df['Total area (sq mi)'] = df['Total area (sq mi)'].apply(lambda x:␣
↪float(''.join(x.split(','))))
    return df
```

[ ]:
```
df = repair_df(df)
```

[ ]:
```
df.describe()
```

[ ]:
```
              Rank   Land area (sq mi)        (km2)  Water area (sq mi)  \
count  150.000000         150.000000   150.000000          150.000000
mean    75.500000         247.013333   639.780000           38.148667
std     43.445368         400.666580  1037.745432          184.266807
min      1.000000          78.500000   203.000000            0.000000
25%     38.250000         101.525000   262.750000            0.700000
50%     75.500000         134.500000   348.500000            2.650000
75%    112.750000         217.700000   564.000000            8.150000
max    150.000000        2870.100000  7434.000000         1945.100000

           (km2).1  Total area (sq mi)       (km2).2  Population (2020)
count   150.000000          150.000000    150.000000       1.500000e+02
mean     98.777467          285.158000    738.560000       4.127783e+05
std     477.247756          560.643139   1452.066648       8.550999e+05
min       0.000000           78.500000    203.000000       3.060000e+02
25%       1.800000          104.325000    270.000000       9.220750e+04
50%       6.900000          142.000000    368.000000       2.006680e+05
75%      20.750000          239.600000    620.500000       4.656162e+05
max    5038.000000         4815.100000  12471.000000       8.804190e+06
```

[ ]:
```
df.dtypes
```

[ ]:
```
Rank                   int64
City                  object
State                 object
Land area (sq mi)    float64
 (km2)                 int64
Water area (sq mi)   float64
 (km2).1             float64
Total area (sq mi)   float64
 (km2).2             float64
Population (2020)      int64
```

```
dtype: object
```

[ ]:
```
def obj_col(column):
    return column[1] == 'object'

col_names = []
for col in list(filter(obj_col, list(zip(list(df.columns), list(df.dtypes))))):
  col_names.append(col[0])
```

[ ]:
```
X_ALL = df.drop(col_names, axis=1)
X_ALL
```

[ ]:
```
     Rank   Land area (sq mi)   (km2)  Water area (sq mi)   (km2).1
0       1              2870.1    7434               1945.1    5038.0  \
1       2              2704.0    7003                550.7    1426.0
2       3              2556.0    6620                920.6    2384.0
3       4              1706.8    4421                239.9     621.0
4       5               778.2    2016                  0.0       0.0
..    …                 …     …                   …       …
145   146               79.6     206                 21.4      55.0
146   147               79.3     205                  0.8       2.1
147   148               79.3     205                 14.6      38.0
148   149               79.1     205                  0.5       1.3
149   150               78.5     203                  0.0       0.0

     Total area (sq mi)   (km2).2  Population (2020)
0                4815.1   12471.0               8458
1                3254.7    8430.0              32255
2                3476.6    9004.0               2127
3                1946.7    5042.0             291247
4                 778.2    2016.0               1182
..                  …       …                 …
145               101.0     262.0             269840
146                80.1     207.0               7396
147                93.9     243.0               8399
148                79.6     206.0             192517
149                78.5     203.0              95342

[150 rows x 8 columns]
```

[ ]:
```
#
#
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=X_ALL.columns)
    return res
```

[ ]:
```
#
X_train, X_test, y_train, y_test = train_test_split(X_ALL, df['Population␣
↪(2020)'],
                                                    test_size=0.2,
                                                    random_state=1)
#              DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

[ ]:
```
((120, 8), (30, 8))
```

---

### 0.1 “StandardScaler”

[ ]:
```
#      StandardScaler
cs11 = StandardScaler()
df_cs11_scaled_temp = cs11.fit_transform(X_ALL)
#       DataFrame
df_cs11_scaled =  arr_to_df(df_cs11_scaled_temp)
df_cs11_scaled
```

[ ]:

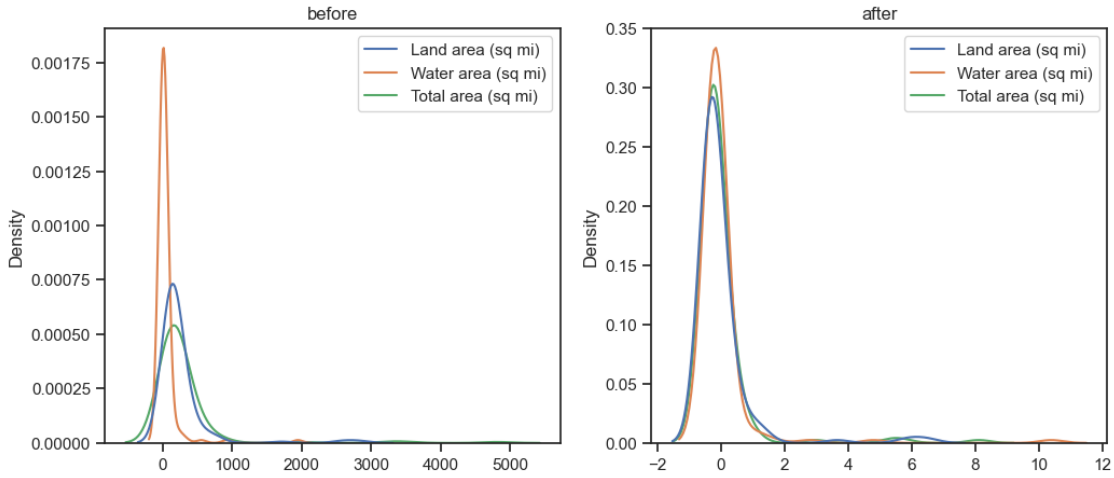
| | Rank | Land area (sq mi) | (km2) | Water area (sq mi) | (km2).1 | \ |
|---|---|---|---|---|---|---|
| 0 | -1.720542 | 6.568739 | 6.569030 | 10.383529 | 10.384060 | |
| 1 | -1.697448 | 6.152791 | 6.152316 | 2.790890 | 2.790309 | |
| 2 | -1.674353 | 5.782169 | 5.782010 | 4.805030 | 4.804377 | |
| 3 | -1.651258 | 3.655601 | 3.655894 | 1.098555 | 1.097904 | |
| 4 | -1.628164 | 1.330199 | 1.330606 | -0.207723 | -0.207667 | |
| .. | … | … | … | … | … | |
| 145 | 1.628164 | -0.419237 | -0.419403 | -0.091198 | -0.092036 | |
| 146 | 1.651258 | -0.419988 | -0.420370 | -0.203367 | -0.203252 | |
| 147 | 1.674353 | -0.419988 | -0.420370 | -0.128225 | -0.127777 | |
| 148 | 1.697448 | -0.420489 | -0.420370 | -0.205001 | -0.204933 | |
| 149 | 1.720542 | -0.421991 | -0.422303 | -0.207723 | -0.207667 | |

| | Total area (sq mi) | (km2).2 | Population (2020) |
|---|---|---|---|
| 0 | 8.106971 | 8.106890 | -0.474418 |
| 1 | 5.314415 | 5.314637 | -0.446495 |
| 2 | 5.711536 | 5.711260 | -0.481847 |
| 3 | 2.973564 | 2.973594 | -0.142601 |
| 4 | 0.882368 | 0.882686 | -0.482956 |
| .. | … | … | … |
| 145 | -0.329577 | -0.329294 | -0.167720 |
| 146 | -0.366980 | -0.367298 | -0.475664 |
| 147 | -0.342283 | -0.342422 | -0.474487 |
| 148 | -0.367875 | -0.367989 | -0.258448 |

```
149           -0.369844 -0.370062          -0.372471

[150 rows x 8 columns]
```

[ ]:
```
#
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    #
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    #
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

[ ]:
```
draw_kde(['Land area (sq mi)', 'Water area (sq mi)', 'Total area (sq mi)'], df,␣
 ↪df_cs11_scaled, 'before', 'after')
```




## 0.2 "Mean Normalisation"

[ ]:
```
#
X_train, X_test, y_train, y_test = train_test_split(X_ALL, df['Population␣
 ↪(2020)'],
                                                    test_size=0.2,
                                                    random_state=1)
#               DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)
```

```
X_train_df.shape, X_test_df.shape
```

[ ]:
```
((120, 8), (30, 8))
```

[ ]:
```
class MeanNormalisation:

    def fit(self, param_df):
        self.means = X_train.mean(axis=0)
        maxs = X_train.max(axis=0)
        mins = X_train.min(axis=0)
        self.ranges = maxs - mins

    def transform(self, param_df):
        param_df_scaled = (param_df - self.means) / self.ranges
        return param_df_scaled

    def fit_transform(self, param_df):
        self.fit(param_df)
        return self.transform(param_df)
```

[ ]:
```
sc21 = MeanNormalisation()
df_cs21_scaled = sc21.fit_transform(X_ALL)
df_cs21_scaled.describe()
```

[ ]:
```
              Rank   Land area (sq mi)       (km2)  Water area (sq mi)  \
count  150.000000         150.000000  150.000000          150.000000   \
mean    -0.007215          -0.004949   -0.004947           -0.002780
std      0.291580           0.143526    0.143513            0.094734
min     -0.507215          -0.065314   -0.065351           -0.022393
25%     -0.257215          -0.057066   -0.057088           -0.022033
50%     -0.007215          -0.045253   -0.045229           -0.021030
75%      0.242785          -0.015450   -0.015427           -0.018203
max      0.492785           0.934686    0.934649            0.977607

          (km2).1  Total area (sq mi)     (km2).2  Population (2020)
count  150.000000          150.000000  150.000000          150.000000
mean    -0.002780           -0.004059   -0.004060           -0.000337
std      0.094730            0.118364    0.118362            0.097128
min     -0.022387           -0.047689   -0.047715           -0.047189
25%     -0.022029           -0.042237   -0.042254           -0.036750
50%     -0.021017           -0.034283   -0.034265           -0.024430
75%     -0.018268           -0.013677   -0.013683            0.005664
max      0.977613            0.952311    0.952285            0.952811
```

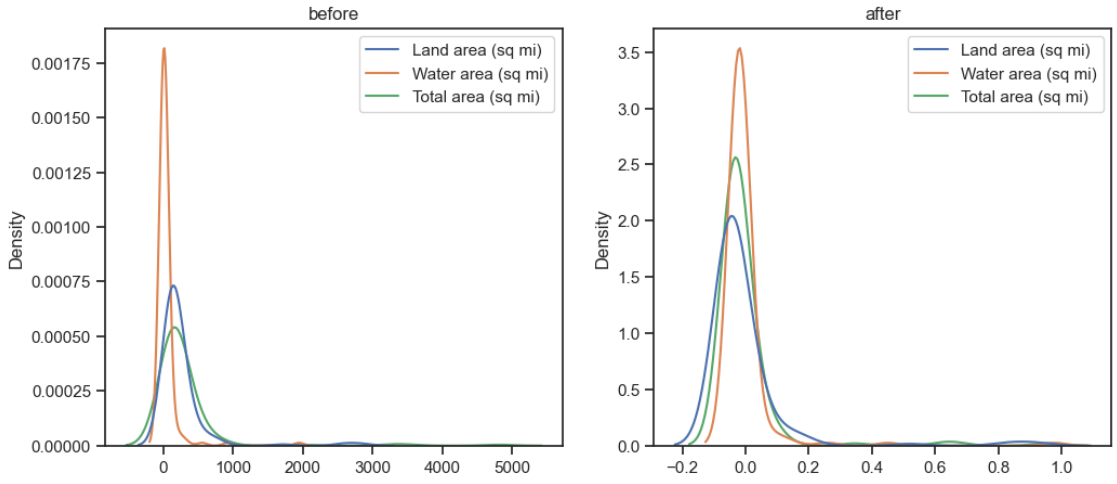
[ ]:
```
cs22 = MeanNormalisation()
cs22.fit(X_train)
df_cs22_scaled_train = cs22.transform(X_train)
```

```
df_cs22_scaled_test = cs22.transform(X_test)
```

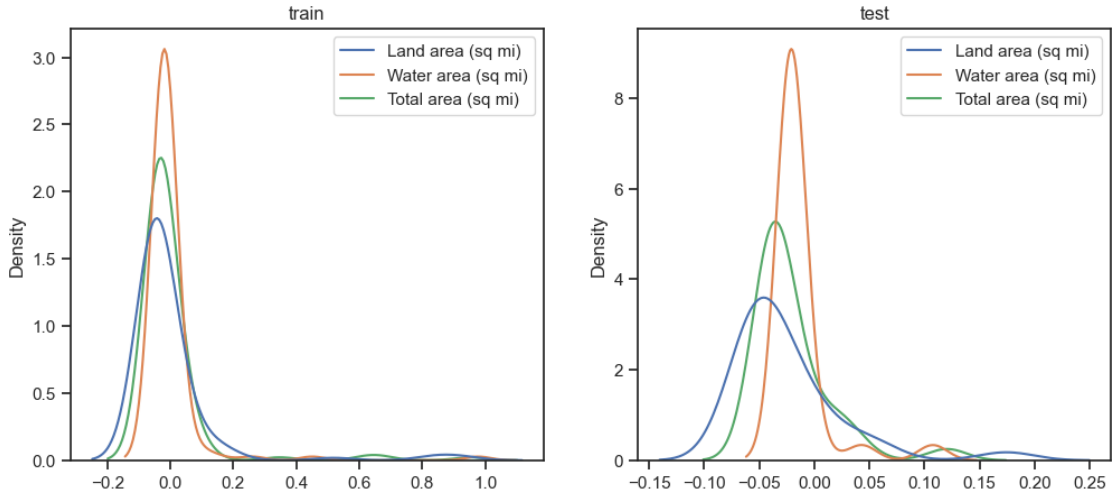
[ ]:
```
df_cs22_scaled_test.describe()
```

[ ]:
```
            Rank   Land area (sq mi)      (km2)  Water area (sq mi)    (km2).1  \
count  30.000000           30.000000  30.000000           30.000000  30.000000
mean   -0.036074           -0.024745  -0.024734           -0.013899  -0.013900
std     0.267647            0.049572   0.049559            0.026031   0.026034
min    -0.473658           -0.065027  -0.065074           -0.022341  -0.022335
25%    -0.263926           -0.053797  -0.053768           -0.022226  -0.022221
50%    -0.010570           -0.045128  -0.045160           -0.021544  -0.021543
75%     0.143792           -0.013479  -0.013456           -0.019655  -0.019707
max     0.472651            0.174262   0.174174            0.107575   0.107625

       Total area (sq mi)    (km2).2  Population (2020)
count           30.000000  30.000000          30.000000
mean            -0.020294  -0.020299          -0.001687
std              0.035161   0.035165           0.066386
min             -0.047351  -0.047389          -0.046957
25%             -0.040595  -0.040603          -0.036715
50%             -0.034219  -0.034184          -0.025034
75%             -0.016015  -0.016047           0.006627
max              0.120364   0.120365           0.264729
```

[ ]:
```
draw_kde(['Land area (sq mi)', 'Water area (sq mi)', 'Total area (sq mi)'], df,␣
↪df_cs21_scaled, 'before', 'after')
```




[ ]:
```
draw_kde(['Land area (sq mi)', 'Water area (sq mi)', 'Total area (sq mi)'],␣
↪df_cs22_scaled_train, df_cs22_scaled_test, 'train', 'test')
```




### 0.3 "MinMax"

[ ]:
```
#      StandardScaler
cs31 = MinMaxScaler()
data_cs31_scaled_temp = cs31.fit_transform(X_ALL)
#       DataFrame
df_cs31_scaled = arr_to_df(data_cs31_scaled_temp)
df_cs31_scaled.describe()
```

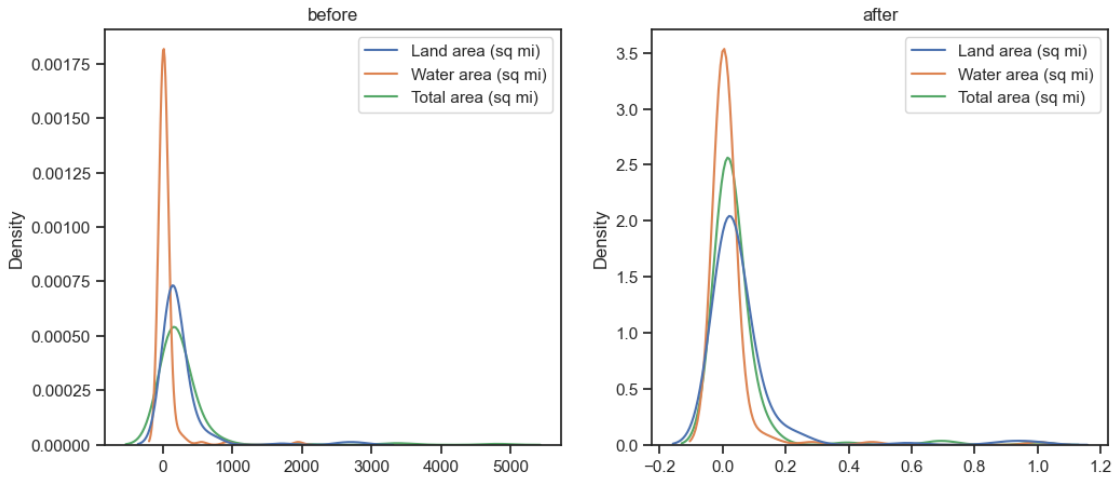
[ ]:

| | Rank | Land area (sq mi) | (km2) | Water area (sq mi) | \ |
|---|---|---|---|---|---|
| count | 150.00000 | 150.000000 | 150.000000 | 150.000000 | |
| mean | 0.50000 | 0.060364 | 0.060404 | 0.019613 | |
| std | 0.29158 | 0.143526 | 0.143513 | 0.094734 | |
| min | 0.00000 | 0.000000 | 0.000000 | 0.000000 | |
| 25% | 0.25000 | 0.008248 | 0.008263 | 0.000360 | |
| 50% | 0.50000 | 0.020060 | 0.020122 | 0.001362 | |
| 75% | 0.75000 | 0.049864 | 0.049924 | 0.004190 | |
| max | 1.00000 | 1.000000 | 1.000000 | 1.000000 | |

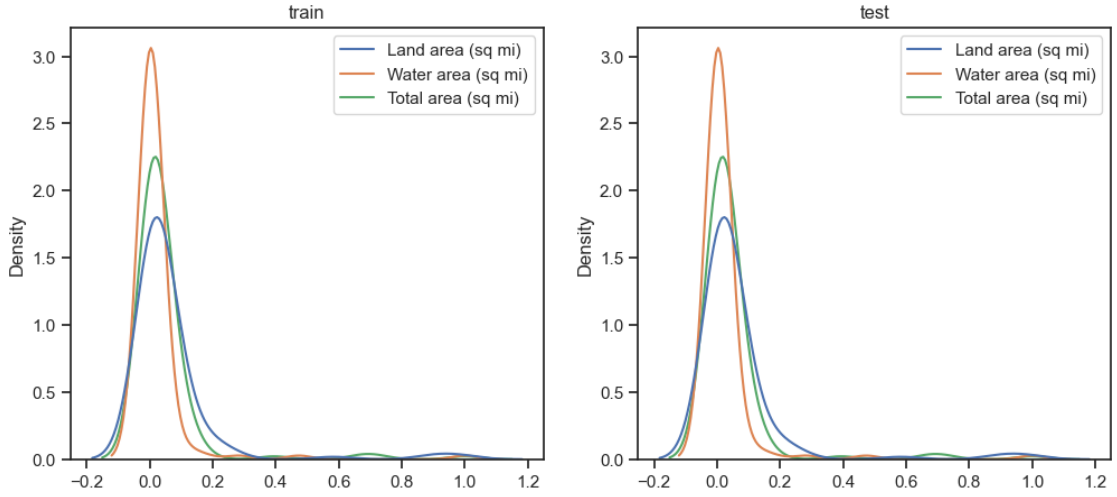
| | (km2).1 | Total area (sq mi) | (km2).2 | Population (2020) |
|---|---|---|---|---|
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 0.019606 | 0.043630 | 0.043655 | 0.046851 |
| std | 0.094730 | 0.118364 | 0.118362 | 0.097128 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000357 | 0.005452 | 0.005461 | 0.010439 |
| 50% | 0.001370 | 0.013406 | 0.013450 | 0.022758 |
| 75% | 0.004119 | 0.034012 | 0.034032 | 0.052853 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

[ ]:
```
cs32 = MinMaxScaler()
cs32.fit(X_train)
df_cs32_scaled_train_temp = cs32.transform(X_train)
df_cs32_scaled_test_temp = cs32.transform(X_test)
#       DataFrame
df_cs32_scaled_train = arr_to_df(df_cs32_scaled_train_temp)
df_cs32_scaled_test = arr_to_df(df_cs32_scaled_train_temp)
```

[ ]:
```
draw_kde(['Land area (sq mi)', 'Water area (sq mi)', 'Total area (sq mi)'], df,␣
↪df_cs31_scaled, 'before', 'after')
```




[ ]:
```
draw_kde(['Land area (sq mi)', 'Water area (sq mi)', 'Total area (sq mi)'],␣
↪df_cs32_scaled_train, df_cs32_scaled_test, 'train', 'test')
```



# 1

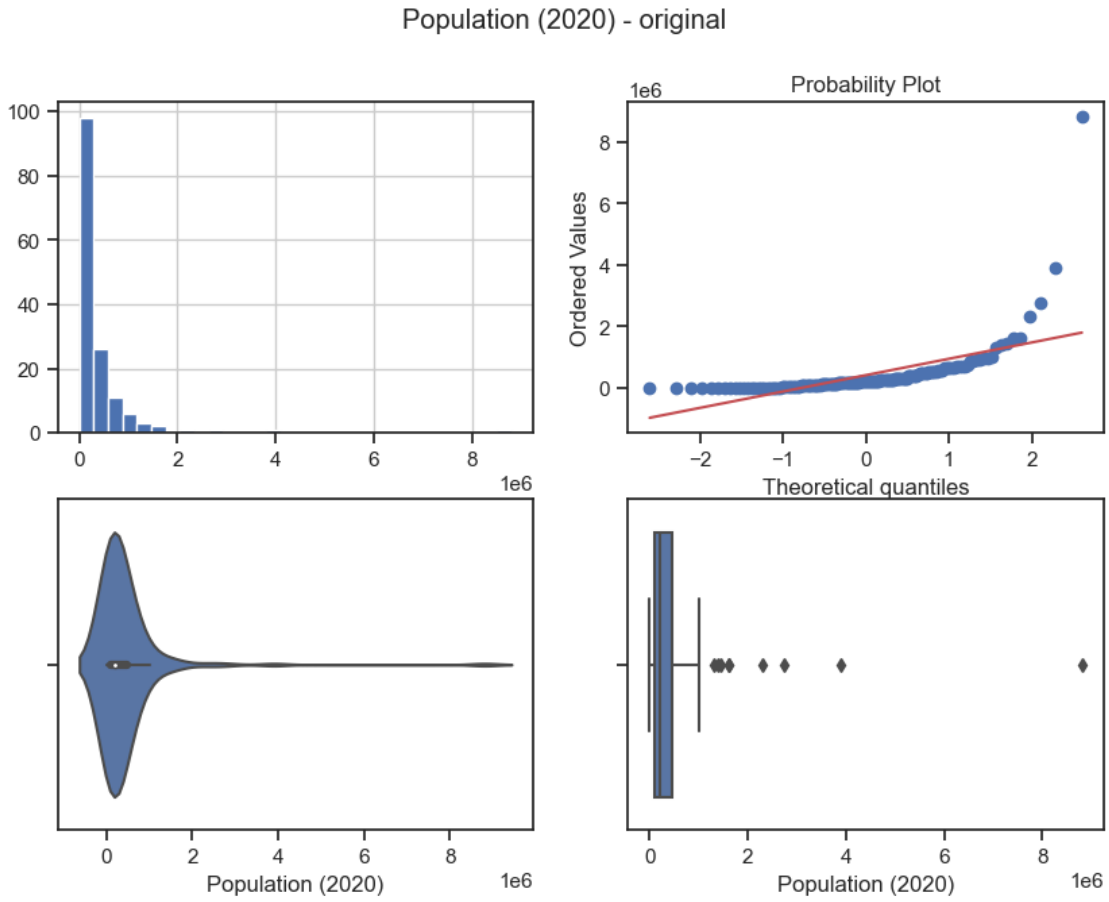
[ ]:
```
df = repair_df(pd.read_csv('dataset.csv'))
df
```

[ ]:
```
     Rank         City         State   Land area (sq mi)   (km2)
0       1        Sitka        Alaska              2870.1    7434  \
1       2       Juneau        Alaska              2704.0    7003
2       3     Wrangell        Alaska              2556.0    6620
3       4    Anchorage        Alaska              1706.8    4421
4       5      Tribune        Kansas               778.2    2016
..    ...          ...           ...                 ...     ...
145   146      Madison     Wisconsin                79.6     206
146   147      Caribou         Maine                79.3     205
147   148    Ellsworth         Maine                79.3     205
148   149  Sioux Falls  South Dakota                79.1     205
149   150   St. George          Utah                78.5     203

     Water area (sq mi)   (km2).1  Total area (sq mi)   (km2).2
0                1945.1    5038.0              4815.1   12471.0  \
1                 550.7    1426.0              3254.7    8430.0
2                 920.6    2384.0              3476.6    9004.0
3                 239.9     621.0              1946.7    5042.0
4                   0.0       0.0               778.2    2016.0
..                  ...       ...                 ...       ...
145                21.4      55.0               101.0     262.0
146                 0.8       2.1                80.1     207.0
147                14.6      38.0                93.9     243.0
148                 0.5       1.3                79.6     206.0
149                 0.0       0.0                78.5     203.0

     Population (2020)
0                 8458
1                32255
2                 2127
3               291247
4                 1182
..                 ...
145             269840
146               7396
147               8399
148             192517
149              95342

[150 rows x 10 columns]
```

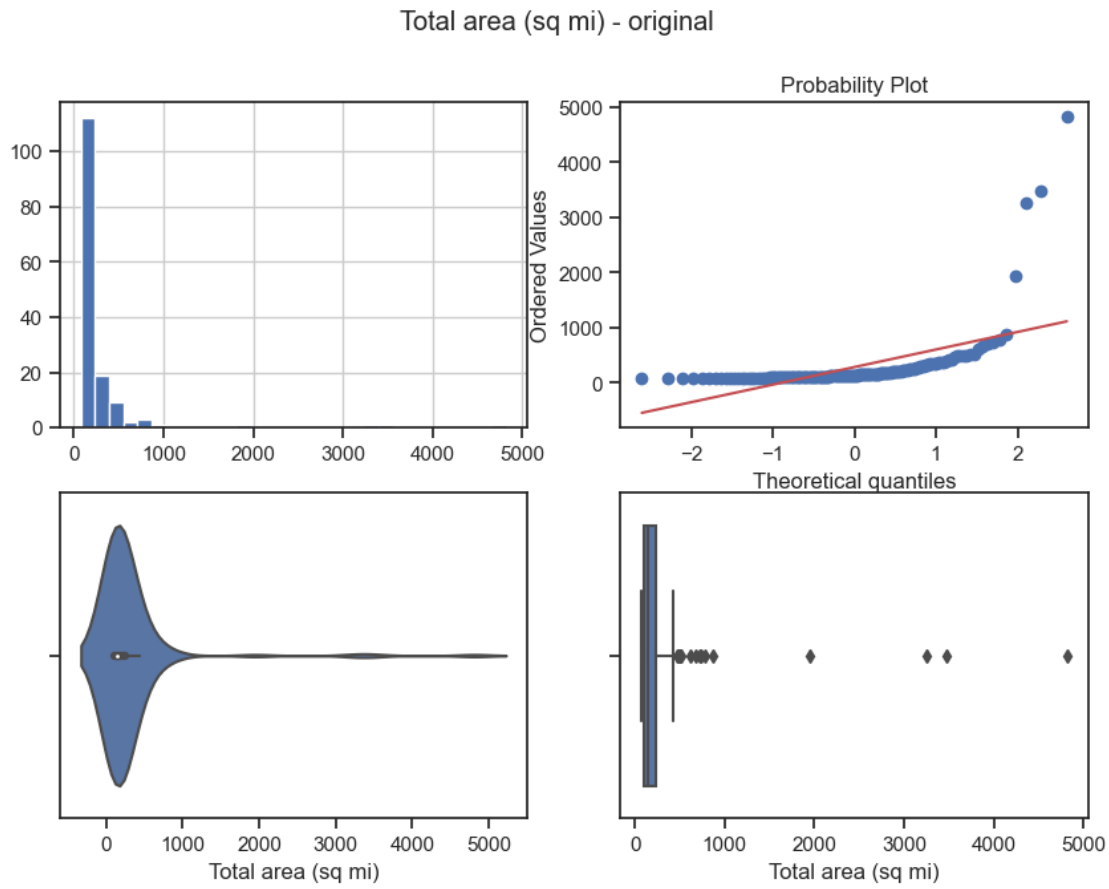
[ ]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    #
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # violinplot
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # boxplot
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

[ ]:
```
diagnostic_plots(df, 'Population (2020)', 'Population (2020) - original')
diagnostic_plots(df, 'Total area (sq mi)', 'Total area (sq mi) - original')
diagnostic_plots(df, 'Water area (sq mi)', 'Water area (sq mi) - original')
diagnostic_plots(df, 'Land area (sq mi)', 'Land area (sq mi) - original')
# diagnostic_plots(df, '', ' - original')
```

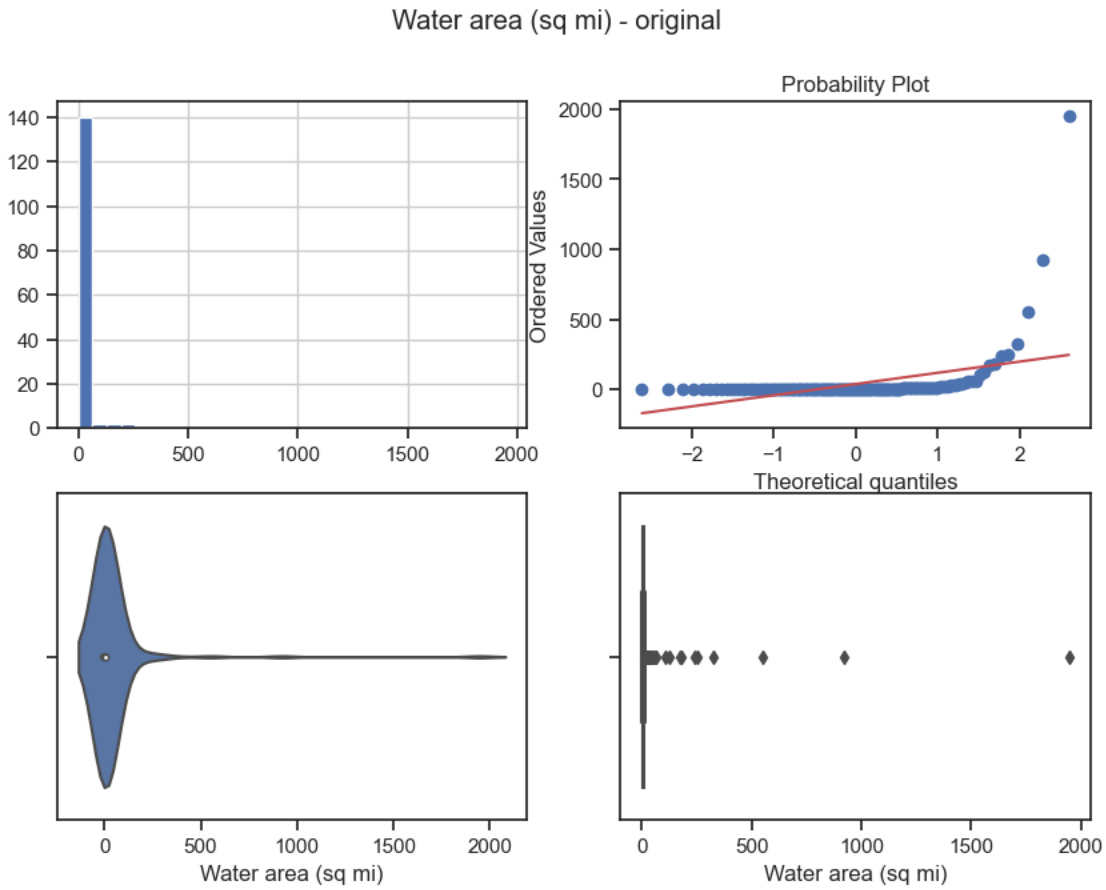
```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



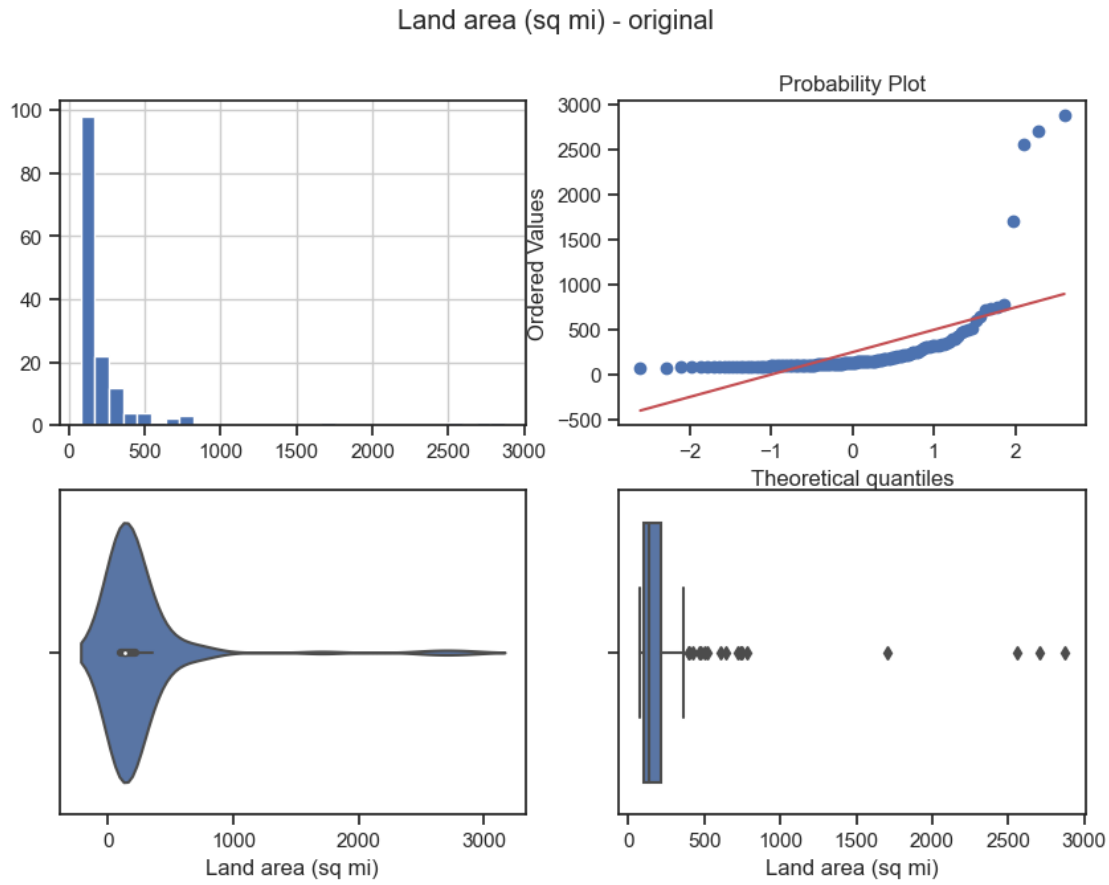

```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```




```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```




```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



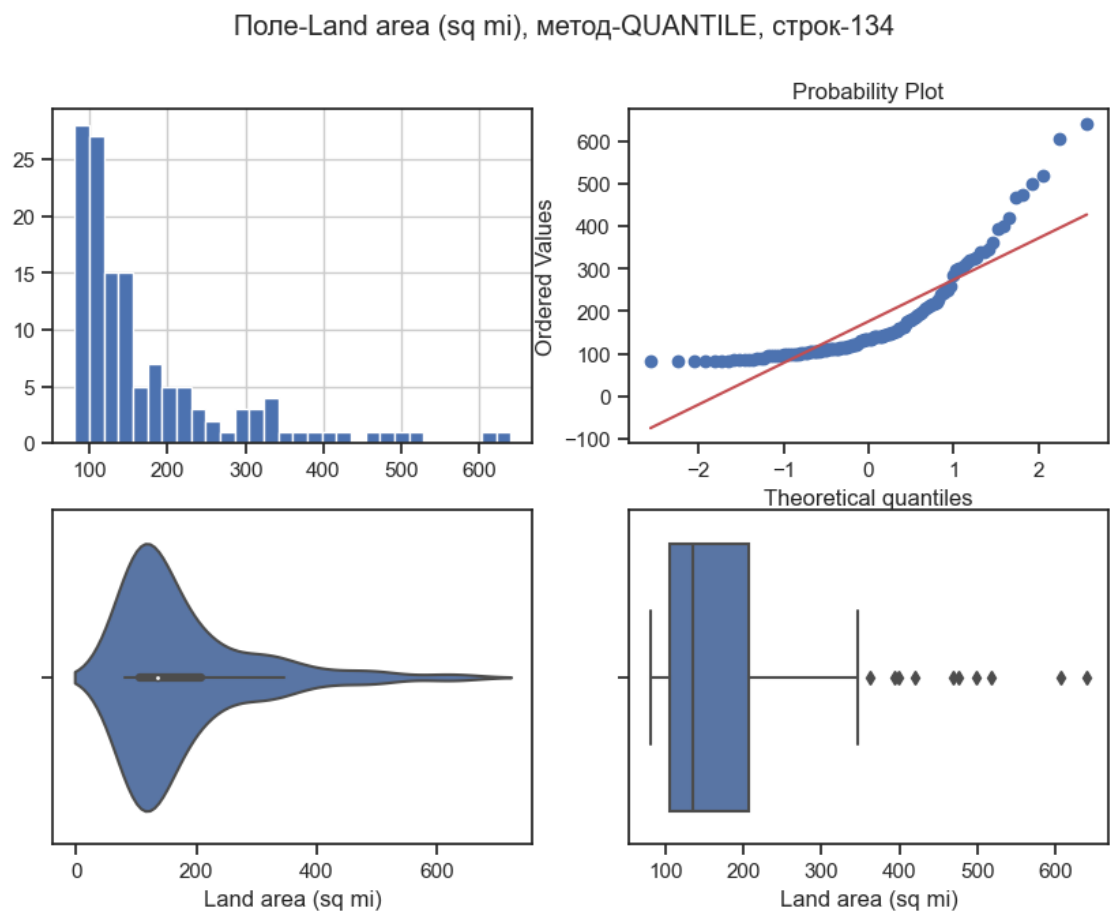

[ ]:
```
#
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
    QUANTILE = 2
    IRQ = 3

#
def get_outlier_boundaries(df, col):
    lower_boundary = df[col].quantile(0.05)
    upper_boundary = df[col].quantile(0.95)
    return lower_boundary, upper_boundary
```

### 1.1 (number_of_reviews)

[ ]:
```
#
lower_boundary, upper_boundary = get_outlier_boundaries(df, "Land area (sq mi)")
#
outliers_temp = np.where(df["Land area (sq mi)"] > upper_boundary, True,
```

```
                        np.where(df["Land area (sq mi)"] < lower_boundary,␣
 ↪True, False))
#
data_trimmed = df.loc[~(outliers_temp), ]
title = '  -{},    -{},    -{}'.format("Land area (sq mi)", "QUANTILE",␣
 ↪data_trimmed.shape[0])
diagnostic_plots(data_trimmed, "Land area (sq mi)", title)
```

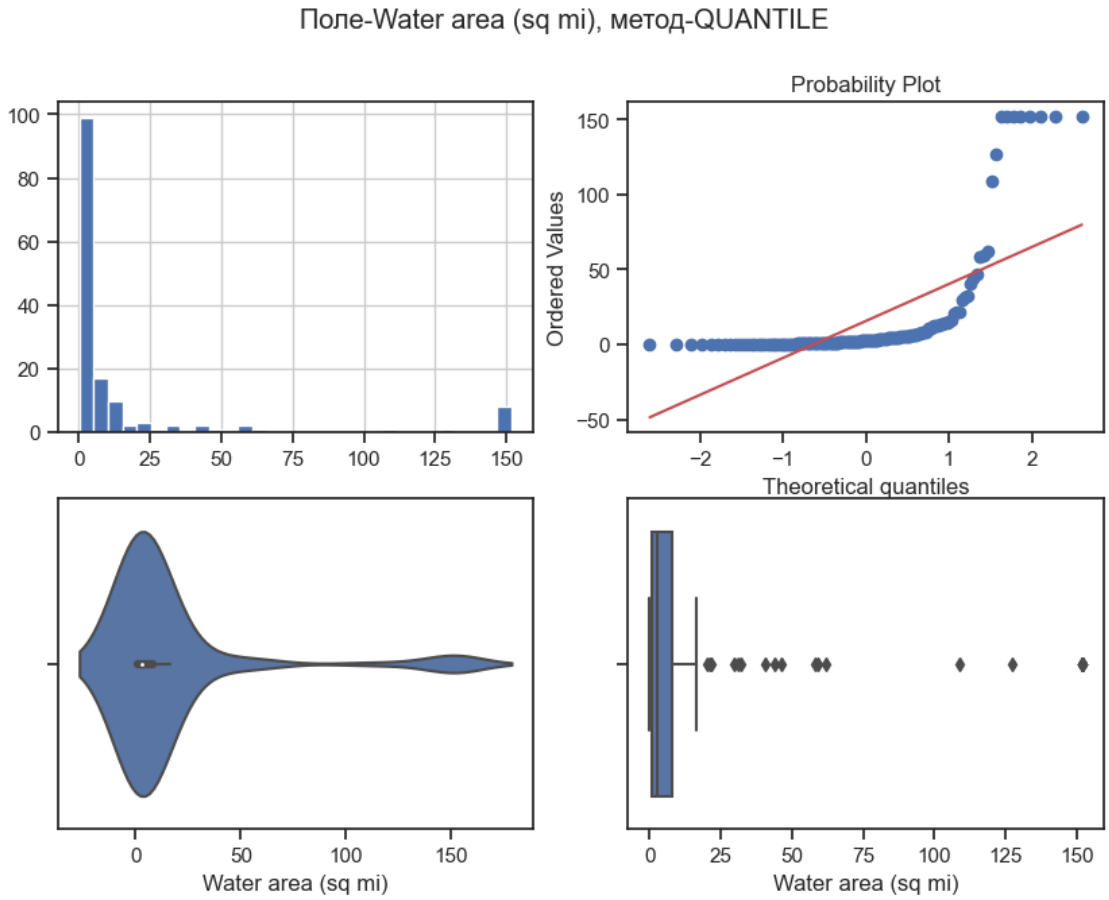
```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



### 1.2

[ ]:
```
#
lower_boundary, upper_boundary = get_outlier_boundaries(df, "Water area (sq␣
 ↪mi)")
#
df["Water area (sq mi)"] = np.where(df["Water area (sq mi)"] > upper_boundary,␣
 ↪upper_boundary,
                    np.where(df["Water area (sq mi)"] < lower_boundary,␣
 ↪lower_boundary, df["Water area (sq mi)"]))
title = '  -{},    -{}'.format("Water area (sq mi)", "QUANTILE")
diagnostic_plots(df, "Water area (sq mi)", title)
```

```
C:\Users\hae19\AppData\Local\Temp\ipykernel_7660\4201870494.py:4:
MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated
since 3.6 and will be removed two minor releases later; explicitly call
ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



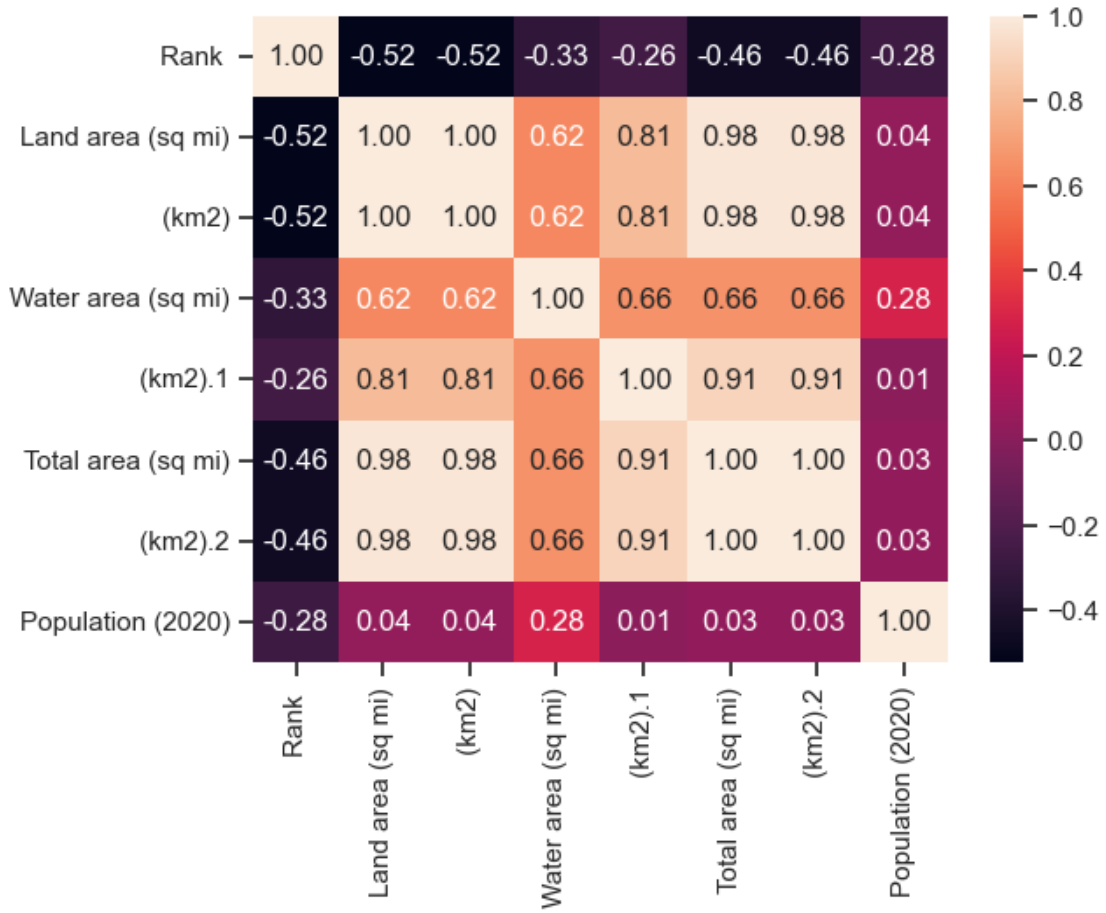
## 2

### 2.1 ( )

[ ]:
```
df_new = df.drop(columns=['City ', 'State '])
```

[ ]:
```
sns.heatmap(df_new.corr(), annot=True, fmt='.2f')
```

[ ]: <Axes: >




[ ]:
```
#         DataFrame
def make_corr_df(df):
    cr = df_new.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.3]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
```

```
    cr.columns = ['f1', 'f2', 'corr']
    return cr

#
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            #
            correlated_block = cr[cr['f1'] == feature]
            cur_dups = list(correlated_block['f2'].unique()) + [feature]
            grouped_feature_list = grouped_feature_list + cur_dups
            correlated_groups.append(cur_dups)
    return correlated_groups
```

[ ]:
```
#
corr_groups(make_corr_df(df_new))
```

[ ]:
```
[[' (km2).2',
  ' (km2)',
  'Land area (sq mi)',
  ' (km2).1',
  'Water area (sq mi)',
  'Rank ',
  'Total area (sq mi)']]
```

### 3

[ ]:
```
df = repair_df(pd.read_csv('dataset.csv'))
df.drop(columns=['City ', 'State '], inplace=True)
df
```

[ ]:
```
     Rank   Land area (sq mi)   (km2)  Water area (sq mi)   (km2).1
0       1              2870.1    7434               1945.1    5038.0  \
1       2              2704.0    7003                550.7    1426.0
2       3              2556.0    6620                920.6    2384.0
3       4              1706.8    4421                239.9     621.0
4       5               778.2    2016                  0.0       0.0
..    …                   …     …                   …        …
145   146                79.6     206                 21.4      55.0
146   147                79.3     205                  0.8       2.1
147   148                79.3     205                 14.6      38.0
148   149                79.1     205                  0.5       1.3
149   150                78.5     203                  0.0       0.0
```

```
     Total area (sq mi)   (km2).2  Population (2020)
0                4815.1   12471.0               8458
1                3254.7    8430.0              32255
2                3476.6    9004.0               2127
3                1946.7    5042.0             291247
4                 778.2    2016.0               1182
..                  …         …                  …
145               101.0     262.0             269840
146                80.1     207.0               7396
147                93.9     243.0               8399
148                79.6     206.0             192517
149                78.5     203.0              95342

[150 rows x 8 columns]
```

[ ]:
```
df.columns
```

[ ]:
```
Index(['Rank ', 'Land area (sq mi)', ' (km2)', 'Water area (sq mi)',
       ' (km2).1', 'Total area (sq mi)', ' (km2).2', 'Population (2020)'],
      dtype='object')
```

[ ]:
```
X3_ALL = df.drop([' (km2)'], axis=1)
```

[ ]:
```
#
X3_train, X3_test, y3_train, y3_test = train_test_split(X3_ALL, df[' (km2)'],
                                                        test_size=0.2,
                                                        random_state=1)
```

[ ]:
```
#       L1-
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1',␣
 ↪max_iter=500, random_state=1)
e_lr1.fit(X3_train, y3_train)
#
e_lr1.coef_
```

```
g:\repos\MMO\venv\lib\site-packages\sklearn\svm\_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[ ]:
```
array([[ 2.43959825e-01, -1.76857746e-01,  0.00000000e+00,
        -2.47978764e+01, -1.62423991e-01, -6.93264753e-02,
         6.49206629e-05],
       [ 2.54069209e-01, -1.79446840e-01, -1.27193654e-01,
         1.90240328e-01, -1.34573722e-01, -6.36154704e-02,
        -4.79787513e-06],
       [ 1.03241273e-01, -9.37380744e-02,  1.07899969e-01,
         2.79630921e-02, -6.59239060e-02, -2.20610335e-02,
```

```
 -3.25168996e-06],
[ 8.45730658e-02, -6.42521574e-02,  4.17525870e-03,
 -1.97538221e-03, -4.95620524e-02, -2.00682776e-02,
  4.76681723e-06],
[ 7.78470858e-01, -5.82456461e-01,  0.00000000e+00,
 -8.71744843e+00, -5.73484920e-01, -2.24471307e-01,
  1.38543948e-04],
[ 2.03996500e-02, -2.04286593e-02,  0.00000000e+00,
 -1.12939288e+01, -2.55088640e-02, -8.15683455e-03,
 -1.66514887e-05],
[ 2.67613828e-02, -3.69925150e-02, -4.00018003e+00,
  1.33120947e+00, -3.80788182e-02, -1.31230224e-02,
  2.59356292e-06],
[ 2.68513008e-02, -3.54389080e-02, -1.64520069e-02,
  1.14779032e-02, -2.77605218e-02, -1.06225368e-02,
 -3.55331883e-06],
[ 1.85153457e-01, -3.44409941e-01,  1.51014095e-01,
  6.18265322e-02, -6.34002326e-02, -2.76286823e-02,
  1.15358981e-05],
[ 3.00911036e-02, -3.99950425e-02, -2.45386101e-02,
 -6.70243884e-02, -3.52686983e-02, -1.34584939e-02,
  6.25700257e-06],
[ 1.90625335e-02, -2.68484911e-02,  7.03569630e-03,
 -2.17972209e-03, -1.90062883e-02, -5.79794488e-03,
 -7.02875139e-06],
[ 1.19024159e-02, -2.67842554e-02,  3.17062304e-01,
 -1.29037997e-01, -2.44063857e-02, -8.16891916e-03,
 -1.37196206e-05],
[ 1.76562133e-02, -1.69324677e-02, -1.33868677e-02,
  3.20137216e-03, -8.31961955e-03, -4.07217536e-03,
 -3.58531217e-06],
[ 2.05995746e-02, -2.13315127e-02, -4.58884308e-02,
  6.84781189e-04, -1.93405894e-02, -4.09898228e-03,
  2.62363581e-06],
[ 2.11645158e-02, -1.73031517e-02,  1.69545793e-02,
  6.62732987e-03, -1.09809659e-02, -4.26093384e-03,
 -2.34077301e-06],
[-3.85257489e-03, -3.27751503e-02,  4.14311535e+00,
 -2.30068527e+00, -1.84957172e-02, -1.55994811e-03,
 -1.34873907e-05],
[ 1.05724535e-02, -1.27621627e-02, -4.45042043e-01,
 -7.18870674e-01, -5.65751565e-03, -1.26536032e-03,
  1.54385745e-06],
[ 1.18032521e-02, -4.48779403e-03,  1.55075594e-02,
 -1.00770802e-01, -7.21417557e-03, -2.45935838e-03,
 -8.93653818e-06],
[ 1.59783784e-02, -1.30727225e-02, -8.22546539e-02,
```

```
 -3.03368175e-02, -1.48394128e-02, -5.71888790e-03,
  4.94110460e-06],
[ 1.56630809e-02, -1.43291413e-01,  0.00000000e+00,
 -2.71687430e+00, -1.47914047e-01, -6.66268013e-02,
  9.46239407e-05],
[ 1.02637703e-02, -8.40187672e-03, -1.08748032e-01,
 -2.42102592e-02, -6.17698030e-03, -1.19623203e-03,
 -2.71845495e-06],
[-4.27028806e-03, -2.79642982e-02, -1.62065350e-01,
 -1.13517430e-01, -3.38534163e-02, -1.41327797e-02,
  1.27757113e-05],
[ 1.09432446e-02, -7.39916556e-03,  8.58817329e-04,
 -1.37824251e-03, -3.93219839e-03, -3.91500313e-04,
 -7.30943619e-06],
[ 4.22421710e-03, -3.98987369e-03,  1.04377263e-01,
 -8.66538081e-02, -5.88708662e-03, -2.83219198e-03,
 -9.48461528e-06],
[-4.76344914e-03, -1.11634110e-02, -7.84808278e-02,
  1.82481268e-02, -9.95716851e-03, -2.96965790e-03,
  2.91359814e-07],
[-2.37553153e-02, -4.56701438e-02,  0.00000000e+00,
 -2.79700697e+01, -5.34352176e-02, -2.31183791e-02,
  9.45964151e-05],
[ 1.05620095e-03, -2.44216024e-02,  1.35190448e-02,
  5.14249911e-03, -4.64111157e-03, -1.22489837e-03,
 -8.31470802e-06],
[ 2.59794521e-03, -2.81896169e-03, -9.29989490e-01,
 -3.31446817e-01, -1.75412330e-03, -1.68115471e-03,
 -4.27056514e-06],
[-6.67922311e-03, -6.36687985e-03, -5.23068554e-02,
  1.66576122e-02, -7.40982430e-03, -2.46683756e-03,
 -1.41976378e-06],
[-6.57661233e-03, -5.11058473e-03, -2.18335060e-02,
  2.62303143e-03, -9.33908287e-03, -1.78293415e-03,
 -5.00450889e-07],
[-1.04401662e-01, -5.70620114e-02,  5.88153548e-02,
  7.99453547e-03, -3.42410811e-02, -1.50312330e-02,
 -8.74277351e-07],
[-9.35111741e-02, -4.15920662e-02, -3.89626127e-01,
  1.57395815e-01, -3.76213368e-02, -1.54343460e-02,
  2.27863740e-06],
[-2.21362339e-01, -1.32603113e-01,  1.18220573e+00,
 -4.04376095e-01, -7.27261958e-02, -2.61120331e-02,
  3.55790642e-06],
[-3.26996938e-01, -1.24199805e-01,  3.29127766e-03,
 -3.35026614e+00, -9.06730455e-02, -4.43425630e-02,
  6.81325485e-06],
```

```
[-1.92131442e-01, -6.76614937e-02, -8.44216084e+00,
  2.94444678e+00, -6.98710527e-02, -2.88838092e-02,
  3.19890913e-06],
[-3.27464742e-01, -2.73418823e-01,  7.99825402e-02,
  2.91426120e-02, -6.11454250e-02, -2.67109848e-02,
  5.31596820e-06],
[-4.82905784e-02, -2.29338459e-02,  1.06302677e-02,
  6.75041876e-03, -1.02967421e-02, -5.04391572e-03,
 -3.09832512e-06],
[-2.78107679e-01, -1.18071424e-01,  1.52764948e+00,
 -8.03278367e-01, -1.03858941e-01, -4.87593825e-02,
  1.58918619e-05],
[-7.86460157e-02, -2.53856434e-02, -4.46041641e-01,
  1.23726471e-01, -2.26235682e-02, -7.93486177e-03,
 -4.34904894e-06],
[-3.60917389e-02, -9.70084497e-03,  2.34042064e-02,
 -8.44391654e-02, -1.08626503e-02, -3.19383084e-03,
 -7.07371643e-07],
[-7.96391651e-02, -2.57245681e-02, -1.23217708e-01,
  4.37225140e-02, -2.17262866e-02, -8.50450230e-03,
 -9.20811962e-07],
[-8.27341090e-02, -2.52000586e-02,  1.70952975e-01,
 -5.92441302e-02, -2.08740945e-02, -9.99092573e-03,
 -1.92252117e-05],
[-1.55775855e-01, -4.36436471e-02,  8.85943638e-02,
 -1.48013835e-02, -3.89729715e-02, -2.05600775e-02,
 -2.95126532e-06],
[-9.24316730e-02, -2.71828778e-02, -2.69373394e-01,
  1.15013176e-03, -2.67365271e-02, -6.81525923e-03,
 -8.67149708e-05],
[-1.08477314e-01, -3.74086895e-02,  6.53258843e-01,
 -2.38217553e-01, -2.35786795e-02, -1.05610282e-02,
  1.04370794e-06],
[-3.55352496e-01, -1.71379450e-01, -2.24434930e-01,
  1.47301479e-01, -1.10662677e-01, -1.93239775e-02,
  4.66060656e-06],
[-3.79567261e-02, -4.78997019e-03, -2.58038488e-01,
 -5.22168184e-02, -5.46132987e-03, -3.78446676e-03,
  4.00666612e-07],
[-5.90318191e-02, -7.05761113e-03, -4.99163012e-02,
  3.61867586e-03, -1.45830041e-02, -3.42957959e-03,
 -3.98223463e-06],
[-1.91010151e-01, -5.61886562e-02, -3.51362950e-01,
  7.92913472e-02, -5.39259707e-02, -2.40827983e-02,
  7.34680704e-06],
[-4.19119349e-02, -1.09628329e-02,  5.29173582e-03,
  3.57880079e-03, -4.58428418e-03, -1.63188155e-03,
```

```
 -2.24603608e-06],
[ 4.37104385e-02, -1.47517305e-02, -6.61842950e+00,
 -6.11092960e+00, -6.25720860e-03, -2.53117172e-04,
 -1.19822085e-03],
[-1.67366271e-01, -4.07854742e-02, -1.44410310e-03,
 -5.02732226e-03, -3.72846565e-02, -1.60147139e-02,
  3.11070190e-06],
[-2.24776361e-01, -5.19230392e-02,  6.37915206e-03,
  2.76564496e-02, -4.39866506e-02, -1.56663211e-02,
 -2.47947994e-06],
[-1.16706441e-01, -2.91143258e-02, -1.30879495e-02,
  1.93495587e-02, -1.70336920e-02, -6.44610215e-03,
 -1.13971215e-05],
[-2.02468689e-01, -4.02501889e-02,  3.44039419e-01,
 -2.01177023e-01, -3.51748630e-02, -1.50080587e-02,
  7.12744051e-07],
[-1.58812677e-01, -2.88956078e-02, -5.54077347e-01,
  1.52287220e-01, -2.86303643e-02, -8.42168614e-03,
 -1.98835109e-06],
[-1.58241252e-01, -2.71449159e-02, -4.64974801e-01,
  3.76287899e-02, -3.33327770e-02, -8.59906483e-03,
  2.14554235e-06],
[-9.24840908e-02, -5.16860011e-03, -7.59174798e-03,
 -6.24678494e-03, -3.75961875e-03, -2.63928742e-03,
 -9.23809894e-04],
[-1.42155624e-01, -2.31961533e-02, -6.43569033e-01,
 -1.20393936e-01, -2.16913433e-02, -8.14655649e-03,
 -1.13671121e-05],
[-1.19295442e+00, -1.96530530e-01, -1.06645571e+01,
 -4.19344902e+00, -2.02229460e-01, -7.13376045e-02,
  9.92849203e-05],
[-5.37449848e-01, -8.59597637e-02, -3.82290397e+01,
  1.12445297e+01, -8.52710993e-02, -3.18678192e-02,
  2.00158364e-05],
[-2.23596930e-01, -6.02522043e-02,  2.27125984e-02,
  8.95527356e-03, -1.72343140e-02, -5.90609914e-03,
  3.12152840e-07],
[-1.46014063e-01, -3.38403642e-02,  1.62091537e-02,
  3.49776855e-03, -7.28368738e-03, -2.33245084e-03,
 -2.12687637e-08],
[-2.60567561e-01, -8.75492134e-02, -2.55920048e+00,
 -1.49588409e+00, -7.99280984e-02, -3.52285390e-02,
  8.42946989e-05],
[-2.36501824e-01, -2.54686904e-02,  6.21943353e-02,
 -8.19986792e-03, -2.36823965e-02, -9.12668845e-03,
 -4.39704392e-06],
[-2.22461391e-01, -2.25661721e-02,  5.76456543e-01,
```

```
 -2.27281310e-01, -1.80749925e-02, -9.41009348e-03,
 -3.47298751e-05],
[-4.04641565e-01, -4.89347574e-02,  4.27683346e+01,
 -1.75764963e+01, -4.68655088e-02, -1.38158751e-02,
  4.02883864e-06],
[-3.60806696e-01, -3.68140210e-02, -3.69220579e-01,
 -6.39226071e-02, -3.68153184e-02, -1.10500608e-02,
  4.05157005e-06],
[-2.68344536e-01, -6.05438112e-03, -7.93790375e-01,
 -4.95625766e+01, -5.88637700e-03, -3.19876146e-03,
  6.44506186e-05],
[-6.54217142e-01, -4.88014729e-02, -4.36077530e+00,
  1.72511840e+00, -5.88447118e-02, -2.59269883e-02,
  7.86553880e-07],
[-1.12873643e-01, -1.12244287e-02, -5.05904450e-01,
 -3.74448159e+01, -8.28622440e-03, -3.26208884e-03,
 -1.54964193e-04],
[-2.97086580e-01, -2.25555713e-02,  2.36290622e-02,
  8.91623684e-03, -4.87082700e-03, -1.87039167e-03,
 -7.21798258e-04],
[-3.79874248e-01, -2.10795298e-02, -8.69313848e-01,
  3.45100463e-01, -2.44092514e-02, -1.03404707e-02,
 -2.69123937e-06],
[-1.45747246e+00, -1.21135357e-01,  6.51231573e+00,
 -2.65814848e+00, -9.64120387e-02, -3.17285116e-02,
 -2.97720618e-06],
[-1.07839967e+00, -8.48333471e-02,  1.55563391e-01,
 -1.64598004e-01, -7.46933122e-02, -2.93683502e-02,
  9.33662625e-06],
[-3.48800999e-01, -5.00082472e-02, -1.23502198e+01,
 -6.42137677e+00, -4.93312087e-02, -1.91181460e-02,
  4.45299606e-05],
[-1.99909575e-01, -4.37065226e-03, -1.20327722e-01,
  1.79948435e-02, -9.67601915e-03, -1.77113721e-03,
 -1.21229956e-04],
[-8.52183118e-01, -4.77571850e-02, -6.48037245e+00,
  1.27355855e+00, -5.13186462e-02, -2.19275108e-02,
  2.12670976e-05],
[-1.74088264e-01, -3.95435437e-03, -2.68175079e+00,
  1.03129546e+00, -6.37751345e-03, -3.06095599e-03,
  1.13811439e-07],
[-1.03999950e-01, -2.82698068e-02,  0.00000000e+00,
  9.61008395e-03, -1.61343746e-02, -7.71671411e-03,
  4.20366526e-06],
[-3.86495653e-01, -9.76715310e-03, -2.84668950e-01,
  3.75340435e-02, -6.43404938e-03, -3.96419396e-03,
 -5.46039780e-06],
```

```
[-1.39368334e+00, -6.01175314e-02, -4.55365481e+00,
  1.34489425e+00, -5.47964502e-02, -1.74166347e-02,
  1.31379611e-05],
[-4.15691080e-01, -1.04838524e-02, -1.48832727e+00,
  4.80443862e-01, -8.92975833e-03, -3.21424903e-03,
 -2.25588539e-06],
[-4.87961878e-01, -2.08910758e-02, -1.25371054e-01,
  2.88948368e-03, -1.79075776e-02, -6.10948274e-03,
  2.22017558e-06],
[-4.12750598e-01, -1.51072199e-02, -6.48323552e-02,
  3.86803079e-02, -1.21335531e-02, -4.89027712e-03,
 -5.92612316e-07],
[-2.15079522e+00, -9.89513913e-02,  1.43418010e-01,
  3.07309456e-02, -6.56651295e-02, -2.34692659e-02,
 -1.10893188e-06],
[-9.83166426e-01, -2.27153364e-02,  2.39238517e-01,
 -1.67360044e-01, -1.83572797e-02, -6.49086092e-03,
 -8.84617064e-06],
[-1.54254122e+00, -5.23813283e-02,  1.53637864e+00,
 -6.02471006e-01, -5.14873083e-02, -1.92933405e-02,
  2.71710480e-06],
[-1.20547119e+00, -2.37363328e-02,  3.26684406e-01,
 -2.83920120e-01, -2.35854442e-02, -9.02421528e-03,
 -2.86172167e-06],
[-1.03578066e+00, -2.60313176e-02,  4.10273110e-02,
  1.77737185e-02, -1.49042736e-02, -5.81615643e-03,
 -3.00296192e-05],
[-2.94383565e-01, -2.05695302e-02, -3.27618992e-01,
 -1.73769806e-01, -1.93679433e-02, -1.00874415e-02,
  1.80938328e-05],
[-2.03466510e+00, -2.55440126e-02,  5.68295329e-02,
  1.56875657e-02, -2.08285811e-02, -6.03270063e-03,
 -1.43051256e-06],
[-7.47960791e-01, -7.84417508e-03, -1.51102180e-01,
  2.21136272e-02, -1.05864685e-02, -2.80105231e-03,
  7.52996588e-07],
[-5.04746956e-01, -2.75023241e-03, -2.07636093e-01,
 -1.13242275e+00, -2.80829011e-03, -2.13893610e-03,
  1.46042652e-05],
[-3.02444144e-01, -1.27586323e-03,  3.36816133e-03,
  8.29488687e-04, -3.10998415e-03, -1.63229831e-04,
 -2.61860095e-07],
[-1.21349021e+01,  2.15267839e-02, -1.84189749e-01,
 -8.74260634e-02,  2.13616757e-02,  9.57441835e-03,
  3.65148329e-05],
[-1.03079936e-01,  1.11248732e-03, -3.88130517e-01,
 -5.26765551e-02,  9.95308804e-04, -2.48490242e-04,
```

```
        -2.53397856e-06],
       [-6.94992446e-02,  2.71972680e-03, -1.11936027e-02,
        -4.25678959e-03, -4.99974767e-05,  4.43769680e-04,
        -3.61668401e-05],
       [-2.88200195e-01,  5.23262655e-03, -1.19945844e+00,
        -6.12513790e-01,  4.90157717e-03,  8.27692786e-04,
        -3.65134757e-04],
       [-6.68438027e+00,  1.80416687e-02, -1.00963964e-01,
        -3.72322478e-02,  1.11275057e-02,  4.03532723e-03,
         1.57691516e-05],
       [-9.53726449e-01,  4.59873764e-03,  1.01403755e-03,
        -8.81460722e-04,  1.20451826e-03,  3.46744796e-04,
        -2.73794499e-03],
       [-3.51760290e+00,  5.16866129e-03, -1.32886864e-02,
        -4.48879649e-03,  2.23234578e-03,  6.98625022e-04,
        -1.59680778e-05],
       [-1.15724271e+00, -1.31942523e-03,  5.55448802e-03,
         2.38342734e-03, -1.19797490e-03,  1.28148038e-06,
         2.14586439e-06]])
```

[ ]:
```
#               "     "
from sklearn.feature_selection import SelectFromModel
sel_e_lr1 = SelectFromModel(e_lr1)
sel_e_lr1.fit(X3_train, y3_train)
sel_e_lr1.get_support()
```

```
g:\repos\MMO\venv\lib\site-packages\sklearn\svm\_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[ ]:
```
array([ True,  True,  True,  True,  True,  True,  True])
```

[ ]:
```
e_lr2 = LinearSVC(C=0.01, penalty="l1", max_iter=2000, dual=False)
e_lr2.fit(X3_train, y3_train)
#
e_lr2.coef_
```

```
g:\repos\MMO\venv\lib\site-packages\sklearn\svm\_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[ ]:
```
array([[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -3.61396843e-03,
        -6.73469694e-07],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         1.50281199e-03,  0.00000000e+00, -3.26928764e-03,
```

```
 -7.57143905e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  3.67111027e-03,  0.00000000e+00, -3.62950258e-03,
 -2.13152837e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.77258942e-03,
  1.48961944e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.40317534e-03,
  5.00255192e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.55586000e-03,
 -7.66740994e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.01767670e-03,
  8.78633104e-08],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.64392718e-03,
 -5.48670790e-07],
[ 0.00000000e+00, -2.88416401e-03,  0.00000000e+00,
  4.46793534e-03,  0.00000000e+00, -3.39033234e-03,
  2.42017400e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.26219128e-03,
  3.83502355e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.51054041e-03,
 -8.02569715e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.88683004e-03,
 -1.52738508e-06],
[-1.29018700e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.96571218e-03,
 -4.05345059e-07],
[-2.19562648e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.76960385e-03,
  8.62963138e-08],
[-2.75831595e-03,  0.00000000e+00,  0.00000000e+00,
  1.25115605e-03,  0.00000000e+00, -2.32366067e-03,
 -1.94813392e-07],
[-6.62383406e-04,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.97662923e-03,
 -1.23620058e-06],
[-2.46649049e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.53606350e-03,
 -4.39459673e-08],
[-2.26074982e-03,  0.00000000e+00,  0.00000000e+00,
```

```
  0.00000000e+00,  0.00000000e+00, -2.34676600e-03,
 -6.11953015e-07],
[-2.45210204e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.80448103e-03,
  2.86561929e-07],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.40309298e-03,
  5.07087269e-07],
[-2.86775444e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.25668671e-03,
 -1.83044085e-07],
[-2.09165535e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.02702074e-03,
  3.20369431e-07],
[-2.64281224e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.95484330e-03,
 -6.39926157e-07],
[-2.00935275e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.92203625e-03,
 -9.81040918e-07],
[-3.41755039e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.10604444e-03,
 -2.42237662e-08],
[-3.33717535e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.25976188e-03,
  1.36304472e-07],
[-2.49792440e-03,  0.00000000e+00,  0.00000000e+00,
  2.96754361e-03,  0.00000000e+00, -2.13389913e-03,
 -7.75382060e-07],
[-3.30136468e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.97976445e-03,
 -3.46166375e-07],
[-3.23891430e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.75809110e-03,
 -1.51247504e-07],
[-3.69582949e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.85790287e-03,
 -4.95672112e-08],
[-3.97555566e-03,  0.00000000e+00,  0.00000000e+00,
  1.75888821e-03,  0.00000000e+00, -1.90162010e-03,
 -5.54728979e-08],
[-3.85313590e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.99138464e-03,
  6.50574613e-08],
[-3.98169529e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.96339382e-03,
  9.47439581e-08],
```

```
[-3.90575968e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.73370160e-03,
 -1.31647492e-07],
[-3.97968365e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.95658039e-03,
  7.85579265e-08],
[-4.03739366e-03,  0.00000000e+00,  0.00000000e+00,
  2.27953661e-03,  0.00000000e+00, -2.01121932e-03,
  6.53134502e-08],
[-4.22484900e-03,  0.00000000e+00,  0.00000000e+00,
  1.79351028e-03,  0.00000000e+00, -1.76683731e-03,
 -1.49746764e-07],
[-3.82818643e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -2.12433098e-03,
  1.99097217e-07],
[-4.12952200e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.71985976e-03,
 -2.02400356e-07],
[-4.20397008e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.78282404e-03,
 -4.48843130e-08],
[-4.24611093e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.76160935e-03,
 -5.34490985e-08],
[-3.81621491e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.24498091e-03,
 -1.60842477e-06],
[-4.39601929e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.64774538e-03,
 -1.56330196e-07],
[-2.93901668e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.16846773e-03,
 -1.35134524e-05],
[-4.50366532e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.72922186e-03,
  2.38866011e-08],
[-4.44818467e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.86506130e-03,
  1.65585101e-07],
[-4.55987602e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.67989025e-03,
 -1.95674742e-08],
[-4.72335029e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.50044217e-03,
 -2.34271568e-07],
[-4.85433114e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.62445729e-03,
```

```
  1.39327313e-07],
[-4.99011494e-03,  0.00000000e+00,  0.00000000e+00,
  1.60250955e-03,  0.00000000e+00, -1.52622086e-03,
 -1.63182255e-07],
[-2.96579545e-03,  0.00000000e+00,  0.00000000e+00,
 -9.97420476e-05,  0.00000000e+00, -8.61387801e-04,
 -3.15965985e-05],
[-4.81698121e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.74297906e-03,
  1.45480255e-07],
[-5.06953644e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.44610845e-03,
 -1.68880241e-07],
[-4.98475207e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.25090119e-03,
 -7.93204238e-07],
[-5.55858089e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.39117691e-03,
  6.74194353e-08],
[-5.21407594e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.43895358e-03,
 -1.28160757e-07],
[-5.27239852e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.58583859e-03,
  9.52864315e-08],
[-3.40591300e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.13286144e-04,
 -8.07367233e-05],
[-5.04646079e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.21688980e-03,
 -9.37000737e-07],
[-5.14585235e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.66630053e-03,
  1.42044962e-07],
[-5.77416123e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.44745093e-03,
  5.54727497e-08],
[-7.13638973e-03, -1.97153591e-03,  0.00000000e+00,
  1.99812332e-03,  0.00000000e+00, -5.10056909e-04,
 -4.96793418e-08],
[-6.38468046e-03,  0.00000000e+00,  0.00000000e+00,
  2.24017074e-03,  0.00000000e+00, -1.44953849e-03,
 -4.07001071e-09],
[-5.85857526e-03,  0.00000000e+00,  0.00000000e+00,
 -8.96413805e-04,  0.00000000e+00, -1.52016657e-03,
  1.75015134e-07],
[-6.52866326e-03,  0.00000000e+00,  0.00000000e+00,
```

```
  0.00000000e+00,  0.00000000e+00, -1.07642745e-03,
 -3.56975307e-07],
[-5.89795387e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.51106603e-04,
 -3.39272304e-06],
[-6.78217646e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.15597350e-03,
 -1.13634446e-07],
[-6.94642398e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.26901036e-03,
  8.73701494e-08],
[-6.73708760e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.00418149e-03,
 -4.86517699e-07],
[-7.47395176e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.13436924e-03,
  1.54097431e-08],
[-6.93718337e-03,  0.00000000e+00,  0.00000000e+00,
 -4.12983403e-04,  0.00000000e+00, -6.10469308e-04,
 -4.12768384e-06],
[-6.43435194e-03,  0.00000000e+00,  0.00000000e+00,
  8.83518682e-04,  0.00000000e+00, -4.12202542e-04,
 -3.29624743e-05],
[-8.34250001e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.86757660e-04,
 -2.03606622e-07],
[-8.40775313e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.87401823e-04,
 -1.89622239e-07],
[-8.43236055e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -1.06618138e-03,
  7.26858870e-08],
[-9.35682014e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -9.36474668e-04,
  4.72224068e-08],
[-7.63155760e-03,  0.00000000e+00,  0.00000000e+00,
 -3.33355695e-04,  0.00000000e+00, -4.31928608e-04,
 -8.04988991e-06],
[-9.80980762e-03,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.97591432e-04,
  4.61719760e-08],
[-1.06042595e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.20881485e-04,
  3.43781088e-08],
[ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -5.53178308e-03,
  8.80542437e-07],
```

```
[-1.10055624e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -5.80113837e-04,
 -3.38694506e-07],
[-1.08391834e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -8.15707253e-04,
  4.06990195e-08],
[-1.12047775e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -7.12562580e-04,
 -6.33949039e-08],
[-1.12274214e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -7.88848307e-04,
  3.97000907e-08],
[-1.17091413e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -7.13257373e-04,
 -1.29591684e-08],
[-1.17965494e-02,  0.00000000e+00,  0.00000000e+00,
  7.61690444e-04,  0.00000000e+00, -7.64253683e-04,
  1.12814047e-08],
[-1.27228122e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.68569573e-04,
 -3.48335101e-07],
[-1.23854911e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -7.11467479e-04,
  2.92443055e-08],
[-1.27045003e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -6.89751832e-04,
  2.56824767e-08],
[-1.26872048e-02,  0.00000000e+00,  0.00000000e+00,
  5.92431721e-04,  0.00000000e+00, -4.02450753e-04,
 -9.04873864e-07],
[-1.30682513e-02,  0.00000000e+00,  0.00000000e+00,
 -1.29485177e-02,  0.00000000e+00, -1.44146439e-03,
  8.13468176e-07],
[-1.56868018e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.43229389e-04,
 -1.36823578e-07],
[-1.55018974e-02,  0.00000000e+00,  0.00000000e+00,
 -6.47978079e-04,  0.00000000e+00, -5.69613031e-04,
  4.97105999e-08],
[-1.61407945e-02,  0.00000000e+00,  0.00000000e+00,
 -2.93624187e-03,  0.00000000e+00, -5.47589386e-04,
  1.12738320e-07],
[-1.69044875e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -3.87449936e-04,
 -1.67781530e-07],
[-1.93233170e-02,  0.00000000e+00,  0.00000000e+00,
  0.00000000e+00,  0.00000000e+00, -4.53005614e-04,
```

```
         2.54261504e-08],
       [-1.32435026e-02,  0.00000000e+00,  0.00000000e+00,
        -1.21218454e-03,  0.00000000e+00, -2.05910234e-04,
        -2.21898210e-06],
       [-1.19797618e-02,  0.00000000e+00,  0.00000000e+00,
        -2.83874420e-04,  0.00000000e+00, -2.65921442e-05,
        -2.22084755e-05],
       [-5.40691871e-03,  0.00000000e+00,  0.00000000e+00,
        -2.52552341e-03,  0.00000000e+00,  6.53597128e-04,
        -3.89720466e-04],
       [-2.05232388e-02,  0.00000000e+00,  0.00000000e+00,
         1.09349966e-04,  0.00000000e+00, -1.15692060e-04,
        -6.20253599e-07],
       [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  8.47129651e-04,
        -3.12238041e-03],
       [-1.68318158e-02,  0.00000000e+00,  0.00000000e+00,
        -1.75645912e-04,  0.00000000e+00,  7.54591423e-06,
        -7.11532048e-06],
       [-3.55409553e-03, -5.03090957e-04,  0.00000000e+00,
         1.99101520e-03,  0.00000000e+00, -4.26980600e-04,
        -2.70101805e-04]])
```

[ ]:
```
#           False  . .
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(X3_train, y3_train)
sel_e_lr2.get_support()
```

```
g:\repos\MMO\venv\lib\site-packages\sklearn\svm\_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

[ ]:
```
array([ True,  True, False,  True, False,  True,  True])
```