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

Лабораторная работа №3
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группы ИУ5-22М
ЧжаоЛян

Москва — 2022 г.

Цель лабораторной работы

изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

In [95]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import datetime
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
import warnings
warnings.simplefilter("ignore", UserWarning)
```

In [96]:
```
dataset = pd.read_csv(r'C:\Users\80667\Desktop\文件\ИУ5\研一下\MMO\lab\lab3\DOGE-USD.csv')
```

In [97]:
```
dataset.head()
```

Out[97]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	0.000293	0.000299	0.000260	0.000268	0.000268	1463600.0
1	2014-09-18	0.000268	0.000325	0.000267	0.000298	0.000298	2215910.0
2	2014-09-19	0.000298	0.000307	0.000275	0.000277	0.000277	883563.0
3	2014-09-20	0.000276	0.000310	0.000267	0.000292	0.000292	993004.0
4	2014-09-21	0.000293	0.000299	0.000284	0.000288	0.000288	539140.0

In [98]:
```
X=dataset.drop('Date',axis=1)
y=dataset['Date']
```

In [99]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y)
```

In [100]:
```
X_train=X_train.drop(['Close'],axis=1)
X_test=X_test.drop(['Close'],axis=1)
X_train
```

Out[100]:

	Open	High	Low	Adj Close	Volume
1288	0.003134	0.003244	0.003114	0.003141	4.912220e+06
2238	0.002582	0.002592	0.002515	0.002517	5.014238e+07
921	0.000298	0.000309	0.000271	0.000272	1.203110e+06
391	0.000119	0.000122	0.000117	0.000121	3.233600e+04
2180	0.002900	0.002934	0.002720	0.002758	6.229589e+07
...	...	...	...	...	...
1504	0.003857	0.003908	0.003819	0.003850	1.182890e+07
756	0.000230	0.000230	0.000224	0.000225	1.365570e+05
426	0.000131	0.000133	0.000129	0.000133	7.167400e+04
977	0.001910	0.003062	0.001910	0.003062	1.054830e+08
2549	0.250362	0.259558	0.249270	0.252596	1.763184e+09

1943 rows × 5 columns

Масштабирование признаков

Методом Z-оценки

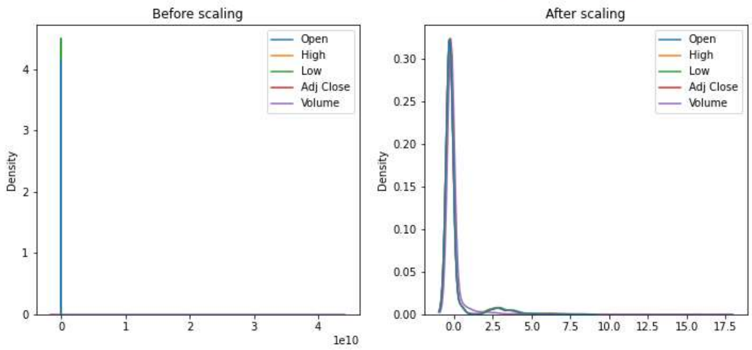
In [101]:
```
def arr_df(a):
    df = pd.DataFrame(a, columns=X_train.columns)
    return df
scaler1 = StandardScaler()
scaled_X_1 = arr_df(scaler1.fit_transform(X_train))
scaled_X_1
```

Out[101]:

	Open	High	Low	Adj Close	Volume
0	-0.268034	-0.267598	-0.268521	-0.268951	-0.206885
1	-0.275110	-0.275368	-0.276847	-0.276944	-0.188235
2	-0.304391	-0.302576	-0.308040	-0.305701	-0.208414
3	-0.306686	-0.304804	-0.310181	-0.307635	-0.208897
4	-0.271034	-0.271292	-0.273998	-0.273857	-0.183224
...	...	...	...	...	...
1938	-0.258765	-0.259685	-0.258721	-0.259870	-0.204033
1939	-0.305263	-0.303517	-0.308693	-0.306303	-0.208854
1940	-0.306532	-0.304673	-0.310014	-0.307481	-0.208881
1941	-0.283726	-0.269767	-0.285257	-0.269963	-0.165417
1942	2.901445	2.787037	3.153138	2.926337	0.518099

1943 rows × 5 columns

In [102]:
```
def data_visualize(columns, df1, df2, label1, label2):
  fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12, 5))
  ax1.set_title(label1)
  sns.kdeplot(data=df1[columns], ax=ax1)
  ax2.set_title(label2)
  sns.kdeplot(data=df2[columns], ax=ax2)
  plt.show()

data_visualize(X_train.columns, X_train, scaled_X_1, 'Before scaling', 'After scaling')
```



Методом MinMaxScaler:

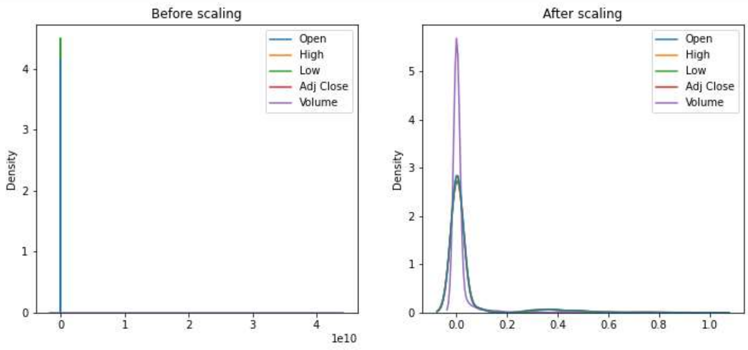
In [103]:
```
scaler2 = MinMaxScaler()
scaled_X_2 = arr_df(scaler2.fit_transform(X_train))
scaled_X_2
```

Out[103]:

	Open	High	Low	Adj Close	Volume
0	0.004428	0.004277	0.004978	0.004645	1.151209e-04
1	0.003625	0.003393	0.003993	0.003696	1.178732e-03
2	0.000304	0.000297	0.000303	0.000281	2.789918e-05
3	0.000044	0.000043	0.000049	0.000052	3.678065e-07
4	0.004087	0.003856	0.004330	0.004062	1.464529e-03
...	...	...	...	...	...
1938	0.005479	0.005177	0.006137	0.005723	2.777703e-04
1939	0.000205	0.000190	0.000225	0.000210	2.818619e-06
1940	0.000061	0.000058	0.000069	0.000070	1.292861e-06
1941	0.002648	0.004030	0.002998	0.004524	2.480096e-03
1942	0.363921	0.351832	0.409786	0.384017	4.146181e-02

1943 rows × 5 columns

In [104]:
```
data_visualize(X_train.columns, X_train, scaled_X_2, 'Before scaling', 'After scaling')
```



Методом RobustScaler:

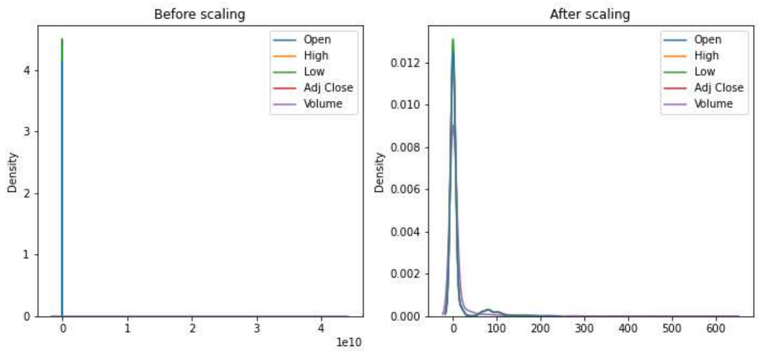
In [105]:
```
scaler3 = RobustScaler()
scaled_X_3 = arr_df(scaler3.fit_transform(X_train))
scaled_X_3
```

Out[105]:

	Open	High	Low	Adj Close	Volume
0	0.316035	0.319731	0.351911	0.319328	-0.103476
1	0.132402	0.110639	0.146157	0.111657	0.565335
2	-0.627412	-0.621502	-0.624646	-0.635494	-0.158322
3	-0.686959	-0.681472	-0.677544	-0.685748	-0.175634
4	0.238190	0.220316	0.216574	0.191863	0.745047
...	...	...	...	...	...
1938	0.556554	0.532671	0.594075	0.555287	-0.001200
1939	-0.650033	-0.646837	-0.640790	-0.651136	-0.174093
1940	-0.682967	-0.677944	-0.673422	-0.681754	-0.175052
1941	-0.091151	0.261365	-0.061657	0.293036	1.383648
1942	82.560878	82.517758	84.905281	83.339546	25.895805

1943 rows × 5 columns

In [106]:
```
data_visualize(X_train.columns, X_train, scaled_X_3, 'Before scaling', 'After scaling')
```



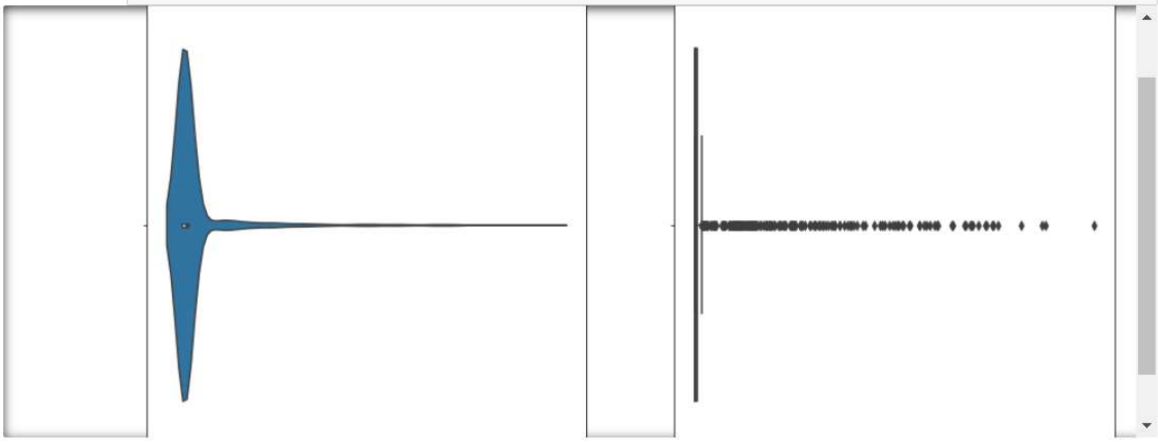
Обработка выбросов для числовых признаков

Удаление выборсов

In [155]:
```
def plot_for_analys(df, variable, title):
    fig, ax = plt.subplots(figsize=(15,7))
    plt.subplot(1, 2, 1)
    sns.violinplot(x=df[variable])
    plt.subplot(1, 2, 2)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

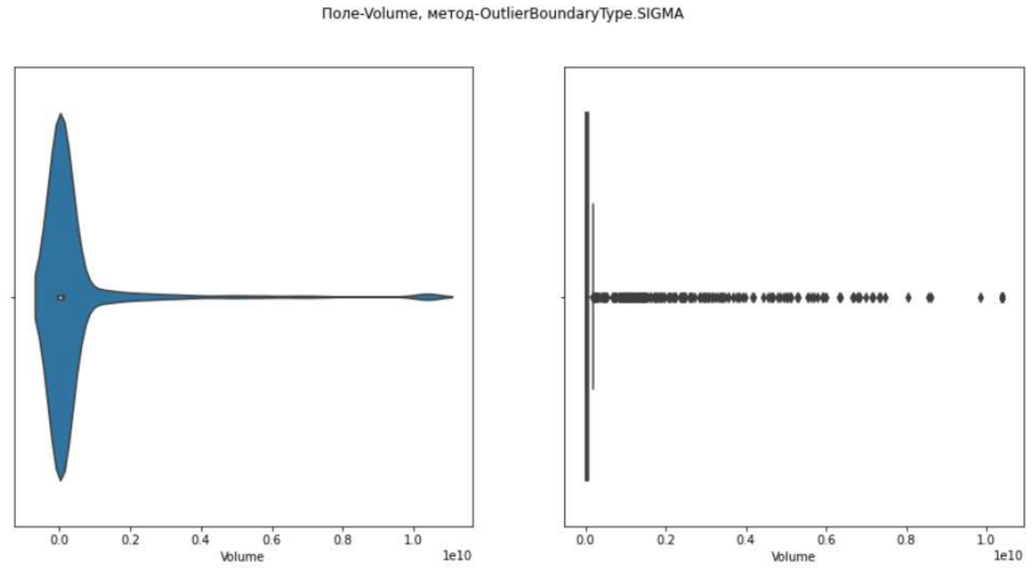
In [156]:
```
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryType):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

    else:
        raise NameError('Unknown Outlier Boundary Type')

    return lower_boundary, upper_boundary
```

In [157]:
```
x_col_list = ['Volume']
data=X_train
for col in x_col_list:
    for obt in OutlierBoundaryType:
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
        # Флаги для удаления выбросов
        outliers_temp = np.where(data[col] > upper_boundary, True,
                                 np.where(data[col] < lower_boundary, True, False))
        # Удаление данных на основе флага
        data_trimmed = data.loc[~(outliers_temp), ]
        title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, data_trimmed.shape[0])
        plot_for_analys(data_trimmed, col, title)
```



Замена выбросов

```
In [158]: for col in x_col_list:
              for obt in OutlierBoundaryType:
                  lower_boundary, upper_boundary = get_outlier_boundaries(data, col, obt)
                  data[col] = np.where(data[col] > upper_boundary, upper_boundary,
                                       np.where(data[col] < lower_boundary, lower_boundary, data[col]))
                  title = 'Поле-{}, метод-{}'.format(col, obt)
                  plot_for_analys(data, col, title)
```



Обработка по крайней мере одного нестандартного признака

In [111]:
```
dataset.head()
```
Out[111]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	0.000293	0.000299	0.000260	0.000268	0.000268	1463600.0
1	2014-09-18	0.000268	0.000325	0.000267	0.000298	0.000298	2215910.0
2	2014-09-19	0.000298	0.000307	0.000275	0.000277	0.000277	883563.0
3	2014-09-20	0.000276	0.000310	0.000267	0.000292	0.000292	993004.0
4	2014-09-21	0.000293	0.000299	0.000284	0.000288	0.000288	539140.0

In [112]:
```
dataset['Date'] = dataset.apply(lambda x: pd.to_datetime(x['Date'], format='%Y-%m-%d'), axis=1)
```
In [113]:
```
dataset.head()
```
Out[113]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	0.000293	0.000299	0.000260	0.000268	0.000268	1463600.0
1	2014-09-18	0.000268	0.000325	0.000267	0.000298	0.000298	2215910.0
2	2014-09-19	0.000298	0.000307	0.000275	0.000277	0.000277	883563.0
3	2014-09-20	0.000276	0.000310	0.000267	0.000292	0.000292	993004.0
4	2014-09-21	0.000293	0.000299	0.000284	0.000288	0.000288	539140.0

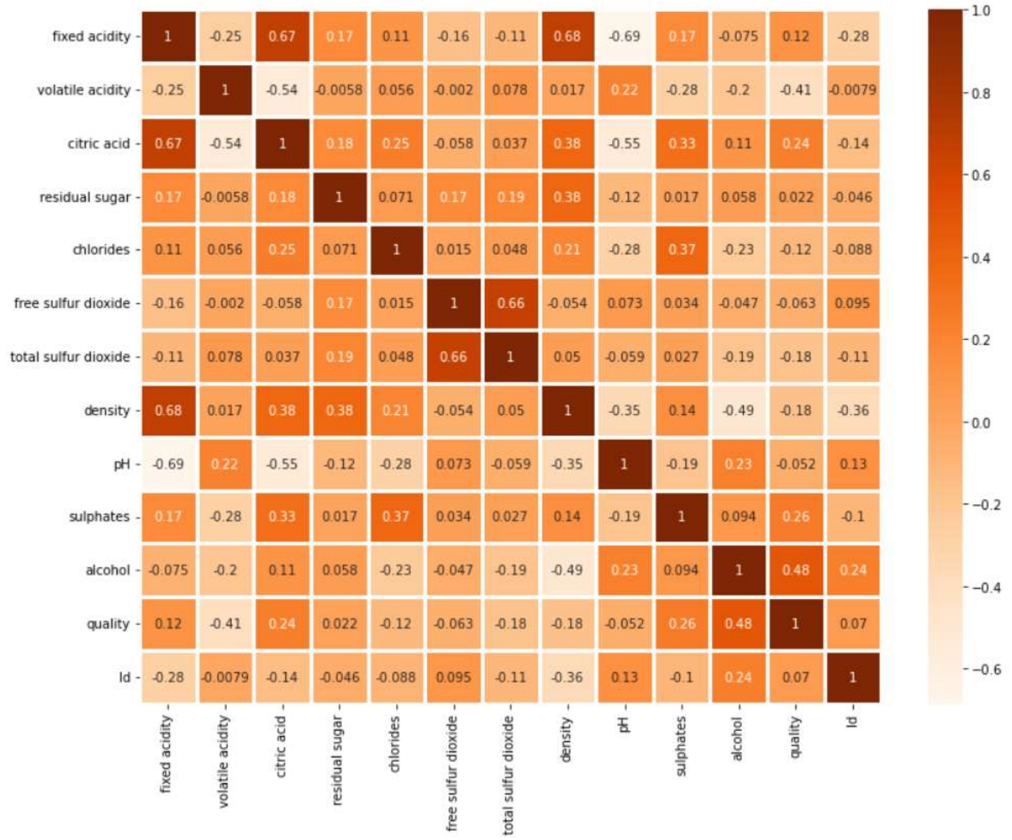
In [114]:
```
dataset['now'] = datetime.datetime.today()
dataset['diff'] = dataset['now'] - dataset['Date']
dataset.dtypes
```
Out[114]:
```
Date          datetime64[ns]
Open                 float64
High                 float64
Low                  float64
Close                float64
Adj Close            float64
Volume               float64
now           datetime64[ns]
diff         timedelta64[ns]
dtype: object
```
In [115]:
```
dataset.head()
```
Out[115]:

	Date	Open	High	Low	Close	Adj Close	Volume	now	diff
0	2014-09-17	0.000293	0.000299	0.000260	0.000268	0.000268	1463600.0	2022-06-06 02:22:24.269973	2819 days 02:22:24.269973
1	2014-09-18	0.000268	0.000325	0.000267	0.000298	0.000298	2215910.0	2022-06-06 02:22:24.269973	2818 days 02:22:24.269973
2	2014-09-19	0.000298	0.000307	0.000275	0.000277	0.000277	883563.0	2022-06-06 02:22:24.269973	2817 days 02:22:24.269973
3	2014-09-20	0.000276	0.000310	0.000267	0.000292	0.000292	993004.0	2022-06-06 02:22:24.269973	2816 days 02:22:24.269973
4	2014-09-21	0.000293	0.000299	0.000284	0.000288	0.000288	539140.0	2022-06-06 02:22:24.269973	2815 days 02:22:24.269973

Отбор признаков из группы методов фильтрации (корреляция признаков)

In [128]:
```python
data_dir2 = r'C:\Users\80667\Desktop\文件\ИУ5\研一下\MMO\数据集\葡萄酒质量数据集\WineQt.csv'
data=pd.read_csv(data_dir2)
plt.figure(figsize=(13,10))
sns.heatmap(data.corr(), cmap="Oranges", annot=True, linewidths=3)
```

Out[128]: <AxesSubplot:>



In [130]:
```
def make_corr_df(df):
    cr = data.corr()
    cr = cr.abs().unstack()
    cr = cr.sort_values(ascending=False)
    cr = cr[cr >= 0.53]
    cr = cr[cr < 1]
    cr = pd.DataFrame(cr).reset_index()
    cr.columns = ['f1', 'f2', 'corr']
    return cr
```

In [131]:
```
make_corr_df(data)
```

Out[131]:

	f1	f2	corr
0	pH	fixed acidity	0.685163
1	fixed acidity	pH	0.685163
2	density	fixed acidity	0.681501
3	fixed acidity	density	0.681501
4	citric acid	fixed acidity	0.673157
5	fixed acidity	citric acid	0.673157
6	free sulfur dioxide	total sulfur dioxide	0.661093
7	total sulfur dioxide	free sulfur dioxide	0.661093
8	pH	citric acid	0.546339
9	citric acid	pH	0.546339
10	volatile acidity	citric acid	0.544187
11	citric acid	volatile acidity	0.544187

In [132]:
```
def corr_groups(cr):
    grouped_feature_list = []
    correlated_groups = []

    for feature in cr['f1'].unique():
        if feature not in grouped_feature_list:
            # находим коррелирующие признаки
            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
```

In [133]:
```
# Группы коррелирующих признаков
corr_groups(make_corr_df(data))
```

Out[133]:
```
[['fixed acidity', 'citric acid', 'pH'],
 ['fixed acidity', 'density'],
 ['total sulfur dioxide', 'free sulfur dioxide'],
 ['citric acid', 'volatile acidity']]
```

In [134]:
```
data=data.drop(['fixed acidity','citric acid'],axis=1)
data.head()
```

Out[134]:

	volatile acidity	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	0.70	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	0.88	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	0.76	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	0.28	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	0.70	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

Отбор признаков из группы методов обертывания (алгоритм полного перебора)

In [135]:

```
pip install mlxtend
```

```
Requirement already satisfied: numpy>=1.16.2 in c:\zl\work\anaconda\anaconda\lib\site-packages (from mlxte
nd) (1.22.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\zl\work\anaconda\anaconda\lib\site-packages (from ma
tplotlib>=3.0.0->mlxtend) (3.0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\zl\work\anaconda\anaconda\lib\site-packages (from m
atplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: pillow>=6.2.0 in c:\zl\work\anaconda\anaconda\lib\site-packages (from matpl
otlib>=3.0.0->mlxtend) (8.4.0)
Requirement already satisfied: cycler>=0.10 in c:\zl\work\anaconda\anaconda\lib\site-packages (from matplo
tlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\zl\work\anaconda\anaconda\lib\site-packages (fro
m matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: six in c:\zl\work\anaconda\anaconda\lib\site-packages (from cycler>=0.10->m
atplotlib>=3.0.0->mlxtend) (1.16.0)
Requirement already satisfied: pytz>=2017.3 in c:\zl\work\anaconda\anaconda\lib\site-packages (from pandas
>=0.24.2->mlxtend) (2021.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\zl\work\anaconda\anaconda\lib\site-packages (fro
m scikit-learn>=1.0.2->mlxtend) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [141]:

```python
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS
data=pd.read_csv(data_dir2)
X=data[['total sulfur dioxide','chlorides','residual sugar','quality','alcohol','free sulfur dioxide']]
y=data[['pH']]
X_train,X_test,y_train,y_test=train_test_split(X,y)
knn = KNeighborsClassifier(n_neighbors=3)
efs1 = EFS(knn,
           min_features=2,
           max_features=4,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs1 = efs1.fit(X_train, y_train, custom_feature_names=X.columns)

print('Best accuracy score: %.2f' % efs1.best_score_)
print('Best subset (indices):', efs1.best_idx_)
print('Best subset (corresponding names):', efs1.best_feature_names_)
```

```
Below are more details about the failures:
--------------------------------------------------------------------------------
5 fits failed with the following error:
Traceback (most recent call last):
  File "C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\model_selection\_validation.py", line 686,
in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py", line 200, in
fit
    return self._fit(X, y)
  File "C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\neighbors\_base.py", line 429, in _fit
    check_classification_targets(y)
  File "C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\utils\multiclass.py", line 200, in check_cl
assification_targets
    raise ValueError("Unknown label type: %r" % y_type)
ValueError: Unknown label type: 'continuous'
```

Отбор признаков из группы методов вложения (линейная регрессия)

```
In [137]: from sklearn.linear_model import Lasso
          # Используем L1-регуляризацию
          e_ls1 = Lasso(random_state=1)
          e_ls1.fit(X_train, y_train)
          # Коэффициенты регрессии
          list(zip(X_train.columns, e_ls1.coef_))

Out[137]: [('volatile acidity', 0.0),
           ('residual sugar', -0.0),
           ('chlorides', -0.0),
           ('free sulfur dioxide', 0.0),
           ('total sulfur dioxide', -0.0),
           ('density', -0.0)]

In [138]: from sklearn.feature_selection import SelectFromModel
          sel_e_ls1 = SelectFromModel(e_ls1)
          sel_e_ls1.fit(X_train, y_train)
          list(zip(X_train.columns, sel_e_ls1.get_support()))

Out[138]: [('volatile acidity', False),
           ('residual sugar', False),
           ('chlorides', False),
           ('free sulfur dioxide', False),
           ('total sulfur dioxide', False),
           ('density', False)]

In [ ]:
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_KNN (дата обращения: 05.04.2019).

[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/stable/ (online; accessed: 20.02.2019).

[3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2019).

[4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).

[5] dronio. Solar Radiation Prediction [Electronic resource] // Kaggle. — 2017. — Access mode: https://www.kaggle.com/dronio/SolarEnergy (online; accessed: 18.02.2019).

[6] Chrétien M. Convert datetime.time to seconds [Electronic resource] // Stack Overflow. — 2017. — Access mode: https://stackoverflow.com/a/44823381 (online; accessed: 20.02.2019).

[7] scikit-learn 0.20.3 documentation [Electronic resource]. — 2019. — Access mode: https://scikit-learn.org/ (online; accessed: 05.04.2019).