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.

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import LinearSVC
from sklearn.feature_selection import SelectFromModel
from category_encoders.count import CountEncoder as ce_CountEncoder
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
```

In [12]:
```
raw_data = pd.read_csv(r'C:\Users\80667\Desktop\文件\IMY5\研一下\MMO\数据集\StudentsPerformance.csv', sep=',')
```

In [13]:
```
raw_data.head()
```

Out[13]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75

In [14]:
```
raw_data.dtypes
```

Out[14]:
```
gender                       object
race/ethnicity               object
parental level of education  object
lunch                        object
test preparation course      object
math score                    int64
reading score                 int64
writing score                 int64
dtype: object
```

In [15]:
```
raw_data_with_na = [c for c in raw_data.columns if raw_data[c].isnull().sum() > 0]
[(c, raw_data[c].isnull().sum()) for c in raw_data_with_na]
```

Out[15]:
```
[]
```

In [16]:
```
raw_data = raw_data.dropna()
```

In [17]:
```
raw_data_with_na = [c for c in raw_data.columns if raw_data[c].isnull().sum() > 0]
[(c, raw_data[c].isnull().sum()) for c in raw_data_with_na]
```

Out[17]:
```
[]
```

In [18]:
```
raw_data.describe()
```

Out[18]:

	math score	reading score	writing score
count	1000.00000	1000.000000	1000.000000
mean	66.08900	69.169000	68.054000
std	15.16308	14.600192	15.195657
min	0.00000	17.000000	10.000000
25%	57.00000	59.000000	57.750000
50%	66.00000	70.000000	69.000000
75%	77.00000	79.000000	79.000000
max	100.00000	100.000000	100.000000

In [19]:
```
# Построение плотности распределения
def draw_kde(col_list, df1, df2, label1, label2):
    fig, (ax1, ax2) = plt.subplots(
        ncols=2, figsize=(12, 5))
    # первый график
    ax1.set_title(label1)
    sns.kdeplot(data=df1[col_list], ax=ax1)
    # второй график
    ax2.set_title(label2)
    sns.kdeplot(data=df2[col_list], ax=ax2)
    plt.show()
```

Масштабирование признаков

In [21]:
```
data_to_sc = raw_data[['math score', 'reading score']]
data_to_sc
```

Out[21]:

	reading score	math score
0	72	72
1	90	69
2	95	90
3	57	47
4	78	76
...	...	...
995	99	88
996	55	62
997	71	59
998	78	68
999	86	77

1000 rows × 2 columns

In [22]:
```
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=data_to_sc.columns)
    return res
```

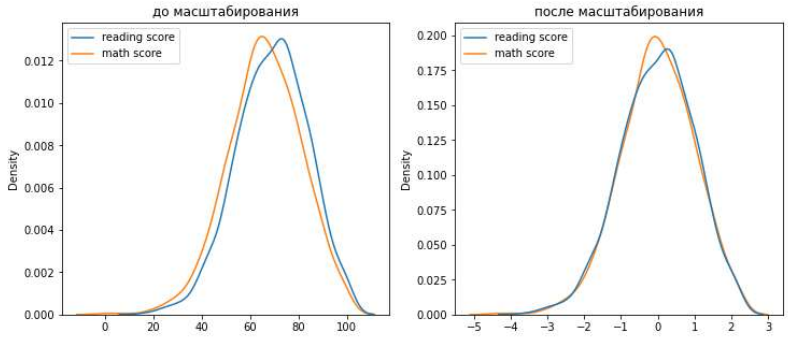
In [23]:
```
#Масштабирование данных на основе Z-оценки
cs1 = StandardScaler()
data_cs1_scaled_temp = cs1.fit_transform(data_to_sc)
data_cs1_scaled =  arr_to_df(data_cs1_scaled_temp)
data_cs1_scaled
```

Out[23]:

	reading score	math score
0	0.193999	0.390024
1	1.427476	0.192076
2	1.770109	1.577711
3	-0.833899	-1.259543
4	0.605158	0.653954
...	...	...
995	2.044215	1.445746
996	-0.970952	-0.269803
997	0.125472	-0.467751
998	0.605158	0.126093
999	1.153370	0.719937

1000 rows × 2 columns

In [26]:
```
draw_kde(['reading score', 'math score'], data_to_sc, data_cs1_scaled,'до масштабирования','после масштабирова
```



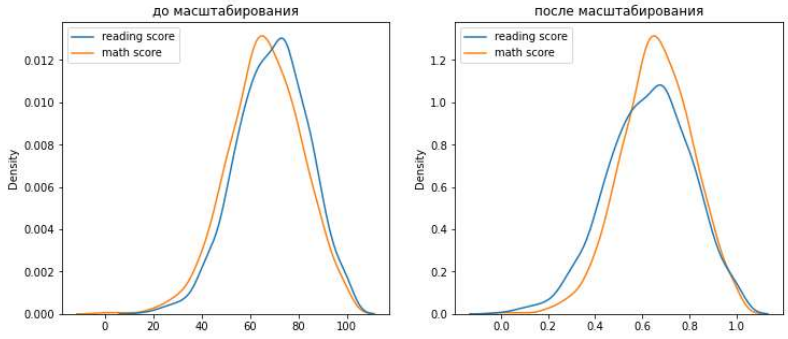
In [27]:
```
#MinMax-масштабирование
cs2 = MinMaxScaler()
data_cs2_scaled_temp = cs2.fit_transform(data_to_sc)
data_cs2_scaled =  arr_to_df(data_cs2_scaled_temp)
data_cs2_scaled
```

Out[27]:

	reading score	math score
0	0.662651	0.72
1	0.879518	0.69
2	0.939759	0.90
3	0.481928	0.47
4	0.734940	0.76
...	...	...
995	0.987952	0.88
996	0.457831	0.62
997	0.650602	0.59
998	0.734940	0.68
999	0.831325	0.77

1000 rows × 2 columns

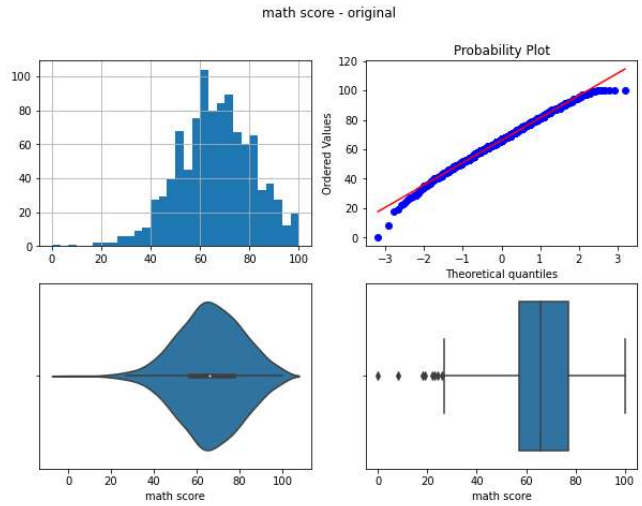
In [48]:
```
draw_kde(['reading score', 'math score'], data_to_sc, data_cs2_scaled, 'до масштабирования', 'после масштабиро
```



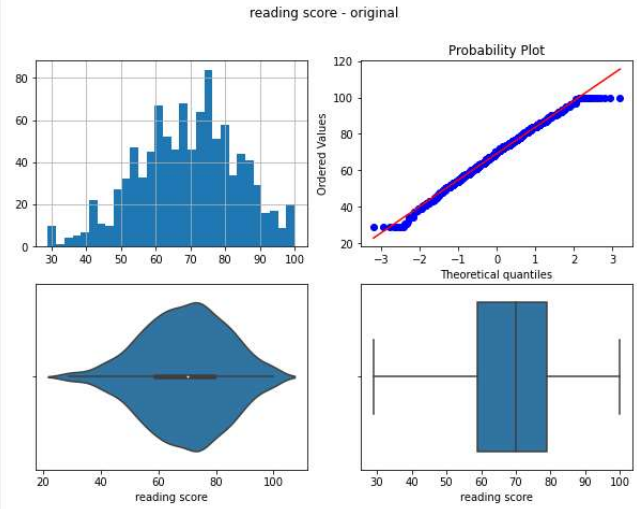
Обработка выбросов

In [49]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    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)
    # ящик с усами
    plt.subplot(2, 2, 3)
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    fig.suptitle(title)
    plt.show()
```

In [50]:
```
diagnostic_plots(data_to_sc, 'math score', 'math score - original')
```

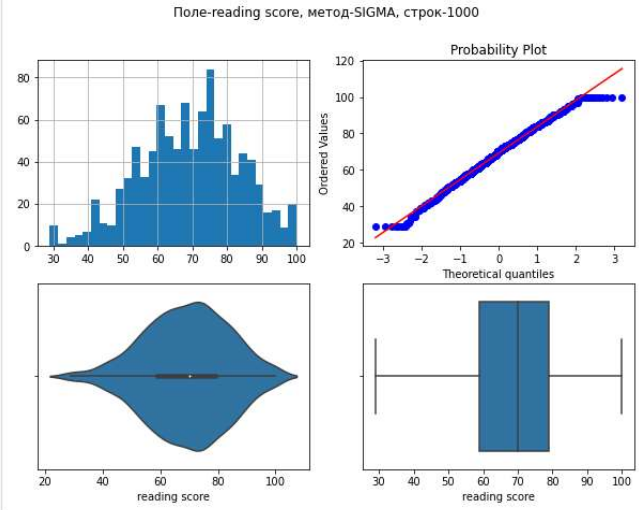


In [51]:
```
diagnostic_plots(raw_data, 'reading score', 'reading score - original')
```



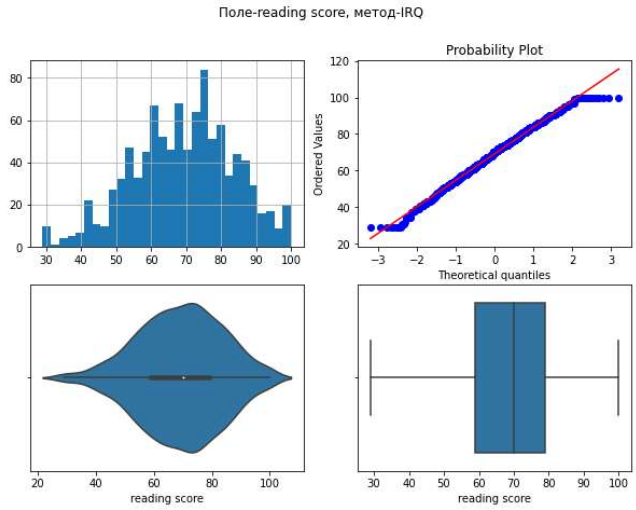
In [52]:
```
#Удаление выбросов методом SIGMA
def del_sigma(data, col):
    K1 = 3
    lower_boundary = data[col].mean() - (K1 * data[col].std())
    upper_boundary = data[col].mean() + (K1 * data[col].std())

    outliers_temp = np.where(data[col] > upper_boundary, True,
                             np.where(data[col] < lower_boundary, True, False))
    data_trimmed = data.loc[~(outliers_temp), ]
    title = 'Поле-{}, Метод-{}, Строк-{}'.format(col, 'SIGMA', data_trimmed.shape[0])
    diagnostic_plots(data_trimmed, col, title)
```

In [53]:
```
del_sigma(raw_data, 'reading score')
```



In [54]:
```
#Замена выбросов
def repl_IRQ(data, col):
    K2 = 1.5
    IQR = data[col].quantile(0.75) - data[col].quantile(0.25)
    lower_boundary = data[col].quantile(0.25) - (K2 * IQR)
    upper_boundary = data[col].quantile(0.75) + (K2 * IQR)
    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, 'IRQ')
    diagnostic_plots(data, col, title)
```

In [55]:
```
repl_IRQ(raw_data, 'reading score')
```



Обработка нестандартного признака

In [71]:
```
raw_data = pd.read_csv(r'C:\Users\80667\Desktop\文件\ИУ5\研一下\MMO\lab\lab3\k_data.csv', sep=',')
```

In [72]:
```
raw_data.head()
```

Out[72]:

	Common.Name	Date	Time	n.observers	County	Sub.cell	Season	DEM	Cell.ID	List.ID
0	Asian Koel	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
1	Black-rumped Flameback	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
2	Black Drongo	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
3	Brahminy Kite	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
4	Common Myna	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1

In [73]:
```
#Обработка Времени
raw_time = raw_data[['Time']]
raw_time
```

Out[73]:

	Time
0	16:30
1	16:30
2	16:30
3	16:30
4	16:30
...	...
300877	10:30
300878	9:05
300879	10:48
300880	10:15
300881	10:00

300882 rows × 1 columns

In [74]:
```
p_time = raw_time
p_time['hour'] = pd.to_datetime(p_time['Time'], format='%H:%M').dt.hour
p_time['minute'] = pd.to_datetime(p_time['Time'], format='%H:%M').dt.minute
p_time
```

```
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/1305754206.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['hour'] = pd.to_datetime(p_time['Time'], format='%H:%M').dt.hour
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/1305754206.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['minute'] = pd.to_datetime(p_time['Time'], format='%H:%M').dt.minute
```

Out[74]:

	Time	hour	minute
0	16:30	16	30
1	16:30	16	30
2	16:30	16	30
3	16:30	16	30
4	16:30	16	30
...	...	...	...
300877	10:30	10	30
300878	9:05	9	5
300879	10:48	10	48
300880	10:15	10	15
300881	10:00	10	0

300882 rows × 3 columns

In [75]:
```
def round_code(v, T, cos_flag = True):
    x = 2*np.pi*v/T
    if cos_flag:
        return np.cos(x)
    else:
        return np.sin(x)
```

In [75]:
```
def round_code(v, T, cos_flag = True):
    x = 2*np.pi*v/T
    if cos_flag:
        return np.cos(x)
    else:
        return np.sin(x)
```

In [76]:
```
p_time['hour_cos'] = p_time.apply(lambda x: round_code(x['hour'], 24), axis=1)
p_time['hour_sin'] = p_time.apply(lambda x: round_code(x['hour'], 24, False), axis=1)
p_time['minute_cos'] = p_time.apply(lambda x: round_code(x['minute'], 60), axis=1)
p_time['minute_sin'] = p_time.apply(lambda x: round_code(x['minute'], 60, False), axis=1)
p_time
```

```
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/3558441377.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['hour_cos'] = p_time.apply(lambda x: round_code(x['hour'], 24), axis=1)
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/3558441377.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['hour_sin'] = p_time.apply(lambda x: round_code(x['hour'], 24, False), axis=1)
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/3558441377.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['minute_cos'] = p_time.apply(lambda x: round_code(x['minute'], 60), axis=1)
C:\Users\80667\AppData\Local\Temp/ipykernel_31756/3558441377.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  p_time['minute_sin'] = p_time.apply(lambda x: round_code(x['minute'], 60, False), axis=1)
```

Out[76]:

	Time	hour	minute	hour_cos	hour_sin	minute_cos	minute_sin
0	16:30	16	30	-0.500000	-0.866025	-1.000000e+00	5.665539e-16
1	16:30	16	30	-0.500000	-0.866025	-1.000000e+00	5.665539e-16
2	16:30	16	30	-0.500000	-0.866025	-1.000000e+00	5.665539e-16
3	16:30	16	30	-0.500000	-0.866025	-1.000000e+00	5.665539e-16
4	16:30	16	30	-0.500000	-0.866025	-1.000000e+00	5.665539e-16
...	...	...	...	...	...	...	...
300877	10:30	10	30	-0.866025	0.500000	-1.000000e+00	5.665539e-16
300878	9:05	9	5	-0.707107	0.707107	8.660254e-01	5.000000e-01
300879	10:48	10	48	-0.866025	0.500000	3.090170e-01	-9.510565e-01
300880	10:15	10	15	-0.866025	0.500000	2.832769e-16	1.000000e+00
300881	10:00	10	0	-0.866025	0.500000	1.000000e+00	0.000000e+00

300882 rows × 7 columns

Отбор признаков

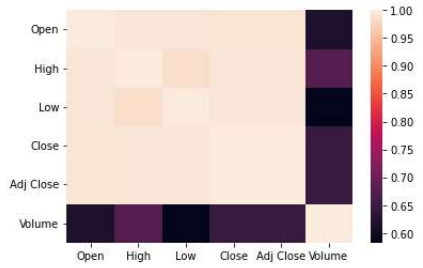
In [78]:
```
#Методы, основанные на корреляции
fs_data = pd.read_csv(r'C:\Users\80667\Desktop\文件\川大5\研一下\MMO\lab\lab3\DOGE-USD.csv', sep=',')
fs_data.head()
```

Out[78]:

	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 [79]:
```
sns.heatmap(fs_data.corr(), annot=False, fmt='.3f')
```

Out[79]: <AxesSubplot:>



In [80]:
```
cr = fs_data.corr()
cr = cr.abs().unstack()
cr = cr.sort_values(ascending=False)
cr = cr[cr >= 0.8]
cr = cr[cr < 1]
cr = pd.DataFrame(cr).reset_index()
cr.columns = ['f1', 'f2', 'corr']
cr
```

Out[80]:

	f1	f2	corr
0	High	Adj Close	0.994930
1	Close	High	0.994930
2	Adj Close	High	0.994930
3	High	Close	0.994930
4	Low	Close	0.994614
5	Low	Adj Close	0.994614
6	Adj Close	Low	0.994614
7	Close	Low	0.994614
8	High	Open	0.993979
9	Open	High	0.993979
10	Open	Low	0.993535
11	Low	Open	0.993535
12	Open	Adj Close	0.992134
13	Adj Close	Open	0.992134
14	Open	Close	0.992134
15	Close	Open	0.992134
16	Low	High	0.986583
17	High	Low	0.986583

In [83]:
```
#Метод обратный Sequential Feature Selector (Методы обертывания)

fs2_data = pd.read_csv(r'C:\Users\80667\Desktop\文件\ИУ5\研一下\MMO\lab\lab3\k_data.csv', sep=',')
fs2_data.head()
```

Out[83]:

	Common.Name	Date	Time	n.observers	County	Sub.cell	Season	DEM	Cell.ID	List.ID
0	Asian Koel	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
1	Black-rumped Flameback	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
2	Black Drongo	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
3	Brahminy Kite	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1
4	Common Myna	7/16/2015	16:30	2.0	Alappuzha	[51,2,2]	Wet	5.0	[76.28,9.84]	List.1

In [42]:
```
raw_data_with_na = [c for c in fs2_data.columns if fs2_data[c].isnull().sum() > 0]
[(c, fs2_data[c].isnull().sum()) for c in raw_data_with_na]
```

Out[42]: []

In [43]:
```
fs2_data = fs2_data.dropna()
```

In [44]:
```
#Кодируем категориальные признаки
CE1 = ce_CountEncoder()
encoded_data = CE1.fit_transform(fs2_data[fs2_data.columns])
encoded_data
```

```
Warning: No categorical columns found. Calling 'transform' will only return input data.
```

Out[44]:

	Year	Population	Yearly % Change	Yearly Change	Migrants (net)	Median Age	Fertility Rate	Density (P/Km²)	Urban Pop %	Urban Population	Country's Share of World Pop	World Population	India Global Rank
0	2020	1380004385	0.99	13586631	-532687	28.4	2.24	464	35.0	483098640	17.70	7794798739	2
1	2019	1366417754	1.02	13775474	-532687	27.1	2.36	460	34.5	471828295	17.71	7713468100	2
2	2018	1352642280	1.04	13965495	-532687	27.1	2.36	455	34.1	460779764	17.73	7631091040	2
3	2017	1338676785	1.07	14159536	-532687	27.1	2.36	450	33.6	449963381	17.74	7547858925	2
4	2016	1324517249	1.10	14364846	-532687	27.1	2.36	445	33.2	439391699	17.75	7464022049	2
5	2015	1310152403	1.20	15174247	-470015	26.8	2.40	441	32.7	429069459	17.75	7379797139	2
6	2010	1234281170	1.47	17334249	-531169	25.1	2.80	415	30.8	380744554	17.74	6956823603	2
7	2005	1147609927	1.67	18206876	-377797	23.8	3.14	386	29.1	334479406	17.54	6541907027	2
8	2000	1056575549	1.85	18530592	-136514	22.7	3.48	355	27.6	291350282	17.20	6143493823	2
9	1995	963922588	1.99	18128958	-110590	21.8	3.83	324	26.5	255558824	16.78	5744212979	2
10	1990	873277798	2.17	17783558	9030	21.1	4.27	294	25.5	222296728	16.39	5327231061	2
11	1985	784360008	2.33	17081433	115942	20.6	4.68	264	24.3	190321782	16.10	4870921740	2
12	1980	698952844	2.32	15169989	222247	20.2	4.97	235	23.0	160941941	15.68	4458003514	2
13	1975	623102897	2.33	13582621	421208	19.7	5.41	210	21.3	132533810	15.27	4079480606	2
14	1970	555189792	2.15	11213294	-68569	19.3	5.72	187	19.7	109388950	15.00	3700437046	2
15	1965	499123324	2.07	9715129	-17078	19.6	5.89	168	18.7	93493844	14.95	3339583597	2
16	1960	450547679	1.91	8133417	-30805	20.2	5.90	152	17.9	80565723	14.85	3034949748	2
17	1955	409880595	1.72	6711079	-21140	20.7	5.90	138	17.6	71958495	14.78	2773019936	2

In [45]:
```
X = encoded_data.drop('Density (P/Km²)', axis=1)
Y = encoded_data['Density (P/Km²)']
```

In [46]:
```
knn = KNeighborsClassifier(n_neighbors=3)

sfs1 = SFS(knn,
           k_features=3,
           forward=True,
           floating=False,
           verbose=2,
           scoring='accuracy',
           cv=0)

sfs1 = sfs1.fit(X, Y)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s finished

[2022-06-09 00:09:09] Features: 1/3 -- score: 0.3333333333333333[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s finished

[2022-06-09 00:09:09] Features: 2/3 -- score: 0.3333333333333333[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent work
ers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s finished

[2022-06-09 00:09:09] Features: 3/3 -- score: 0.3333333333333333
```

In [47]:
```
sfs1.k_feature_names_
```

Out[47]:
```
('Year', 'Yearly % Change', 'Yearly Change')
```

In [48]:
```
#Линейный классификатор на основе SVM (Методы вложений)

e_lr2 = LinearSVC(C=0.01, penalty="l1", max_iter=2000, dual=False)
e_lr2.fit(X, Y)
# Коэффициенты регрессии
e_lr2.coef_
```

```
C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\svm\_base.py:1225: ConvergenceWarning: Liblinear failed to converge, increase the num
ber of iterations.
  warnings.warn(
```

Out[48]:
```
array([[ 2.69459308e-03,  8.29659277e-09,  0.00000000e+00,
        -1.50767581e-06,  1.29966431e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -9.30788654e-09,
         0.00000000e+00,  1.16380121e-09,  0.00000000e+00],
       [ 4.56395119e-04,  7.99228789e-09,  0.00000000e+00,
         5.88737679e-07, -1.80162982e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -1.59333795e-07,
         0.00000000e+00,  9.46213273e-10,  0.00000000e+00],
       [ 4.91750196e-06, -1.16150536e-09,  0.00000000e+00,
         8.12416221e-07, -9.84968197e-07,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -8.00631289e-08,
         0.00000000e+00, -5.14799852e-11,  0.00000000e+00],
       [-5.78224550e-04, -9.57195522e-09,  0.00000000e+00,
```

```
         5.60270598e-07, -6.05536677e-06,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -1.98097233e-08,
         0.00000000e+00, -9.57694083e-11,  0.00000000e+00],
       [-6.54500500e-03,  4.51426414e-09,  0.00000000e+00,
         3.36349768e-07,  6.99973171e-06,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  1.10291527e-08,
         0.00000000e+00, -1.92834986e-11,  0.00000000e+00],
       [-2.25452947e-03, -1.37828732e-09,  0.00000000e+00,
         6.52611249e-07, -1.04448833e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -2.18810932e-08,
         0.00000000e+00, -3.77338798e-10,  0.00000000e+00],
       [ 0.00000000e+00, -3.83963604e-09,  0.00000000e+00,
         1.84384061e-07, -2.20455835e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -1.48608670e-08,
         0.00000000e+00, -5.05343180e-10,  0.00000000e+00],
       [-5.54252147e-04, -3.85192449e-10,  0.00000000e+00,
         8.06978585e-10,  1.40493787e-07,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  2.14807873e-09,
         0.00000000e+00, -2.91811007e-12,  0.00000000e+00],
       [ 0.00000000e+00,  1.64649099e-09,  0.00000000e+00,
        -2.77614553e-07, -1.49709912e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -1.22885322e-08,
         0.00000000e+00, -1.73146438e-10,  0.00000000e+00],
       [ 0.00000000e+00,  9.17952937e-10,  0.00000000e+00,
        -2.38002154e-07, -1.00752934e-05,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -8.81609266e-09,
         0.00000000e+00,  1.99990439e-11,  0.00000000e+00],
       [ 0.00000000e+00, -9.03242074e-11,  0.00000000e+00,
        -1.51901199e-07, -4.26491495e-06,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00, -1.51527600e-09,
         0.00000000e+00,  9.09598766e-12,  0.00000000e+00],
       [ 0.00000000e+00,  7.06465951e-10,  0.00000000e+00,
        -3.84415182e-07,  5.88110555e-07,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  1.07579679e-08,
         0.00000000e+00, -1.08946280e-10,  0.00000000e+00],
       [ 0.00000000e+00,  2.61962899e-09,  0.00000000e+00,
        -1.34318378e-06,  7.95379272e-06,  0.00000000e+00,
         0.00000000e+00,  0.00000000e+00,  3.61636258e-08,
         0.00000000e+00,  2.25554937e-10,  0.00000000e+00]])
```

In [49]:
```
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(X,Y)
sel_e_lr2.get_support()
```

```
C:\ZL\Work\Anaconda\Anaconda\lib\site-packages\sklearn\svm\_base.py:1225: ConvergenceWarning: Liblinear failed to converge, increase the num
ber of iterations.
  warnings.warn(
```

Out[49]:
```
array([ True, False, False,  True,  True, False, False, False, False,
       False, False, 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).