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

Отчет
Лабораторная работа № 3
По курсу «Методы машинного обучения»

Обработка признаков часть 3

ИСПОЛНИТЕЛЬ:
Попов Илья Андреевич
Группа ИУ5-23М

____

"__"___________2022 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

____

"__"___________2022 г.

Москва 2022

Задание

1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
   i. масштабирование признаков (не менее чем тремя способами);
   ii. обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
   iii. обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
   iv. отбор признаков:
      - один метод из группы методов фильтрации (filter methods);
      - один метод из группы методов обертывания (wrapper methods);
      - один метод из группы методов вложений (embedded methods).

Выполнение

Popov I.A. IU5-23M lab3

In [62]:
```
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 [3]:
```
raw_data = pd.read_csv('kba_data.csv', sep=',')
```

In [4]:
```
raw_data.head()
```

Out[4]:

	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 [5]:
```
raw_data.dtypes
```

Out[5]:
```
Common.Name     object
Date            object
Time            object
n.observers    float64
County          object
Sub.cell        object
Season          object
DEM            float64
Cell.ID         object
List.ID         object
dtype: object
```

In [6]:
```
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[6]:
```
[('Sub.cell', 127), ('Cell.ID', 127), ('List.ID', 127)]
```

In [7]:
```
raw_data = raw_data.dropna()
```

In [8]:
```
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[8]:
```
[]
```

In [9]:
```
raw_data.describe()
```

Out[9]:

	n.observers	DEM
count	300755.000000	300755.000000
mean	2.351103	225.631464
std	1.024697	346.708288
min	1.000000	0.000000
25%	2.000000	26.000000
50%	2.000000	69.000000
75%	3.000000	203.000000
max	15.000000	2410.000000

In [10]:
```python
# 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 [11]:
```python
data_to_sc = raw_data[{'n.observers', 'DEM'}]
data_to_sc
```

Out[11]:

	DEM	n.observers
0	5.0	2.0
1	5.0	2.0
2	5.0	2.0
3	5.0	2.0
4	5.0	2.0
...	...	...
300750	860.0	4.0
300751	860.0	4.0
300752	860.0	4.0
300753	860.0	4.0
300754	860.0	4.0

300755 rows × 2 columns

In [12]:
```python
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=data_to_sc.columns)
    return res
```

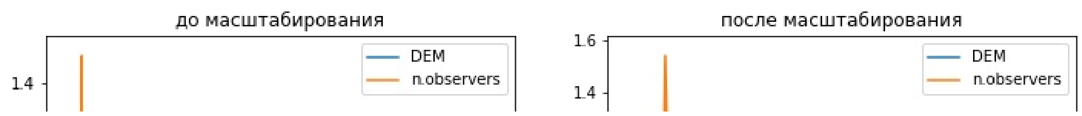
In [13]:
```python
#Масштабирование данных на основе Z-оценки
cs1 = StandardScaler()
data_cs1_scaled_temp = cs1.fit_transform(data_to_sc)
data_cs1_scaled =  arr_to_df(data_cs1_scaled_temp)
data_cs1_scaled
```

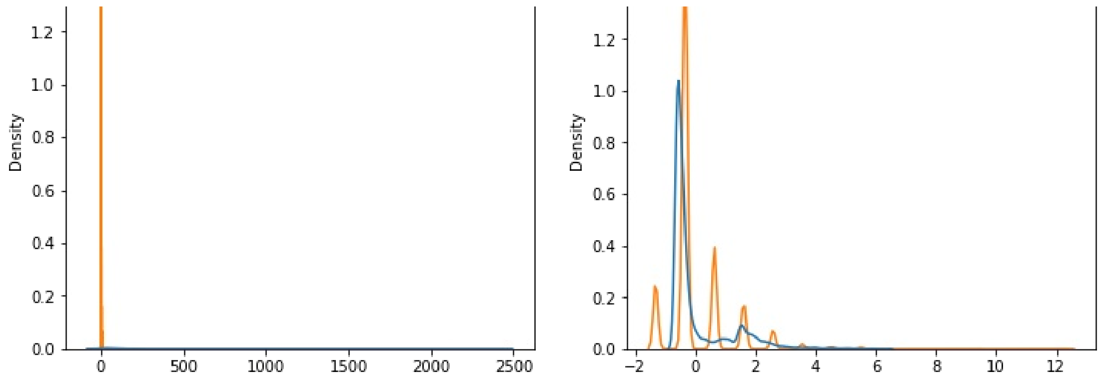
Out[13]:

	DEM	n.observers
0	-0.636362	-0.342641
1	-0.636362	-0.342641
2	-0.636362	-0.342641
3	-0.636362	-0.342641
4	-0.636362	-0.342641
...	...	...
300750	1.829693	1.609158
300751	1.829693	1.609158
300752	1.829693	1.609158
300753	1.829693	1.609158
300754	1.829693	1.609158

300755 rows × 2 columns

In [14]:
```python
draw_kde(['DEM', 'n.observers'], data_to_sc, data_cs1_scaled, 'до масштабирования', 'после масштабирования')
```





In [15]:

```
#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
```

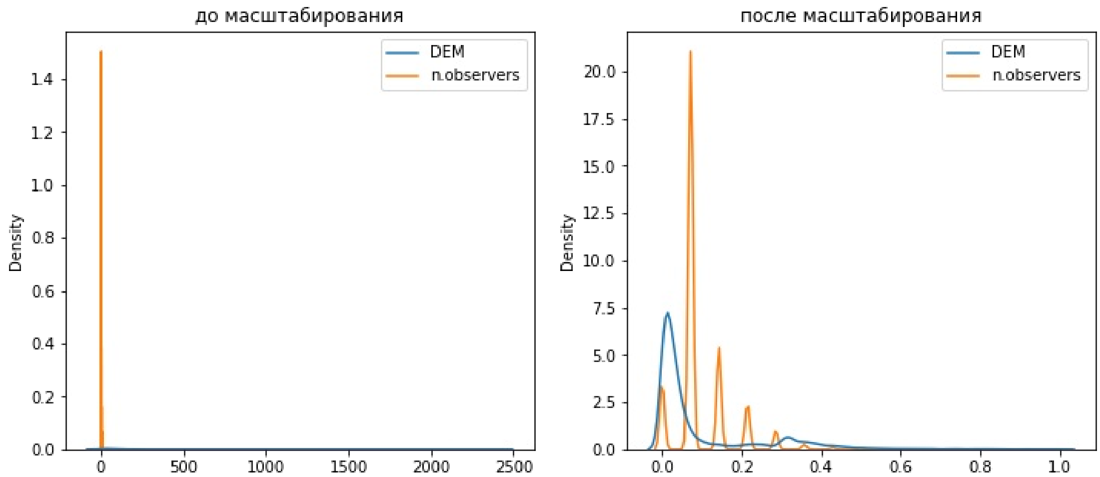
Out[15]:

	DEM	n.observers
0	0.002075	0.071429
1	0.002075	0.071429
2	0.002075	0.071429
3	0.002075	0.071429
4	0.002075	0.071429
...	...	...
300750	0.356846	0.214286
300751	0.356846	0.214286
300752	0.356846	0.214286
300753	0.356846	0.214286
300754	0.356846	0.214286

300755 rows × 2 columns

In [16]:

```
draw_kde(['DEM', 'n.observers'], data_to_sc, data_cs2_scaled, 'до масштабирования', 'после масштабирования')
```



In [17]:

```
#Масштабирование по медиане
cs3 = RobustScaler()
data_cs3_scaled_temp = cs3.fit_transform(data_to_sc)
data_cs3_scaled =  arr_to_df(data_cs3_scaled_temp)
data_cs3_scaled
```

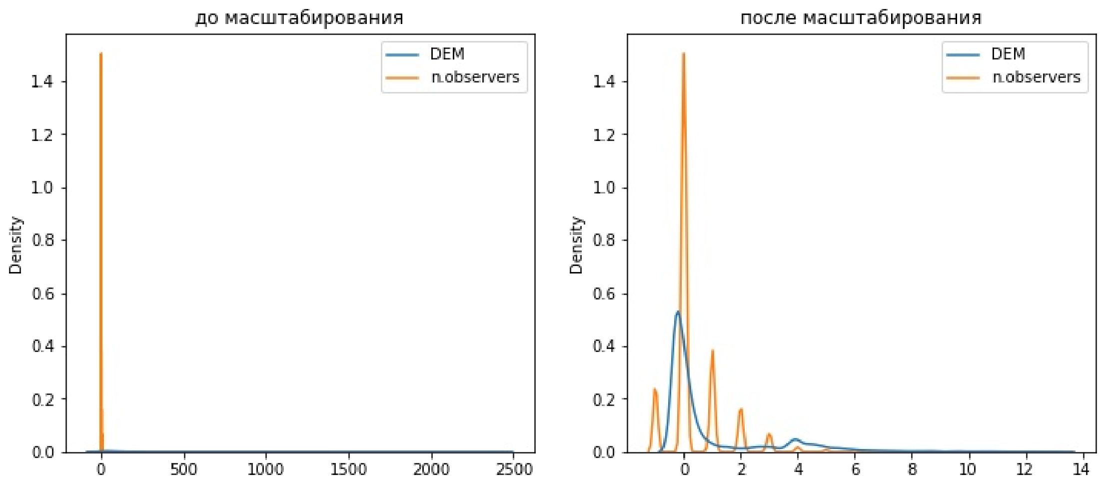
Out[17]:

	DEM	n.observers
0	-0.361582	0.0
1	-0.361582	0.0
2	-0.361582	0.0
3	-0.361582	0.0

4	-0.361582	0.0
...	...	...
300750	4.468927	2.0
300751	4.468927	2.0
300752	4.468927	2.0
300753	4.468927	2.0
300754	4.468927	2.0

300755 rows × 2 columns

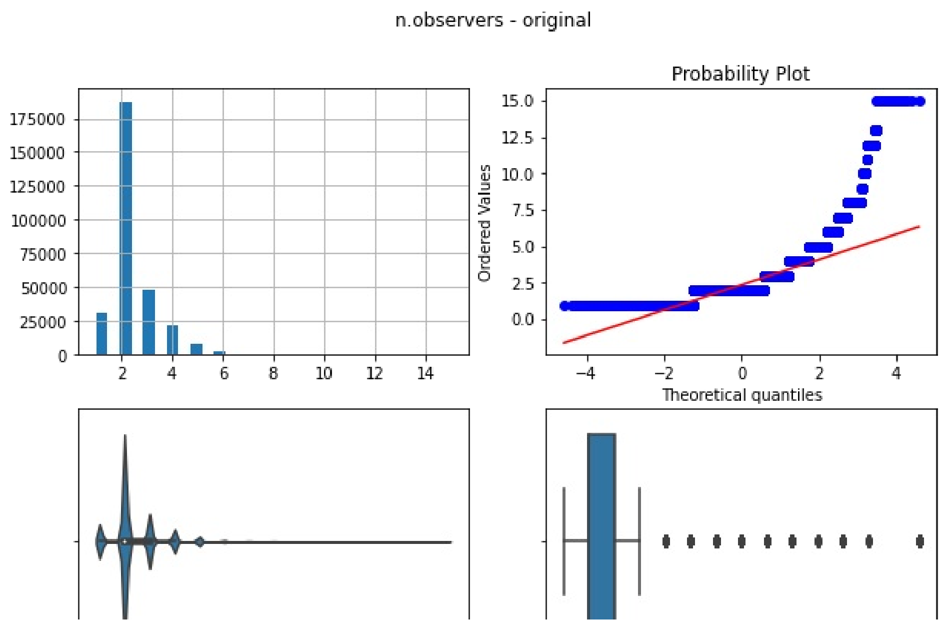
In [18]:
```
draw_kde(['DEM', 'n.observers'], data_to_sc, data_cs3_scaled, 'до масштабирования', 'после масштабирования')
```

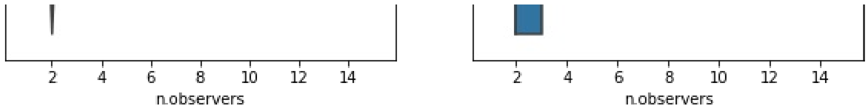


Обработка выбросов

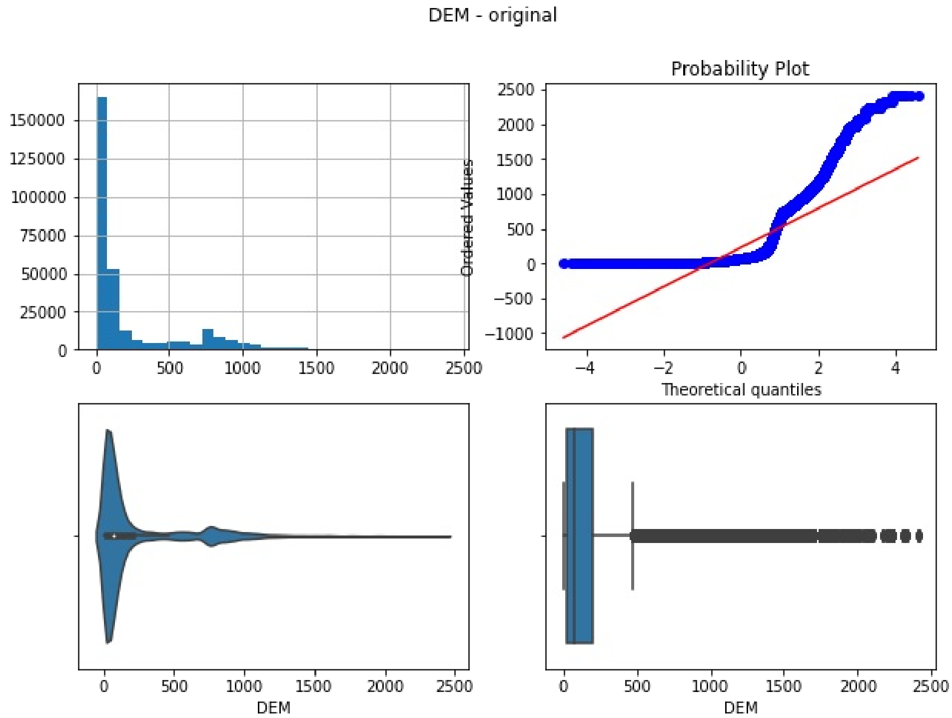
In [19]:
```
def diagnostic_plots(df, variable, title):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 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 [20]:
```
diagnostic_plots(data_to_sc, 'n.observers', 'n.observers - original')
```



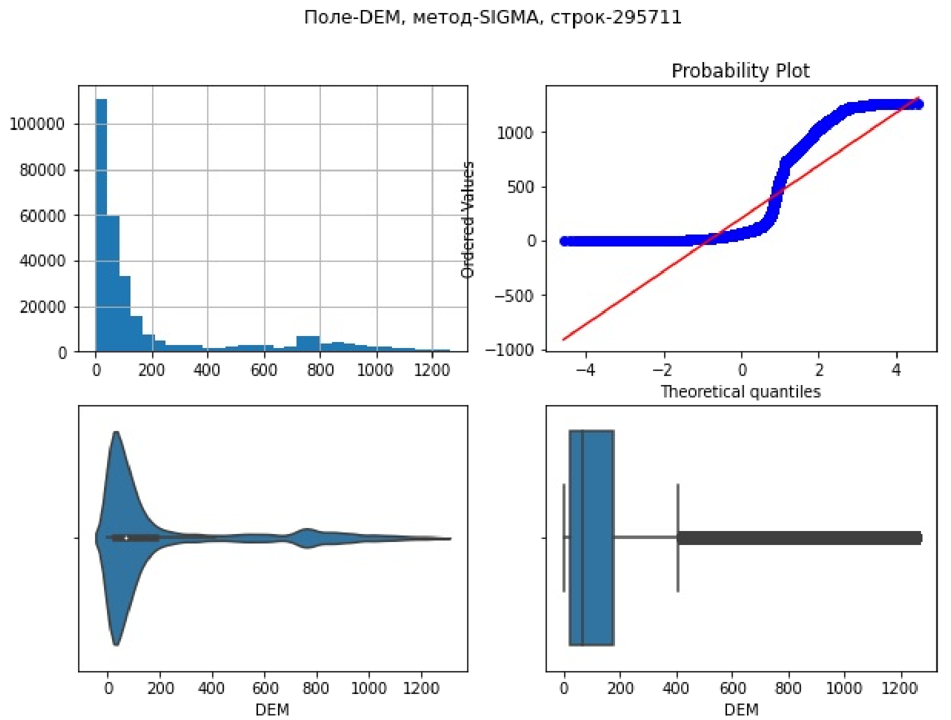


In [21]:
```
diagnostic_plots(raw_data, 'DEM', 'DEM - original')
```



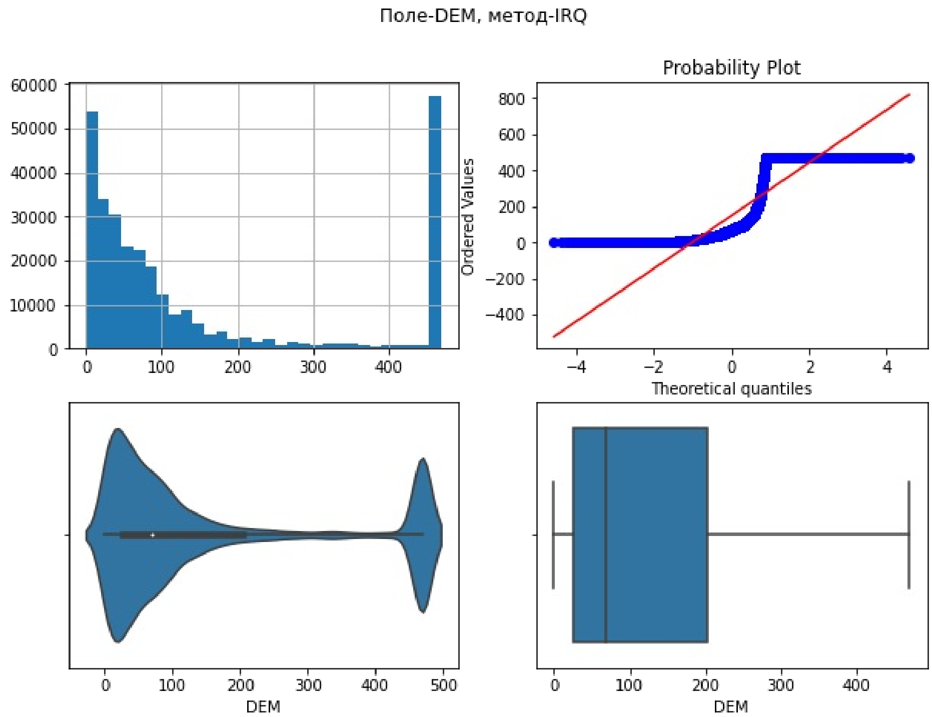
In [22]:
```
#Удаление выбросов методом SIGMA
def del_sigma(data, col):
    K1 = 3
    lower_boundary = data[col].mean() - (K1 * data[col].std())
    upper_boundary = data[col].mean() + (K1 * data[col].std())

    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 [23]:
```
del_sigma(raw_data, 'DEM')
```



In [24]:
```
#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 [25]:
```
repl_IRQ(raw_data, 'DEM')
```



Обработка нестандартного признака

In [26]:
```
#Обработка времени
raw_time = raw_data[{'Time'}]
raw_time
```

Out[26]:

	Time
0	16:30
1	16:30
2	16:30
3	16:30
4	16:30
...	...
300750	7:17
300751	7:17
300752	7:17
300753	7:17
300754	7:17

300755 rows × 1 columns

In [27]:
```
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\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy

c:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[27]:

	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
...	...	...	...
300750	7:17	7	17
300751	7:17	7	17
300752	7:17	7	17
300753	7:17	7	17
300754	7:17	7	17

300755 rows × 3 columns

In [28]:

```
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 [29]:

```
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\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
c:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy

c:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports until
c:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopy
Warning:
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#ret
urning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

Out[29]:

	Time	hour	minute	hour_cos	hour_sin	minute_cos	minute_sin
0	16:30	16	30	-0.500000	-0.866025	-1.000000	5.665539e-16
1	16:30	16	30	-0.500000	-0.866025	-1.000000	5.665539e-16
2	16:30	16	30	-0.500000	-0.866025	-1.000000	5.665539e-16
3	16:30	16	30	-0.500000	-0.866025	-1.000000	5.665539e-16
4	16:30	16	30	-0.500000	-0.866025	-1.000000	5.665539e-16
...	...	...	...	...	...	...	...
300750	7:17	7	17	-0.258819	0.965926	-0.207912	9.781476e-01
300751	7:17	7	17	-0.258819	0.965926	-0.207912	9.781476e-01
300752	7:17	7	17	-0.258819	0.965926	-0.207912	9.781476e-01
300753	7:17	7	17	-0.258819	0.965926	-0.207912	9.781476e-01
300754	7:17	7	17	-0.258819	0.965926	-0.207912	9.781476e-01

300755 rows × 7 columns

Отбор признаков

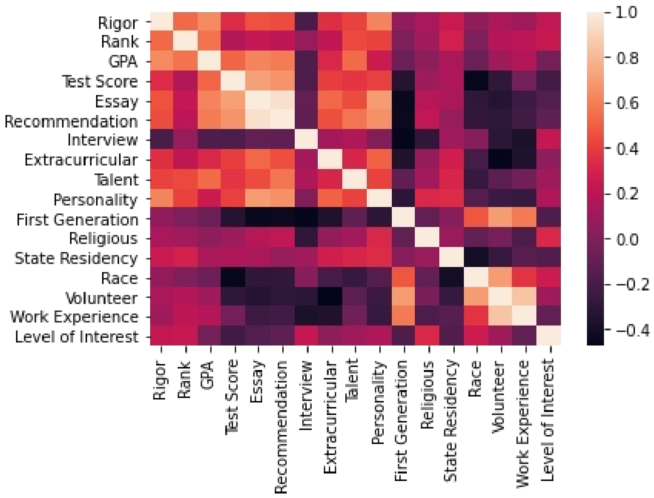
In [30]:
```
#Методы, основанные на корреляции
fs_data = pd.read_csv('Admission_Data.csv', sep=',')
fs_data.head()
```

Out[30]:

	Name of University	Rigor	Rank	GPA	Test Score	Essay	Recommendation	Interview	Extracurricular	Talent	Personality	First Generation	Religious	Res
0	Princeton University	3	3	3	3	3	3	1	3	3	3	1	0.0	
1	Brown University	3	3	3	3	3	3	1	2	3	3	1	0.0	
2	California Institute of Technology (Caltech)	3	2	2	3	3	3	0	2	1	3	1	0.0	
3	Cornell University	3	2	3	3	3	3	1	3	3	3	1	NaN	
4	Dartmouth College	3	3	3	3	3	3	1	3	2	3	0	1.0	

In [31]:
```
sns.heatmap(fs_data.corr(), annot=False, fmt='.3f')
```

Out[31]: <AxesSubplot:>



In [32]:
```
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[32]:

	f1	f2	corr
0	Recommendation	Essay	0.959114
1	Essay	Recommendation	0.959114
2	Volunteer	Work Experience	0.848387
3	Work Experience	Volunteer	0.848387

In [59]:

```
#Метод обратный Sequential Feature Selector (Методы обертывания)

fs2_data = pd.read_csv('wines_SPA.csv', sep=',')
fs2_data.head()
```

Out[59]:

	winery	wine	year	rating	num_reviews	country	region	price	type	body	acidity
0	Teso La Monja	Tinto	2013	4.9	58	Espana	Toro	995.00	Toro Red	5.0	3.0
1	Artadi	Vina El Pison	2018	4.9	31	Espana	Vino de Espana	313.50	Tempranillo	4.0	2.0
2	Vega Sicilia	Unico	2009	4.8	1793	Espana	Ribera del Duero	324.95	Ribera Del Duero Red	5.0	3.0
3	Vega Sicilia	Unico	1999	4.8	1705	Espana	Ribera del Duero	692.96	Ribera Del Duero Red	5.0	3.0
4	Vega Sicilia	Unico	1996	4.8	1309	Espana	Ribera del Duero	778.06	Ribera Del Duero Red	5.0	3.0

In [50]:

```
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[50]:

```
[('year', 2), ('type', 545), ('body', 1169), ('acidity', 1169)]
```

In [52]:

```
fs2_data = fs2_data.dropna()
```

In [53]:

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

Out[53]:

	winery	wine	year	rating	num_reviews	country	region	price	type	body	acidity
0	15	53	58	4.9	58	6329	264	995.00	261	5.0	3.0
1	239	16	752	4.9	31	6329	239	313.50	267	4.0	2.0
2	95	41	39	4.8	1793	6329	1280	324.95	1277	5.0	3.0
3	95	41	10	4.8	1705	6329	1280	692.96	1277	5.0	3.0
4	95	41	11	4.8	1309	6329	1280	778.06	1277	5.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...
7495	414	422	810	4.2	392	6329	2221	19.98	2143	4.0	3.0
7496	201	199	752	4.2	390	6329	622	16.76	620	4.0	3.0
7497	201	200	652	4.2	390	6329	201	24.45	787	4.0	3.0
7498	206	415	1078	4.2	389	6329	1280	64.50	1277	5.0	3.0
7499	204	201	810	4.2	388	6329	1280	31.63	1277	5.0	3.0

6329 rows × 11 columns

In [54]:

```
X = encoded_data.drop('wine', axis=1)
Y = encoded_data['wine']
```

In [55]:

```
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:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    2.7s finished

[2022-06-01 23:02:23] Features: 1/3 -- score: 0.6958445252014537[Parallel(n_jobs=1)]: Using backend SequentialB
ackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.6s finished

[2022-06-01 23:02:25] Features: 2/3 -- score: 0.8996681940274925[Parallel(n_jobs=1)]: Using backend SequentialB
ackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.4s finished

[2022-06-01 23:02:26] Features: 3/3 -- score: 0.9270026860483489
```

In [56]:

```
sfs1.k_feature_names_
```

Out[56]:

```
('winery', 'price', 'type')
```

In [60]:

```
#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:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\sklearn\svm\_base.py:1208: ConvergenceWa
rning: Liblinear failed to converge, increase the number of iterations.
  ConvergenceWarning,
```

Out[60]:

```
array([[-4.14722706e-03, -1.87267540e-05,  0.00000000e+00,
        -9.62723602e-05,  1.22223923e-05, -1.06654764e-04,
        -1.14004799e-03, -3.69213146e-05, -3.70210947e-02,
         0.00000000e+00],
       [-1.97898498e-03,  1.17959306e-04,  0.00000000e+00,
        -2.10716906e-05, -1.10699056e-04, -5.62420368e-05,
        -3.87233829e-04,  7.25856322e-05,  0.00000000e+00,
         0.00000000e+00],
       [-1.33135871e-03,  2.99970309e-05,  0.00000000e+00,
         1.03043963e-05, -1.18424528e-04, -2.79947576e-05,
        -5.95992293e-05,  8.17499446e-06,  0.00000000e+00,
         0.00000000e+00],
       [-8.06972526e-04,  4.52158355e-06,  0.00000000e+00,
        -1.60281193e-06, -1.30094168e-04, -3.23688982e-05,
        -5.37869160e-05,  2.12967257e-05,  0.00000000e+00,
         0.00000000e+00],
       [-8.40528153e-04,  1.61444515e-05,  0.00000000e+00,
        -2.47825654e-06, -1.45886294e-04, -1.18966455e-05,
        -5.40899939e-05,  4.95265144e-05,  0.00000000e+00,
         0.00000000e+00],
       [-4.48969713e-04, -1.50715117e-05,  0.00000000e+00,
         8.80785329e-06, -1.43474930e-04, -1.55120875e-05,
         3.70883928e-05,  3.25527533e-05,  0.00000000e+00,
         0.00000000e+00],
       [-3.09145368e-04, -5.07858242e-05,  0.00000000e+00,
        -1.66936222e-05, -1.48969936e-04, -5.58886784e-06,
        -1.17390445e-05,  3.60335141e-05,  0.00000000e+00,
         0.00000000e+00],
       [-1.98322663e-05, -5.10818189e-05,  0.00000000e+00,
         2.85246756e-05, -1.55926494e-04,  9.19221178e-06,
         7.78473502e-05,  5.60317884e-08,  0.00000000e+00,
         0.00000000e+00],
       [-1.76042940e-04, -9.36123886e-06,  0.00000000e+00,
         1.38040528e-05, -1.51738149e-04,  1.74848496e-07,
        -4.27304720e-05, -3.98140334e-06,  0.00000000e+00,
         0.00000000e+00],
       [-3.92045944e-04, -4.91063502e-05,  0.00000000e+00,
        -1.56693306e-05, -1.75799073e-04,  4.03502559e-05,
        -9.28907035e-05,  5.86673704e-05,  0.00000000e+00,
         0.00000000e+00],
       [-2.79020688e-04, -3.18026677e-05,  0.00000000e+00,
         1.49750933e-06, -1.53467901e-04,  1.01881252e-05,
        -2.43839169e-05,  8.38150948e-06,  0.00000000e+00,
         0.00000000e+00],
       [-1.02201907e-03, -1.87101908e-04,  0.00000000e+00,
         3.07566010e-05, -8.06007156e-04,  1.03365452e-03,
         8.86321461e-04,  8.65998719e-04,  0.00000000e+00,
         0.00000000e+00],
       [-2.37951907e-04, -5.85316433e-05,  0.00000000e+00,
```

```
       -5.17576327e-06, -1.43925680e-04, -2.55347405e-06,
       -2.80491158e-05, -1.82938259e-05,  0.00000000e+00,
        0.00000000e+00],
      [ 3.08266397e-02, -2.82702735e-04,  0.00000000e+00,
       -8.03348919e-04, -7.31670483e-04, -9.17214610e-03,
        2.79149810e-03, -4.23052565e-03,  0.00000000e+00,
        0.00000000e+00],
      [-4.88595998e-03, -7.68569238e-07,  0.00000000e+00,
        6.18026384e-06, -2.44271111e-04,  1.93374391e-04,
        6.58008076e-04,  6.34045552e-05,  0.00000000e+00,
        0.00000000e+00],
      [ 2.12789279e-04,  3.03796600e-06,  0.00000000e+00,
       -1.42203163e-03, -1.48540819e-04,  8.34710020e-06,
        6.04750067e-04, -3.35944334e-04,  0.00000000e+00,
        0.00000000e+00],
      [-2.45004745e-05, -4.19997018e-05,  0.00000000e+00,
        4.65135570e-06, -1.69951317e-04,  5.98621973e-05,
       -9.37500694e-05, -3.98857188e-06,  0.00000000e+00,
        0.00000000e+00],
      [-1.55909644e-04, -2.73627094e-05,  0.00000000e+00,
        2.55500579e-05, -1.62498598e-04,  8.23409006e-06,
        1.71675262e-04,  1.97155680e-06,  1.01376063e-02,
        0.00000000e+00],
      [-3.90579776e-04, -4.69116363e-05,  0.00000000e+00,
       -6.38678100e-08, -1.45444480e-04, -1.46474067e-06,
        3.62039431e-05, -2.47622432e-05,  0.00000000e+00,
        0.00000000e+00],
      [ 5.29870723e-06, -1.77778292e-04,  0.00000000e+00,
        3.47574588e-05, -1.50556170e-04,  1.14481018e-05,
        2.57954108e-05, -1.03181939e-05,  0.00000000e+00,
        0.00000000e+00],
      [ 6.59823326e-04, -6.60094758e-04,  0.00000000e+00,
        2.89128015e-05, -1.64437505e-04,  0.00000000e+00,
        4.42845320e-04,  1.63262738e-05,  0.00000000e+00,
        0.00000000e+00],
      [-5.06717788e-04,  3.75401081e-05,  0.00000000e+00,
        1.00495147e-05, -1.49611348e-04, -4.31767573e-05,
        7.56816674e-05,  2.77940534e-05,  0.00000000e+00,
        0.00000000e+00],
      [ 1.94344924e-02,  4.51059283e-04,  0.00000000e+00,
        2.08042189e-04, -8.50632661e-04,  2.21382190e-03,
       -2.39274910e-03, -4.19920615e-03,  0.00000000e+00,
        0.00000000e+00],
      [ 9.65949398e-04, -1.63644675e-03,  0.00000000e+00,
       -1.01380875e-04, -3.03189019e-04,  3.48772785e-04,
       -6.66310043e-04,  4.12092412e-04,  0.00000000e+00,
        0.00000000e+00],
      [ 2.19456404e-02,  5.84086379e-03,  0.00000000e+00,
       -1.05548972e-03, -6.47450828e-04, -4.18071954e-03,
       -4.67463420e-01,  1.09503315e-03,  1.51678081e+00,
        0.00000000e+00],
      [ 4.11858946e-04, -5.57022581e-04,  0.00000000e+00,
       -2.31526344e-04, -5.14404310e-05, -8.12974710e-04,
       -3.36167454e-02,  1.34456488e-03,  0.00000000e+00,
        0.00000000e+00],
      [ 7.47625982e-04,  7.54027917e-05, -1.03608134e-01,
       -2.26323863e-05, -7.68557108e-05,  1.35557169e-04,
       -4.32750995e-04, -1.27814527e-04, -9.76273961e-02,
        1.69793747e-01],
      [ 1.47712975e-02,  6.47707850e-04,  0.00000000e+00,
        1.23492591e-04, -1.16035576e-03,  3.33123089e-04,
       -9.30285353e-04, -1.93861342e-03,  1.24527823e+00,
       -3.25708725e-01],
      [ 5.05396867e-03, -3.87775689e-03,  0.00000000e+00,
        4.05529620e-05, -3.66984602e-04,  6.07365775e-04,
       -1.20028155e-02,  6.01038299e-04,  0.00000000e+00,
        0.00000000e+00],
      [ 2.78707539e-04,  3.14800202e-03,  0.00000000e+00,
       -1.73961862e-04, -1.33537127e-03,  2.69358145e-04,
        0.00000000e+00, -3.29599733e-04,  1.08670612e+00,
        0.00000000e+00],
      [ 3.61422078e-03,  2.39041676e-04,  0.00000000e+00,
       -1.57105679e-05, -3.39437722e-04, -3.61791676e-05,
       -2.85778732e-02,  4.30147752e-04,  1.32912805e-01,
        0.00000000e+00]])
```

In [63]:

```
sel_e_lr2 = SelectFromModel(e_lr2)
sel_e_lr2.fit(X,Y)
sel_e_lr2.get_support()
```

```
c:\users\ilya\appdata\local\programs\python\python37\lib\site-packages\sklearn\svm\_base.py:1208: ConvergenceWa
```

```
rning: Liblinear failed to converge, increase the number of iterations.
  ConvergenceWarning,
```

Out[63]:

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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js