МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

Лабораторная работа №3
по курсу «Методы машинного обучения»

ИСПОЛНИТЕЛЬНИЦА: Абросимова Н.Г.
ИУ5-24М

_____
подпись

"__"_____2021 г.

ПРЕПОДАВАТЕЛЬ: _____
ФИО

_____
подпись

"__"_____2021 г.

Москва - 2021

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).

Текст программы и экранные формы

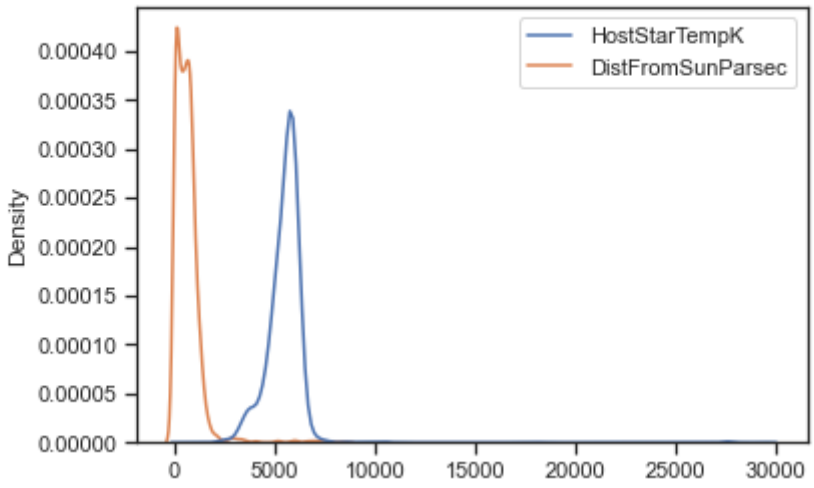
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
data=pd.read_csv('oec.csv', sep=",")
```

Масштабирование признаков

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScal
er
```

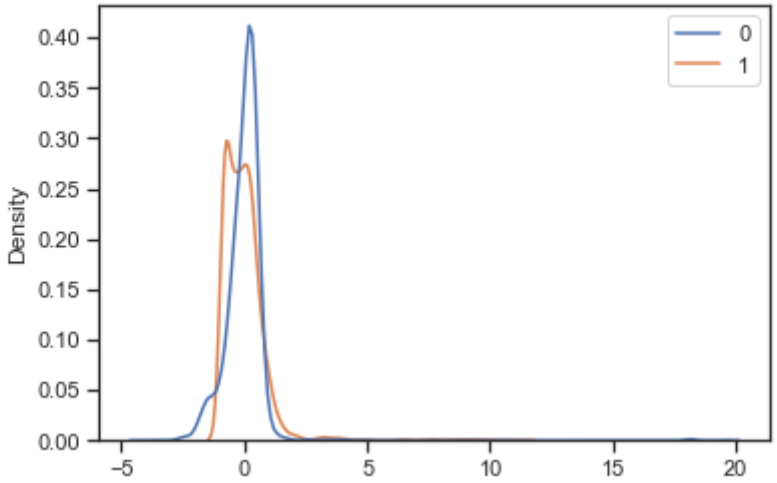
```
#масштабирование на основе Z-оценки sc1 = StandardScaler() sc1_data =
sc1.fit_transform(data[['HostStarTempK', 'DistFromSunParsec']])
sns.kdeplot(data=data[['HostStarTempK', 'DistFromSunParsec']])
```

```
Out[33]: <AxesSubplot:ylabel='Density'>
```




```
sns.kdeplot(data=sc1_data)
```

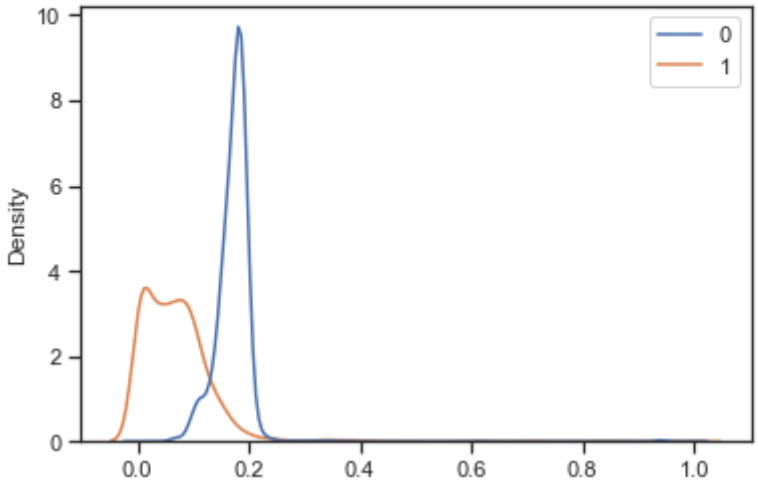
Out[34]: <AxesSubplot:ylabel='Density'>




```
#MinMax-масштабирование
sc2 = MinMaxScaler()
sc2_data = sc2.fit_transform(data[['HostStarTempK', 'DistFromSunParsec']])
```

```
sns.kdeplot(data=sc2_data)
```

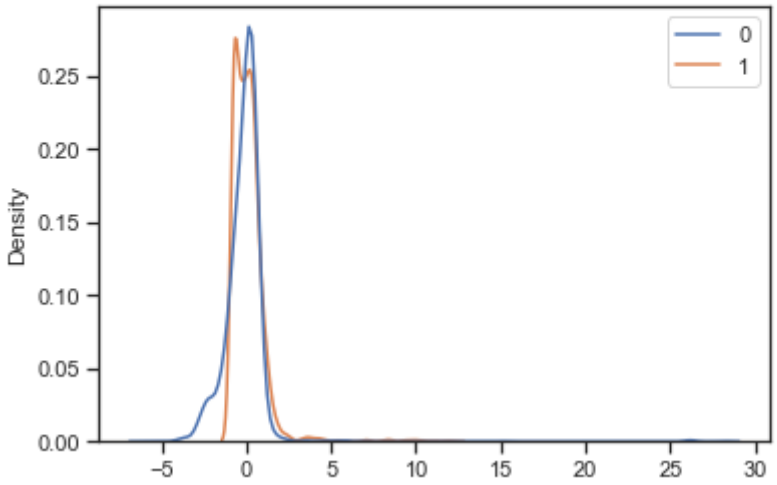
<AxesSubplot:ylabel='Density'>




```
#масштабирование по медиане
sc3 = RobustScaler()
sc3_data = sc3.fit_transform(data[['HostStarTempK', 'DistFromSunParsec']])
```

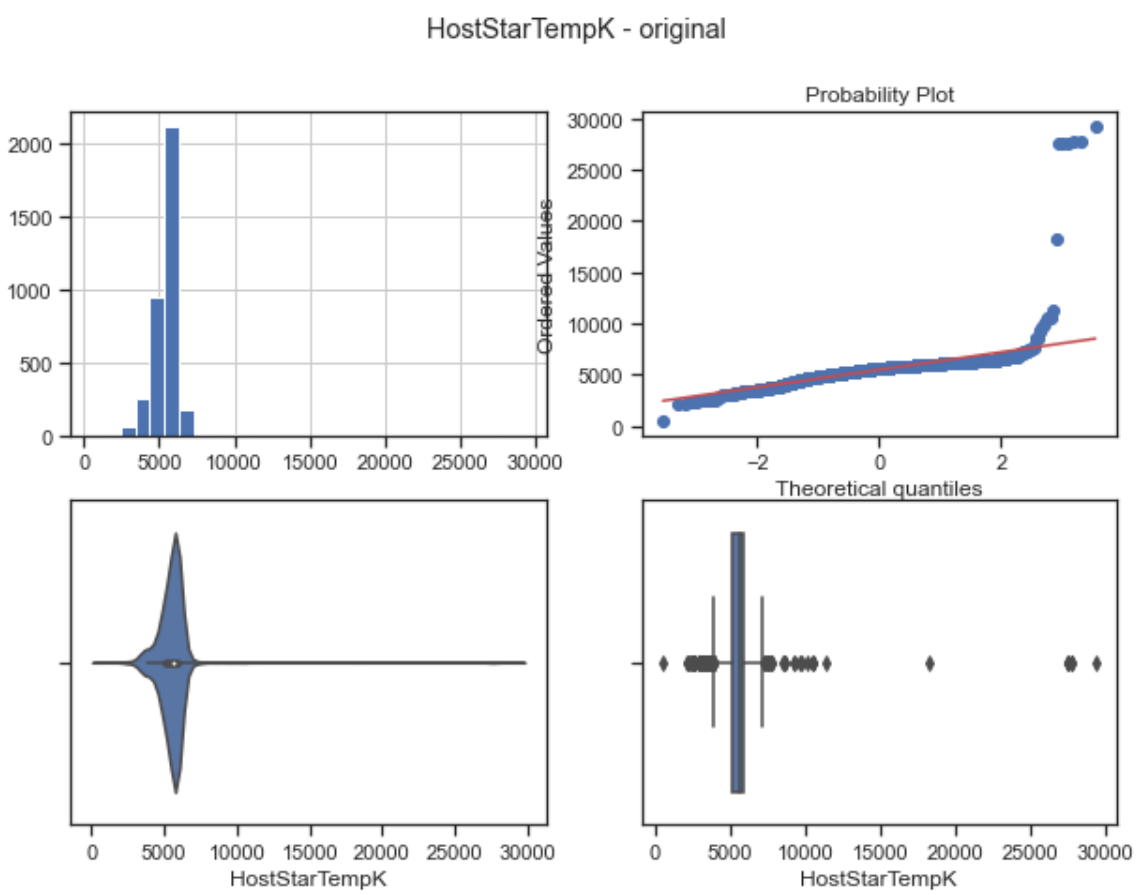
```
: sns.kdeplot(data=sc3_data)
```

```
: <AxesSubplot:ylabel='Density'>
```

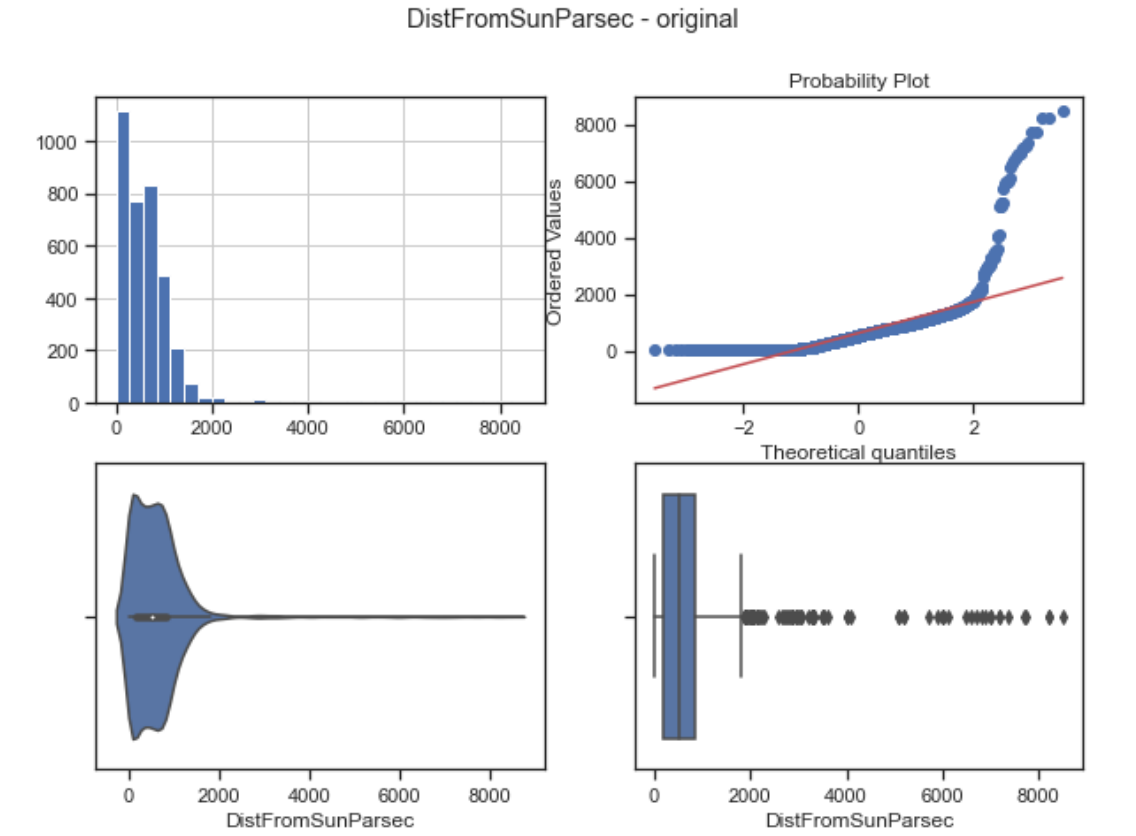


Обработка выбросов

```
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()
diagnostic_plots(data, 'HostStarTempK', 'HostStarTempK - original')
```




```
diagnostic_plots(data, 'DistFromSunParsec', 'DistFromSunParsec - original'
)
```




```
# Тип вычисления верхней и нижней границы выбросов
from enum import Enum
class OutlierBoundaryType(Enum):
    SIGMA = 1
```

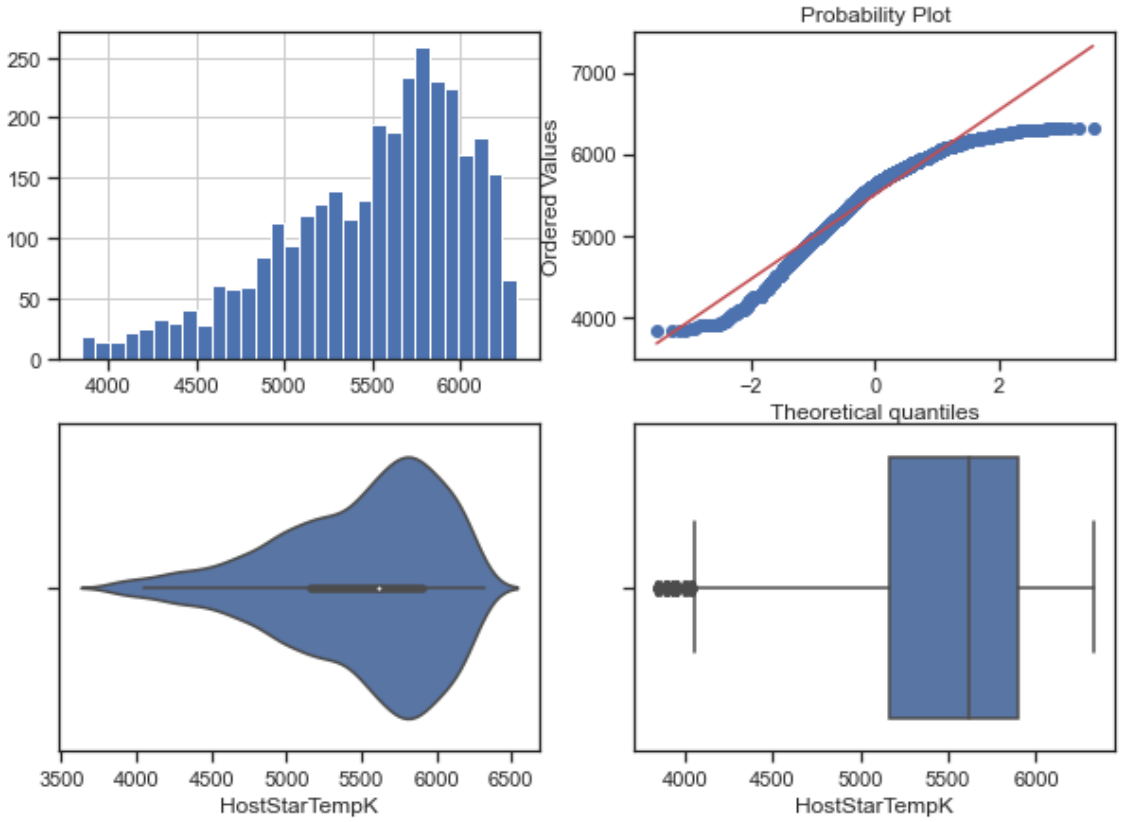
```
    QUANTILE = 2
    IRQ = 3
```

:

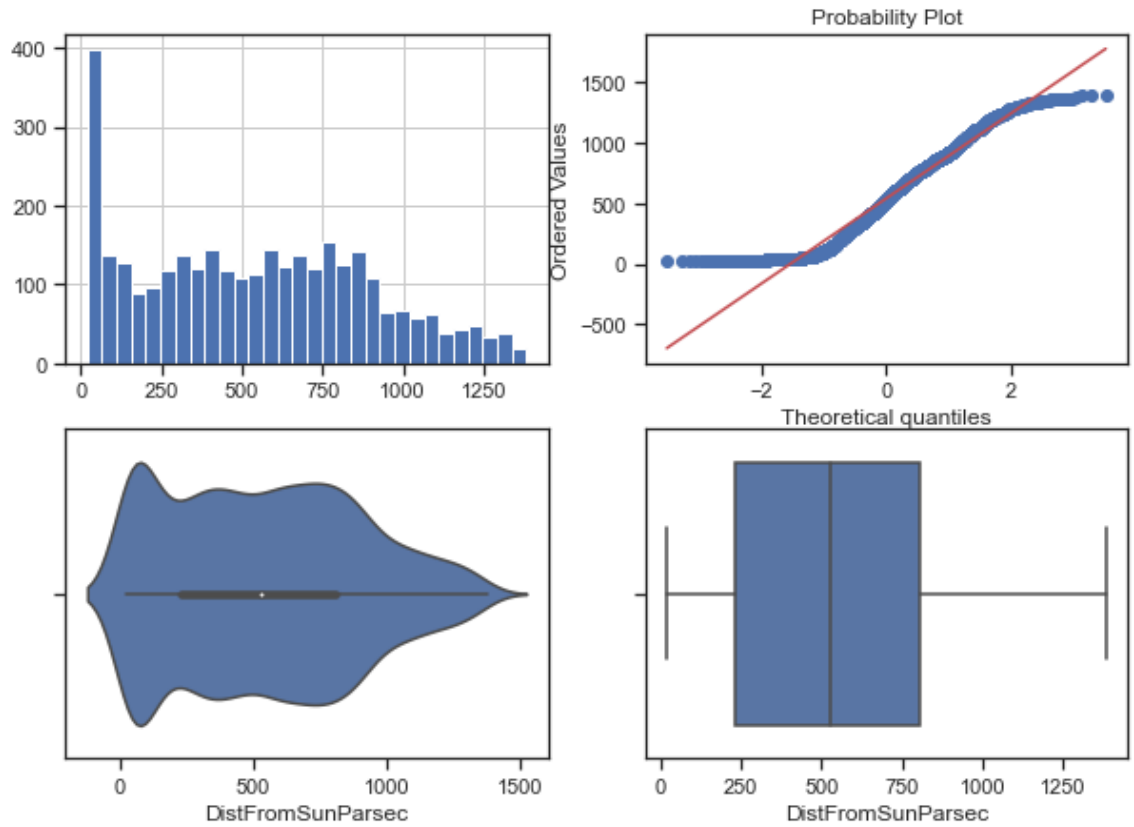
```
# Функция вычисления верхней и нижней границы выбросов
def get_outlier_boundaries(df, col, outlier_boundary_type: OutlierBoundaryTyp
e):
    if outlier_boundary_type == OutlierBoundaryType.SIGMA:
        K1 = 3
        lower_boundary = df[col].mean() - (K1 * df[col].std())
        upper_boundary = df[col].mean() + (K1 * df[col].std())

    elif outlier_boundary_type == OutlierBoundaryType.QUANTILE:
        lower_boundary = df[col].quantile(0.05)
        upper_boundary = df[col].quantile(0.95)

    elif outlier_boundary_type == OutlierBoundaryType.IRQ:
        K2 = 1.5
        IQR = df[col].quantile(0.75) - df[col].quantile(0.25)
        lower_boundary = df[col].quantile(0.25) - (K2 * IQR)
        upper_boundary = df[col].quantile(0.75) + (K2 * IQR)

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

    return lower_boundary, upper_boundary
```

```
for col in x_col_list:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
        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_trimme
d.shape[0])
        diagnostic_plots(data_trimmed, col, title)
```

Поле-HostStarTempK, метод-OutlierBoundaryType.QUANTILE, строк-3225

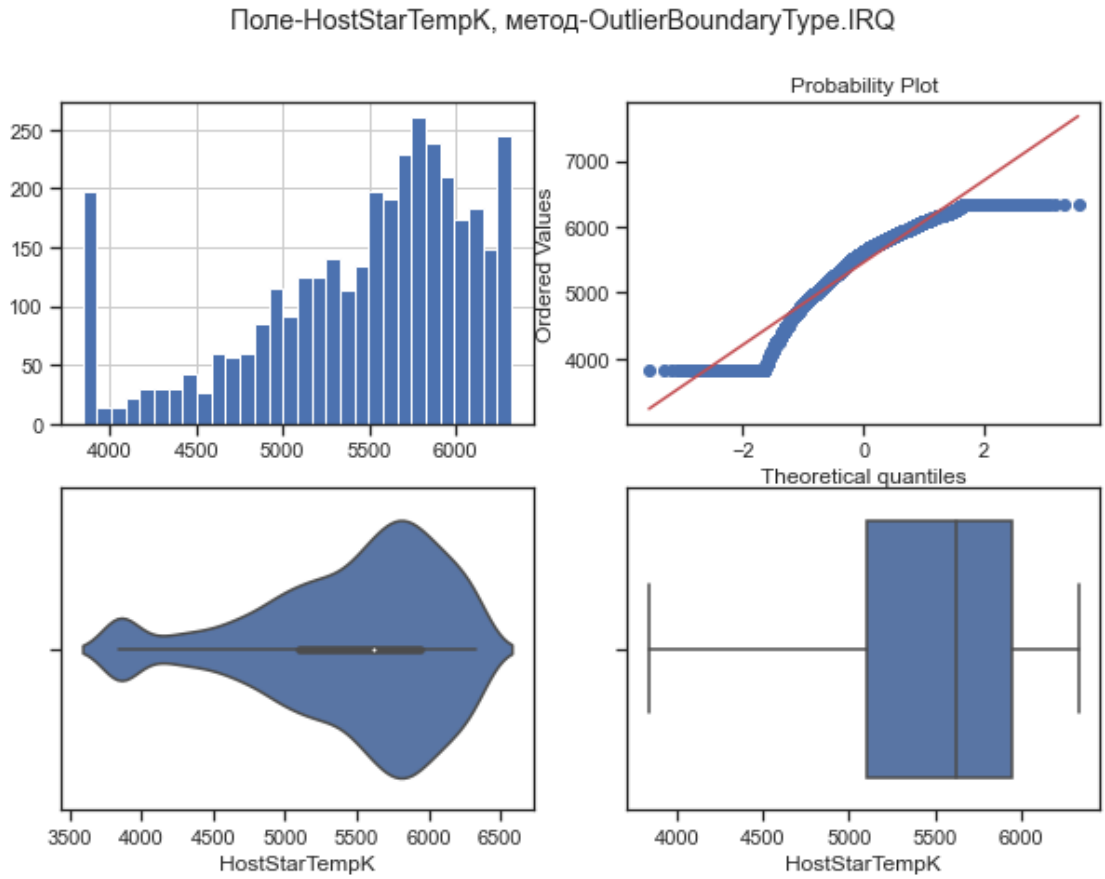


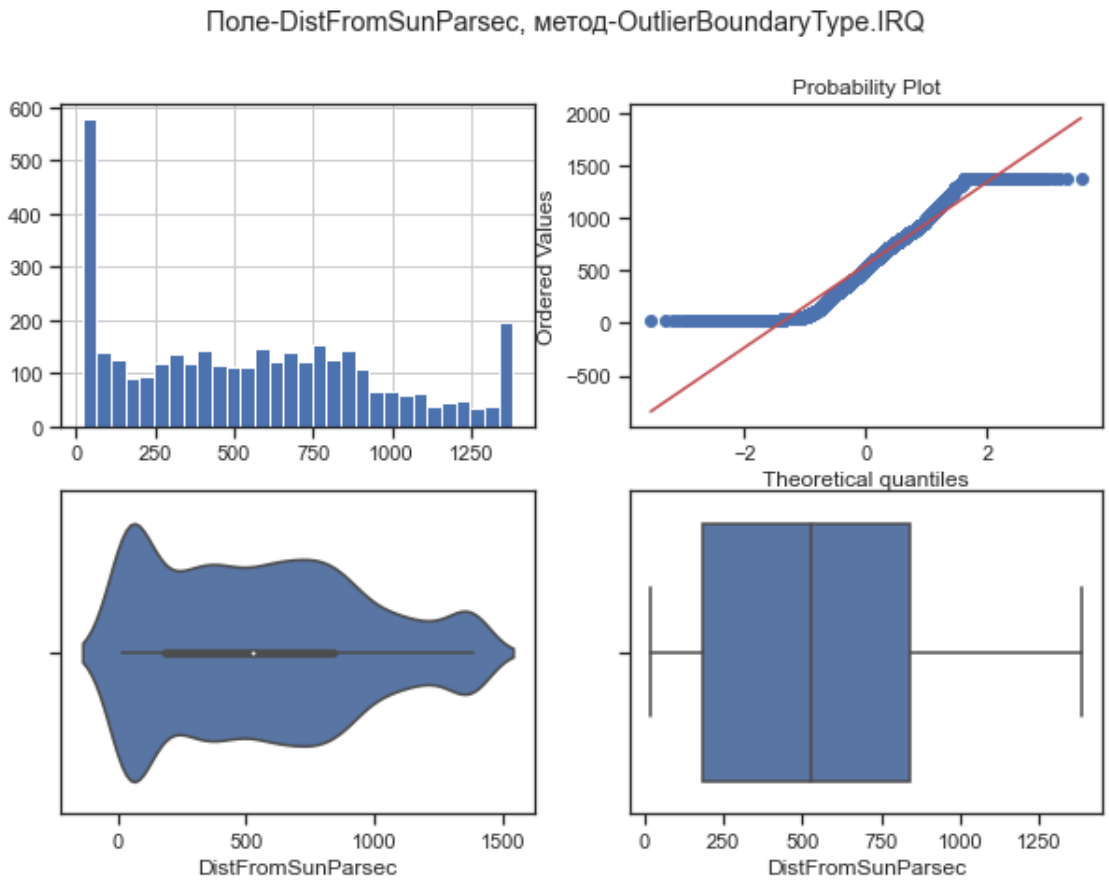
Поле-DistFromSunParsec, метод-OutlierBoundaryType.QUANTILE, строк-3225




```
for col in x_col_list:
    for obt in OutlierBoundaryType:
        # Вычисление верхней и нижней границы
```

```
        lower_boundary, upper_boundary = get_outlier_boundaries(data, col,
obt)
        # Изменение данных
        data[col] = np.where(data[col] > upper_boundary, upper_boundary,
                             np.where(data[col] < lower_boundary, lowe
r_boundary, data[col]))
        title = 'Поле-{}, метод-{}'.format(col, obt)
        diagnostic_plots(data, col, title)
```





Отбор признаков

```python
#filter methods
def get_duplicates(X):
    """
    Поиск дубликатов в колонках
    X - DataFrame
    """

    pairs = {}
    dups = []

    # Перебор всех колонок (внешний)
    for i in range(X.shape[1]):

        # текущая колонка
        feat_outer = X.columns[i]

        # если текущая колонка не является дублем
        if feat_outer not in dups:

            # создаем запись в словаре, колонка является ключом
            pairs[feat_outer] = []

            # Перебор оставшихся колонок (внутренний)
            for feat_inner in X.columns[i + 1:]:

                # Если колонки идентичны
                if X[feat_outer].equals(X[feat_inner]):

                    # добавление в словарь и список дубликатов
                    pairs[feat_outer].append(feat_inner)
                    dups.append(feat_inner)
```

```
    return pairs
```

```
get_duplicates(data)
```

```
{'PlanetIdentifier': [],
 'TypeFlag': [],
 'RadiusJpt': [],
 'PeriodDays': [],
 'DiscoveryMethod': [],
 'DiscoveryYear': [],
 'LastUpdated': [],
 'RightAscension': [],
 'Declination': [],
 'DistFromSunParsec': [],
 'HostStarMassSlrMass': [],
 'HostStarRadiusSlrRad': [],
 'HostStarMetallicity': [],
 'HostStarTempK': [],
 'ListsPlanetIsOn': []}
```

```
efs2 = EFS(knn,
           min_features=1,
           max_features=2,
           scoring='accuracy',
           print_progress=True,
           cv=5)

efs2 = efs2.fit(data[['HostStarMassSlrMass','HostStarRadiusSlrRad', 'DistF
romSunParsec']], data[['TypeFlag']])

print('Best accuracy score: %.2f' % efs2.best_score_)
print('Best subset (indices):', efs2.best_idx_)
print('Best subset (corresponding names):', efs2.best_feature_names_)
```

```
Best accuracy score: 0.94
Best subset (indices): (2,)
Best subset (corresponding names): ('DistFromSunParsec',)
```

```
#embedded methods
from sklearn.linear_model import LogisticRegression
# Используем L1-регуляризацию
e_lr1 = LogisticRegression(C=1000, solver='liblinear', penalty='l1', max_iter
=500, random_state=1)
e_lr1.fit(data[['HostStarMassSlrMass','HostStarRadiusSlrRad', 'DistFromSunPar
sec']], data[['TypeFlag']])
# Коэффициенты регрессии
e_lr1.coef_
```

```
array([[-6.90454280e-02, -7.89441404e-03,  2.89928264e-03],
       [-1.08644978e+00,  3.13511851e-02,  1.53656587e-04],
       [ 1.92457113e-01,  2.94652166e-03, -3.94425882e-03],
       [ 1.03228784e+00, -8.67972471e-02, -5.87089970e-02]])
```

```
from sklearn.feature_selection import SelectFromModel
# Все 4 признака являются "хорошими"
sel_e_lr1 = SelectFromModel(e_lr1)
```

```
sel_e_lr1.fit(data[['HostStarMassSlrMass','HostStarRadiusSlrRad', 'DistFromSu
nParsec']], data[['TypeFlag']])
sel_e_lr1.get_support()
array([ True,  True,  True])
```

```
data = pd.read_csv('IOT-temp.csv', sep=',')
```

In [15]:
```
data.head()
```

Out[15]:

	id	room_id/id	noted_date	temp	out/in
0	__export__.temp_log_196134_bd201015	Room Admin	08-12-2018 09:30	29	In
1	__export__.temp_log_196131_7bca51bc	Room Admin	08-12-2018 09:30	29	In
2	__export__.temp_log_196127_522915e3	Room Admin	08-12-2018 09:29	41	Out
3	__export__.temp_log_196128_be0919cf	Room Admin	08-12-2018 09:29	41	Out
4	__export__.temp_log_196126_d30b72fb	Room Admin	08-12-2018 09:29	31	In

In [16]:
```
data.dtypes
```

Out[16]:
```
id             object
room_id/id     object
noted_date     object
temp            int64
out/in         object
dtype: object
```

```
# Сконвертируем дату и время в нужный формат data['dt'] =
data.apply(lambda x: pd.to_datetime(x['noted_date'], format='%d-%m-%Y
%H:%M'), axis=1)
```

In [19]:
```
data.head()
```

Out[19]:

	id	room_id/id	noted_date	temp	out/in	dt
0	__export__.temp_log_196134_bd201015	Room Admin	08-12-2018 09:30	29	In	2018-12-08 09:30:00
1	__export__.temp_log_196131_7bca51bc	Room Admin	08-12-2018 09:30	29	In	2018-12-08 09:30:00
2	__export__.temp_log_196127_522915e3	Room Admin	08-12-2018 09:29	41	Out	2018-12-08 09:29:00
3	__export__.temp_log_196128_be0919cf	Room Admin	08-12-2018 09:29	41	Out	2018-12-08 09:29:00
4	__export__.temp_log_196126_d30b72fb	Room Admin	08-12-2018 09:29	31	In	2018-12-08 09:29:00

```
data.dtypes
```

Out[20]:
```
id                     object
room_id/id             object
noted_date             object
temp                    int64
out/in                 object
dt             datetime64[ns]
dtype: object
```

In [21]:
```
# День
data['day'] = data['dt'].dt.day
# Месяц
data['month'] = data['dt'].dt.month
```

```
# Год
data['year'] = data['dt'].dt.year
# Часы
data['hour'] = data['dt'].dt.hour
#Минуты
data['minute'] = data['dt'].dt.minute
#Неделя года
data['week'] = data['dt'].dt.isocalendar().week
#Квартал
data['quarter'] = data['dt'].dt.quarter
#День недели
data['dayofweek'] = data['dt'].dt.dayofweek
#Выходной день
data['day_name'] = data['dt'].dt.day_name()
data['is_holiday'] = data.apply(lambda x: 1 if x['dt'].dayofweek in [5,6] els
e 0, axis=1)
```

In [22]:

```
data.head()
```

Out[22]:

	id	room_id/id	noted_date	temp	out/in	dt	day	month	year	hour	minute	week	quarter	dayofweek	day_na
0	__export__.temp_log_196134_bd201015	Room Admin	08-12-2018 09:30	29	In	2018-12-08 09:30:00	8	12	2018	9	30	49	4	5	Saturda
1	__export__.temp_log_196131_7bca51bc	Room Admin	08-12-2018 09:30	29	In	2018-12-08 09:30:00	8	12	2018	9	30	49	4	5	Saturda
2	__export__.temp_log_196127_522915e3	Room Admin	08-12-2018 09:29	41	Out	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturda
3	__export__.temp_log_196128_be0919cf	Room Admin	08-12-2018 09:29	41	Out	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturda
4	__export__.temp_log_196126_d30b72fb	Room Admin	08-12-2018 09:29	31	In	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturda

In [23]:

```
# Разница между датами
data['now'] = datetime.datetime.today()
data['diff'] = data['now'] - data['dt']
data.dtypes
```

Out[23]:

```
id                        object
room_id/id                object
noted_date                object
temp                       int64
out/in                    object
dt              datetime64[ns]
day                        int64
month                      int64
year                       int64
hour                       int64
minute                     int64
week                      UInt32
quarter                    int64
dayofweek                  int64
```

```
day_name                  object
is_holiday                 int64
now              datetime64[ns]
diff            timedelta64[ns]
dtype: object
```

In [24]:

```
data.head()
```

Out[24]:

_id/id	noted_date	temp	out/in	dt	day	month	year	hour	minute	week	quarter	dayofweek	day_name	is_holiday	now	diff
	08-12-2018 09:30	29	In	2018-12-08 09:30:00	8	12	2018	9	30	49	4	5	Saturday	1	2021-04-04 01:20:23.302556	847 days 15:50:23.302556
	08-12-2018 09:30	29	In	2018-12-08 09:30:00	8	12	2018	9	30	49	4	5	Saturday	1	2021-04-04 01:20:23.302556	847 days 15:50:23.302556
	08-12-2018 09:29	41	Out	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturday	1	2021-04-04 01:20:23.302556	847 days 15:51:23.302556
	08-12-2018 09:29	41	Out	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturday	1	2021-04-04 01:20:23.302556	847 days 15:51:23.302556
	08-12-2018 09:29	31	In	2018-12-08 09:29:00	8	12	2018	9	29	49	4	5	Saturday	1	2021-04-04 01:20:23.302556	847 days 15:51:23.302556