|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

**ФАКУЛЬТЕТ** ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

**КАФЕДРА** СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

**Отчёт к лабораторным работам по курсу**

**«Методы машинного обучения»**

**Лабораторная работа №3 «Обработка признаков (часть 2)»**

**Выполнил:**

студент(ка) группы ИУ5И-21М Лю Бэйбэй

подпись, дата

**Проверил:**

к.т.н., доц., Виноградовой М.В.

подпись, дата

Москва, 2022 г.

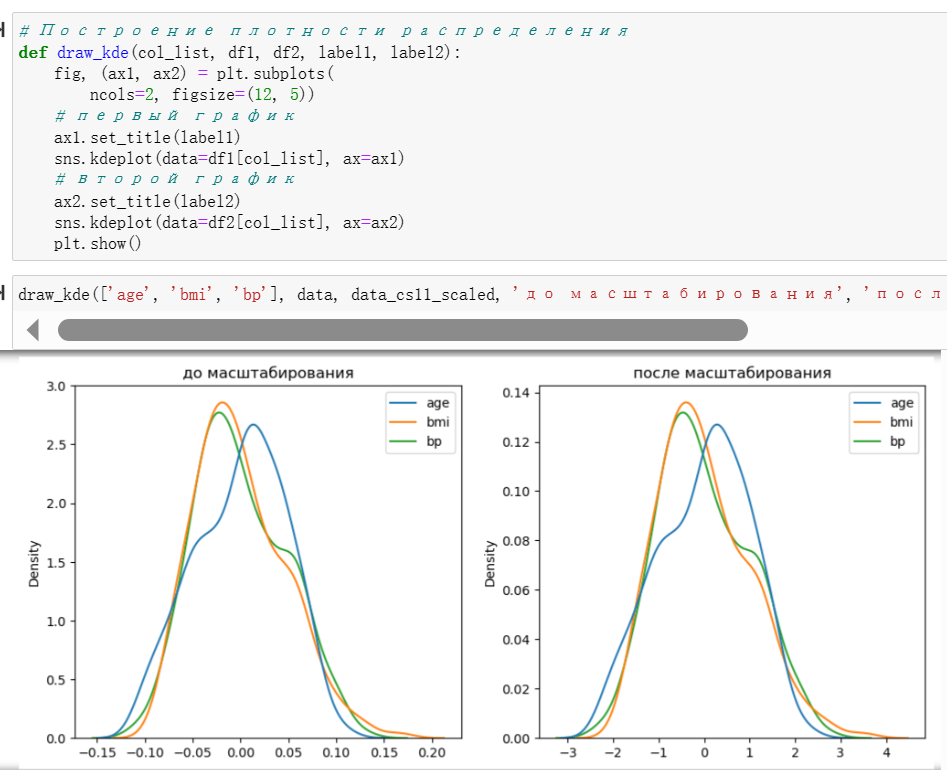
# 1. описание задания

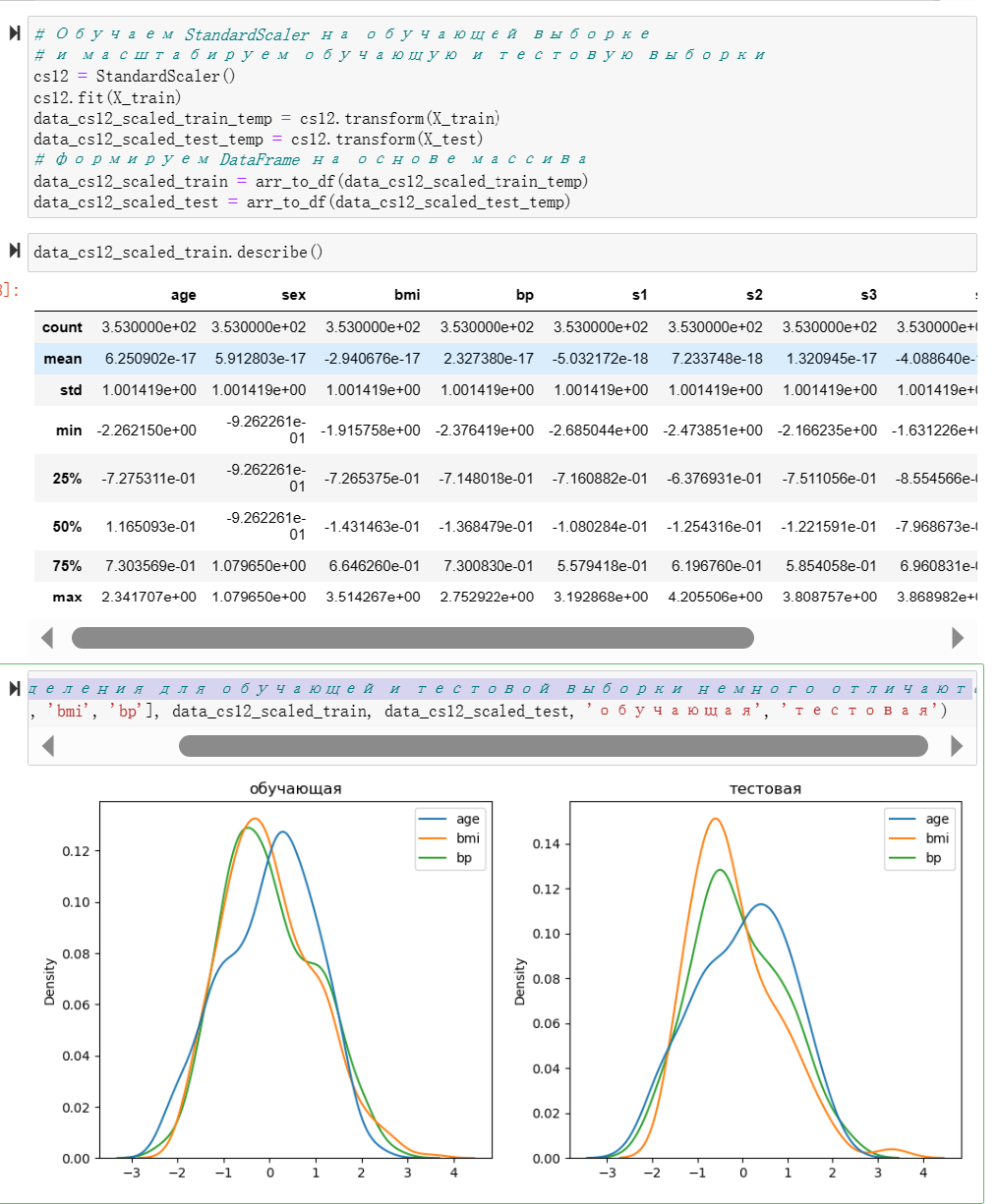
1. Выбрать один или несколько наборов данных (датасетов) для решения следующих задач. Каждая задача может быть решена на отдельном датасете, или несколько задач могут быть решены на одном датасете. Просьба не использовать датасет, на котором данная задача решалась в лекции.
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
   1. масштабирование признаков (не менее чем тремя способами);
   2. обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);
   3. обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);
   4. 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).

# 2. Текст программы и экранные формы с примерами выполнения программы.

масштабирование признаков.

Масштабирование данных на основе Z-оценки





Масштабирование "Mean Normalisation"

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):

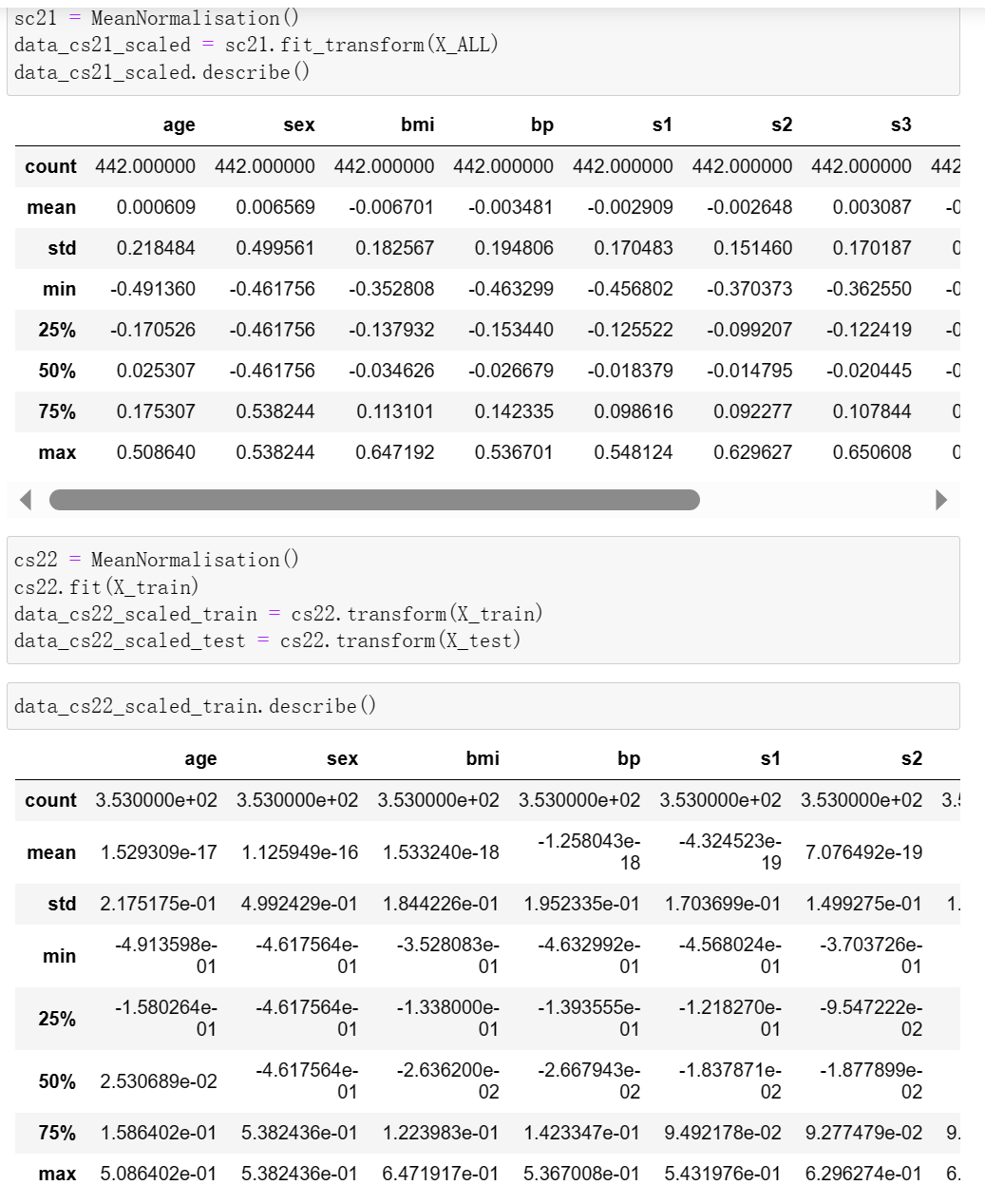
param\_df\_scaled = (param\_df - self.means) / self.ranges

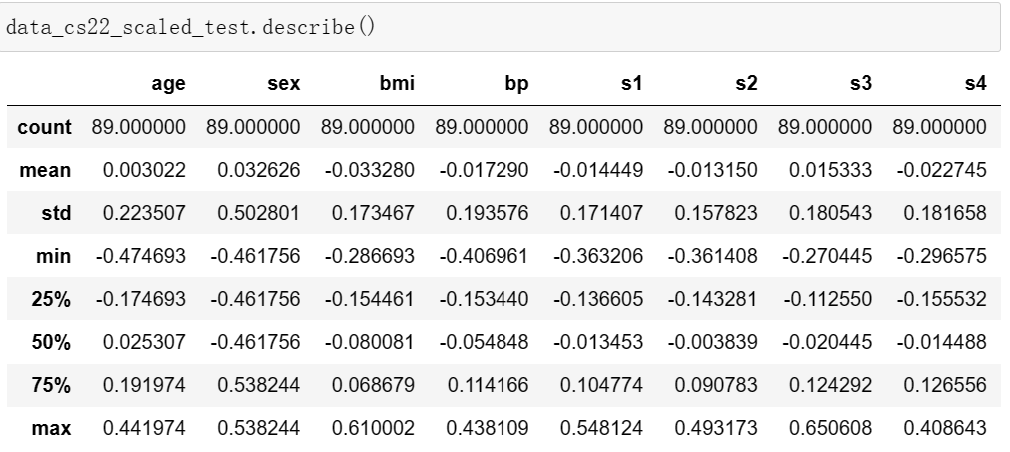
return param\_df\_scaled

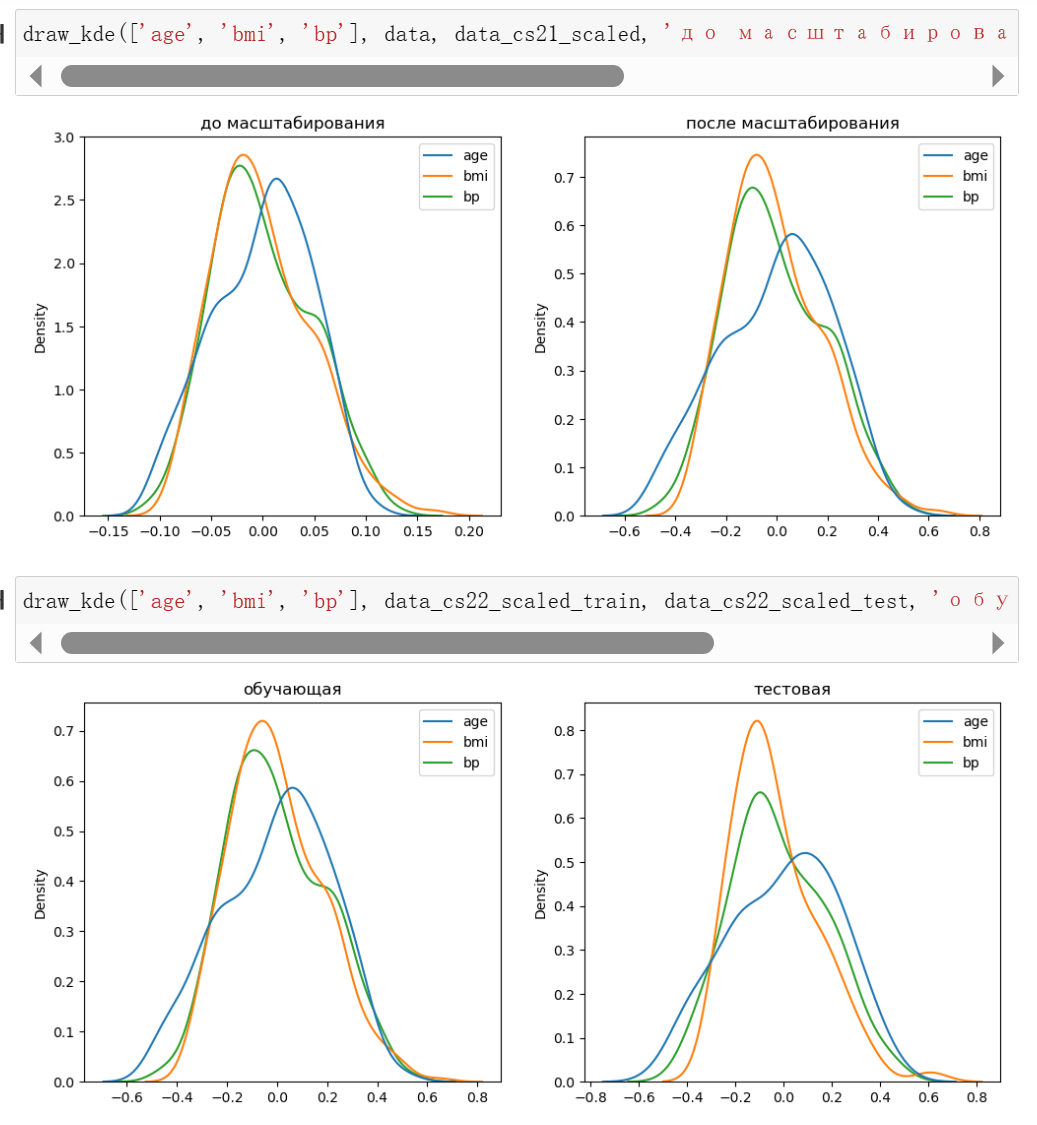
def fit\_transform(self, param\_df):

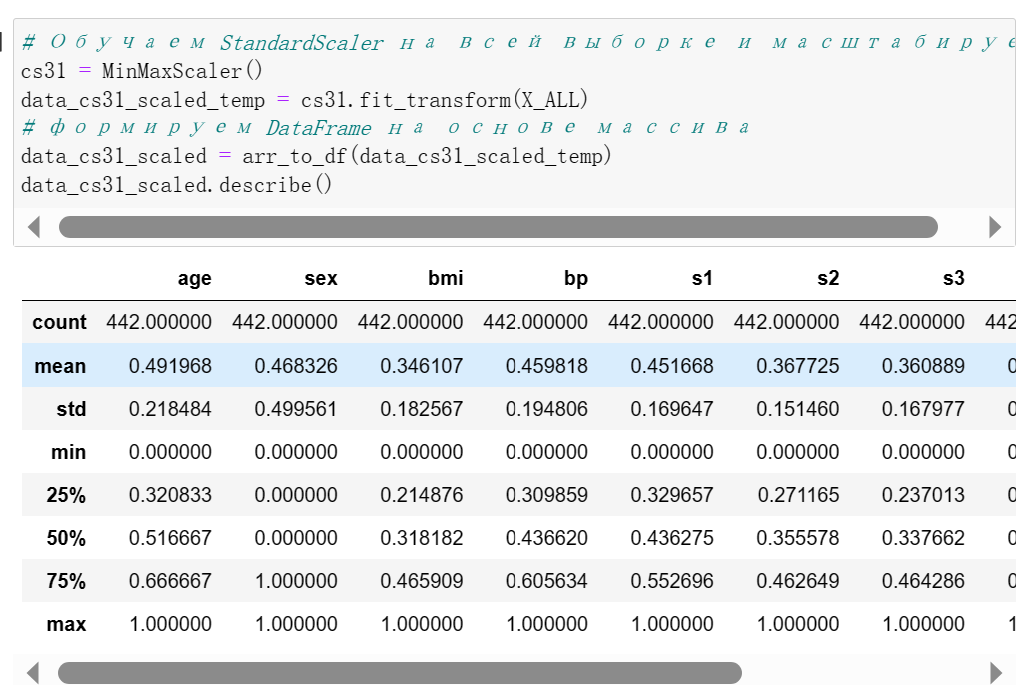
self.fit(param\_df)

return self.transform(param\_df)







MinMax-масштабирование

cs32 = MinMaxScaler()

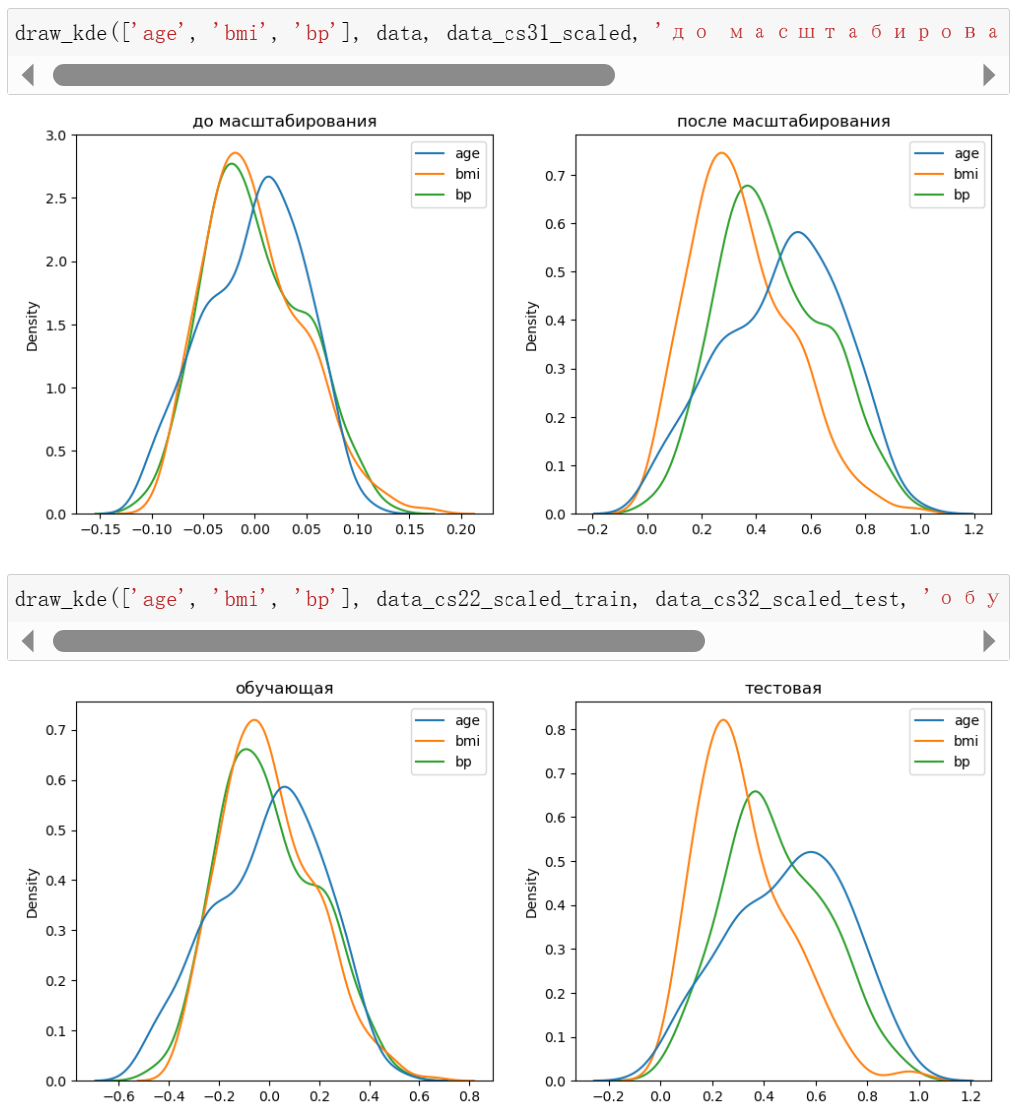
cs32.fit(X\_train)

data\_cs32\_scaled\_train\_temp = cs32.transform(X\_train)

data\_cs32\_scaled\_test\_temp = cs32.transform(X\_test)

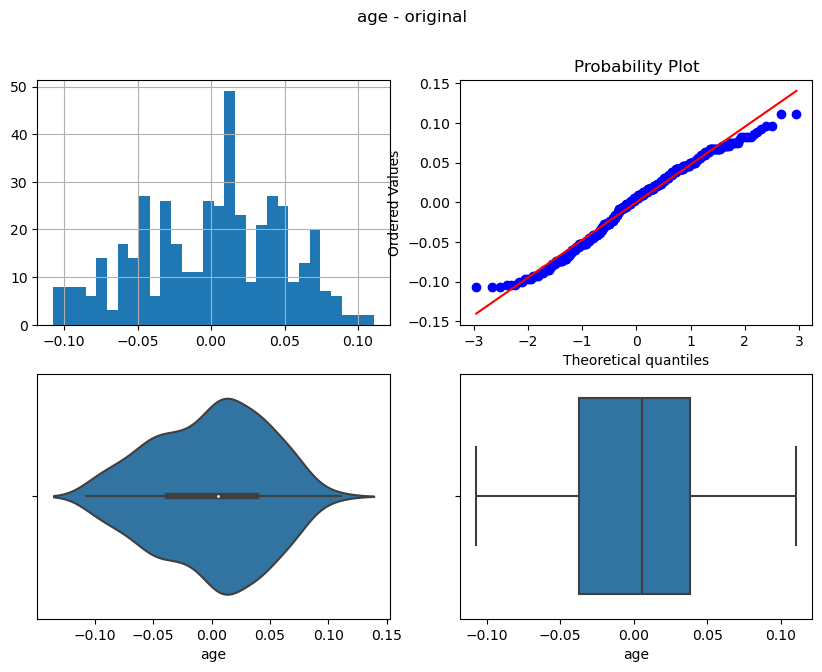
# формируем DataFrame на основе массива

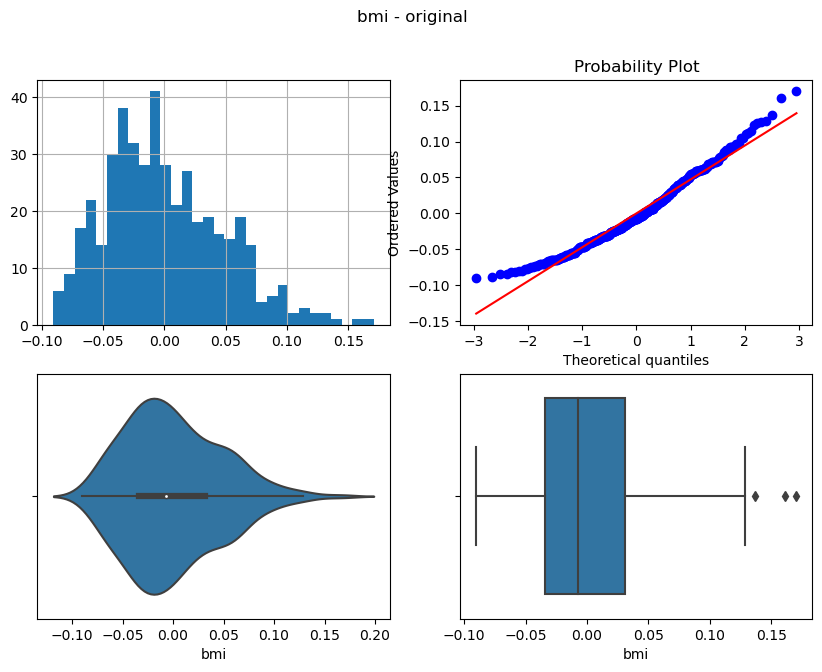
data\_cs32\_scaled\_train = arr\_to\_df(data\_cs32\_scaled\_train\_temp)

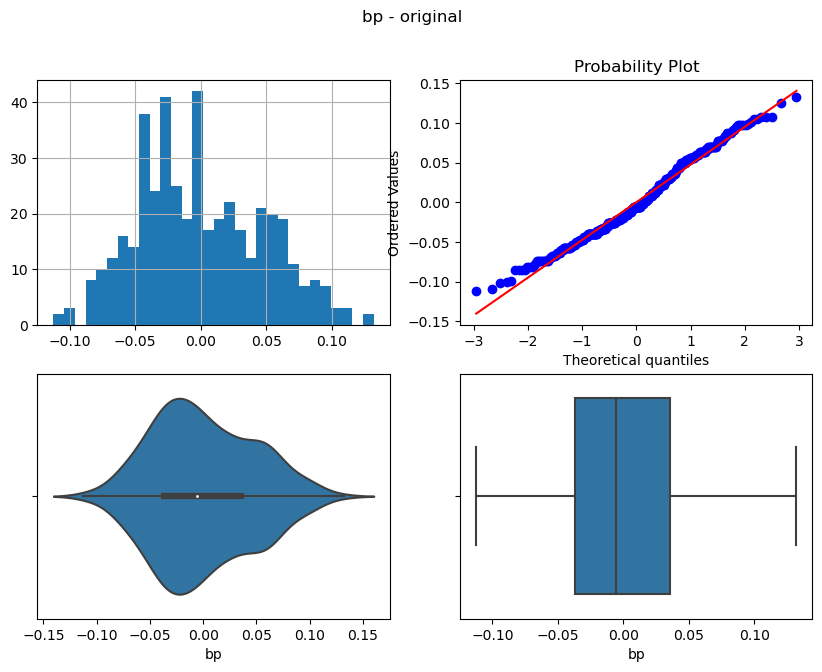
data\_cs32\_scaled\_test = arr\_to\_df(data\_cs32\_scaled\_test\_temp) 

обработку выбросов для числовых признаков (по одному способу для удаления выбросов и для замены выбросов);

Необработанные данные：







Удаление выбросов

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,

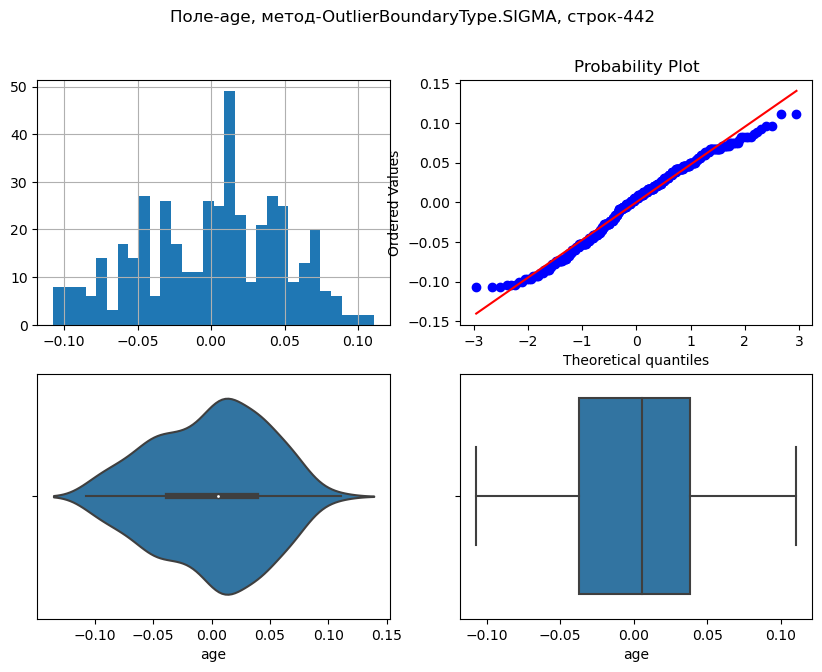
np.where(data[col] < lower\_boundary, True, False))

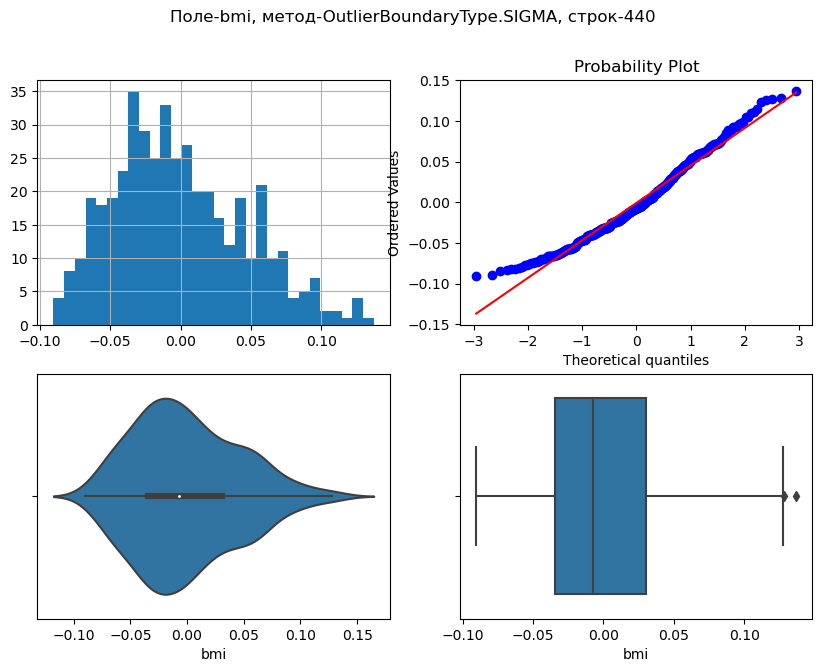
# Удаление данных на основе флага

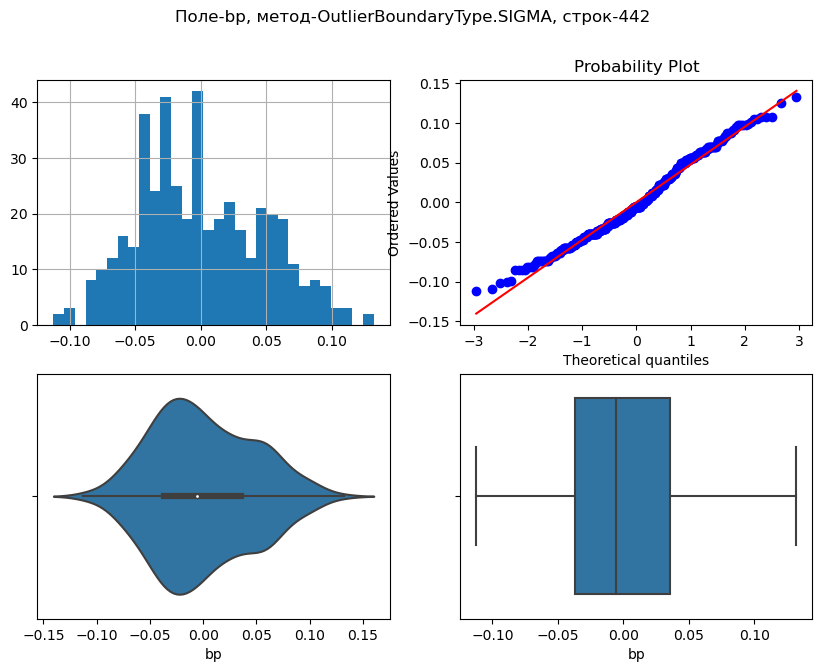
data\_trimmed = data.loc[~(outliers\_temp), ]

title = 'Поле-{}, метод-{}, строк-{}'.format(col, obt, data\_trimmed.shape[0])

diagnostic\_plots(data\_trimmed, col, title)







замены выбросов：

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)

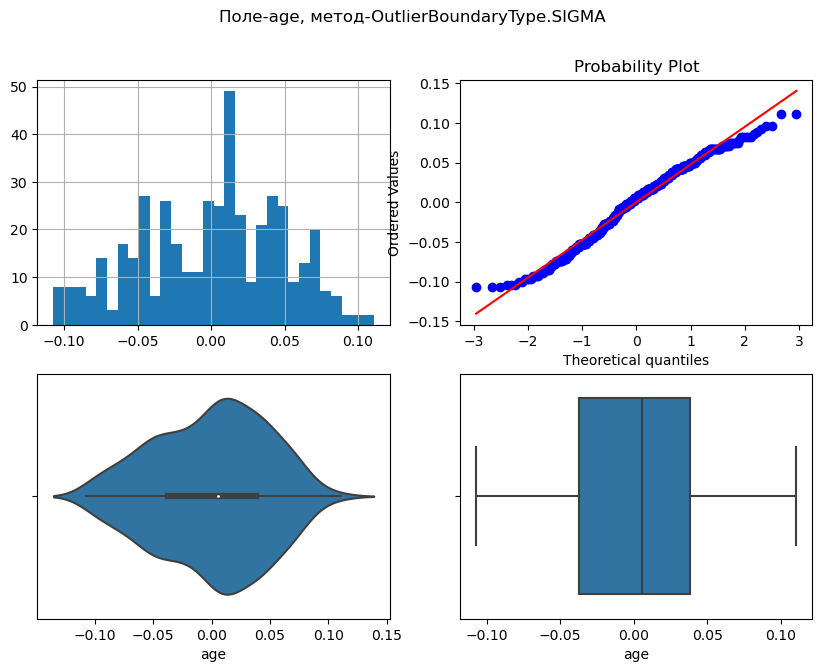
# Изменение данных

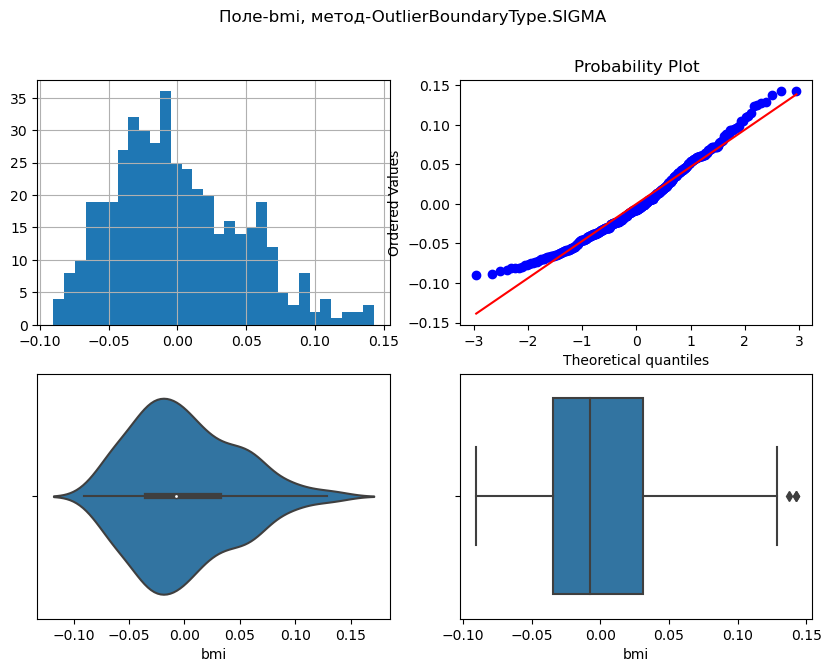
data[col] = np.where(data[col] > upper\_boundary, upper\_boundary,

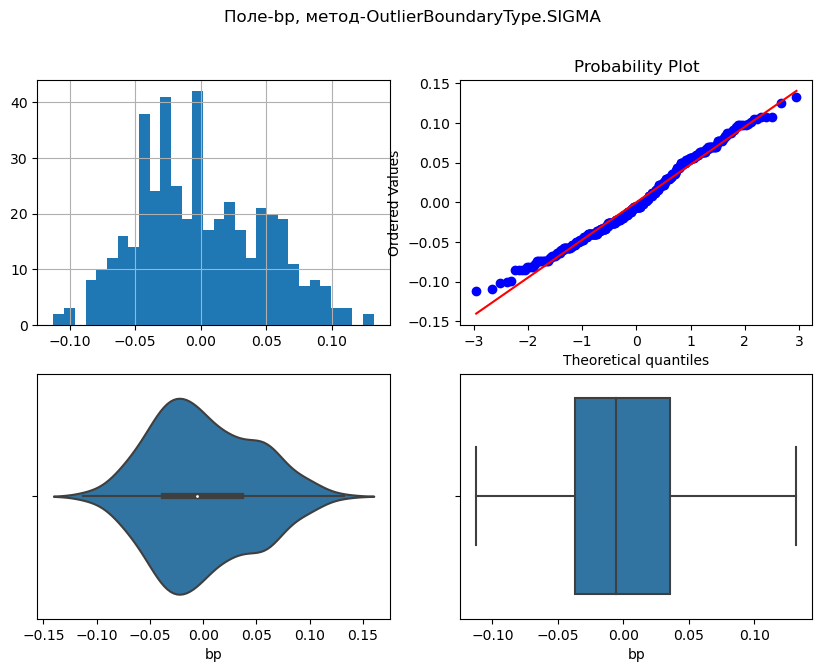
np.where(data[col] < lower\_boundary, lower\_boundary, data[col]))

title = 'Поле-{}, метод-{}'.format(col, obt)

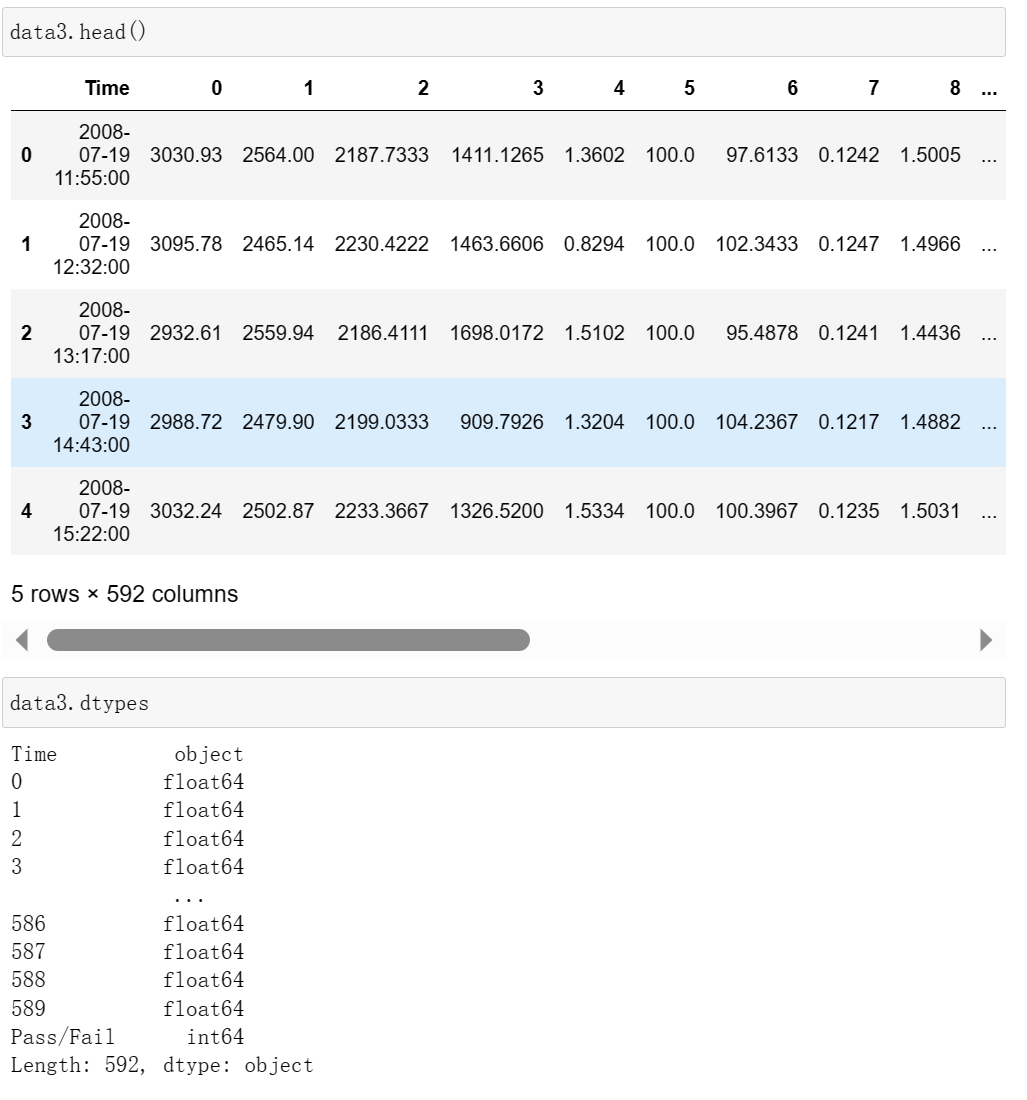
diagnostic\_plots(data, col, title)







обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);



data3['dt'] = data3.apply(lambda x: pd.to\_datetime(x['Time'], format='%Y/%m/%d %H:%M:%S'), axis=1)

# День

data3['day'] = data3['dt'].dt.day

# Месяц

data3['month'] = data3['dt'].dt.month

# Год

data3['year'] = data3['dt'].dt.year

# Часы

data3['hour'] = data3['dt'].dt.hour

#Минуты

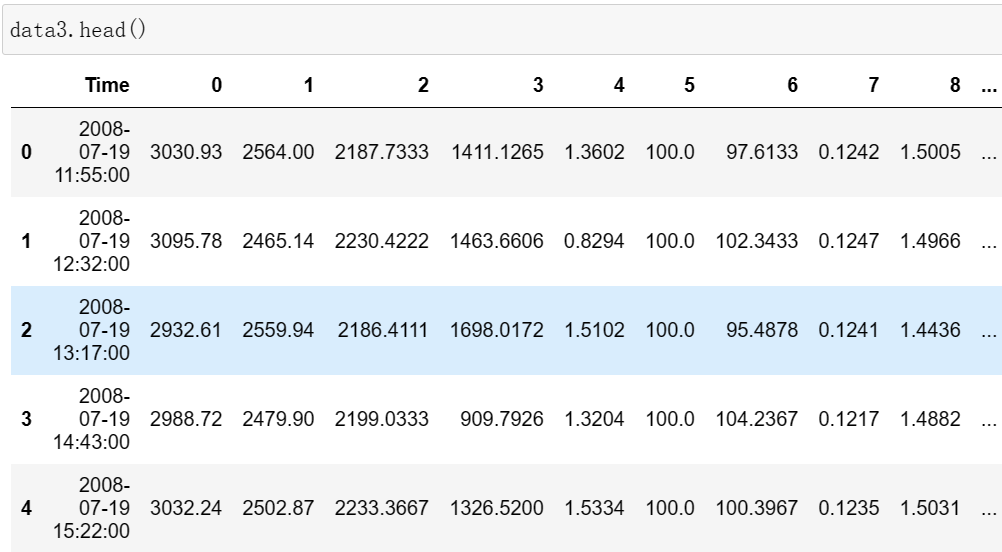
data3['minute'] = data3['dt'].dt.minute

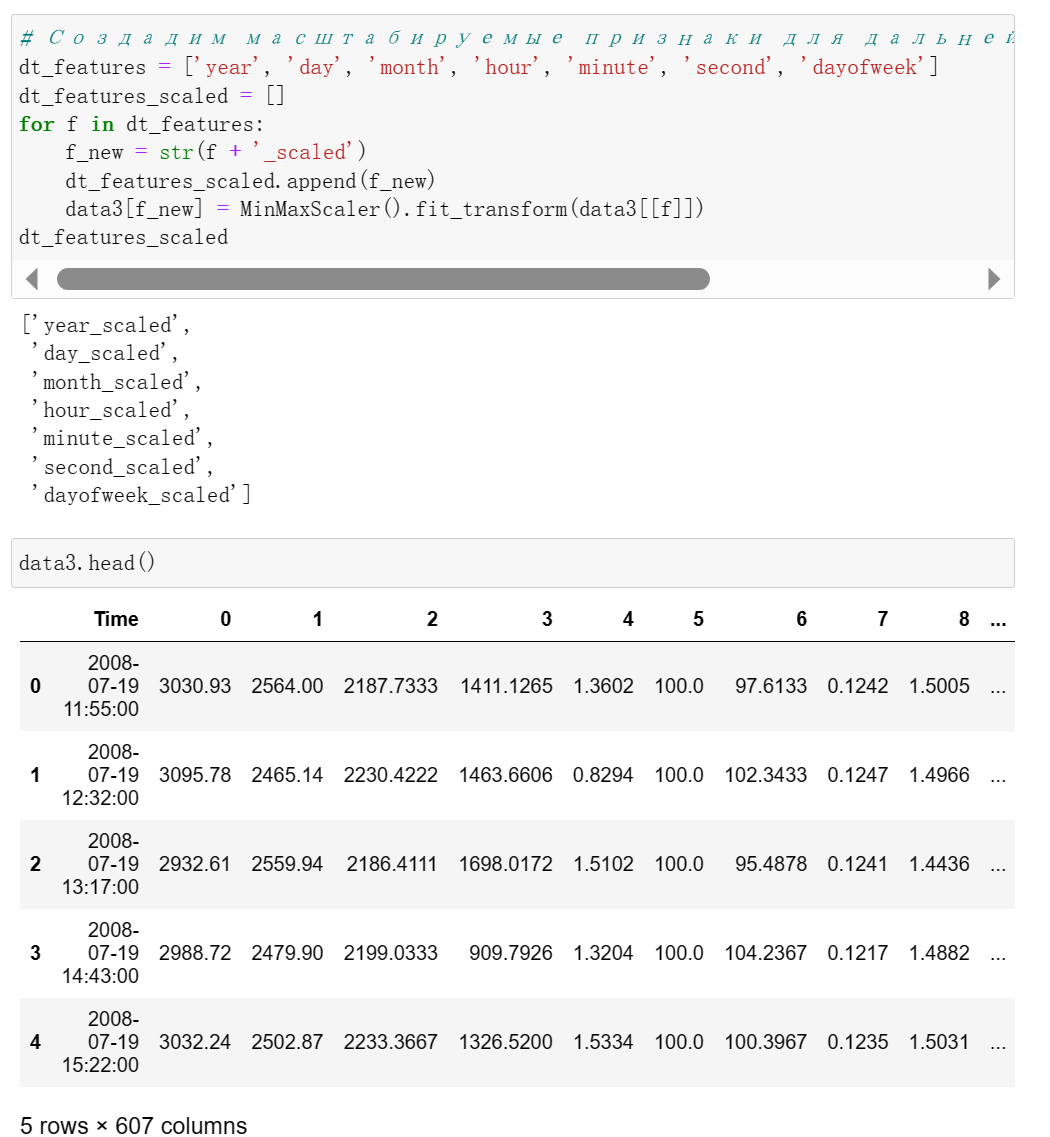
#Секунды

data3['second'] = data3['dt'].dt.second

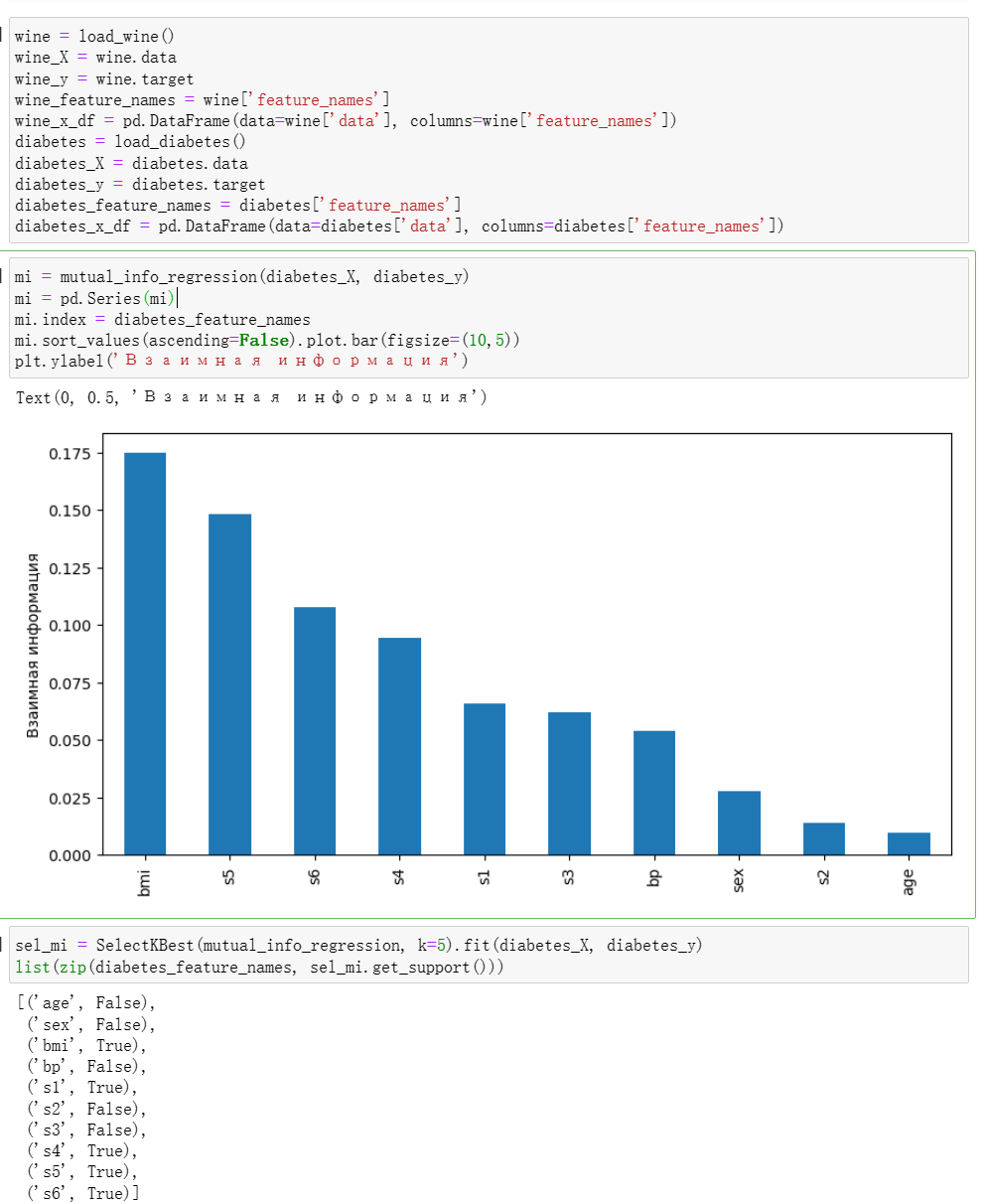
#День недели

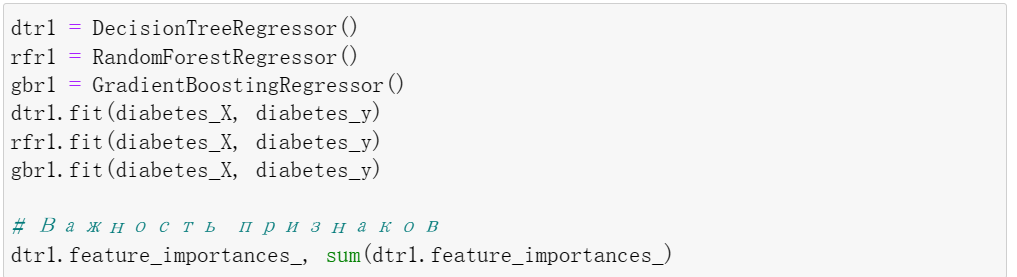
data3['dayofweek'] = data3['dt'].dt.dayofweek





отбор признаков：





from operator import itemgetter

def draw\_feature\_importances(tree\_model, X\_dataset, title, figsize=(7,4)):

"""

Вывод важности признаков в виде графика

"""

# Сортировка значений важности признаков по убыванию

list\_to\_sort = list(zip(X\_dataset.columns.values, tree\_model.feature\_importances\_))

sorted\_list = sorted(list\_to\_sort, key=itemgetter(1), reverse = True)

# Названия признаков

labels = [x for x,\_ in sorted\_list]

# Важности признаков

data = [x for \_,x in sorted\_list]

# Вывод графика

fig, ax = plt.subplots(figsize=figsize)

ax.set\_title(title)

ind = np.arange(len(labels))

plt.bar(ind, data)

plt.xticks(ind, labels, rotation='vertical')

# Вывод значений

for a,b in zip(ind, data):

plt.text(a-0.1, b+0.005, str(round(b,3)))

plt.show()

return labels, data

