Рубежный контроль №1 по курсу «Методы машинного обучения»

Коротков Никита, ИУ5-23М

Вариант задания

Номер варианта	Задание 1	Задание 2	Доп. требование
6	6	26	для произвольной колонки данных построить парные диаграммы (pairplot)

Импорт библиотек

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
import seaborn as sns
import scipy.stats as stats
```

Задание 1. Для набора данных проведите устранение пропусков для одного (произвольного) числового признака с использованием метода заполнения средним значением.

```
# Загрузка датасета
df = pd.read_csv('cherry_blossom_forecasts.csv', index_col=0)
df.head(10)
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 52142,\n \"fields\": [\n   {\n   \"column\": \"place_code\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12088950,\n   \"min\": 1370010,\n   \"max\": 46370019,\n   \"num_unique_values\":

```
1004,\n        \"samples\": [\n          40370032,\n          25370025,\n          26370021\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"date\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 53,\n        \"samples\": [\n          \"2024-02-20\",\n          \"2024-03-13\",\n          \"2024-03-19\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"mankai_date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"num_unique_values\": 63,\n        \"samples\": [\n          \"2024-03-18\",\n          \"2024-05-14\",\n          \"2024-05-12\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"kaika_date\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"num_unique_values\": 67,\n        \"samples\": [\n          \"2024-04-01\",\n          \"2024-04-24\",\n          \"2024-05-09\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"meter\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 22.588689050745174,\n        \"min\": 0.0,\n        \"max\": 200.0,\n        \"num_unique_values\": 171,\n        \"samples\": [\n          99.0,\n          54.0,\n          64.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"tavg\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 3.9677239463397767,\n        \"min\": -14.5,\n        \"max\": 20.6,\n        \"num_unique_values\": 286,\n        \"samples\": [\n          -3.2,\n          19.9,\n          2.1\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"tmin\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4.101461978826027,\n        \"min\": -22.2,\n        \"max\": 17.8,\n        \"num_unique_values\": 326,\n        \"samples\": [\n          -12.2,\n          13.1,\n          11.9\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"tmax\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 4.535723446523079,\n        \"min\": -6.0,\n        \"max\": 24.8,\n        \"num_unique_values\": 285,\n        \"samples\": [\n          1.9,\n          23.2,\n          -3.6\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"prcp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 8.567467208446796,\n        \"min\": 0.0,\n        \"max\": 140.5,\n        \"num_unique_values\": 437,\n        \"samples\": [\n          36.8,\n          10.1,\n          82.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

Проверим датасет на наличие пропусков в данных

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 52142 entries, 1370053 to 43370005
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   date         52142 non-null  object 
 1   mankai_date  52141 non-null  object 
 2   kaika_date   52141 non-null  object 
 3   meter        52141 non-null  float64
 4   tavg         45926 non-null  float64
 5   tmin         45926 non-null  float64
 6   tmax         45926 non-null  float64
 7   prcp         43966 non-null  float64
dtypes: float64(5), object(3)
memory usage: 3.6+ MB
```

В числовом признаке tavg много пропусков, заполним их средним значением

```
def impute_column(dataset, column, strategy_param,
fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                            fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data

all_data, filled_data, missed_data = impute_column(df, 'tavg', 'mean')
```

Заменяем исходный столбец на него же с заполненными пропусками

```
df['tavg'] = all_data
df.head(10)
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 52142,\n \"fields\": [\n   {\n   \"column\": \"place_code\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12088950,\n   \"min\":

```
1370010,\n        \"max\": 46370019,\n        \"num_unique_values\":
1004,\n        \"samples\": [\n          40370032,\n
25370025,\n          26370021\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"date\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 53,\n
\"samples\": [\n          \"2024-02-20\",\n          \"2024-03-13\",\n
\"2024-03-19\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"mankai_date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 63,\n        \"samples\":
[\n          \"2024-03-18\",\n          \"2024-05-14\",\n
\"2024-05-12\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"kaika_date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 67,\n        \"samples\":
[\n          \"2024-04-01\",\n          \"2024-04-24\",\n
\"2024-05-09\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"meter\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 22.588689050745174,\n        \"min\": 0.0,\n        \"max\":
200.0,\n        \"num_unique_values\": 171,\n        \"samples\": [\n
99.0,\n          54.0,\n          64.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tavg\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3.7237143035798392,\n
\"min\": -14.5,\n        \"max\": 20.6,\n
\"num_unique_values\": 287,\n        \"samples\": [\n          -4.3,\n
17.8,\n          2.1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"tmin\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 4.101461978826027,\n        \"min\": -22.2,\n        \"max\":
17.8,\n        \"num_unique_values\": 326,\n        \"samples\": [\n
-12.2,\n          13.1,\n          11.9\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tmax\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 4.535723446523079,\n
\"min\": -6.0,\n        \"max\": 24.8,\n        \"num_unique_values\":
285,\n        \"samples\": [\n          1.9,\n          23.2,\n
-3.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"prcp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 8.567467208446796,\n        \"min\": 0.0,\n        \"max\":
140.5,\n        \"num_unique_values\": 437,\n        \"samples\": [\n
36.8,\n          10.1,\n          82.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 52142 entries, 1370053 to 43370005
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         52142 non-null  object
 1   mankai_date  52141 non-null  object
 2   kaika_date   52141 non-null  object
 3   meter        52141 non-null  float64
 4   tavg         52142 non-null  float64
 5   tmin         45926 non-null  float64
 6   tmax         45926 non-null  float64
 7   prcp         43966 non-null  float64
dtypes: float64(5), object(3)
memory usage: 3.6+ MB
```

Как можно видеть, пропуски в признаки tavg были заполнены средним значением

Задание 2. Для набора данных для одного (произвольного) числового признака проведите обнаружение и замену (найденными верхними и нижними границами) выбросов на основе правила трех сигм.

```
# Загрузка датасета
df = pd.read_csv('cherry_blossom_forecasts.csv')
df.head(10)
```


{"summary":"{\n \"name\": \"df\",\n \"rows\": 72185,\n \"fields\": [\n   {\n   \"column\": \"place_code\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12107919,\n   \"min\": 1370010,\n   \"max\": 46370019,\n   \"num_unique_values\": 1004,\n   \"samples\": [\n   40370032,\n   25370025,\n   26370021\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"date\",\n   \"properties\": {\n   \"dtype\": \"object\",\n   \"num_unique_values\": 72,\n   \"samples\": [\n   \"2024-02-05\",\n   \"2024-04-03\",\n   \"2024-02-19\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"mankai_date\",\n   \"properties\": {\n   \"dtype\": \"object\",\n   \"num_unique_values\": 63,\n   \"samples\":

```
[\n          \"2024-03-18\",\n          \"2024-05-14\",\n
\"2024-05-12\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"kaika_date\",\n      \"properties\": {\n        \"dtype\":
\"object\",\n        \"num_unique_values\": 68,\n        \"samples\":
[\n          \"2024-03-27\",\n          \"2024-04-24\",\n
\"2024-05-09\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"meter\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 56,\n        \"min\": 0,\n        \"max\": 200,\n
\"num_unique_values\": 199,\n        \"samples\": [\n          75,\n
7,\n          110\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"tavg\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 4.695801898829381,\n        \"min\": -14.5,\n        \"max\":
20.9,\n        \"num_unique_values\": 302,\n        \"samples\": [\n
-2.1,\n          14.4,\n          15.6\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"tmin\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 4.829221288836313,\n
\"min\": -22.2,\n        \"max\": 19.4,\n
\"num_unique_values\": 339,\n        \"samples\": [\n          16.2,\n
-4.1,\n          -11.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"tmax\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 5.183663673005783,\n        \"min\": -6.0,\n        \"max\":
28.0,\n        \"num_unique_values\": 302,\n        \"samples\": [\n
2.5,\n          19.6,\n          -4.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"prcp\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 10.94973160928017,\n
\"min\": 0.0,\n        \"max\": 140.5,\n        \"num_unique_values\":
579,\n        \"samples\": [\n          40.3,\n          76.3,\n
62.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```


Функция для построения нескольких графиков

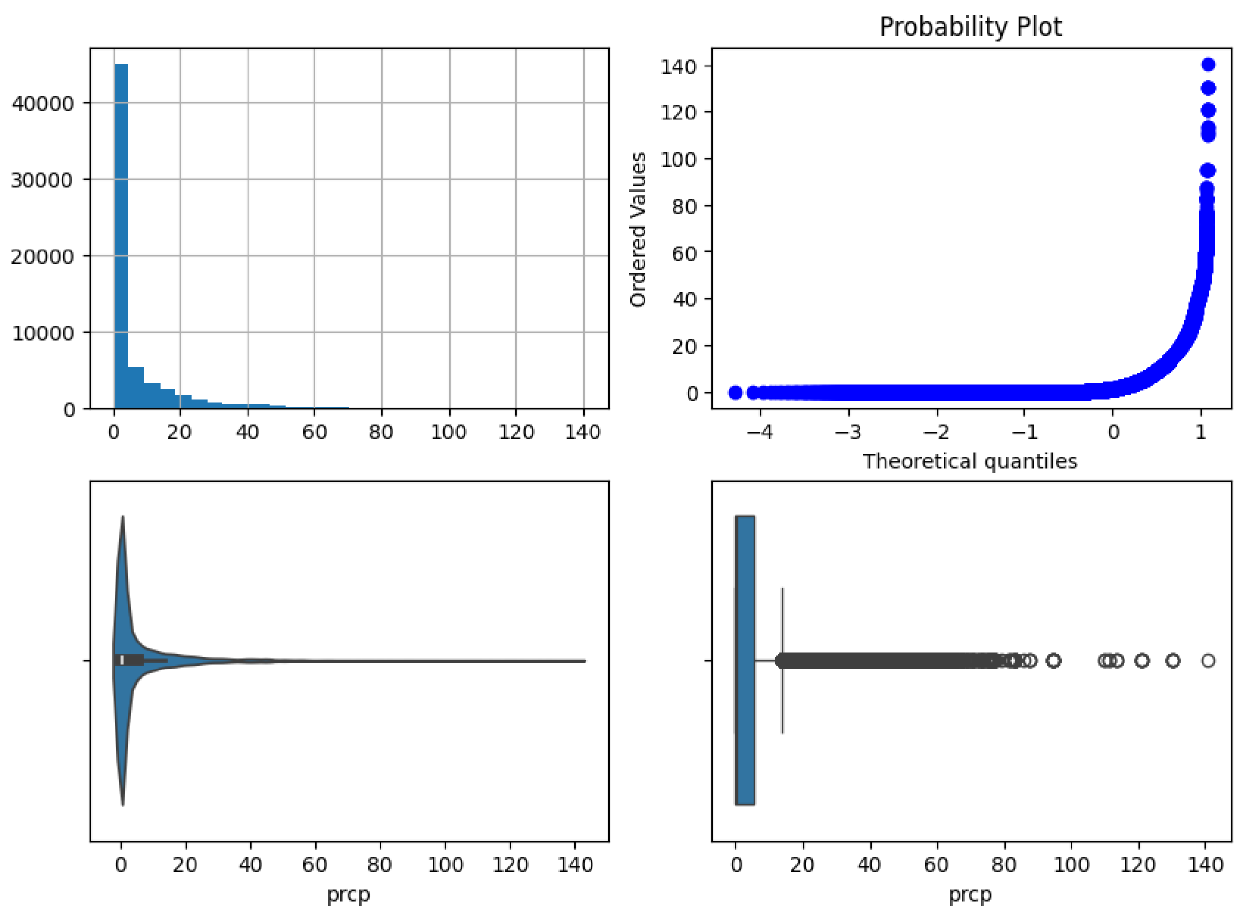
```
def diagnostic_plots(df, variable):
    fig, ax = plt.subplots(figsize=(10,7))
    # гистограмма
    plt.subplot(2, 2, 1)
    df[variable].hist(bins=30)
    ## Q-Q plot
    plt.subplot(2, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    # ящик с усами
    plt.subplot(2, 2, 3)
```

```
    sns.violinplot(x=df[variable])
    # ящик с усами
    plt.subplot(2, 2, 4)
    sns.boxplot(x=df[variable])
    plt.show()
```

Выявляем при помощи графиков выбросы в признаке fixed acidity

```
diagnostic_plots(df, 'prcp')
```

```
<ipython-input-20-c2e6bc34c22e>:4: MatplotlibDeprecationWarning: Auto-
removal of overlapping axes is deprecated since 3.6 and will be
removed two minor releases later; explicitly call ax.remove() as
needed.
  plt.subplot(2, 2, 1)
```




```
df.shape
```

```
(72185, 9)
```

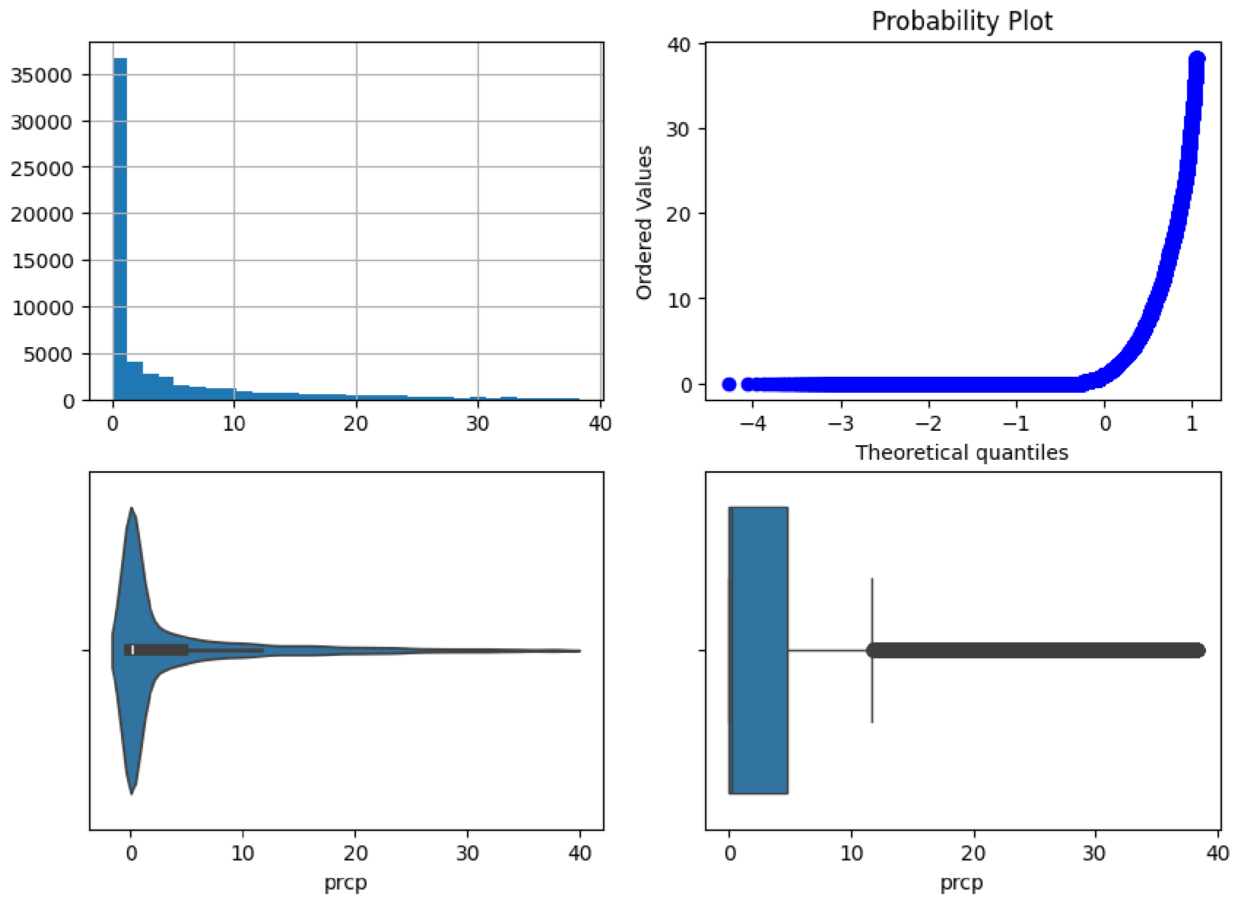
Удаляем выбросы. Как можно увидеть, были удаленны только далекие от медианы выбросы, а группа ближайших (на ящике с усами) - осталась. Это показывает, что распределение было немного ассиметричным (наклоненным). Но метод сработал хорошо и не удалил группу значений, не являющуюся выбросами

```
col = 'prcp'

# Вычисление верхней и нижней границы
lower_boundary = df[col].mean() - (3 * df[col].std())
upper_boundary = df[col].mean() + (3 * df[col].std())
# Флаги для удаления выбросов
outliers_temp = np.where(df[col] > upper_boundary, True,
                          np.where(df[col] < lower_boundary, True,
False))
# Удаление данных на основе флага
data_trimmed = df.loc[~(outliers_temp), ]

diagnostic_plots(data_trimmed, col)
```

```
<ipython-input-20-c2e6bc34c22e>:4: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(2, 2, 1)
```



Количество строк уменьшилось

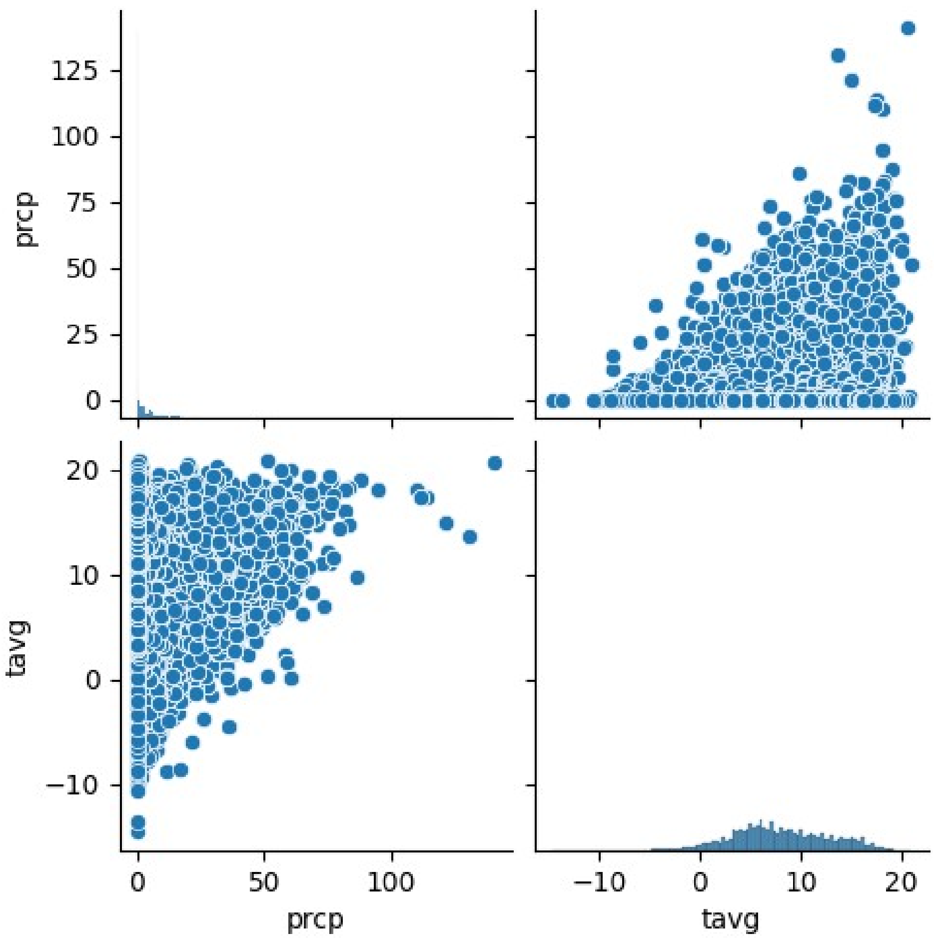
```
data_trimmed.shape
```

```
(70289, 9)
```

Построение графика по варианту

```
sns.pairplot(df, vars=['prcp', 'tavg'])
```

```
<seaborn.axisgrid.PairGrid at 0x79df0d2472b0>
```



prcp
125
100
75
50
25
0
tavg
20
10
0
−10
0
50
100
prcp
−10
0
10
20
tavg