МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Лабораторная работа №\_\_3\_\_**

по дисциплине«Методы машинного обучения»

Тема: «Обработка признаков (часть 2).»

ИСПОЛНИТЕЛЬ: Лу Сяои

ФИО

группа ИУ5И-22М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"1" Июнь 2023 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_2023 г.

Москва - 2023

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## **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:
      1. один метод из группы методов фильтрации (filter methods);
      2. один метод из группы методов обертывания (wrapper methods);
      3. один метод из группы методов вложений (embedded methods).

**текст программы и экранные формы с примерами выполнения**

Использование набора данных Sleep\_Efficiency.csv

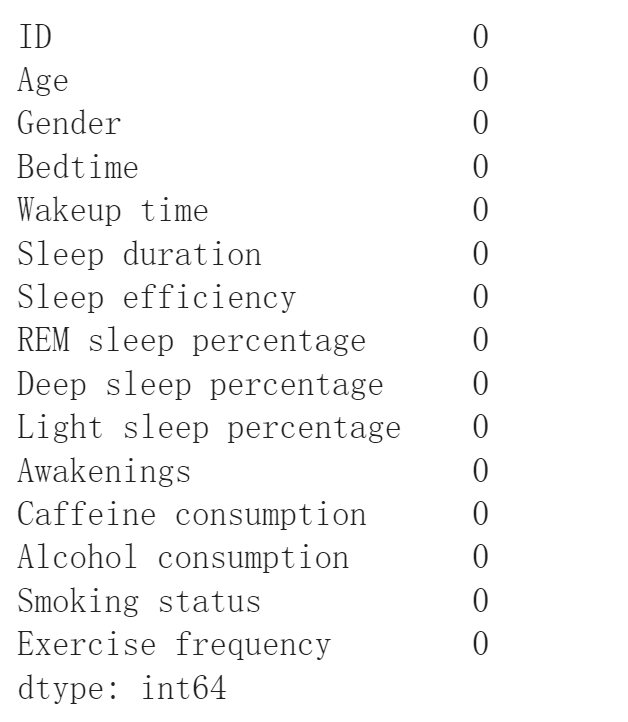
# подключение библиотек  
import numpy as np # linear algebra  
import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)  
import random  
import math as math  
import seaborn as sns #  
import matplotlib.pyplot as plt  
import missingno as msno  
import plotly.graph\_objs as go  
import plotly.express as px #  
plt.style.use('seaborn-dark')  
plt.style.context('grayscale')  
%matplotlib inline  
import re  
from wordcloud import WordCloud, STOPWORDS

from google.colab import drive  
drive.mount('/content/drive')  
df = pd.read\_csv('/content/drive/My Drive/Sleep\_Efficiency.csv')

Mounted at /content/drive

# replace null values with mean  
df['Awakenings'].fillna(df['Awakenings'].mean(), inplace=True)  
df['Caffeine consumption'].fillna(df['Caffeine consumption'].mean(), inplace=True)  
df['Alcohol consumption'].fillna(df['Alcohol consumption'].mean(), inplace=True)  
df['Exercise frequency'].fillna(df['Exercise frequency'].mean(), inplace=True)

# replace null values with mean  
df['Awakenings'].fillna(df['Awakenings'].mean(), inplace=True)  
df['Caffeine consumption'].fillna(df['Caffeine consumption'].mean(), inplace=True)  
df['Alcohol consumption'].fillna(df['Alcohol consumption'].mean(), inplace=True)  
df['Exercise frequency'].fillna(df['Exercise frequency'].mean(), inplace=True)  
# checking for null values  
df.isnull().sum()



df.columns = [col.replace(' ', '\_').lower() for col in df.columns]  
print(df.columns)

Index(['id', 'age', 'gender', 'bedtime', 'wakeup\_time', 'sleep\_duration',

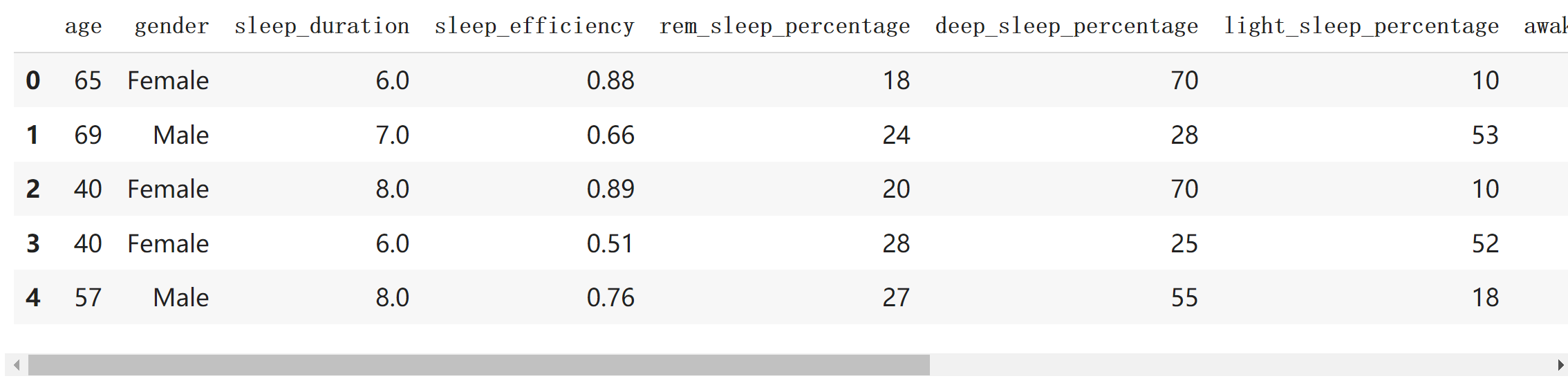
'sleep\_efficiency', 'rem\_sleep\_percentage', 'deep\_sleep\_percentage',

'light\_sleep\_percentage', 'awakenings', 'caffeine\_consumption',

'alcohol\_consumption', 'smoking\_status', 'exercise\_frequency'],

dtype='object')

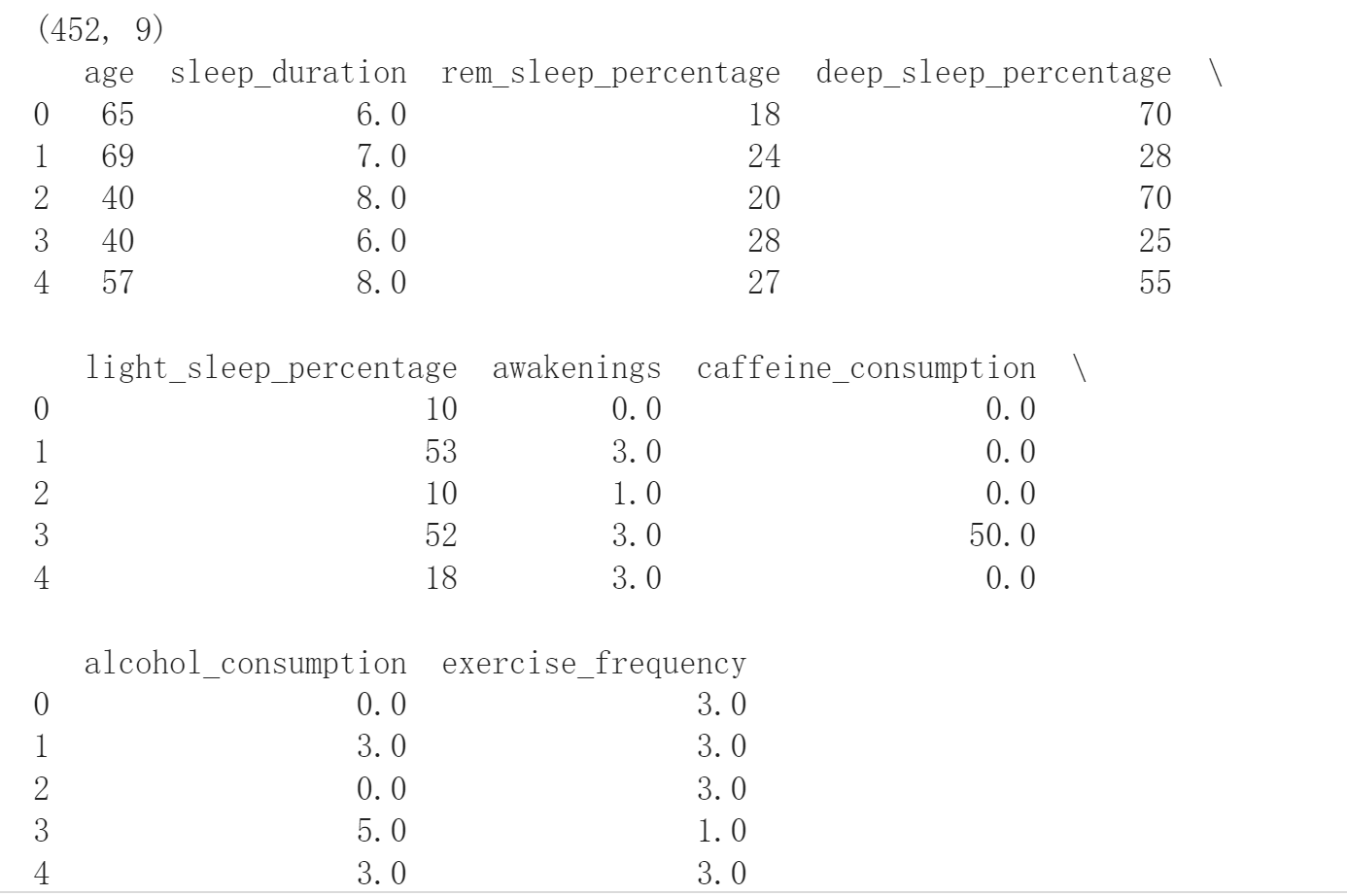
df = df.drop(['id', 'bedtime', 'wakeup\_time'], axis='columns')  
df.head()



# 从 DataFrame 对象中提取特征矩阵 X  
X\_cate = df.loc[:, ['gender', 'smoking\_status']]  
  
# 查看特征矩阵的维度大小  
print(X\_cate.shape)

(452, 2)

X\_num = df.loc[:, ['age','sleep\_duration','rem\_sleep\_percentage', 'deep\_sleep\_percentage',  
       'light\_sleep\_percentage', 'awakenings', 'caffeine\_consumption',  
       'alcohol\_consumption','exercise\_frequency']]  
print(X\_num.shape)  
print(X\_num.head())



# масштабирование признаков (не менее чем тремя способами);

## 1.Min-Max

from sklearn.preprocessing import MinMaxScaler

# 创建特征缩放器对象

scaler = MinMaxScaler()

# 对特征矩阵进行缩放

X\_num\_scaled = scaler.fit\_transform(X\_num)

import numpy as np  
  
print("Mean of X\_num\_scaled:", np.mean(X\_num\_scaled, axis=0))  
print("Standard deviation of X\_num\_scaled:", np.std(X\_num\_scaled, axis=0))  
print("Minimum value of X\_num\_scaled:", np.min(X\_num\_scaled, axis=0))  
print("Maximum value of X\_num\_scaled:", np.max(X\_num\_scaled, axis=0))

Mean of X\_num\_scaled: [ 1.24086187e-02 -3.42920354e-02 1.43173198e-01 -5.99133873e-01

5.57702727e-01 3.20601852e-01 -3.92999301e-18 6.22706422e-01

-6.95067265e-02]

Standard deviation of X\_num\_scaled: [0.57207264 0.86566573 0.56686698 1.32293529 1.23542896 0.66243491

0.58642512 0.80606826 0.47234421]

Minimum value of X\_num\_scaled: [-1.34782609 -2.5 -1. -3.40425532 -0.89795918 -0.5

-0.47306792 0. -0.66666667]

Maximum value of X\_num\_scaled: [1.26086957 2.5 1.14285714 1.27659574 3.10204082 1.5

3.52693208 2.5 1. ]

## 2.Z-Score

from sklearn.preprocessing import StandardScaler  
  
# 创建特征缩放器对象  
scaler = StandardScaler()  
  
# 对特征矩阵进行缩放  
X\_num\_scaled = scaler.fit\_transform(X\_num)

import numpy as np  
  
print("Mean of X\_num\_scaled:", np.mean(X\_num\_scaled, axis=0))  
print("Standard deviation of X\_num\_scaled:", np.std(X\_num\_scaled, axis=0))  
print("Minimum value of X\_num\_scaled:", np.min(X\_num\_scaled, axis=0))  
print("Maximum value of X\_num\_scaled:", np.max(X\_num\_scaled, axis=0))

Mean of X\_num\_scaled: [-4.91249126e-17 -2.90819483e-16 -1.19864787e-16 1.33619762e-16

-4.71599161e-17 -2.20079608e-16 -7.85998602e-17 -7.85998602e-18

0.00000000e+00]

Standard deviation of X\_num\_scaled: [1. 1. 1. 1. 1. 1. 1. 1. 1.]

Minimum value of X\_num\_scaled: [-2.37773074 -2.84833728 -2.0166516 -2.12037692 -1.17826436 -1.23876602

-0.8066979 -0.77252319 -1.2642474 ]

Maximum value of X\_num\_scaled: [2.18234688 2.92756425 1.76352475 1.4178544 2.05947745 1.78039855

6.01429225 2.32895114 2.26425286]

## 3.Robust Scaling

from sklearn.preprocessing import RobustScaler  
  
# 创建特征缩放器对象  
scaler = RobustScaler()  
  
# 对特征矩阵进行缩放  
X\_num\_scaled = scaler.fit\_transform(X\_num)

import numpy as np  
  
print("Mean of X\_num\_scaled:", np.mean(X\_num\_scaled, axis=0))  
print("Standard deviation of X\_num\_scaled:", np.std(X\_num\_scaled, axis=0))  
print("Minimum value of X\_num\_scaled:", np.min(X\_num\_scaled, axis=0))  
print("Maximum value of X\_num\_scaled:", np.max(X\_num\_scaled, axis=0))

Mean of X\_num\_scaled: [ 1.24086187e-02 -3.42920354e-02 1.43173198e-01 -5.99133873e-01

5.57702727e-01 3.20601852e-01 -3.92999301e-18 6.22706422e-01

-6.95067265e-02]

Standard deviation of X\_num\_scaled: [0.57207264 0.86566573 0.56686698 1.32293529 1.23542896 0.66243491

0.58642512 0.80606826 0.47234421]

Minimum value of X\_num\_scaled: [-1.34782609 -2.5 -1. -3.40425532 -0.89795918 -0.5

-0.47306792 0. -0.66666667]

Maximum value of X\_num\_scaled: [1.26086957 2.5 1.14285714 1.27659574 3.10204082 1.5

3.52693208 2.5 1. ]

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

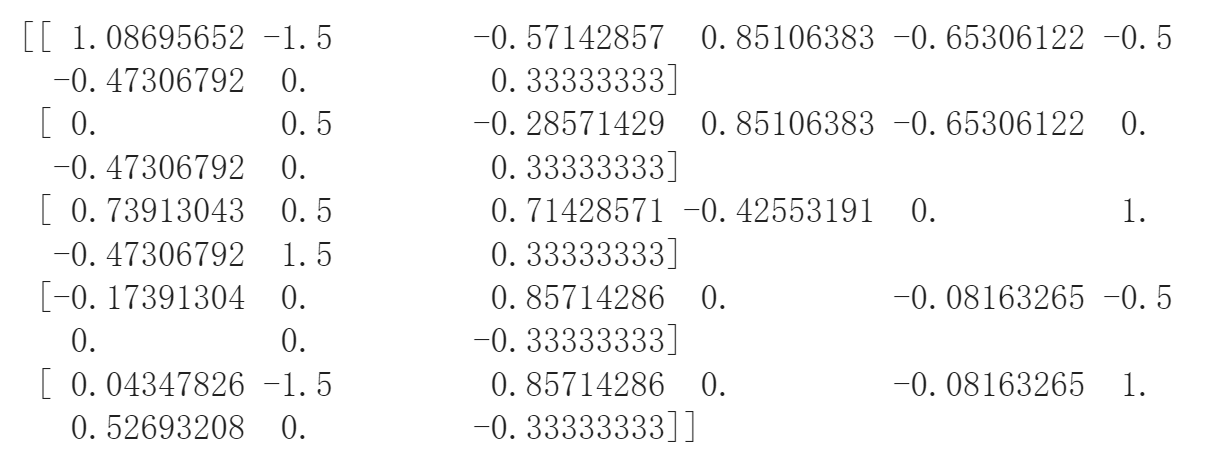
## 1.Удаление выбросов:

Выбросы можно обнаружить и удалить из числовых атрибутов с помощью среднего и стандартного отклонения. Если значение числового атрибута после масштабирования отличается от 2,5 стандартных отклонений, он может быть помечен как выброс и удален из набора данных.

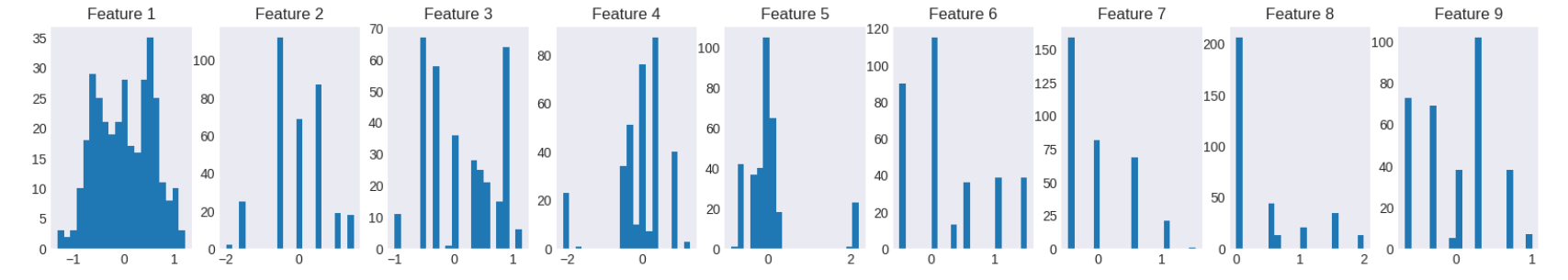
Сначала мы вычисляем, превышает ли масштабированное значение каждого признака 2,5 стандартных отклонения, и создаем маску outlier\_mask для определения всех значений, которые превышают 2,5 стандартных отклонения в масштабированном значении. Затем мы используем функцию all для вычисления маски для матрицы признаков и используем outlier\_mask для удаления промахов из матрицы признаков.

import numpy as np  
  
# 计算每个特征在缩放后的值是否超过2.5个标准差  
outlier\_mask = (np.abs(X\_num\_scaled) < 2.5).all(axis=1)  
  
# 从特征矩阵中删除异常值  
X\_num\_cleaned = X\_num\_scaled[outlier\_mask, :]

print(X\_num\_cleaned[:5,:])



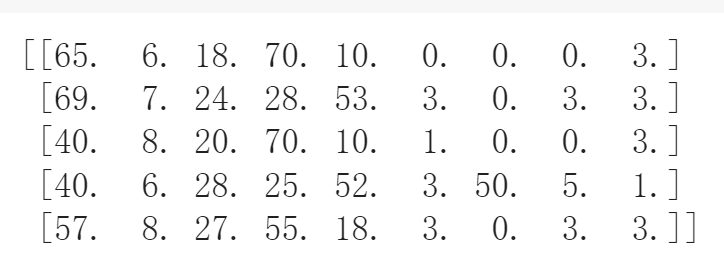
fig, axs = plt.subplots(1, X\_num\_cleaned.shape[1], figsize=(20, 3))  
  
for i in range(X\_num\_cleaned.shape[1]):  
    axs[i].hist(X\_num\_cleaned[:, i], bins=20)  
    axs[i].set\_title("Feature " + str(i+1))  
  
plt.show()



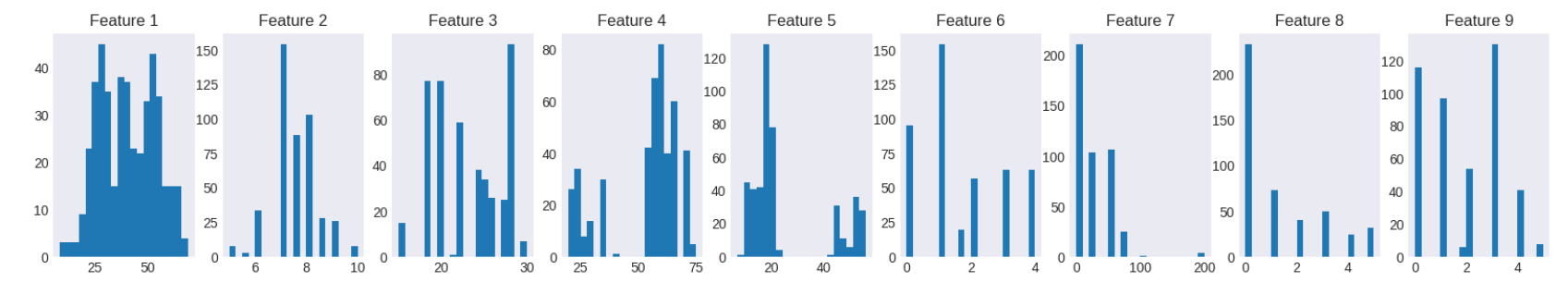
## 2.Замена выбросов:

С выбросами можно работать с помощью методов, основанных на интерполяции, таких как интерполяция среднего, интерполяция медианы, интерполяция K-ближайшего соседа и т.д. Например, мы можем использовать медиану для замены выбросов в числовых атрибутах. В этом примере мы сначала импортировали класс SimpleImputer и создали объект интерполятора, используя стратегию медианы. Затем мы используем метод fit\_transform для интерполяции медианы для значений выбросов в числовом атрибуте, сохраняя результат в X\_num\_imputed.

from sklearn.impute import SimpleImputer  
  
# 创建插补器对象  
imputer = SimpleImputer(strategy='median')  
  
# 使用中位数插补替换异常值  
X\_num\_imputed = imputer.fit\_transform(X\_num)

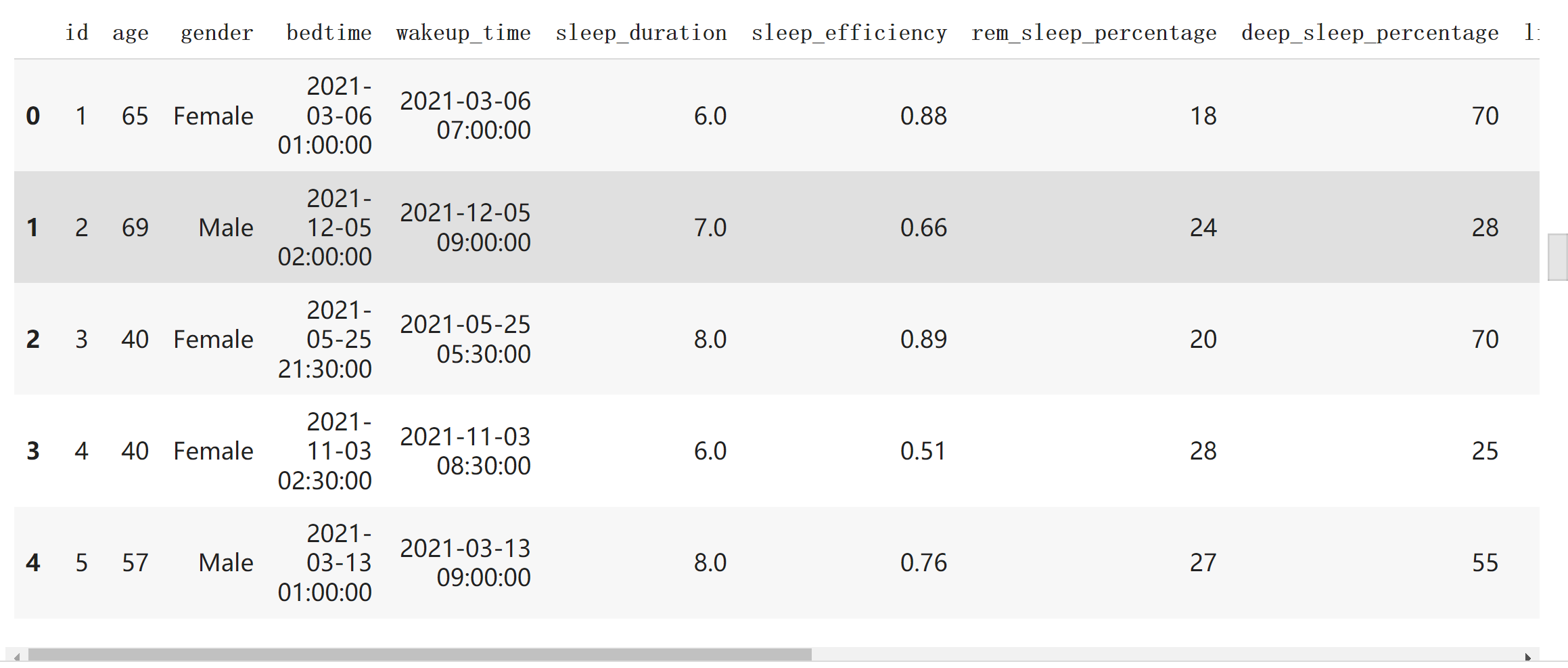


# 使用matplotlib进行可视化  
import matplotlib.pyplot as plt  
  
fig, axs = plt.subplots(1, X\_num\_imputed.shape[1], figsize=(20, 3))  
  
for i in range(X\_num\_imputed.shape[1]):  
    axs[i].hist(X\_num\_imputed[:, i], bins=20)  
    axs[i].set\_title("Feature " + str(i+1))  
  
plt.show()

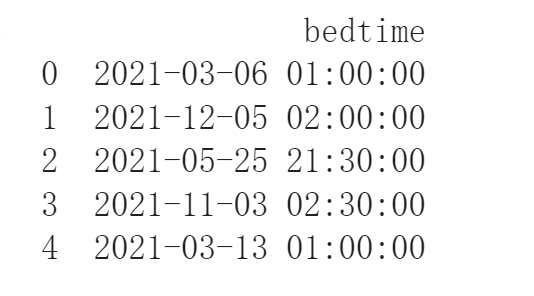


# обработку по крайней мере одного нестандартного признака (который не является числовым или категориальным);

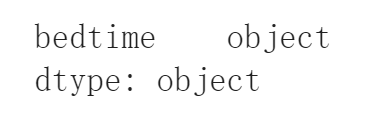
df.head()



X\_t = df.loc[:, ['bedtime']]  
print(X\_t.head())



X\_t.dtypes



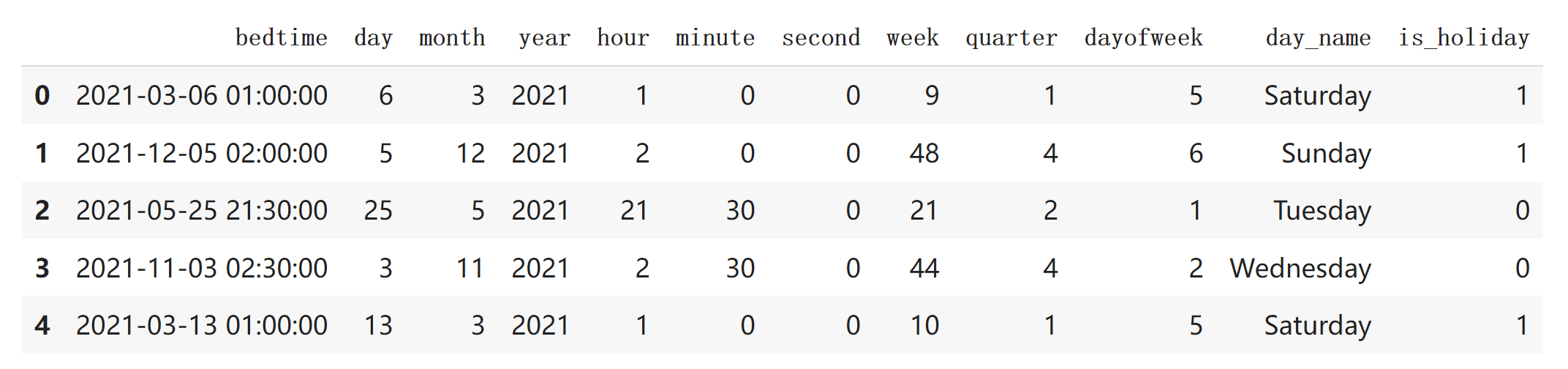
X\_t['bedtime'] = pd.to\_datetime(X\_t['bedtime'])  
X\_t.dtypes



## Выделение стандартных признаков даты и времени

# День  
X\_t['day'] = X\_t['bedtime'].dt.day  
# Месяц  
X\_t['month'] = X\_t['bedtime'].dt.month  
# Год  
X\_t['year'] = X\_t['bedtime'].dt.year  
# Часы  
X\_t['hour'] = X\_t['bedtime'].dt.hour  
#Минуты  
X\_t['minute'] = X\_t['bedtime'].dt.minute  
#Секунды  
X\_t['second'] = X\_t['bedtime'].dt.second  
#Неделя года  
X\_t['week'] = X\_t['bedtime'].dt.isocalendar().week  
#Квартал  
X\_t['quarter'] = X\_t['bedtime'].dt.quarter  
#День недели   
X\_t['dayofweek'] = X\_t['bedtime'].dt.dayofweek  
#Выходной день  
X\_t['day\_name'] = X\_t['bedtime'].dt.day\_name()  
X\_t['is\_holiday'] = X\_t.apply(lambda x: 1 if x['bedtime'].dayofweek in [5,6] else 0, axis=1)

X\_t.head()



from sklearn.preprocessing import MinMaxScaler  
# Создадим масштабируемые признаки для дальнейших экспериментов  
dt\_features = ['year', 'day', 'month', 'hour', 'minute', 'second', 'week', 'quarter', 'dayofweek']  
dt\_features\_scaled = []  
for f in dt\_features:  
    f\_new = str(f + '\_scaled')  
    dt\_features\_scaled.append(f\_new)  
    X\_t[f\_new] = MinMaxScaler().fit\_transform(X\_t[[f]])  
dt\_features\_scaled

['year\_scaled',

'day\_scaled',

'month\_scaled',

'hour\_scaled',

'minute\_scaled',

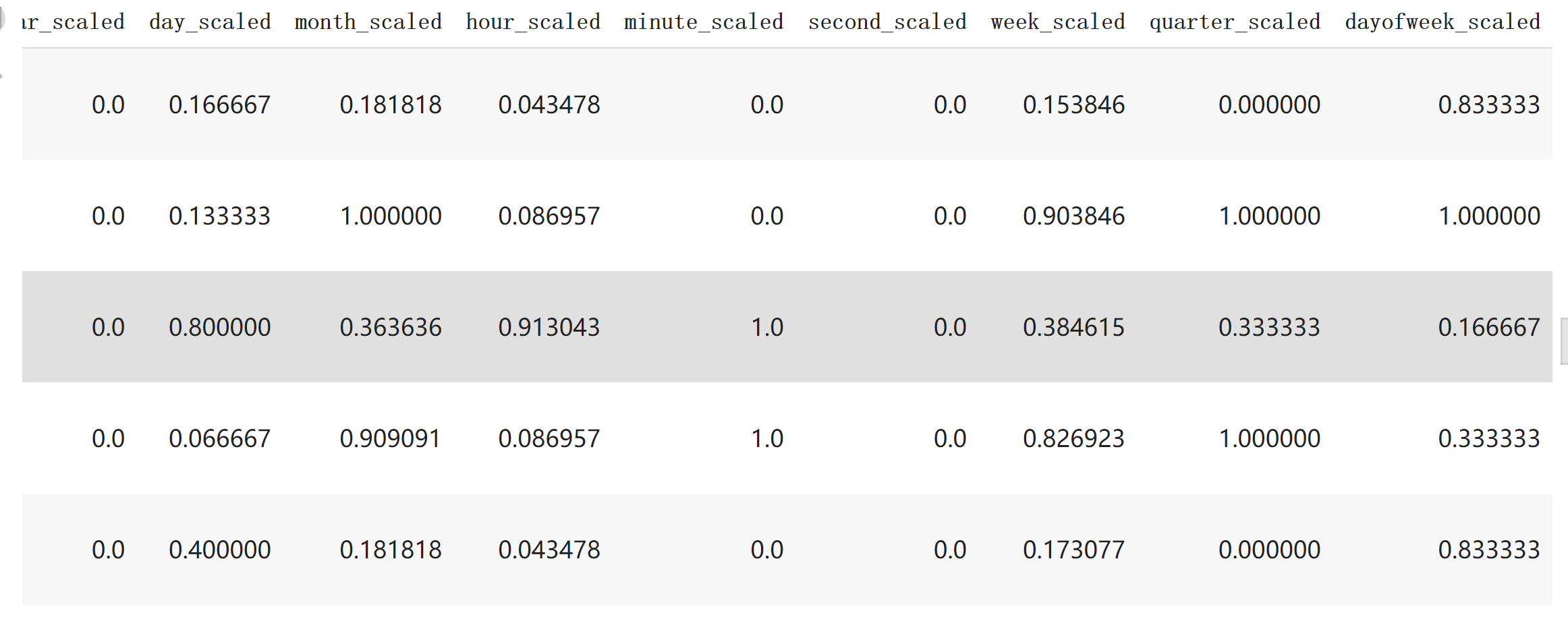
'second\_scaled',

'week\_scaled',

'quarter\_scaled',

'dayofweek\_scaled']

X\_t.head()



## Временные интервалы

import datetime  
# Разница между датами   
X\_t['now'] = datetime.datetime.today()  
X\_t['diff'] = X\_t['now'] - X\_t['bedtime']  
X\_t.dtypes

bedtime datetime64[ns]

day int64

month int64

year int64

hour int64

minute int64

second int64

week UInt32

quarter int64

dayofweek int64

day\_name object

is\_holiday int64

year\_scaled float64

day\_scaled float64

month\_scaled float64

hour\_scaled float64

minute\_scaled float64

second\_scaled float64

week\_scaled float64

quarter\_scaled float64

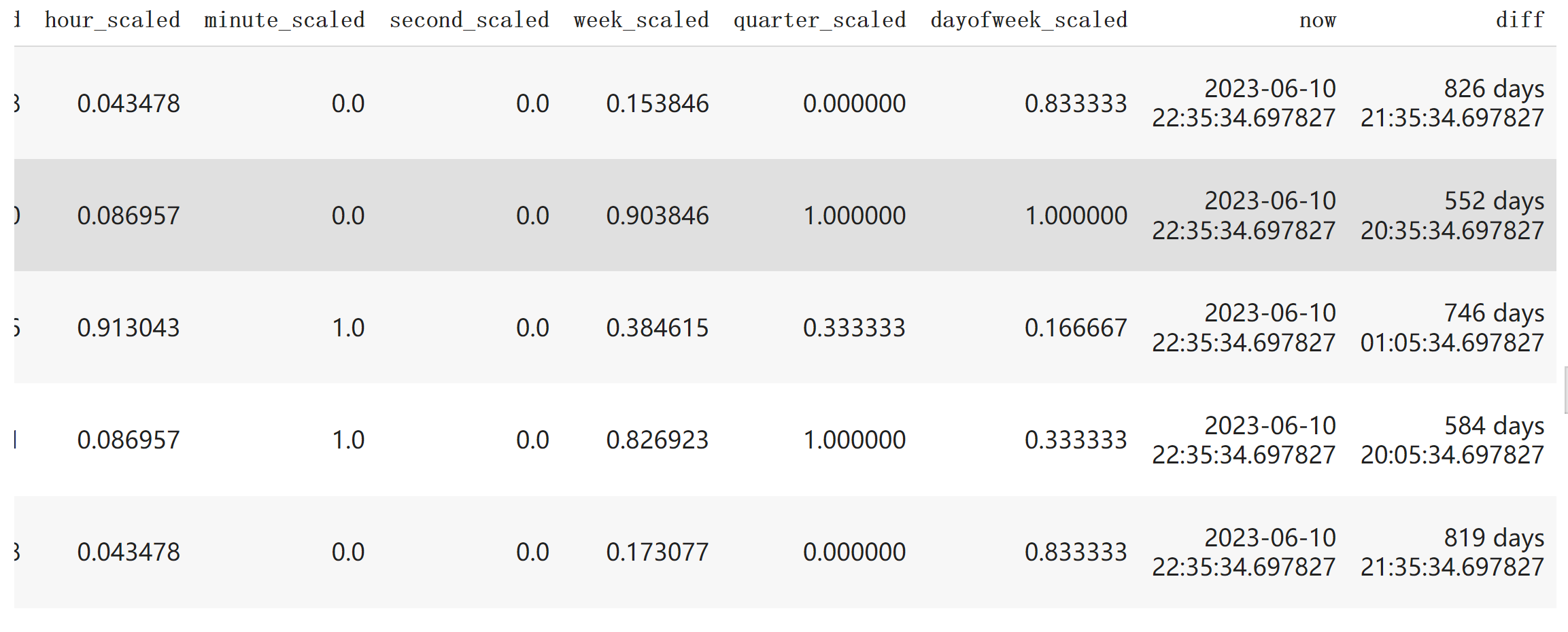
dayofweek\_scaled float64

now datetime64[ns]

diff timedelta64[ns]

dtype: object

X\_t.head()



# Чтобы получить разницу между датами в заданных единицах   
# (минутах, днях и т.д.) нужно разделить timedelta64 на длительность  
X\_t['diff\_days']=X\_t['diff']/np.timedelta64(1,'D')  
X\_t.dtypes

bedtime datetime64[ns]

day int64

month int64

year int64

hour int64

minute int64

second int64

week UInt32

quarter int64

dayofweek int64

day\_name object

is\_holiday int64

year\_scaled float64

day\_scaled float64

month\_scaled float64

hour\_scaled float64

minute\_scaled float64

second\_scaled float64

week\_scaled float64

quarter\_scaled float64

dayofweek\_scaled float64

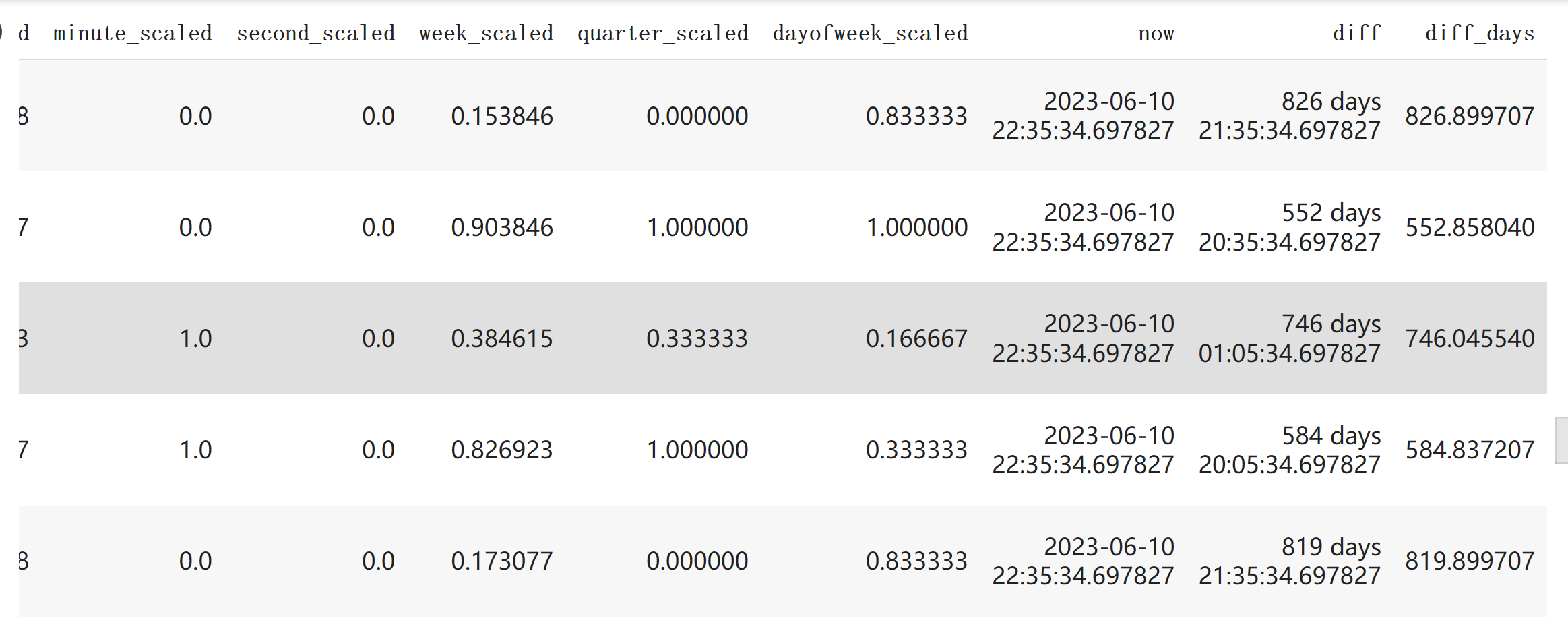
now datetime64[ns]

diff timedelta64[ns]

diff\_days float64

dtype: object

X\_t.head()



## Отображение на круг

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)

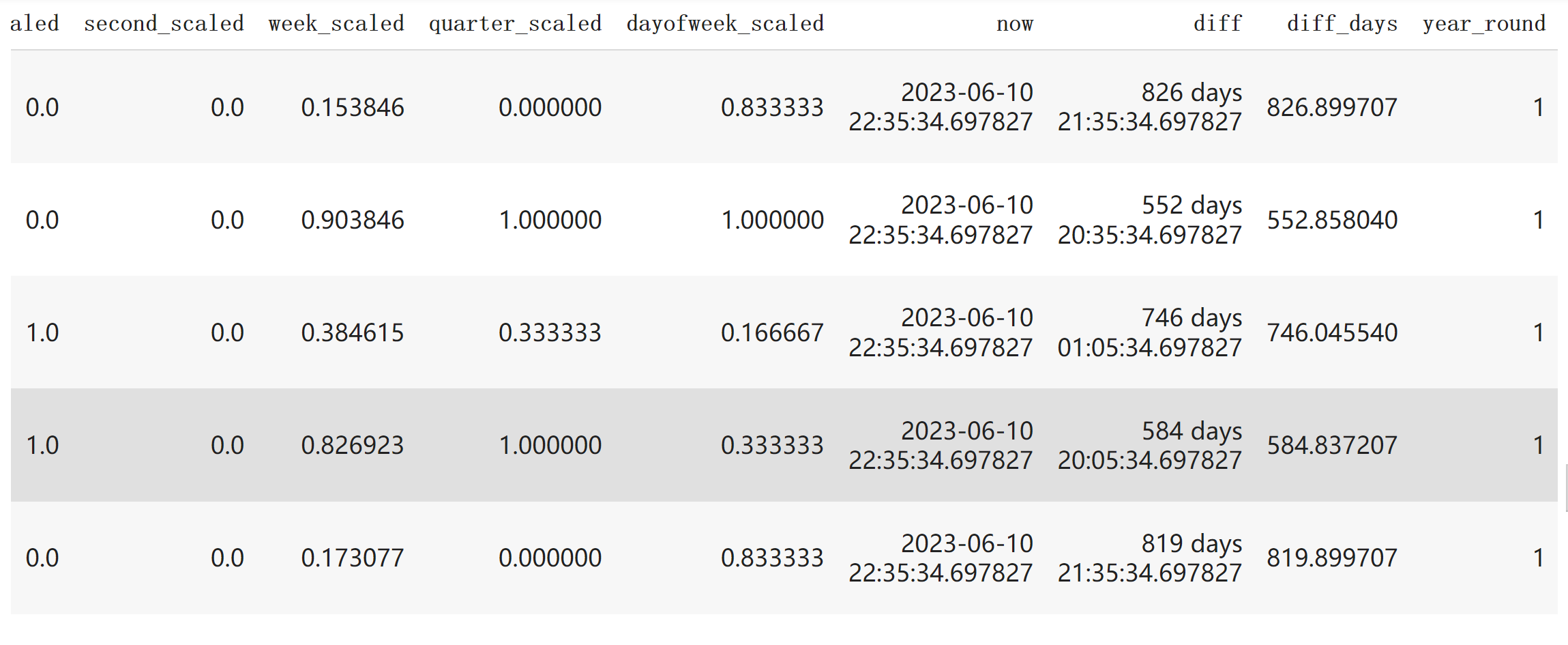
# Опеределим периоды для признаков  
for f in dt\_features:  
    print(f, X\_t[f].min(), X\_t[f].max())



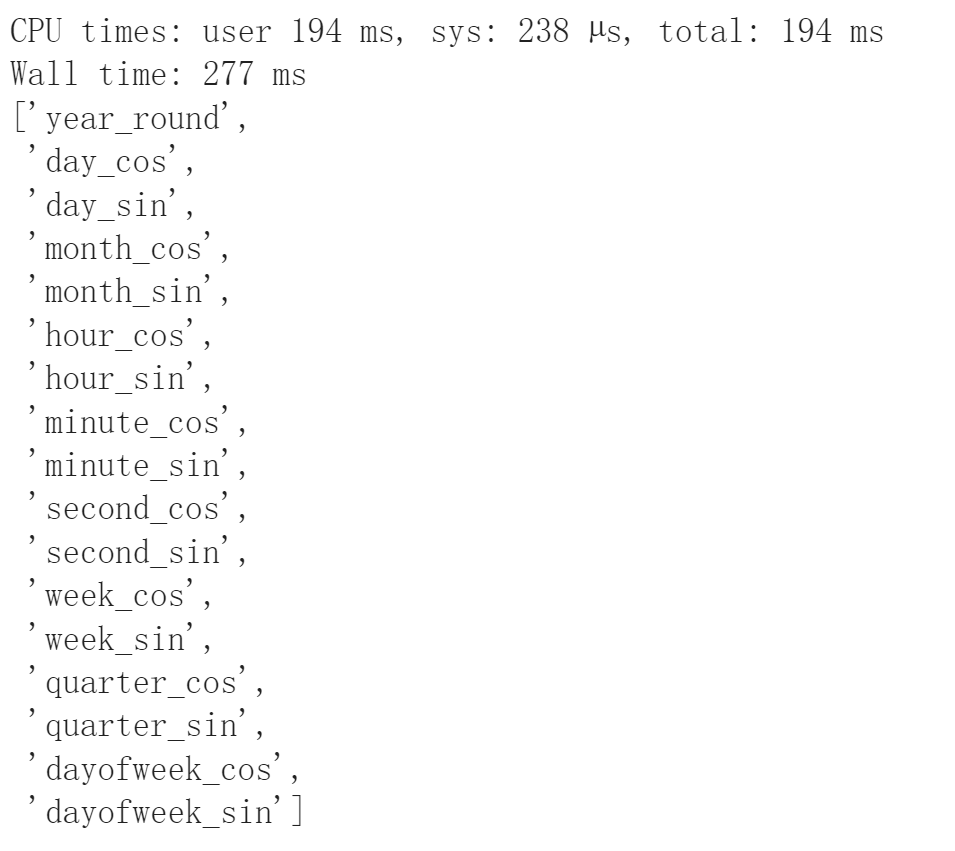
# периоды для признаков  
dt\_features\_periods = [0, 31, 12, 24, 60, 60, 52, 4, 7]

X\_t['year\_round'] = X\_t.apply(lambda x: 1 if x['year']==2021 else 0, axis=1)  
dt\_features\_round = ['year\_round']

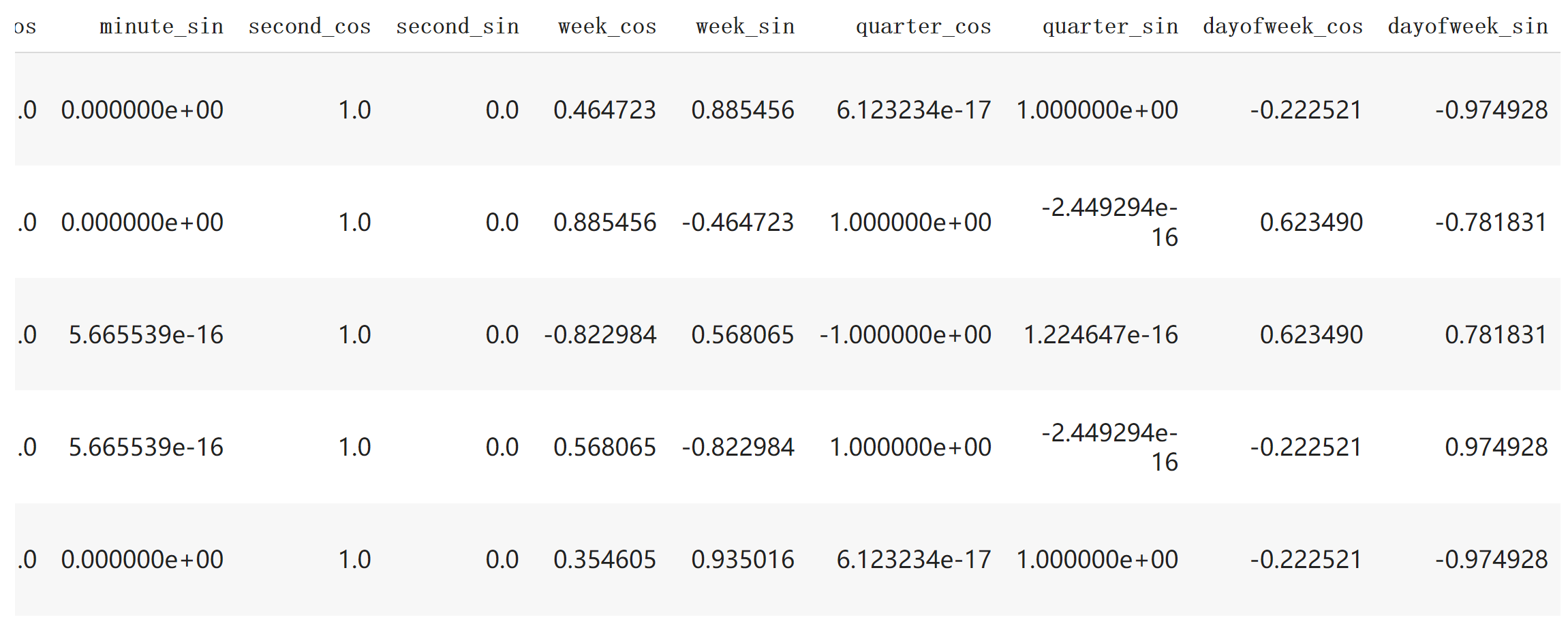
X\_t.head()



%%time  
# Построим отображение признаков на круг, год пропускаем  
for f,p in zip(dt\_features[1:], dt\_features\_periods[1:]):  
    f\_cos = str(f + '\_cos')  
    f\_sin = str(f + '\_sin')  
    X\_t[f\_cos] = X\_t.apply(lambda x: round\_code(x[f], p), axis=1)  
    X\_t[f\_sin] = X\_t.apply(lambda x: round\_code(x[f], p, False), axis=1)  
    dt\_features\_round.append(f\_cos)  
    dt\_features\_round.append(f\_sin)  
dt\_features\_round

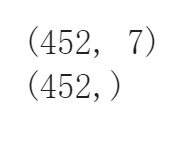


X\_t.head()



# отбор признаков:

# 将所有字符串类型的特征进行独热编码  
from sklearn.feature\_selection import VarianceThreshold  
  
# 将gender, smoking\_status进行特征转换  
gender\_new = pd.Series(df['gender'].replace({'Male': 0, 'Female': 1}), name='gender\_new')  
smoking\_status\_new = pd.Series(df['smoking\_status'].replace({'Yes': 0, 'No': 1}), name='smoking\_status\_new')  
  
# 创建新的X\_tFrame对象并与原始数据拼接  
df\_new = pd.concat([df, gender\_new, smoking\_status\_new], axis=1)  
  
X= df\_new.loc[:, ['gender\_new', 'smoking\_status\_new', 'age', 'sleep\_duration', 'caffeine\_consumption',  
       'alcohol\_consumption','exercise\_frequency']]  
print(X.shape)  
y = df\_new.loc[:, 'sleep\_efficiency']  
print(y.shape)

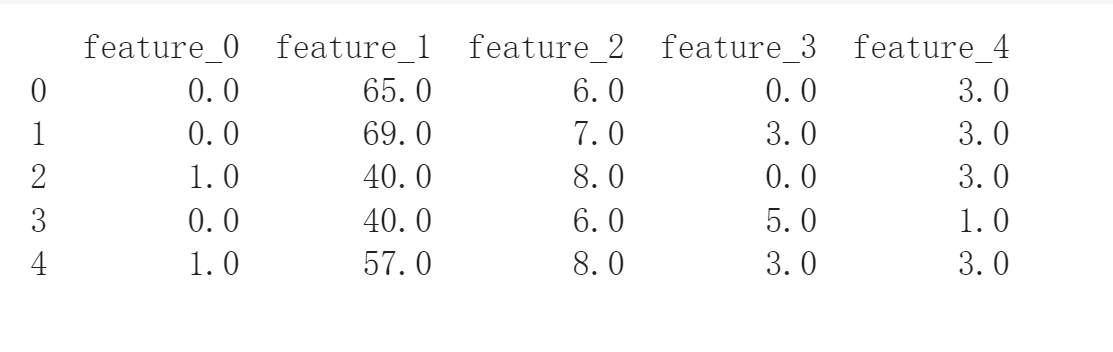


## 1.один метод из группы методов фильтрации (filter methods);

Метод отбора по дисперсии, метод отбора по дисперсии может быть использован для отбора признаков путем удаления признаков с малой дисперсией

from sklearn.feature\_selection import VarianceThreshold  
  
selector = VarianceThreshold(threshold=0.1)  # 方差阈值为0.1，即删除方差小于0.1的特征  
X\_new = selector.fit\_transform(X)  # 进行特征选择

# 创建特征矩阵的列名列表  
features = ['feature\_{}'.format(i) for i in range(X\_new.shape[1])]  
  
# 将选择后的特征矩阵Datanew转化为DataFrame对象  
df\_X\_new = pd.DataFrame(X\_new, columns=features)  
  
# 查看选择后的特征矩阵X\_new  
print(df\_X\_new.head())  
  
# # 可视化选择后的特征矩阵X\_new  
# import seaborn as sns  
  
# sns.pairplot(df\_X\_new)

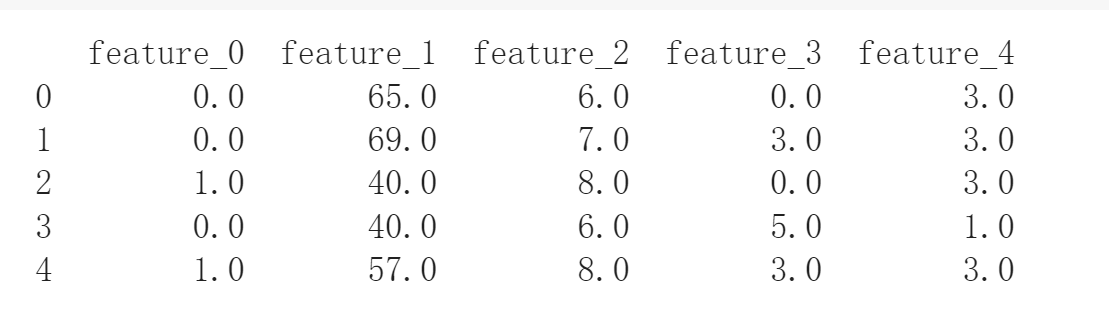


## 2.один метод из группы методов обертывания (wrapper methods);

Рекурсивное устранение признаков (RFE) - это метод отбора признаков на основе модели, который можно использовать для постепенного устранения признаков путем обучения модели и последующего выбора оптимального подмножества признаков. Мы использовали модуль RFE библиотеки scikit-learn, задали базовую модель в виде случайного леса и выбрали количество признаков для отбора (т.е. 5 признаков).

from sklearn.feature\_selection import RFE  
from sklearn.ensemble import RandomForestRegressor  
  
estimator = RandomForestRegressor(n\_estimators=50, random\_state=0)  # 利用随机森林作为基础模型  
selector = RFE(estimator, n\_features\_to\_select=5, step=1)  # 选择5个特征  
X\_new = selector.fit\_transform(X, y)  # 进行特征选择

# 创建特征矩阵的列名列表  
features = ['feature\_{}'.format(i) for i in range(X\_new.shape[1])]  
  
# 将选择后的特征矩阵X\_new转化为DataFrame对象  
df\_X\_new = pd.DataFrame(X\_new, columns=features)  
  
# 查看选择后的特征矩阵X\_new  
print(df\_X\_new.head())



## 3.один метод из группы методов вложений (embedded methods).

Выбор признаков на основе дерева - это метод встраивания, при котором древовидная модель может быть обучена для выбора оптимального подмножества признаков. Базовая модель задается в виде случайного леса, а для отбираемых признаков выбирается порог важности, т.е. признаки со средней важностью, превышающей это значение, сохраняются.

from sklearn.feature\_selection import SelectFromModel  
from sklearn.ensemble import RandomForestRegressor  
  
estimator = RandomForestRegressor(n\_estimators=50, random\_state=0)  # 利用随机森林作为基础模型  
selector = SelectFromModel(estimator, threshold='mean')  # 选择平均重要性大于该值的特征  
X\_new = selector.fit\_transform(X, y)  # 进行特征选择

# 创建特征矩阵的列名列表  
features = ['feature\_{}'.format(i) for i in range(X\_new.shape[1])]  
  
# 将选择后的特征矩阵X\_new转化为DataFrame对象  
df\_X\_new = pd.DataFrame(X\_new, columns=features)  
  
# 查看选择后的特征矩阵X\_new  
print(df\_X\_new.head())

