Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет»

Институт Компьютерных Наук и Технологий

Лабораторная работа № 4

по дисциплине

«Введение в анализ данных»

Отчет

Студент Панькова Светлана

Группа ИТ-13-2023

Пермь 2025

Задача №1.

1. Импортируем нужные нам библиотеки

```
import numpy

import pandas
```

2. Сгенерировать случайным образом массив numpy из 1000 значений нормально распределенной случайной величины (numpy.random.normal) с мат. ожиданием M = 1.0 и стандартным отклонением s = 1.0. Преобразовать его в объект Series.

```
# 1. Сгенерировать случайным образом массив numpy из 1000 значений нормально распределенной случайной величины (numpy.random.normal) с мат. ожиданием M = 1.0 и стандартным отклонением s = 1.0

# Преобразовать его в объект Series

M = 1.0

s = 1.0

data = numpy.random.normal(M, s, 1000)

# Преобразование в объект Series

data_series = pandas.Series(data)

print('Массив, преобразованный в Series')

print(data_series)
```

```
Массив, преобразованный в Series
0     -0.534770
1      0.512389
2     -0.397345
3      1.324712
4      0.319313
          ...
995    2.378723
996    1.100152
997    1.426004
998    1.894008
999    1.603863
Length: 1000, dtype: float64
```

3. Вычислить, какая доля всех значений находится в диапазоне (M-s; M+s).

```
# 2. Вычислить, какая доля всех значений находится в диапазоне (M-s; M+s)

dolya = ((data_series > M - s) & (data_series < M + s)).mean()

print('\nДоля значений в диапазоне (M-s; M+s)')

print(dolya)
```

```
Доля значений в диапазоне (M-s; M+s)
0.684
```

4. Вычислить, какая доля всех значений находится в диапазоне (M-3s; M+3s). Какой должна быть эта доля теоретически? Насколько реальный результат соответствует теории?

```
# 3. Вычислить, какая доля всех значений находится в диапазоне (M-3s; M+3s)

# Какой должна быть эта доля теоретически? Насколько реальный результат соответствует теории?

dolya3 = ((data_series > M - 3 * s) & (data_series < M + 3 * s)).mean()

print('Доля значений в диапазоне (M-3s; M+3s)')
```

```
print(dolya3)

# Теоретическая доля для диапазона (M-3s; M+3s) равна примерно 0.997

# Правило 68-95-99 гласит, что для данных с нормальным распределением:

# Распределение данных составляет примерно 68% в пределах одного стандартного отклонения от среднего.

# Два стандартных отклонения от среднего значения охватывают около 95% данных.

# Три стандартных отклонения от среднего значения включают 99,7% данных.

theor_dolya = 0.997

print('Теоритическая доля:', theor_dolya, 'близка к программно полученному значению доли\n')
```

```
Доля значений в диапазоне (M-3s; M+3s)
0.999
Теоритическая доля: 0.997 близка к программно полученному значению доли
```

Иногда значение программно полученной доли не совпадает полностью, а только близко к значению теоретически полученной доли.

5. Заменить каждое значение x в серии на его квадратный корень (numpy.sqrt(x)). Результат записать в новый объект Series. Почему возникает предупреждение, и что происходит с теми значениями, для которых возникает предупреждение?

```
# 4. Заменить каждое значение x в серии на его квадратный корень (numpy.sqrt(x)).

# Результат записать в новый объект Series. Почему возникает предупреждение,

# и что происходит с теми значениями, для которых возникает предупреждение?

sqrt_data_series = numpy.sqrt(data_series)

print('Замена значений в серии на их квадратный корень')

print(sqrt_data_series)
```

```
# Предупреждение возникает для отрицательных значений, так как корень из отрицательного числа

# не определен в действительных числах. Эти значения заменяются на NaN.
```

```
Замена значений в серии на их квадратный корень
0           NaN
1      0.715814
2           NaN
3      1.150961
4      0.565078
          ...
995    1.542311
996    1.048881
997    1.194154
998    1.376230
999    1.266437
Length: 1000, dtype: float64
```

6. Посчитать среднее арифметическое для получившихся значений. Отсутствующие значения (NaN) учитываться не должны.

```
# 5. Посчитать среднее арифметическое для получившихся значений.

# Отсутствующие значения (NaN) учитываться не должны. Если True, то значения NAN/null исключаются из расчета.

mean_sqrt_data_series = sqrt_data_series.mean(skipna=True)

print('\nСреднее арифметическое для получившихся значений.')

print(mean_sqrt_data_series, '\n')
```

```
Среднее арифметическое для получившихся значений.
1.076167479990794
```

7. На основе двух объектов Series (исходного и полученного на шаге 4) создать DataFrame с двумя столбцами. Названия (явные индексы) для столбцов: «number» и «root» соответственно. Явные индексы для строк не задавать. Вывести первые 6 строк из созданного датафрейма.

```
# 6. На основе двух объектов Series (исходного и полученного на шаге 4 создать DataFrame с двумя столбцами. Названия (явные индексы) для столбцов: «number» и «root» соответственно. Явные индексы для строк не задавать. Вывести первые 6 строк из созданного датафрейма.

dataframe = pandas.DataFrame({
    'number': data_series,
    'root': sqrt_data_series})


# Вывод первых 6 строк из созданного датафрейма
print('Первые 6 строк из датафрейма')
print(dataframe.head(6))
```

```
Первые 6 строк из датафрейма
     number      root
0 -0.566780       NaN
1  1.659991  1.288407
2 -0.004580       NaN
3  2.317385  1.522296
4  0.408329  0.639006
5 -0.271405       NaN
```

8. С помощью функции query найти в датафрейме записи, в которых значение квадратного корня находится в диапазоне от 1.8 до 1.9. Вывести результат.

```
# 7. С помощью функции query найти в датафрейме записи, в которых
# значение квадратного корня находится в диапазоне от 1.8 до 1.9.
result_query = dataframe.query('1.8 <= root <= 1.9')
```

```
print('\nЗаписи, где значение квадратного корня находится
в диапазоне от 1.8 до 1.9: ')

print(result_query)
```

```
Записи, где значение квадратного корня находится в диапазоне от 1.8 до 1.9:
       number      root
159  3.288474  1.813415
499  3.533054  1.879642
852  3.498104  1.870322
884  3.313654  1.820345
901  3.275958  1.809961
```

Задача №2.

1. Загрузите данные из файла «athlete_events.csv» о спортсменах – участниках олимпийских игр (ОИ).

```
import pandas as pd

import matplotlib.pyplot as plt

import os


# 1: Загрузка данных с проверкой наличия файлов

if os.path.exists('athlete_events.csv'):

   data = pd.read_csv('athlete_events.csv')

else:

   print("Файл 'athlete_events.csv' не найден.")
```

2. Определите количество значений каждого из признаков в загруженных данных. По каким значениям имеются не все данные? По какому значению отсутствующих данных больше всего? Подсказка: воспользуйтесь функцией count или info.

```
# 2.Определите количество значений каждого из признаков в
загруженных данных.
```

```python
# По каким значениям имеются не все данные? По какому значению отсутствующих данных больше всего?

# Подсказка: воспользуйтесь функцией count или info.


# Количество значений каждого из признаков

print("Количество значений каждого из признаков:")

print(data.count())


# Поиск наибольшего количества отсутствующих значений

not_values = data.isnull().sum()  # Считаем количество пропущенных значений в каждом столбце

print("\nПризнаки с отсутствующими данными:")

print(not_values[not_values > 0])

print(f"\nПризнак с наибольшим количеством отсутствующих данных: {not_values.idxmax()} ({not_values.max()})")
```

```
Количество значений каждого из признаков:
ID        271116
Name      271116
Sex       271116
Age       261642
Height    210945
Weight    208241
Team      271116
NOC       271116
Games     271116
Year      271116
Season    271116
City      271116
Sport     271116
Event     271116
Medal      39783
dtype: int64
```

```
Признаки с отсутствующими данными:
Age         9474
Height     60171
Weight     62875
Medal     231333
dtype: int64
```

```
Признак с наибольшим количеством отсутствующих данных: Medal (231333)
```

3. Выведите статистическую информацию (среднее значение, стандартное отклонение, минимальное и максимальное значение, значение квартилей) по полям: возраст, рост, вес. Подсказка: воспользуйтесь функцией describe.

```
#  3.  Выведите  статистическую  информацию  стандартное
отклонение, минимальное и максимальное значение,

#  значение  квартилей)  по  полям:  возраст,  рост,  вес.
Подсказка: воспользуйтесь функцией describe.

print("\nСтатистическая информация по возрасту, росту и
весу:")
```

```
print(data[['Age', 'Height', 'Weight']].describe())
```

```
Статистическая информация по возрасту, росту и весу:
                Age         Height         Weight
count  261642.000000  210945.000000  208241.000000
mean       25.556898     175.338970      70.702393
std         6.393561      10.518462      14.348020
min        10.000000     127.000000      25.000000
25%        21.000000     168.000000      60.000000
50%        24.000000     175.000000      70.000000
75%        28.000000     183.000000      79.000000
max        97.000000     226.000000     214.000000
```

4. Сколько лет было самому молодому участнику олимпийских игр в 1992 году? Как звали этого участника и в какой дисциплине он(а) участвовал(а)?

```
# 1) Сколько лет было самому молодому участнику олимпийских игр в 1992 году?

# Как звали этого участника и в какой дисциплине он(а) участвовал(а)?

# Фильтруем данные по 1992 году, получаем одну строку с наименьшим значением в столбце возраст

young_1992 = data[data['Year'] == 1992].nsmallest(1, 'Age')

print(f"\nСамый молодой участник олимпийских игр в 1992 году: {young_1992['Name'].values[0]}, "

      f"{young_1992['Age'].values[0]} лет, \nдисциплина: {young_1992['Event'].values[0]}")
```

```
Самый молодой участник олимпийских игр в 1992 году: Carlos Bienvenido Front Barrera, 11.0 лет,
дисциплина: Rowing Men's Coxed Eights
```

5. Выведите список всех видов спорта, которые когда-либо входили в программу олимпийских игр. (Каждый вид спорта должен присутствовать в списке один раз.)

```python
# 2) Выведите список всех видов спорта, которые когда-либо входили в программу олимпийских игр.

# (Каждый вид спорта должен присутствовать в списке один раз.)

sports_list = data['Sport'].unique()  # Возвращаем список уникальных

print("\nСписок всех видов спорта:")

print(sports_list)
```

```
Список всех видов спорта:
['Basketball' 'Judo' 'Football' 'Tug-Of-War' 'Speed Skating'
 'Cross Country Skiing' 'Athletics' 'Ice Hockey' 'Swimming' 'Badminton'
 'Sailing' 'Biathlon' 'Gymnastics' 'Art Competitions' 'Alpine Skiing'
 'Handball' 'Weightlifting' 'Wrestling' 'Luge' 'Water Polo' 'Hockey'
 'Rowing' 'Bobsleigh' 'Fencing' 'Equestrianism' 'Shooting' 'Boxing'
 'Taekwondo' 'Cycling' 'Diving' 'Canoeing' 'Tennis' 'Modern Pentathlon'
 'Figure Skating' 'Golf' 'Softball' 'Archery' 'Volleyball'
 'Synchronized Swimming' 'Table Tennis' 'Nordic Combined' 'Baseball'
 'Rhythmic Gymnastics' 'Freestyle Skiing' 'Rugby Sevens' 'Trampolining'
 'Beach Volleyball' 'Triathlon' 'Ski Jumping' 'Curling' 'Snowboarding'
 'Rugby' 'Short Track Speed Skating' 'Skeleton' 'Lacrosse' 'Polo'
 'Cricket' 'Racquets' 'Motorboating' 'Military Ski Patrol' 'Croquet'
 'Jeu De Paume' 'Roque' 'Alpinism' 'Basque Pelota' 'Aeronautics']
```

6. Каков средний рост теннисисток (пол – женский, вид спорта – большой теннис), участвовавших в играх 2000 года?

```python
# 3) Каков средний рост теннисисток (пол - женский, вид спорта - большой теннис), участвовавших в играх 2000 года?

height_2000 = data[(data['Year'] == 2000) & (data['Sex'] == 'F') & (data['Sport'] == 'Tennis')]['Height'].mean()

print(f"\nСредний рост теннисисток, участвовавших в играх 2000 года: {height_2000:.2f} см")
```

```
Средний рост теннисисток, участвовавших в играх 2000 года: 171.79 см
```

7. Сколько золотых медалей в настольном теннисе выиграл Китай на ОИ в 2008 году?

```
# 4) Сколько золотых медалей в настольном теннисе выиграл Китай на ОИ в 2008 году?

gold_medals = data[

   (data['Year'] == 2008) & (data['NOC'] == 'CHN') & (data['Sport'] == 'Table Tennis') & (data['Medal'] == 'Gold')]

print(f"\nСколько золотых медалей в настольном теннисе \nвыиграл Китай на ОИ в 2008 году?:
{gold_medals.shape[0]}")
```

```
Сколько золотых медалей в настольном теннисе
выиграл Китай на ОИ в 2008 году?: 8
```

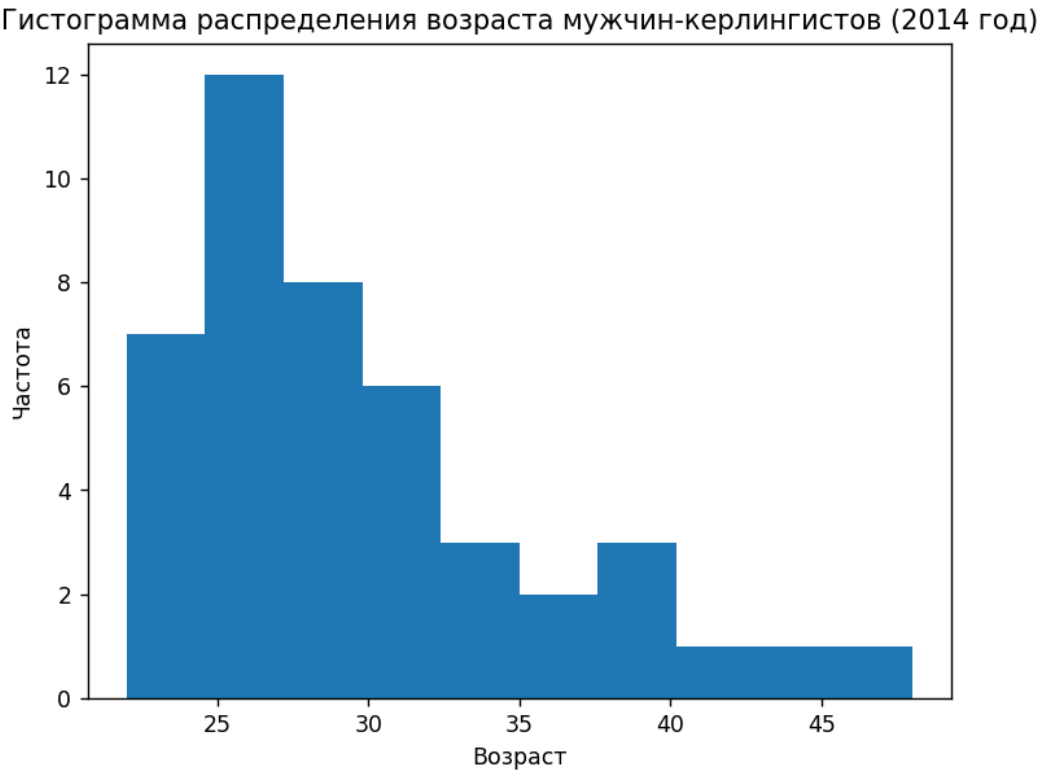
8. Как изменилось количество видов спорта на летних ОИ в 2004 году по сравнению с летними ОИ в 1988 году?

```
# 5) Как изменилось количество видов спорта на летних ОИ в 2004 году по сравнению с летними ОИ в 1988 году?

sports_2004 = data[(data['Year'] == 2004) & (data['Season'] == 'Summer')]['Sport'].nunique()
# Считаем сразу количество

sports_1988 = data[(data['Year'] == 1988) & (data['Season'] == 'Summer')]['Sport'].nunique()
# Считаем сразу количество

print(f"\nИзменение количества видов спорта на летних ОИ в 2004 году по сравнению с 1988 годом: "

     f"\nКоличество в 2004 году {sports_2004}"

     f"\nКоличество в 1988 году {sports_1988}"

     f"\nРазница:{sports_2004 - sports_1988}")
```

```
Изменение количества видов спорта на летних ОИ в 2004 году по сравнению с 1988 годом:
Количество в 2004 году 34
Количество в 1988 году 27
Разница:7
```

9. Постройте гистограмму распределения возраста мужчин- керлингистов (Sport == ‘Curling’), участвовавших в олимпиаде 2014 года. Подсказка: для построения гистограммы можно использовать функцию hist() из библиотеки matplotlib с параметрами по умолчанию (либо можете использовать любую другую функцию на свое усмотрение).

```
# 6) Постройте гистограмму распределения возраста мужчин-керлингистов (Sport == ‘Curling’),

# участвовавших в олимпиаде 2014 года. Подсказка: для построения гистограммы можно использовать

# функцию hist() из библиотеки matplotlib с параметрами по умолчанию

curling_2014 = data[(data['Sport'] == 'Curling') & (data['Year'] == 2014) & (data['Sex'] == 'M')]

plt.hist(curling_2014['Age'])

plt.title('Гистограмма распределения возраста мужчин-керлингистов (2014 год)')

plt.xlabel('Возраст')

plt.ylabel('Частота')

plt.show()
```



10. Рассмотрим зимнюю олимпиаду 2006 года. Сгруппируйте данные по стране (используйте признак «NOC») и посчитайте для каждой страны количество завоеванных медалей и средний возраст спортсменов. Выведите только те страны, которые завоевали хотя бы одну медаль.

```
# 7) Рассмотрим зимнюю олимпиаду 2006 года. Сгруппируйте
данные по стране (используйте признак «NOC»)

# и посчитайте для каждой страны количество завоеванных
медалей и средний возраст спортсменов.

# Выведите только те страны, которые завоевали хотя бы
одну медаль.

medals_age_2006 = data[(data['Year'] == 2006) &
(data['Season'] == 'Winter')].groupby('NOC').agg(

    {'Medal': 'count', 'Age': 'mean'})

medals_age_2006 = medals_age_2006[medals_age_2006['Medal']
> 0]

print("\nКоличество завоеванных медалей и средний возраст
спортсменов зимней олимпиады 2006 года:")

print(medals_age_2006)
```

```
Количество завоеванных медалей и средний возраст спортсменов зимней олимпиады 2006 года:
     Medal        Age
NOC
AUS      2  25.711111
AUT     30  27.704545
BLR      1  27.142857
BUL      1  26.181818
CAN     69  25.481967
CHN     13  23.534247
CRO      3  22.760870
CZE     27  26.276471
EST      3  25.634921
FIN     41  26.614286
FRA     15  26.283019
GBR      1  26.851852
GER     54  27.376426
ITA     25  26.727586
JPN      1  26.515789
KOR     19  21.564706
LAT      1  26.380435
NED     13  25.873016
NOR     23  28.186335
POL      2  25.219780
RUS     41  25.784452
SUI     21  26.475138
SVK      1  25.517544
SWE     64  26.791667
UKR      3  25.614679
USA     52  25.818462
```

11. Продолжим рассматривать зимнюю олимпиаду 2006 года. Посчитайте, сколько медалей каждого достоинства завоевала каждая из стран-участниц (страны, не завоевавшие ни одной медали, можно не выводить). Для этого сгруппируйте данные по стране и по виду медали. Представьте данные в виде сводной таблицы (pivot_table). В сводной таблице не должно быть отсутствующих значений (NaN), замените их на 0.

```
# 8) Продолжим рассматривать зимнюю олимпиаду 2006 года.
Посчитайте, сколько медалей каждого достоинства завоевала

# каждая из стран-участниц (страны, не завоевавшие ни
одной медали, можно не выводить).

# Для этого сгруппируйте данные по стране и по виду
медали. Представьте данные в виде сводной

# таблицы (pivot_table). В сводной таблице не должно быть
отсутствующих значений (NaN), замените их на 0.
```

```
medals_count_2006 = data[(data['Year'] == 2006) &
(data['Season'] == 'Winter')].pivot_table(

   # Фильтруем данные для зимней Олимпиады 2006 года и
создаем сводную таблицу

   index='NOC',  # Группируем по стране

   columns='Medal',  # Группируем по виду медали (столбцы)

   values='ID',  # Столбец для агрегирования
(идентификаторы спортсменов)

   aggfunc='count',  # Подсчитываем количество медалей
каждого типа

   fill_value=0  # Заполняем отсутствующие значения нулями

)

print("\nКоличество медалей каждого достоинства для зимней
олимпиады 2006 года:")

print(medals_count_2006)
```

```
Количество медалей каждого достоинства для зимней олимпиады 2006 года:
Medal  Bronze  Gold  Silver
NOC
AUS         1     1       0
AUT         7    16       7
BLR         0     0       1
BUL         0     0       1
CAN        11    30      28
CHN         6     2       5
CRO         0     1       2
CZE        24     1       2
EST         0     3       0
FIN         7     0      34
FRA        10     3       2
GBR         0     0       1
GER         6    23      25
ITA        14    11       0
JPN         0     1       0
KOR         2    14       3
LAT         1     0       0
NED         8     3       2
NOR        12     2       9
POL         1     0       1
RUS        13    16      12
SUI         9     5       7
SVK         0     0       1
SWE         8    35      21
UKR         3     0       0
USA        32     9      11
```

Задача №3

1. Загрузите данные из файла «telecom_churn.csv» о клиентах оператора сотовой связи.

```
import pandas as pd

import os


# 1. Загрузка данных с проверкой наличия файлов
```

```
if os.path.exists('telecom_churn.csv'):
    data = pd.read_csv('telecom_churn.csv')
else:
    print("Файл 'telecom_churn.csv' не найден.")
```

2. Выведите общую информацию о датафрейме с помощью методов info или describe. Есть ли отсутствующие данные?

```
# 2. Выведите общую информацию о датафрейме с помощью методов info или describe. Есть ли отсутствующие данные?
print(data.info())


# Проверка на наличие отсутствующих данных
not_values = data.isnull().sum()  # Считаем количество пропущенных значений в каждом столбце
print("\nОтсутствующие данные:")
print(not_values[not_values > 0])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Account length          3333 non-null   int64
 2   Area code               3333 non-null   int64
 3   International plan      3333 non-null   object
 4   Voice mail plan         3333 non-null   object
 5   Number vmail messages   3333 non-null   int64
 6   Total day minutes       3333 non-null   float64
 7   Total day calls         3333 non-null   int64
 8   Total day charge        3333 non-null   float64
 9   Total eve minutes       3333 non-null   float64
 10  Total eve calls         3333 non-null   int64
 11  Total eve charge        3333 non-null   float64
 12  Total night minutes     3333 non-null   float64
 13  Total night calls       3333 non-null   int64
 14  Total night charge      3333 non-null   float64
 15  Total intl minutes      3333 non-null   float64
 16  Total intl calls        3333 non-null   int64
 17  Total intl charge       3333 non-null   float64
 18  Customer service calls  3333 non-null   int64
 19  Churn                   3333 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB
None
```

```
Отсутствующие данные:
Series([], dtype: int64)
```

Отсутствующих данных нет

3. С помощью метода value_counts определите, сколько клиентов активны, а сколько потеряно. Сколько процентов клиентов в имеющихся данных активны, а сколько потеряны?

```python
# 2. С помощью метода value_counts определите, сколько клиентов активны, а сколько потеряно.

# Сколько процентов клиентов в имеющихся данных активны, а сколько потеряны?

# Churn - отток клиентов (False - клиент активен, true - клиент потерян, то есть расторг договор)

val_counts = data['Churn'].value_counts()

print("\nКоличество клиентов:\n", val_counts)


# Процент активных и потерянных клиентов

active = val_counts[False] / val_counts.sum() * 100

lost = val_counts[True] / val_counts.sum() * 100

print(f"Процент активных клиентов: {active:.2f}%")

print(f"Процент потерянных клиентов: {lost:.2f}%")
```

```
Количество клиентов:
 Churn
False    2850
True      483
Name: count, dtype: int64
Процент активных клиентов: 85.51%
Процент потерянных клиентов: 14.49%
```

4. Добавьте дополнительный столбец в датафрейм продолжительность одного звонка (вычислить как суммарная продолжительность всех звонков, деленная на суммарное количество всех звонков). Отсортируйте данные по этому значению по убыванию и выведите 10 первых записей.

```python
# 3. Добавьте дополнительный столбец в датафрейм продолжительность одного звонка

# (вычислить как суммарная продолжительность всех звонков, деленная на суммарное количество всех звонков).
```

```
# Отсортируйте данные по этому значению по убыванию и выведите 10 первых записей.

data['Duration of one call'] = (

        (data['Total day minutes'] + data['Total eve minutes'] + data['Total night minutes']) /

        (data['Total day calls'] + data['Total eve calls'] + data['Total night calls'])

)


# Сортировка по убыванию и вывод первых 10 записей

sorted_data = data.sort_values(by='Duration of one call', ascending=False)

print('\n10 первых записей продолжительности одного звонка')

print(sorted_data[['Duration of one call']].head(10))
```

```
10 первых записей продолжительности одного звонка
      Duration of one call
985               3.693644
2824              3.599519
244               3.509402
2321              3.429258
2033              3.351121
1709              3.347872
2536              3.325806
1686              3.222624
649               3.197368
2289              3.181498
```

5. Сгруппируйте данные по значению поля «Churn» и вычислите среднюю продолжительность одного звонка в каждой категории. Есть ли существенная разница в средней продолжительности одного звонка между активными и потерянными клиентами?

```
# 4. Сгруппируйте данные по значению поля «Churn» и вычислите среднюю продолжительность одного звонка

# в каждой категории. Есть ли существенная разница в средней продолжительности одного звонка

# между активными и потерянными клиентами?

av_duration = data.groupby('Churn')['Duration of one call'].mean()

print("\nСредняя продолжительность одного звонка по категориям Churn:\n", av_duration)

print('Существенной разницы в средней продолжительности одного звонка между активными и потерянными клиентами нет\n')
```

```
Средняя продолжительность одного звонка по категориям Churn:
 Churn
False    1.938102
True     2.091193
Name: Duration of one call, dtype: float64
Существенной разницы в средней продолжительности одного звонка между активными и потерянными клиентами нет
```

6. Сгруппируйте данные по значению поля «Churn» и вычислите среднее количество звонков в службу поддержки в каждой категории. Есть ли существенная разница между активными и потерянными клиентами?

```
# 5. Сгруппируйте данные по значению поля «Churn» и вычислите среднее количество звонков в службу поддержки

# в каждой категории. Есть ли существенная разница между активными и потерянными клиентами?

av_service = data.groupby('Churn')['Customer service calls'].mean()

print("\nСреднее количество звонков в службу поддержки по категориям Churn:\n", av_service)

print('Имеется довольно существенная разница в среднем количестве звонков в службу поддержки между активными и '

      'потерянными клиентами\n')
```

```
Среднее количество звонков в службу поддержки по категориям Churn:
 Churn
False    1.449825
True     2.229814
Name: Customer service calls, dtype: float64
Имеется довольно существенная разница в среднем количестве звонков в службу поддержки между активными и потерянными клиентами
```

7. Исследуйте подробнее связь между параметрами «Churn» и «Customer service calls», построив таблицу сопряженности (факторную таблицу) по этим признакам. При каком количестве звонков в службу поддержки процент оттока становится существенно выше, чем в целом по датафрейму? (можете использовать «более 40%».)

```
# 6. Исследуйте подробнее связь между параметрами «Churn» и «Customer service calls»,

# построив таблицу сопряженности (факторную таблицу) по этим признакам.

# Подсказка: используйте функцию crosstab. При каком количестве звонков в службу поддержки процент

# оттока становится существенно выше, чем в целом по датафрейму? (В качестве уточнения фразы «существенно выше»

# можете использовать «более 40%».)

crosstab_calls = pd.crosstab(data['Customer service calls'], data['Churn'])

print("\nТаблица сопряженности между Churn и Customer service calls:\n", crosstab_calls)



# Процент оттока для каждого значения "Customer service calls"

# axis=1 указывает, что сумма должна быть рассчитана по строкам

# Количество оттока делим на количества звонков пользователей (0 697, 1 1181 и тд)

crosstab_calls['Churn rate'] = crosstab_calls[True] / crosstab_calls.sum(axis=1) * 100



# Определение порога, при котором процент оттока выше 40%
```

```
high_churn = crosstab_calls[crosstab_calls['Churn rate'] >
40]

print("\nКоличество звонков в службу поддержки при
проценте оттока выше 40%:\n", high_churn)
```

```
Таблица сопряженности между Churn и Customer service calls:
 Churn                   False  True
Customer service calls
0                         605    92
1                        1059   122
2                         672    87
3                         385    44
4                          90    76
5                          26    40
6                           8    14
7                           4     5
8                           1     1
9                           0     2

Количество звонков в службу поддержки при проценте оттока выше 40%:
 Churn                   False  True  Churn rate
Customer service calls
4                          90    76   45.783133
5                          26    40   60.606061
6                           8    14   63.636364
7                           4     5   55.555556
8                           1     1   50.000000
9                           0     2  100.000000
```

8. Аналогично предыдущему пункту исследуйте связь между параметрами «Churn» и «International plan». Можно ли утверждать, что процент оттока среди клиентов, использующих международный роуминг, существенно выше или ниже, чем среди клиентов, не использующих его?

```
# 7. Аналогично предыдущему пункту исследуйте связь между
параметрами «Churn» и «International plan».

# Можно ли утверждать, что процент оттока среди клиентов,
использующих международный роуминг, существенно
```

```
# выше или ниже, чем среди клиентов, не использующих его?

crosstab_plan = pd.crosstab(data['International plan'],
data['Churn'])

print("\nТаблица сопряженности между Churn и International
plan:\n", crosstab_plan)



# Процент оттока для каждого значения "International plan"

# axis=1 указывает, что сумма должна быть рассчитана по
строкам

# Количество оттока делим на всех, кто использует/не
использует роуминг (No 3010, Yes 323)

crosstab_plan['Churn rate'] = crosstab_plan[True] /
crosstab_plan.sum(axis=1) * 100

print("\nПроцент оттока по наличию международного
роуминга:\n", crosstab_plan[['Churn rate']])

print('У людей, у которых не подключена услуга
международного роуминга, процент оттока ниже, чем у людей,
у которых '

      'подключена эта услуга')
```

```
Таблица сопряженности между Churn и International plan:
 Churn              False  True
International plan
No                  2664    346
Yes                  186    137

Процент оттока по наличию международного роуминга:
 Churn              Churn rate
International plan
No                  11.495017
Yes                 42.414861
У людей, у которых не подключена услуга международного роуминга, процент оттока ниже, чем у людей, у которых подключена эта услуга
```

9. Добавьте в датафрейм столбец «Прогнозируемый отток», заполнив его на основе значений столбцов «Customer service calls» и «International plan». Сравните значение в этом столбце со значением столбца «Churn». Если мы будем пользоваться построенным прогнозом, то какой процент ошибок первого и второго рода (ложноположительных и ложноотрицательных) мы получим?

```
# 8. Добавьте в датафрейм столбец «Прогнозируемый отток», заполнив его на основе значений столбцов

# «Customer service calls» и «International plan».

# Сравните значение в этом столбце со значением столбца «Churn». Если мы будем пользоваться построенным прогнозом,

# то какой процент ошибок первого и второго рода (ложноположительных и ложноотрицательных) мы получим?

data['Projected Churn'] = ((data['Customer service calls'] >= 4) | (data['International plan'] == 'Yes')).astype(bool)


# Сравнение значений столбца "Predicted Churn" со столбцом "Churn"

true_true = ((data['Projected Churn'] == True) & (data['Churn'] == True)).sum()

true_false = ((data['Projected Churn'] == True) & (data['Churn'] == False)).sum()

false_false = ((data['Projected Churn'] == False) & (data['Churn'] == False)).sum()

false_true = ((data['Projected Churn'] == False) & (data['Churn'] == True)).sum()


total_predictions = len(data)

error_1 = true_false / total_predictions * 100

print('\nПроцент ошибок первого рода', error_1)

error_2 = false_true / total_predictions * 100

print('Процент ошибок второго рода', error_2)
```

```
Процент ошибок первого рода 9.18091809180918
Процент ошибок второго рода 6.810681068106811
```