Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет»

Институт компьютерных наук и технологий

Отчет
по лабораторной работе № 4

по дисциплине
«Введение в анализ данных»

Студент Власова Елизавета Александровна

Группа ИТ–14–2023

Пермь 2025

Оглавление

Задание 1 (2 балла)

1. Сгенерировать случайным образом массив numpy из 1000 значений нормально распределенной случайной величины (numpy.random.normal) с мат. ожиданием. Преобразовать его в объект Series.

Решение:

```
# 1
# генерация выборки
M = 1   # математическое ожидание
s = 1   # стандартное отклонение

# создаем массив из 1000 нормально распределенных значений
arr = np.random.normal(M, s, 1000)

# преобразуем в series
series = pd.Series(arr)
```

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

Решение:

```
# 2
# доля значений в диапазоне (M - s; M + s) - считаем среднее
within_1sigma = ((series > (M - s)) & (series < (M + s))).mean()
print(f"Доля в диапазоне (M - s; M + s): {within_1sigma:.3f}")
```

Результаты работы программы:

```
Доля в диапазоне (M - s; M + s): 0.687
```

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

Решение:

```python
# 3
# доля значений в диапазоне (M - 3s; M + 3s) - считаем среднее
within_3sigma = ((series > (M - 3 * s)) & (series < (M + 3 * s))).mean()
print(f"Доля в диапазоне (M - 3s; M + 3s): {within_3sigma:.3f}")
```

Результаты работы программы:

```
Доля в диапазоне (M - 3s; M + 3s): 0.998
```

Для нормального распределения действует так называемое "правило сигм" (или эмпирическое правило 68 - 95 - 99,7). Согласно ему, примерно 68% всех наблюдений располагаются в пределах одного стандартного отклонения от среднего значения, около 95% - в пределах двух, и примерно 99,7% - в пределах трех стандартных отклонений. То есть:

- в интервале ±1σ должно быть около 68% данных;
- в интервале ±3σ должно быть около 99.7%.

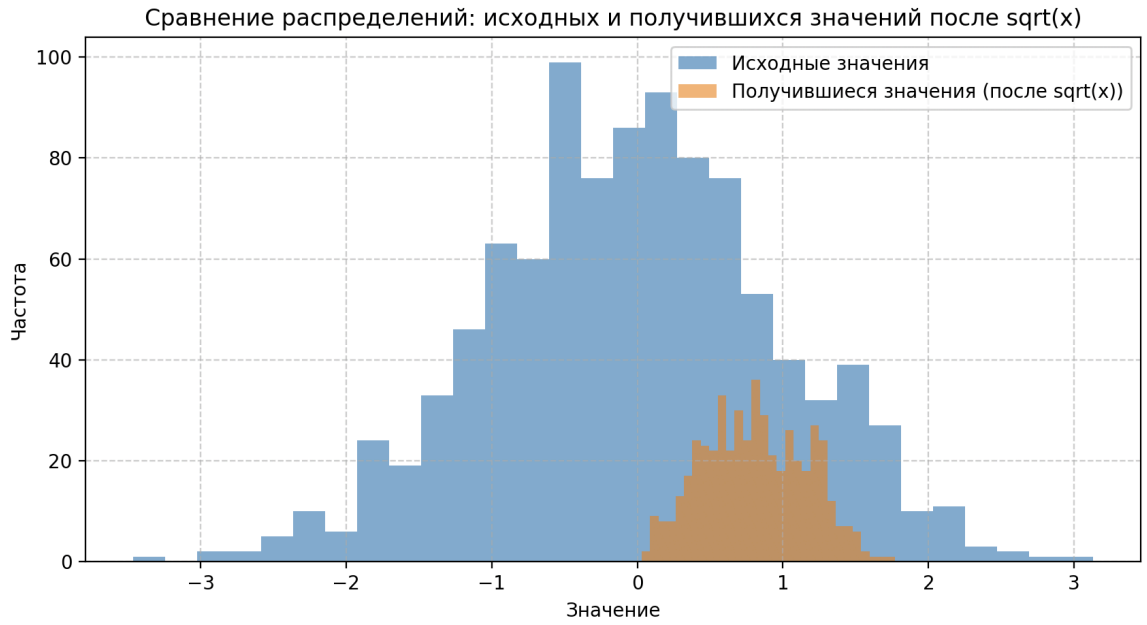
Таким образом, полученные результаты соответствует теории.

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?

Решение:

```python
# 4
# замена каждого x в серии на sqrt(x)
root_series = pd.Series(np.sqrt(series))
```

Результаты работы программы:




```
/Users/lisa/Downloads/lab4_vad/venv/lib/python3.14/site-packages/pandas/core/a
rraylike.py:399: RuntimeWarning: invalid value encountered in sqrt
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Предупреждение (RuntimeWarning: invalid value encountered in sqrt) возникает потому, что функция np.sqrt() не определена для отрицательных чисел. Для отрицательных элементов результат становится NaN (отсутствующее значение).

5. Подсчитать среднее арифметическое для получившихся значений. Отсутствующие значения (NaN) учитываться не должны.

Решение:

```python
# 5
# среднее арифметическое после sqrt
mean_root = root_series.mean(skipna=True)
print(f"Среднее значение для получившихся значений: {mean_root:.3f}")
```

Результаты работы программы:

```
Среднее значение для получившихся значений: 0.809
```

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.

Решение:

```python
# 6
# создание DataFrame
df = pd.DataFrame({
    "number": series, # столбцы
    "root": root_series
})

print("\nПервые 6 строк DataFrame:")
print(df.head(6))
```

Результаты работы программы:

```
Первые 6 строк DataFrame:
     number      root
0  1.473317  1.213803
1  0.476769  0.690484
2  0.366179  0.605127
3 -1.297955       NaN
4  1.070754  1.034772
5 -1.481829       NaN
```

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

Решение:

```
# 7
# поиск записей, где sqrt(x) ∈ [1.8, 1.9]
result = df.query("root >= 1.8 and root <= 1.9")
print("\nЗаписи, где root ∈ [1.8; 1.9]:")
print(result)
```

Результаты работы программы:

```
Записи, где root ∈ [1.8; 1.9]:
      number     root
962  3.379052  1.83822
```

Задание 2 (4 балла)

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

Данные содержат следующие признаки:

- ID – уникальный идентификатор спортсмена;
- Name – имя спортсмена;
- Sex – пол (M – мужской, F – женский);
- Age – возраст (полных лет, целое число);
- Height – рост в сантиметрах;
- Weight – вес в килограммах;
- Team – название команды (страны);

- NOC – трехбуквенное обозначение страны (по стандарту МОК);
- Games – год и вид ОИ (летние или зимние);
- Year – год проведения ОИ;
- Season – вид ОИ (летние или зимние);
- City – город проведения ОИ;
- Sport – вид спорта;
- Event – соревнование (дисциплина);
- Medal – завоеванная медаль (Gold, Silver, Bronze или NA).

Решение:

```
import pandas as pd
import matplotlib.pyplot as plt

# 1
# загрузка данных
df = pd.read_csv("athlete_events.csv")
```

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

Решение:

```python
# 2
# информация о данных
print("\nИнформация о данных:")
print(df.info())

# количество непустых значений по каждому столбцу
print("\nКоличество непустых значений:")
print(df.count())

# количество пропущенных данных
missing = df.isna().sum()
print("\nКоличество пропусков в каждом столбце:")
print(missing)

print("\nБольше всего пропусков в:", missing.idxmax(), "(", missing.max(), "пропусков)")
```

Результаты работы программы:

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      271116 non-null  int64
 1   Name    271116 non-null  object
 2   Sex     271116 non-null  object
 3   Age     261642 non-null  float64
 4   Height  210945 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object
 7   NOC     271116 non-null  object
 8   Games   271116 non-null  object
 9   Year    271116 non-null  int64
 10  Season  271116 non-null  object
 11  City    271116 non-null  object
 12  Sport   271116 non-null  object
 13  Event   271116 non-null  object
 14  Medal   39783 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
None
```

```
Количество непустых значений:
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
```

```
Количество пропусков в каждом столбце:
ID             0
Name           0
Sex            0
Age         9474
Height     60171
Weight     62875
Team           0
NOC            0
Games          0
Year           0
Season         0
City           0
Sport          0
Event          0
Medal     231333
dtype: int64

Больше всего пропусков в: Medal ( 231333 пропусков)
```

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

Решение:

```
# 3
# статистическая информация по возрасту, росту и весу
print("\nСтатистика по возрасту, росту и весу:")
print(df[["Age", "Height", "Weight"]].describe())
```

Результаты работы программы:

```
Статистика 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. Ответьте на вопросы, написав соответствующий код:

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

Решение:

```python
# 4.1
# самый молодой участник в 1992 году
young_1992 = df[df["Year"] == 1992]
min_age = young_1992["Age"].min()
youngest = young_1992[young_1992["Age"] == min_age][["Name", "Age", "Event"]]
print(f"\n4.1) Самый молодой участник в 1992 году:")
print(youngest)
```

Результаты работы программы:

```
4.1) Самый молодой участник в 1992 году:
                                    Name   Age                        Event
73461  Carlos Bienvenido Front Barrera  11.0  Rowing Men's Coxed Eights
```

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

Решение:

```python
# 4.2
# все виды спорта
sports = df["Sport"].unique()
print("\n 4.2) Все виды спорта, которые были на Олимпиадах:")
print(sorted(sports))
```

Результаты работы программы:

```
 4.2) Все виды спорта, которые были на Олимпиадах:
['Aeronautics', 'Alpine Skiing', 'Alpinism', 'Archery', 'Art Competitions', 'Athletics', 'Badminton', 'Baseball', 'B
asketball', 'Basque Pelota', 'Beach Volleyball', 'Biathlon', 'Bobsleigh', 'Boxing', 'Canoeing', 'Cricket', 'Croquet'
, 'Cross Country Skiing', 'Curling', 'Cycling', 'Diving', 'Equestrianism', 'Fencing', 'Figure Skating', 'Football',
'Freestyle Skiing', 'Golf', 'Gymnastics', 'Handball', 'Hockey', 'Ice Hockey', 'Jeu De Paume', 'Judo', 'Lacrosse', 'L
uge', 'Military Ski Patrol', 'Modern Pentathlon', 'Motorboating', 'Nordic Combined', 'Polo', 'Racquets', 'Rhythmic G
ymnastics', 'Roque', 'Rowing', 'Rugby', 'Rugby Sevens', 'Sailing', 'Shooting', 'Short Track Speed Skating', 'Skeleto
n', 'Ski Jumping', 'Snowboarding', 'Softball', 'Speed Skating', 'Swimming', 'Synchronized Swimming', 'Table Tennis',
 'Taekwondo', 'Tennis', 'Trampolining', 'Triathlon', 'Tug-Of-War', 'Volleyball', 'Water Polo', 'Weightlifting', 'Wre
stling']
```

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

Решение:

```python
# 4.3
# средний рост теннисисток
tennis_2000 = df[(df["Year"] == 2000) & (df["Sex"] == "F") & (df["Sport"] == "Tennis")]
mean_height_tennis = tennis_2000["Height"].mean()
print(f"\n4.3) Средний рост теннисисток 2000 года: {mean_height_tennis:.2f} см")
```

Результаты работы программы:

```
4.3) Средний рост теннисисток 2000 года: 171.79 см
```

4.4) Сколько золотых медалей в настольном теннисе выиграл Китай на ОИ в 2008 году?

Решение:

```python
# 4.4
# сколько золотых медалей в настольном теннисе выиграл Китай в 2008 году
china_gold_2008 = df[
    (df["Year"] == 2008)
    & (df["Team"] == "China")
    & (df["Sport"] == "Table Tennis")
    & (df["Medal"] == "Gold")
]
print(f"\n4.4) Китай выиграл в настольном теннисе в 2008 году золотых медалей: {len(china_gold_2008)}")
```

Результаты работы программы:

```
4.4) Китай выиграл в настольном теннисе в 2008 году золотых медалей: 8
```

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

Решение:

```python
# 4.5
# изменение количества видов спорта: 2004 vs 1988 (летние)
sports_1988 = df[(df["Year"] == 1988) & (df["Season"] == "Summer")]["Sport"].nunique()
sports_2004 = df[(df["Year"] == 2004) & (df["Season"] == "Summer")]["Sport"].nunique()
print(f"\n4.5) В 1988 году — {sports_1988} видов спорта, в 2004 — {sports_2004}.")
print(f"Изменение: {sports_2004 - sports_1988} видов.")
```

Результаты работы программы:

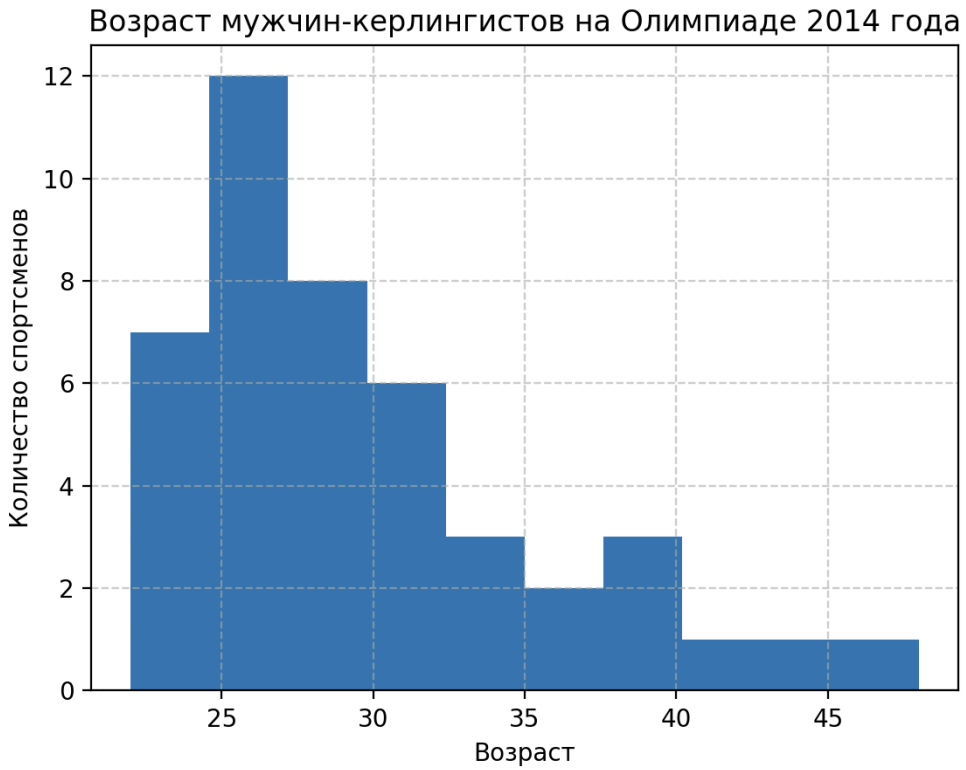
```
4.5) В 1988 году – 27 видов спорта, в 2004 – 34.
Изменение: 7 видов.
```

4.6) 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 (либо можете использовать любую другую функцию на свое усмотрение).

Решение:

```
# 4.6
# гистограмма возраста мужчин-керлингистов на олимпиаде 2014 года
curling_2014 = df[(df["Year"] == 2014) & (df["Sport"] == "Curling") & (df["Sex"] == "M")]
plt.hist(curling_2014["Age"].dropna(), bins=10)
plt.title("Возраст мужчин-керлингистов на Олимпиаде 2014 года")
plt.xlabel("Возраст")
plt.ylabel("Количество спортсменов")
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Результаты работы программы:



4.7) 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.

Решение:

```
# 4.7
# зимняя олимпиада 2006 — медали и средний возраст по странам
winter_2006 = df[(df["Year"] == 2006) & (df["Season"] == "Winter")]

medals_age = (
    winter_2006.groupby("NOC") # группируем по странам
    .agg(
        medals=("Medal", lambda x: x.notna().sum()),
        avg_age=("Age", "mean")
    )
    .query("medals > 0")
    .sort_values("medals", ascending=False)
)

print("\n4.7) Количество медалей и средний возраст спортсменов:")
print(medals_age)
```

Результаты работы программы:

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

4.8) 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.

Решение:

```python
# 4.8
# сводная таблица: сколько медалей каждого достоинства у каждой страны (2006, зимние)
pivot_medals = (
    winter_2006.pivot_table(
        index="NOC",
        columns="Medal",
        values="ID",
        aggfunc="count",
        fill_value=0
    )
)
print("\n4.8) Сводная таблица по медалям:")
print(pivot_medals)
```

Результаты работы программы:

```
4.8) Сводная таблица по медалям:
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
```

Задание 3 (3 балла)

Загрузите данные из файла «telecom_churn.csv» о клиентах оператора сотовой связи. Данные содержат, в числе прочих, следующие признаки:

- State – обозначение территории (штата);
- Area code – код города;
- International plan – подключена ли услуга международного роуминга;
- Number vmail messages – количество голосовых сообщений;

- Total day minutes – суммарная продолжительность дневных звонков (в минутах);
- Total day calls – количество дневных звонков;
- Total eve minutes, Total eve calls – аналогичные показатели по вечерним звонкам;
- Total night minutes, Total night calls – аналогично по ночным звонкам;
- Customer service calls – количество звонков в службу поддержки;
- Churn – отток клиентов (False – клиент активен, true – клиент потерян, то есть расторг договор).

Остальные столбцы можно сразу проигнорировать при загрузке данных.

Решение:

```
# загрузка данных
df = pd.read_csv("telecom_churn.csv")

# оставляем только нужные столбцы
df = df[[
    "State", "Area code", "International plan", "Number vmail messages",
    "Total day minutes", "Total day calls",
    "Total eve minutes", "Total eve calls",
    "Total night minutes", "Total night calls",
    "Customer service calls", "Churn"
]]
```

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

Решение:

```python
# 1
# информация о данных
print("Информация о данных:")
print(df.info())

# проверка наличия пропусков
print("\nКоличество пропусков в каждом столбце:")
print(df.isna().sum())
```

Результаты работы программы:

```
Информация о данных:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3333 non-null   object
 1   Area code               3333 non-null   int64
 2   International plan      3333 non-null   object
 3   Number vmail messages   3333 non-null   int64
 4   Total day minutes       3333 non-null   float64
 5   Total day calls         3333 non-null   int64
 6   Total eve minutes       3333 non-null   float64
 7   Total eve calls         3333 non-null   int64
 8   Total night minutes     3333 non-null   float64
 9   Total night calls       3333 non-null   int64
 10  Customer service calls  3333 non-null   int64
 11  Churn                   3333 non-null   bool
dtypes: bool(1), float64(3), int64(6), object(2)
memory usage: 289.8+ KB
None
```

```
Количество пропусков в каждом столбце:
State                     0
Area code                 0
International plan        0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total eve minutes         0
Total eve calls           0
Total night minutes       0
Total night calls         0
Customer service calls    0
Churn                     0
dtype: int64
```

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

Решение:

```python
# 2
# количество активных и потерянных клиентов
# False - активный клиент
# True - потерянный клиент
print("\nКоличество клиентов по категориям Churn:")
churn_counts = df["Churn"].value_counts()
print(churn_counts)

# в процентах
churn_percent = df["Churn"].value_counts(normalize=True) * 100
print("\nПроцентное распределение:")
print(churn_percent)
```

Результаты работы программы:

```
Количество клиентов по категориям Churn:
Churn
False    2850
True      483
Name: count, dtype: int64

Процентное распределение:
Churn
False    85.508551
True     14.491449
Name: proportion, dtype: float64
```

3. Добавьте дополнительный столбец в датафрейм – средняя 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.

Решение:

```python
# 3
# добавим столбец «Продолжительность одного звонка»
df["Total calls"] = df["Total day calls"] + df["Total eve calls"] + df["Total night calls"]
df["Total minutes"] = df["Total day minutes"] + df["Total eve minutes"] + df["Total night minutes"]

# избегаем деления на ноль
df["Avg call duration"] = df["Total minutes"] / df["Total calls"]

# сортировка по убыванию
print("\nТоп-10 клиентов по средней длительности звонка:")
print(df.sort_values("Avg call duration", ascending=False).head(10))
```

Результаты работы программы:

```
Топ-10 клиентов по средней длительности звонка:
     State  Area code International plan  Number vmail messages  ...  Churn  Total calls  Total minutes  Avg call duration
985     NY        415                Yes                      0  ...   True          236          871.7           3.693644
2824    OR        415                 No                      0  ...   True          208          748.7           3.599519
244     VA        408                 No                      0  ...   True          234          821.2           3.509402
2321    AR        408                 No                      0  ...  False          229          785.3           3.429258
2033    NJ        408                 No                      0  ...   True          223          747.3           3.351121
1709    NV        408                 No                      0  ...  False          188          629.4           3.347872
2536    CT        408                 No                      0  ...   True          217          721.7           3.325806
1686    CT        408                 No                     40  ...  False          221          712.2           3.222624
649     MO        408                 No                      0  ...   True          228          729.0           3.197368
2289    MS        415                Yes                     23  ...  False          227          722.2           3.181498
```

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?

Решение:

```
# 4
# средняя продолжительность звонка по группам Churn
avg_duration_by_churn = df.groupby("Churn")["Avg call duration"].mean()
print("\nСредняя продолжительность одного звонка (по Churn):")
print(avg_duration_by_churn)

diff_duration = avg_duration_by_churn[True] - avg_duration_by_churn[False]
print(f"Разница между потерянными и активными клиентами: {diff_duration:.2f} мин.")
```

Результаты работы программы:

```
Средняя продолжительность одного звонка (по Churn):
Churn
False    1.938102
True     2.091193
Name: Avg call duration, dtype: float64
Разница между потерянными и активными клиентами: 0.15 мин.
```

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?

Решение:

```
# 5
# среднее количество звонков в поддержку по Churn
support_calls = df.groupby("Churn")["Customer service calls"].mean()
print("\nСреднее количество звонков в поддержку (по Churn):")
print(support_calls)
```

Результаты работы программы:

```
Среднее количество звонков в поддержку (по Churn):
Churn
False    1.449825
True     2.229814
Name: Customer service calls, dtype: float64
```

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%».)

Решение:

```python
# 6
# таблица сопряженности по Churn и Customer service calls
ct = pd.crosstab(df["Customer service calls"], df["Churn"], normalize='index') * 100
print("\nПроцент оттока в зависимости от количества звонков в поддержку:")
print(ct)

# при каком количестве звонков процент оттока > 40%
high_churn_calls = ct[ct[True] > 40]
print("\nКоличество звонков в поддержку, при которых отток > 40%:")
print(high_churn_calls)
```

Результаты работы программы:

```
Процент оттока в зависимости от количества звонков в поддержку:
Churn                         False       True
Customer service calls
0                         86.800574   13.199426
1                         89.669771   10.330229
2                         88.537549   11.462451
3                         89.743590   10.256410
4                         54.216867   45.783133
5                         39.393939   60.606061
6                         36.363636   63.636364
7                         44.444444   55.555556
8                         50.000000   50.000000
9                          0.000000  100.000000

Количество звонков в поддержку, при которых отток > 40%:
Churn                         False       True
Customer service calls
4                         54.216867   45.783133
5                         39.393939   60.606061
6                         36.363636   63.636364
7                         44.444444   55.555556
8                         50.000000   50.000000
9                          0.000000  100.000000
```

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?

Решение:

```python
# 7
# таблица сопряженности Churn и International plan
ct_plan = pd.crosstab(df["International plan"], df["Churn"], normalize='index') * 100
print("\nПроцент оттока в зависимости от наличия международного плана:")
print(ct_plan)

# вывод
if ct_plan.loc["Yes", True] > ct_plan.loc["No", True]:
    print("\n---> Процент оттока среди клиентов с международным планом существенно выше.")
else:
    print("\n---> Процент оттока среди клиентов без международного плана выше или равен.")
```

Результаты работы программы:

```
Процент оттока в зависимости от наличия международного плана:
Churn                   False      True
International plan
No                  88.504983  11.495017
Yes                 57.585139  42.414861

---> Процент оттока среди клиентов с международным планом существенно выше.
```

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?

Решение:

```python
# 8
# прогнозируемый отток
# если есть международный план ИЛИ больше 3 звонков в поддержку → прогнозируем отток (True)
df["Predicted churn"] = (df["International plan"] == "Yes") | (df["Customer service calls"] > 3)

# сравнение реального и прогнозного оттока
confusion = pd.crosstab(df["Churn"], df["Predicted churn"], rownames=["Реальный"], colnames=["Прогнозируемый"])

print("\nТаблица сравнения (реальный vs прогнозируемый):")
print("\nТаблица 1: Клиенты, которые остались (Churn = False)")
print(confusion.loc[False])

print("\nТаблица 2: Клиенты, которые ушли (Churn = True)")
print(confusion.loc[True])

# подсчет ошибок
# ошибка 1 рода (ложноотрицательная): клиент ушёл, но модель сказала False
false_negative = confusion.loc[True, False]
# ошибка 2 рода (ложноположительная): клиент остался, но модель сказала True
false_positive = confusion.loc[False, True]

total_true = confusion.loc[True].sum()
total_false = confusion.loc[False].sum()

error_type1 = false_negative / total_true * 100
error_type2 = false_positive / total_false * 100

print(f"\nОшибка 1 рода (ушел, но не предсказали): {error_type1:.2f}%")
print(f"Ошибка 2 рода (не ушел, но предсказали): {error_type2:.2f}%")
```

Результаты работы программы:

```
Таблица сравнения (реальный vs прогнозируемый):

Таблица 1: Клиенты, которые остались (Churn = False)
Прогнозируемый
False    2544
True      306
Name: False, dtype: int64

Таблица 2: Клиенты, которые ушли (Churn = True)
Прогнозируемый
False    227
True     256
Name: True, dtype: int64

Ошибка 1 рода (ушел, но не предсказали): 47.00%
Ошибка 2 рода (не ушел, но предсказали): 10.74%
```