Содержание

Защита персональных данных клиентов

Нам нужно защитить данные клиентов страховой компании. Необходимо разработать такой метод преобразования данных, чтобы по ним было сложно восстановить персональную информацию.

Нужно защитить данные, чтобы при преобразовании качество моделей машинного обучения не ухудшилось. Подбирать наилучшую модель не требуется.

Цель исследования — разработать такой метод преобразования данных, чтобы по ним было сложно восстановить персональную информацию.

Данные получим из файла *insurance.csv*.

Исследование пройдёт в пять этапов:

- Загрузка данных
- Изучение данных
- Умножение матриц
- Выбор алгоритма преобразования
- Проверка алгоритма

Загрузка данных

In [1]:
```
# Импорт необходимых библиотек


import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [2]:
```
# Константы


r_state = 12345
```

In [3]:
```
# Функции и классы


# Функция для получения общих сведений о данных

def data_info(title, data):
    print('Общие сведения "{}":'.format(title))
    print()

    data.info()
```

```
    print()
    print()

    print('Пример данных (случайные 5 строк):')
    display(data.sample(5, random_state=r_state))

    print()
    print()

    print('Количество пропусков по столбцам:')
    print()
    for col in data.columns:
        nmv = data[col].isna().sum()
        pmv = nmv/len(data)

        if pmv == 0:
            print('\033[0m{} - {} шт. - {:.2%}'.format(col, nmv, pmv))
        elif pmv <= 0.1:
            print('\033[0m{} - \033[43m{} шт.\033[0m - \033[43m{:.2%}'.format(col, nmv, pmv))
        else:
            print('\033[0m{} - \033[41m{} шт.\033[0m - \033[41m{:.2%}'.format(col, nmv, pmv))
        print('\033[0m')

    print()

    print('Количество уникальных значений в столбцах:')
    print()
    for col in data.columns:
        print('{} - {}'.format(col, data[col].nunique()))

    print()
    print()

    print('Количество явных дубликатов: {} шт.'.format(data.duplicated().sum()))

# Функция для корреляции признаков

def corr_info(data, col_del):
    print('Корреляция признаков:')
    print()
    corr_matrix = data.drop(columns=col_del).corr()
    sns.set(font_scale=1.15)
    plt.figure(figsize=(8,4))
    sns.heatmap(
            corr_matrix,
            cmap='RdBu_r',
            annot=True,
            vmin=-1, vmax=1);
    plt.show()
    print()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            cor = corr_matrix.iloc[i, j]
            print('Корреляция между {} и {}:'.format(corr_matrix.columns[i], corr_matrix.columns[j]))
            if cor == 0:
                print(f'Коэффициент корреляции: {cor:.6f} \n Свзяь отсутствует')
            elif 0 < cor <= 0.3:
                print(f'Коэффициент корреляции: {cor:.6f} \n Слабая прямая связь')
            elif -0.3 <= cor < 0:
                print(f'Коэффициент корреляции: {cor:.6f} \n Слабая обратная связь')
            elif 0.3 < cor <= 0.7:
                print(f'Коэффициент корреляции: {cor:.6f} \n Средняя прямая связь')
            elif -0.7 <= cor < -0.3:
                print(f'Коэффициент корреляции: {cor:.6f} \n Средняя обратная связь')
            elif 0.7 < cor < 1:
                print(f'Коэффициент корреляции: {cor:.6f} \n Сильная прямая связь')
            elif -1 < cor < -0.7:
                print(f'Коэффициент корреляции: {cor:.6f} \n Сильная обратная связь')
            elif cor == 1:
                print(f'Коэффициент корреляции: {cor:.6f} \n Полная прямая связь')
            elif cor == -1:
                print(f'Коэффициент корреляции: {cor:.6f} \n Полная обратная связь')
            else:
                print('Введен неверный коэффициент')
            print()

# Функция разбивки данных для обучения

def split_data(data, target, test_size):
    features = data.drop([target] , axis=1)
    target = data[target]
```

```
    features_train, features_test, target_train, target_test = train_test_split(
    features, target, test_size=test_size, random_state=r_state)

    features_list = {'features_train' : features_train, 'features_test' : features_test}
    targets_list = {'target_train' : target_train, 'target_test' : target_test}

    print('Вспомогательные признаки:')
    print()
    for i in features_list:
        sh = features_list[i].shape
        psh = sh[0]/len(data)
        print('{} - Объектов: {} шт., признаков: {} шт. - {:.2%}'.format(i, sh[0], sh[1], psh))

    print()

    print('Целевые признаки:')
    print()
    for i in targets_list:
        sh = targets_list[i].shape
        psh = sh[0]/len(data)
        print('{} - Объектов: {} шт. - {:.2%}'.format(i, sh[0], psh))

    return features_train, features_test, target_train, target_test
```

In [4]:
```
data = pd.read_csv('datasets/insurance.csv')
```

Изучение данных

Первичный анализ

In [5]:
```
data_info('data', data)
print()
print()
col_del = ['Пол']
corr_info(data, col_del)
```

Общие сведения "data":

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Пол                5000 non-null   int64
 1   Возраст            5000 non-null   float64
 2   Зарплата           5000 non-null   float64
 3   Члены семьи        5000 non-null   int64
 4   Страховые выплаты  5000 non-null   int64
dtypes: float64(2), int64(3)
memory usage: 195.4 KB
```

Пример данных (случайные 5 строк):

	Пол	Возраст	Зарплата	Члены семьи	Страховые выплаты
3183	0	33.0	39000.0	4	0
1071	0	50.0	43100.0	2	2
2640	1	39.0	42100.0	0	0
2282	0	20.0	34800.0	0	0
1595	0	41.0	40000.0	4	0

Количество пропусков по столбцам:

Пол - 0 шт. - 0.00%

Возраст - 0 шт. - 0.00%

Зарплата - 0 шт. - 0.00%

Члены семьи - 0 шт. - 0.00%

Страховые выплаты - 0 шт. - 0.00%

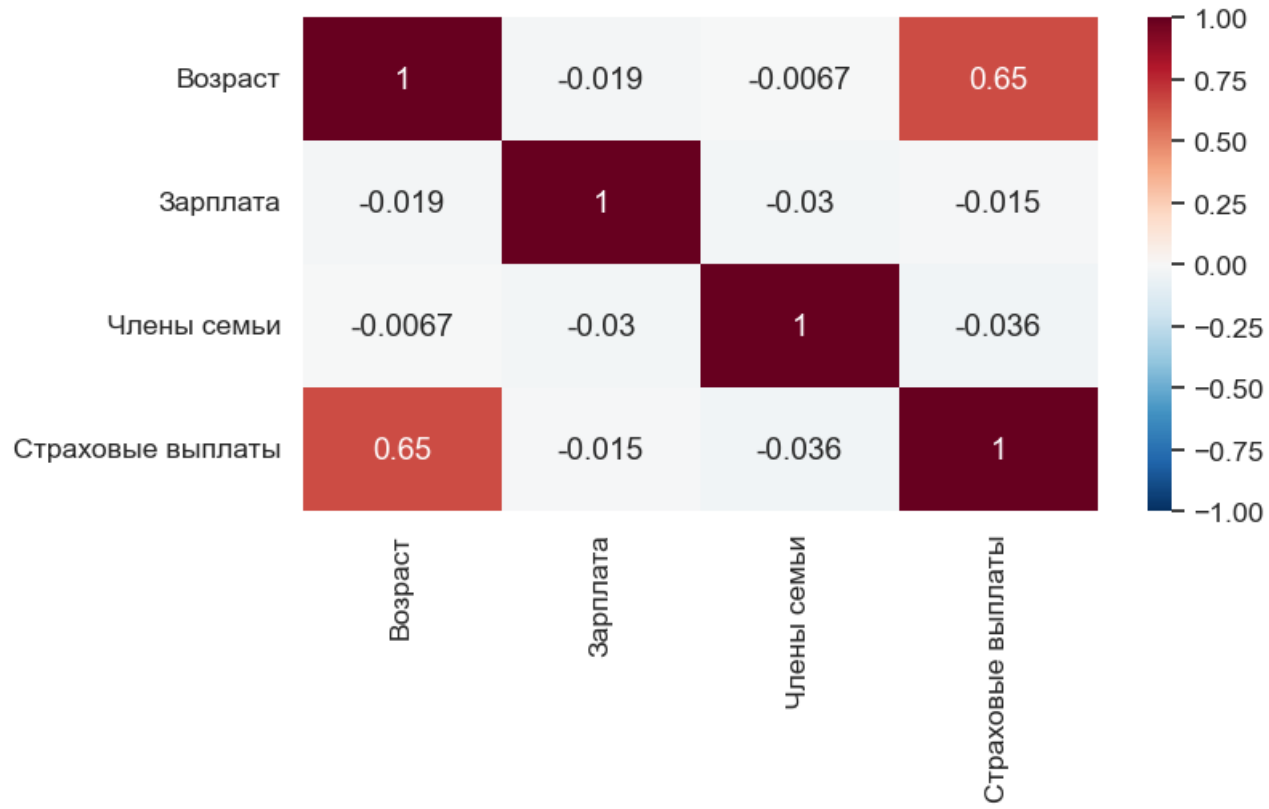
Количество уникальных значений в столбцах:

Пол - 2
Возраст - 46

Возраст - 46
Зарплата - 524
Члены семьи - 7
Страховые выплаты - 6

Количество явных дубликатов: 153 шт.

Корреляция признаков:



Корреляция между Зарплата и Возраст:
Коэффициент корреляции: -0.019093
Слабая обратная связь

Корреляция между Члены семьи и Возраст:
Коэффициент корреляции: -0.006692
Слабая обратная связь

Корреляция между Члены семьи и Зарплата:
Коэффициент корреляции: -0.030296
Слабая обратная связь

Корреляция между Страховые выплаты и Возраст:
Коэффициент корреляции: 0.651030
Средняя прямая связь

Корреляция между Страховые выплаты и Зарплата:
Коэффициент корреляции: -0.014963
Слабая обратная связь

Корреляция между Страховые выплаты и Члены семьи:
Коэффициент корреляции: -0.036290
Слабая обратная связь

Описание данных:

- Данные состоят из 5000 объектов
- Имееют 5 признаков:
  - Вспомогательные признаки:
    - пол
    - возраст
    - зарплата застрахованного
    - количество членов его семьи
  - Целевой признак:
    - количество страховых выплат клиенту за последние 5 лет
- Пропуски отсутствуют
- Данные в признаках "Возраст" и "Зарплата" имеют формат *float*, но целочисленные значения
- Все признаки имеют малые целочисленные значения - переведем их в формат *int32* для уменьшения памяти и быстродействия
- Содержат 153 явных дубликата (На решение задачи исследования не влияет, оставим эти данные для обучения)
- Проверим признаки на наличие аномалий:
  - "Пол" - должен содержать 2 уникальных значения
  - "Возраст" - проверим по критериям 18 <= Возраст <= 100
  - "Зарплата" - не должен быть равен 0
  - "Члены семьи" - не должен быть меньше 0
  - "Страховые выплаты" - не должен быть меньше 0

Предобработка данных

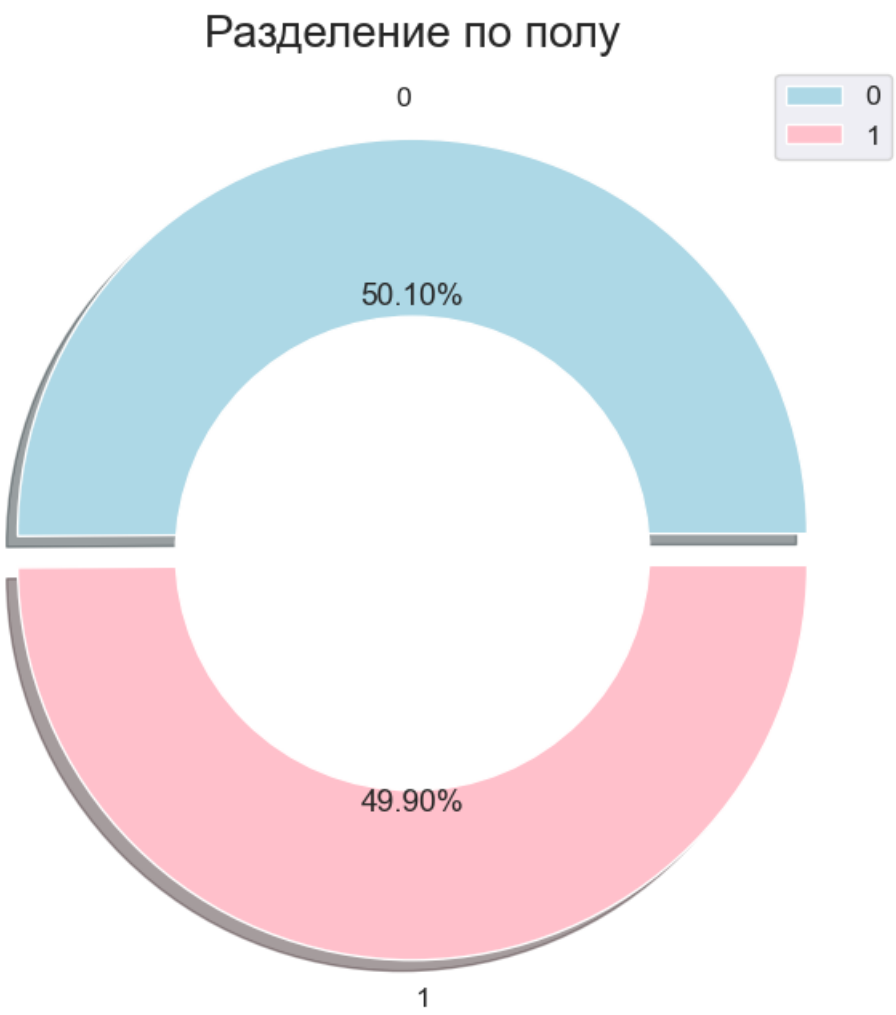
In [6]:
```
data = data.astype('int32')
data.dtypes
```

Out[6]:
```
Пол                 int32
Возраст             int32
Зарплата            int32
Члены семьи         int32
Страховые выплаты   int32
dtype: object
```

Проверка на наличие аномалий

Признак "Пол"

In [7]:
```
x = data['Пол'].value_counts()
colors = ['lightblue', 'pink']
labels = x.index
explode = [0.05, 0.03]
donut = plt.Circle((0,0), 0.6, color = 'white')
plt.figure(figsize=(8,8))
plt.pie(x, colors=colors, labels=labels, shadow=True, explode=explode, autopct='%.2f%%')

plt.title('Разделение по полу', fontsize = 20)
p = plt.gcf()
p.gca().add_artist(donut)
plt.legend()
plt.show()

print(x)
```



0 2505
1 2495
Name: Пол, dtype: int64
Аномалий не обнаружено

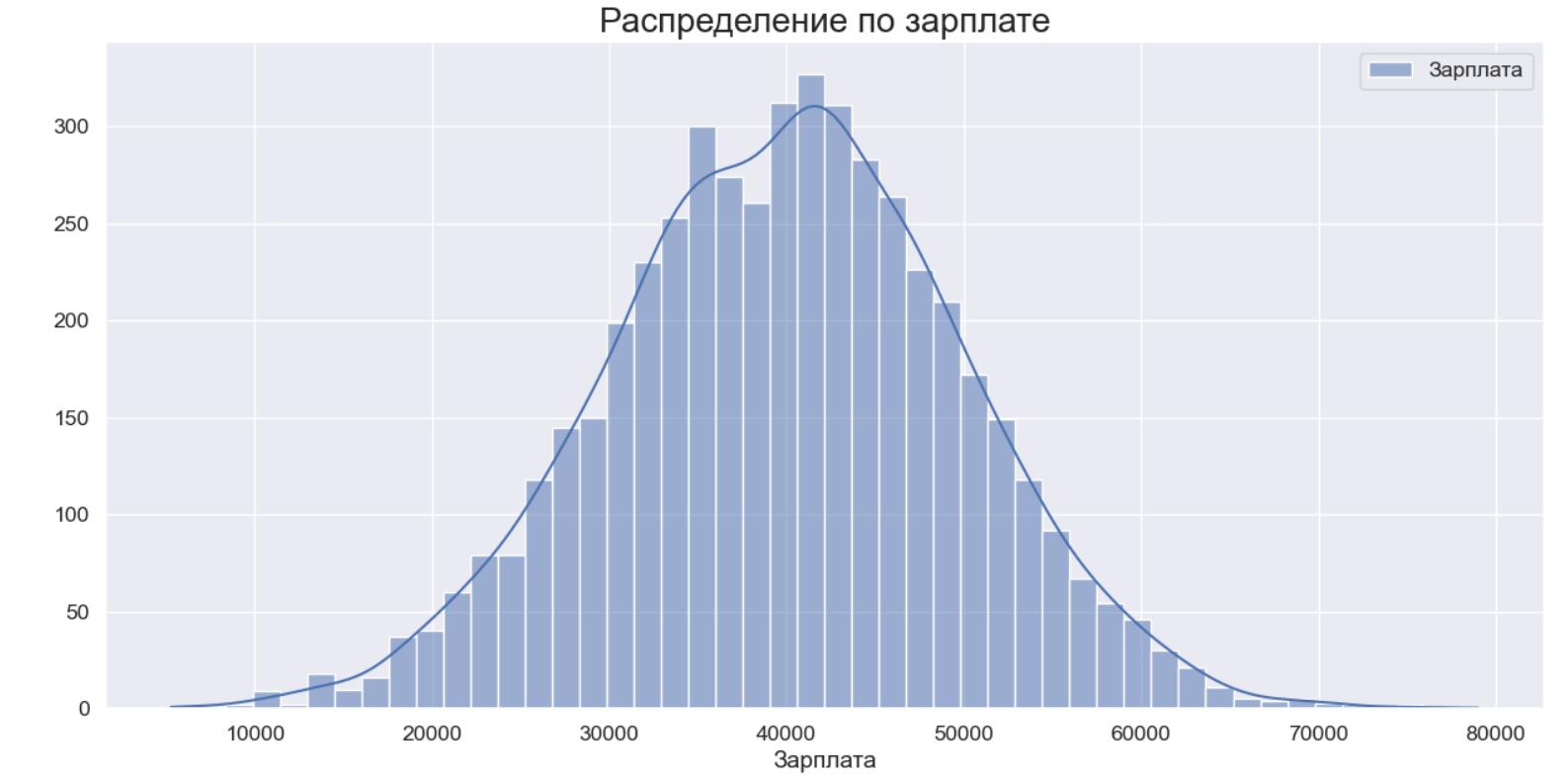
Признак "Возраст"

In [8]:
```
x = data['Возраст']
plt.figure(figsize=(15,7))
sns.histplot(x, label="Возраст", kde=True);
plt.legend();
plt.title(label='Распределение по возрасту', fontsize=20)
plt.xlabel('Возраст')
plt.ylabel(' ')
plt.show()

print("Минимальный возраст клиента:", data['Возраст'].min())
print("Максимальный возраст клиента:", data['Возраст'].max())
```

Минимальный возраст клиента: 18
Максимальный возраст клиента: 65
Аномалий не обнаружено

Признак "Зарплата"

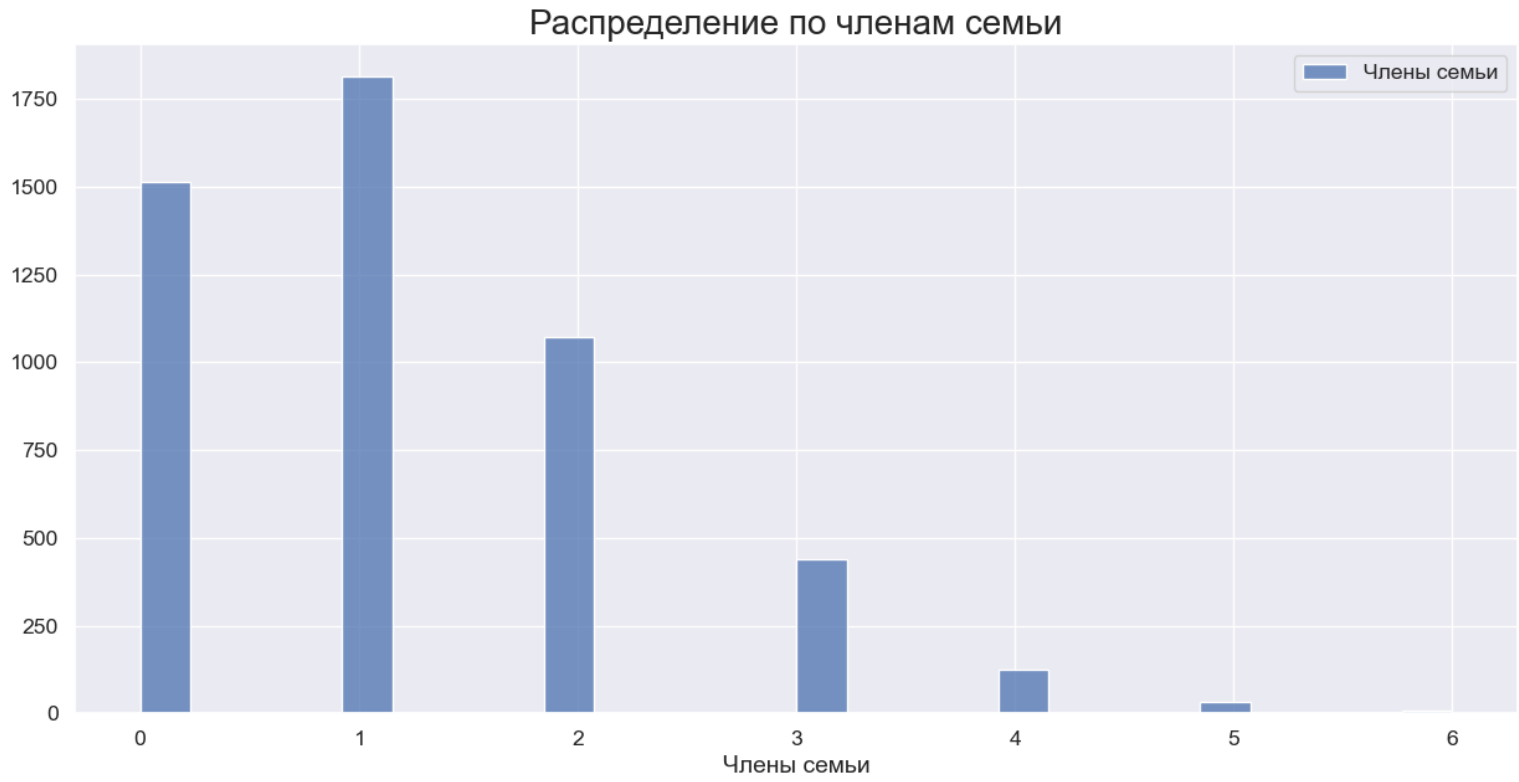
In [9]:
```
x = data['Зарплата']
plt.figure(figsize=(15,7))
sns.histplot(x, label="Зарплата", kde=True);
plt.legend();
plt.title(label='Распределение по зарплате', fontsize=20)
plt.xlabel('Зарплата')
plt.ylabel(' ')
plt.show()

print("Минимальная зарплата клиента:", data['Зарплата'].min())
print("Максимальная зарплата клиента:", data['Зарплата'].max())
```



Минимальная зарплата клиента: 5300
Максимальная зарплата клиента: 79000
Аномалий не обнаружено

Признак "Члены семьи"

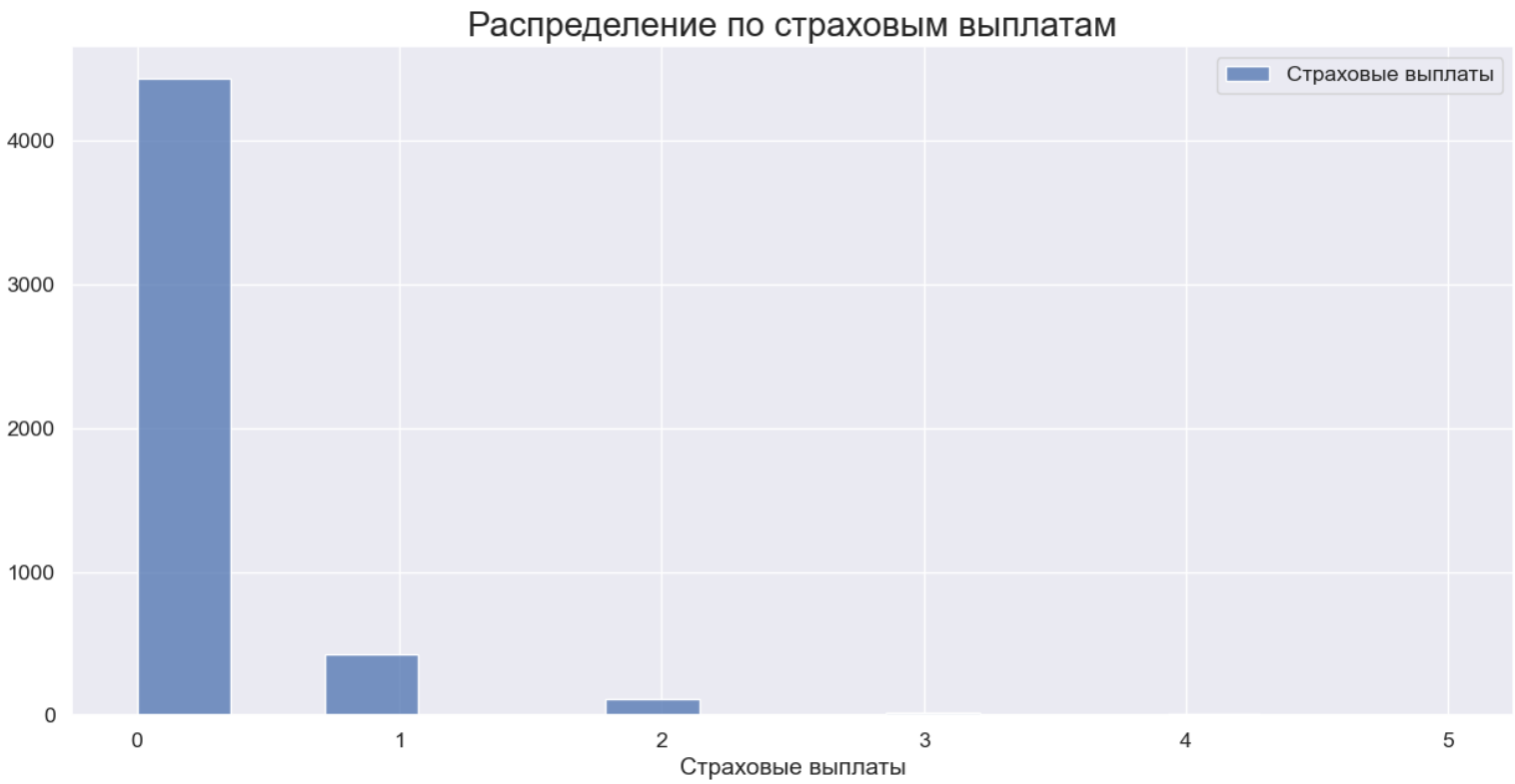
In [10]:
```
x = data['Члены семьи']
plt.figure(figsize=(15,7))
sns.histplot(x, label="Члены семьи");
plt.legend();
plt.title(label='Распределение по членам семьи', fontsize=20)
plt.xlabel('Члены семьи')
plt.ylabel(' ')
plt.show()

print("Минимальное количество членов семьи клиента:", data['Члены семьи'].min())
print("Максимальное количество членов семьи клиента:", data['Члены семьи'].max())
```



Минимальное количество членов семьи клиента: 0
Максимальное количество членов семьи клиента: 6
Аномалий не обнаружено

Признак "Страховые выплаты"

In [11]:
```
x = data['Страховые выплаты']
plt.figure(figsize=(15,7))
sns.histplot(x, label="Страховые выплаты");
plt.legend();
plt.title(label='Распределение по страховым выплатам', fontsize=20)
plt.xlabel('Страховые выплаты')
plt.ylabel(' ')
plt.show()

print("Минимальное количество страховых выплаты клиенту:", data['Страховые выплаты'].min())
print("Максимальное количество страховых выплаты клиенту:", data['Страховые выплаты'].max())
```



Минимальное количество страховых выплаты клиенту: 0
Максимальное количество страховых выплаты клиенту: 5
Аномалий не обнаружено

Умножение матриц

Обозначения:

- X — матрица признаков (нулевой столбец состоит из единиц)
- y — вектор целевого признака
- P — матрица, на которую умножаются признаки
- w — вектор весов линейной регрессии (нулевой элемент равен сдвигу)

Предсказания:

$$ a = Xw $$
Задача обучения:

$$ w = \arg\min_w MSE(Xw, y) $$
Формула обучения:

$$ w = (X^T X)^{-1} X^T y $$

Обоснование:

Нам необходимо доказать что $a = a_1$

1. Представим новую матрицу признаков X_1 как произведение исходной матрицы X на обратимую матрицу P:

$$ X_1 = X P $$

1. Подставим новую матрицу X_1 в формулу обучения w_1:

$$ w_1 = ((X P)^T X P)^{-1} (X P)^T y $$

1. Раскроем скобки у произведения $(XP)^T$:

$$ w_1 = (X^T P^T X P)^{-1} X^T P^T y $$

1. Перегруппируем множители в скобках $(X^T P^T X P)^{-1}$:

$$ w_1 = ((X^T X) (P^T P))^{-1} X^T P^T y $$

1. Раскроем скобки у произведения $((X^T X) (P^T P))^{-1}$:

$$ w_1 = (X^T X)^{-1} (P^T P)^{-1} X^T P^T y = P^{-1} (X^T X)^{-1} (P^T)^{-1} P^T X^T y $$

1. P - обратимая матрица $\Rightarrow$ $(P^T)^{-1} P^T = E$, где E - еденичная матрица:

$$ w_1 = P^{-1} (X^T X)^{-1} E X^T y = P^{-1} (X^T X)^{-1} X^T y $$

1. Исходя из формулы обучения заменим $(X^T X)^{-1} X^T y$ на w:

$$ w_1 = P^{-1} w $$

1. Подставим новое значение w_1 в формулу для предсказаний линейной регрессии $a = Xw$:

$$ a_1 = X_1 w_1 = X P P^{-1} w $$

1. P - обратимая матрица $\Rightarrow$ $P P^{-1} = E$, где E - еденичная матрица:

$$ a_1 = X_1 w_1 = X P P^{-1} w = X E w = X w = a $$
Мы доказали, что предсказания a_1 для матрицы признаков X, умноженных на обратимую матрицу P, равны предсказаниям a

Ответ:

Качество линейной регрессии при умножении признаков X на обратимую матрицу P не изменится

Параметры линейной регрессии в исходной задаче w и в преобразованной w_1 связаны по формуле:
$$ w_1 = P^{-1} w $$

Алгоритм преобразования

Алгоритм:

Исходя из проведенного выше исследования, в качестве алгоритма преобразования данных можно предложить умножение исходных данных на обратимую матрицу со случайными величинами.

Для создания обратимой матрицы со случайными величинами воспользуемся функцией *numpy.random.randn()*

Обоснование:

Разобьем данные на обучающую и тестовую выборки в соотношении 75:25

In [12]:
```
features_train, features_test, target_train, target_test = split_data(data, 'Страховые выплаты', 0.25)
```

Вспомогательные признаки:

features_train - Объектов: 3750 шт., признаков: 4 шт. - 75.00%
features_test - Объектов: 1250 шт., признаков: 4 шт. - 25.00%

Целевые признаки:

target_train - Объектов: 3750 шт. - 75.00%
target_test - Объектов: 1250 шт. - 25.00%

Преобразуем полученные выборки со вспомагательными признаками в матрицы

In [13]:
```
matrix_features_train = features_train.values
matrix_features_test = features_test.values
print(matrix_features_train.shape)
print(matrix_features_test.shape)
```

(3750, 4)
(1250, 4)

Создадим квадратную обратимую матрицу со случайными величинами

Размерность должна быть равна количеству признаков выборок *features*

In [14]:
```
size_matrix = matrix_features_train.shape[1]
random_matrix = np.random.randn(size_matrix, size_matrix)
random_matrix
```

Out[14]:
```
array([[-1.07753811,  0.06994202, -0.29724028,  0.32602498],
       [ 1.10416681,  1.30864958,  0.42693268,  1.73720317],
       [ 1.5251875 , -0.04444712, -1.4732031 ,  0.29587006],
       [ 2.96854916,  0.44813659, -0.49827889,  0.16626662]])
```

Проверим получившеюся матрицу на обратимость

Проверим двумя способами:

- Функцией *numpy.linalg.inv()*
  - Если матрица обратима, то функция создаст обратную матрицу
  - Если матрица необратима, то функция вызовет ошибку
- Умножим нашу матрицу на обратную ей матрицу
  - В результате мы должны получить единичную матрицу (матрица, у которой на главной диаганали единицы, а остальные элементы - нули)

Способ *numpy.linalg.inv()*

In [15]:
```
random_matrix_inv = np.linalg.inv(random_matrix)
random_matrix_inv
```

Out[15]:
```
array([[-1.02528749,  0.15967003,  0.34514283, -0.27201713],
       [ 5.90492881, -1.00685012, -2.82588549,  3.96979162],
       [-1.90793121,  0.42238636,  0.14033195, -0.92175874],
       [-3.32766835,  1.12881506,  1.87490235, -2.59105267]])
```

Способ "обратной матрицы"

In [16]:
```
np.round(random_matrix.dot(random_matrix_inv)).astype(int)
```

Out[16]:
```
array([[1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]])
```

Оба способа показали что матрица *random_matrix* является обратимой

Сравним данные на разных этапах преобразования

Для проверки возьмем данные из *features_train*

Закодируем данные нашим методом

In [17]:
```
encoded_matrix_features_train = matrix_features_train.dot(random_matrix)
encoded_matrix_features_test = matrix_features_test.dot(random_matrix)
```

Декодируем данные нашим методом

In [18]:
```
decoded_matrix_features_train = encoded_matrix_features_train.dot(random_matrix_inv)
decoded_matrix_features_test = encoded_matrix_features_test.dot(random_matrix_inv)
```

Сравним данные на примере *features_train*

In [19]:
```
encoded_features_train = pd.DataFrame(
                            encoded_matrix_features_train,
                            columns=features_train.columns,
                            index=features_train.index)

decoded_features_train = pd.DataFrame(
                            decoded_matrix_features_train,
                            columns=features_train.columns,
                            index=features_train.index)

decoded_features_train = round(decoded_features_train).astype(int)

print('Исходные данные:')
display(features_train.head())
print()
print('Кодированные данные:')
display(encoded_features_train.head())
print()
print('Декодированные данные:')
display(decoded_features_train.head())
```

Исходные данные:

	Пол	Возраст	Зарплата	Члены семьи
3369	1	43	36200	1
1441	1	34	57600	0
571	0	32	41100	1
225	0	36	45100	1
2558	0	33	50600	2

Кодированные данные:

	Пол	Возраст	Зарплата	Члены семьи
3369	55261.157841	-1552.195641	-53312.389720	10785.688198
1441	87887.264384	-2515.589937	-84842.280225	17101.506386
571	62723.508316	-1784.451604	-60535.483940	12216.016232
225	68828.675000	-1957.005476	-66426.588619	13406.445284
2558	77216.862323	-2204.942434	-74530.984759	15028.685271

Декодированные данные:

	Пол	Возраст	Зарплата	Члены семьи
3369	1	43	36200	1
1441	1	34	57600	0
571	0	32	41100	1
225	0	36	45100	1
2558	0	33	50600	2

- После кодирования-декодированния данные не изменились
- По кодированным данным сложно восстановить персональную информацию

Проверка алгоритма

Проверим, что качество линейной регрессии из sklearn не отличается до и после преобразования. Применим метрику R2

In [20]:
```
model = LinearRegression()

model.fit(matrix_features_train, target_train)
predicted_original = model.predict(features_test)
r2_original = model.score(features_test, target_test)
```

```
c:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but LinearRegression was fitted without feature name
s
  warnings.warn(
c:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:443: UserWarning: X has feature names, but LinearRegression was fitted without feature name
s
  warnings.warn(
```

In [21]:
```
model.fit(encoded_matrix_features_train, target_train)
predicted_encoded = model.predict(encoded_matrix_features_test)
r2_encoded = model.score(encoded_matrix_features_test, target_test)
```

In [22]:
```
model.fit(decoded_matrix_features_train, target_train)
predicted_decoded = model.predict(decoded_matrix_features_test)
r2_decoded = model.score(decoded_matrix_features_test, target_test)
```

In [23]:
```
print('R2 score исходных данных: {:.6f}'.format(r2_original))
print('R2 score кодированных данных: {:.6}'.format(r2_encoded))
print('R2 score декодированных данных: {:.6f}'.format(r2_decoded))
```

```
R2 score исходных данных: 0.435228
R2 score кодированных данных: 0.435228
R2 score декодированных данных: 0.435228
```

In [24]:
```
r2_original = round(r2_original, 12)
r2_encoded = round(r2_encoded, 12)
r2_decoded = round(r2_decoded, 12)

if (r2_original == r2_encoded) & (r2_original == r2_decoded):
    print('R2 score равны. Алгоритм кодирования работает')
else:
    print('R2 score не равны. Алгоритм кодирования не работает')
```

```
R2 score равны. Алгоритм кодирования работает
```

Общий вывод

Проведено исследование с целью разработать такой метод преобразования данных, чтобы по ним было сложно восстановить персональную информацию и при преобразовании качество моделей машинного обучения не ухудшилось.

Данные получили из файла insurance.csv.

Исследование проходило в пять этапов:

- Загрузка данных
  - Добавили библиотеки и функции, необходимые для нашего исследования
  - Загрузили данные
- Изучение данных
  - Данные состоят из 5000 объектов
  - Имееют 5 признаков:
    - Вспомогательные признаки:
      - пол
      - возраст
      - зарплата застрахованного
      - количество членов его семьи
    - Целевой признак:
      - количество страховых выплат клиенту за последние 5 лет
  - Пропуски отсутствуют
  - Данные в признаках "Возраст" и "Зарплата" имели формат *float*, но целочисленные значения
  - Все признаки имели малые целочисленные значения - мы преобразовали их в формат *int32* для уменьшения памяти и быстродеятсвия
  - Содержат 153 явных дубликата (На решение задачи исследования не влияет, мы оставили эти данные для обучения)
  - Проверили признаки на наличие аномалий:
    - "Пол" - должен содержать 2 уникальных значения
    - "Возраст" - проверим по критериям 18 <= Возраст <= 100
    - "Зарплата" - не должен быть равен 0
    - "Члены семьи" - не должен быть меньше 0
    - "Страховые выплаты" - не должен быть меньше 0
    - Аномалий не обнаружено
- Умножение матриц
  - Доказали что качество линейной регрессии при умножении признаков X на обратимую матрицу P не изменится
  - Параметры линейной регрессии в исходной задаче w и в преобразованной w_1 связаны по формуле: $w_1 = P^{-1} w$
- Выбор алгоритма преобразования
  - В качестве алгоритма преобразования данных предложили умножение исходных данных на обратимую матрицу со случайными величинами
  - Для создания обратимой матрицы со случайными величинами воспользовались функцией numpy.random.randn()
  - После кодирования-декодированния данные не изменились
  - По кодированным данным сложно восстановить персональную информацию
- Проверка алгоритма
  - Проверили, что качество линейной регрессии из sklearn не отличается до и после преобразования
    - Применим метрику R2
      - R2 score исходных данных: 0.435228
      - R2 score кодированных данных: 0.435228
      - R2 score декодированных данных: 0.435228
    - R2 score равны. Алгоритм кодирования работает