МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3

по дисциплине «Технологии и инструментарий анализа больших данных»

Выполнил студент группы ИКБО-20-21 Фомичев Р.А.

Проверил ассистент кафедры ПМ ИИТ Тетерин Н.Н.

Москва 2024

1. Загрузить данные из файла “insurance.csv”. С помощью метода describe() посмотреть статистику по данным. Сделать выводы.

Код программы представлен на рисунке 1.

```
df = pd.read_csv('prak3/insurance.csv')
print(df.describe())
```

Рисунок 1 – Код программы

Результат работы программы представлен на рисунке 2.

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

Рисунок 2 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что средний возраст опрошенных 39 лет, средние выплаты составляют 13270 у.е., в среднем опрошенные имеют не более 2 детей.

2. Построить гистограммы для числовых показателей. Сделать выводы.

Код представлен на рисунке 3.

```
fix, ax = plt.subplots(1, 4, figsize=(20, 10))
ax[0].hist(df['age'], label='age')
ax[0].legend()
ax[1].hist(df['bmi'], label='bmi')
ax[1].legend()
ax[2].hist(df['children'], label='children')
ax[2].legend()
ax[3].hist(df['charges'], label='charges')
ax[3].legend()
plt.show()
```

Рисунок 3 – Код программы

Результат работы программы представлен на рисунке 4.

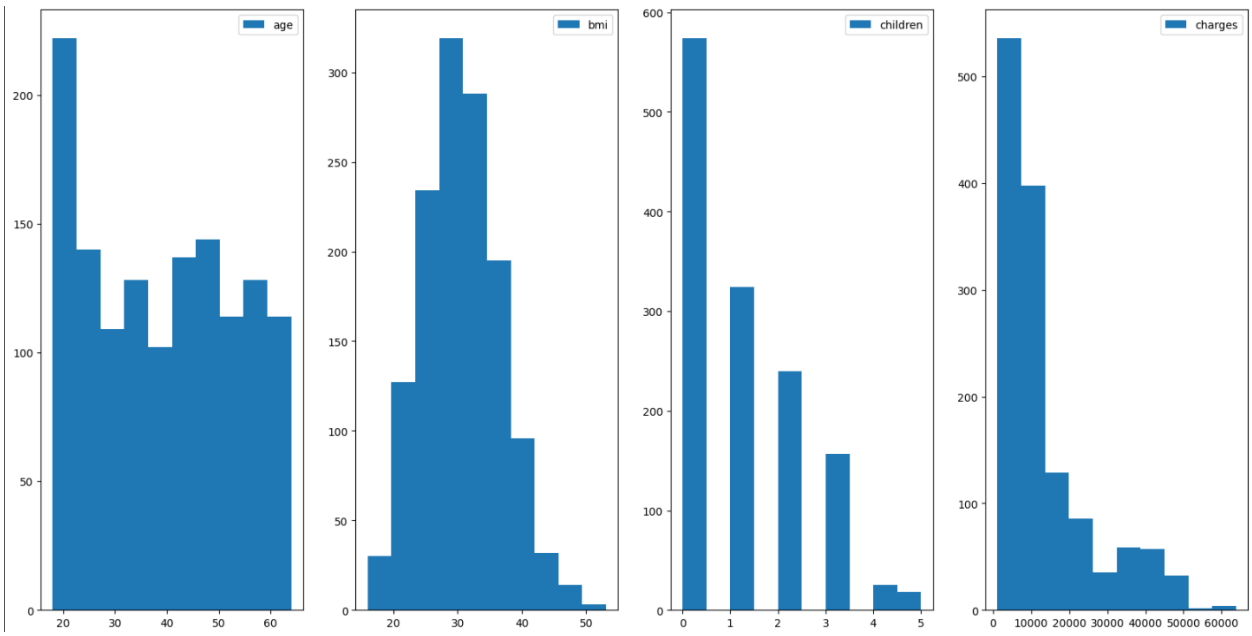

Рисунок 4 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что больше всего опрошенных было возраста до 20 лет, имт больше всего от 25 до 30, больше всего людей из выборки без детей, а самая частая выплатой является до 10000 долларов.

3. Найти меры центральной тенденции и меры разброса для индекса массы тела (bmi) и расходов (charges). Отобразить результаты в виде текста и на гистограммах (3 вертикальные линии). Добавить легенду на графики. Сделать выводы.

Код программы представлен на рисунке 5.

```python
bmi_mean = df['bmi'].mean()
bmi_median = df['bmi'].median()
bmi_mode = df['bmi'].mode()[0]
bmi_std = df['bmi'].std()

charges_mean = df['charges'].mean()
charges_median = df['charges'].median()
charges_mode = df['charges'].mode()[0]
charges_std = df['charges'].std()

print(f"иМТ: \n"
    f"Среднее: {bmi_mean}, \n"
    f"медиана: {bmi_median}, \n"
    f"Мода: {bmi_mode}, \n"
    f"Стандартное отклонение: {bmi_std}\n")
print("Charges: \n"
    f"Среднее: {charges_mean}, \n"
    f"Медиана: {charges_median}, \n"
    f"МоДа: {charges_mode}, \n"
    "Стандартное отклонение: {charges_std}\n")

plt.figure(figsize=(10, 5))
sns.histplot(df['bmi'], bins=15, kde=True)
plt.axvline(bmi_mean, color='red', label=f'Среднее = {bmi_mean}')
plt.axvline(bmi_median, color='green', label=f'Медиана {bmi_median}')
plt.axvline(bmi_mode, color='blue', label=f'Мода = {bmi_mode}')
plt.title("ИМТ")
plt.legend()
plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(df['charges'], bins=15, kde=True)
plt.axvline(charges_mean, color='red', label=f'Среднее {charges_mean}')
plt.axvline(charges_median, color='green', label='Медиана = {charges_median}')
plt.axvline(charges_mode, color='blue', label=f'Мода = {charges_mode}')
plt.title('Charges')
plt.legend()
plt.show()
```

Рисунок 5 – Код программы

Результат представлен на рисунке 6.

```
ИМТ:
Среднее: 30.66339686098655,
медиана: 30.4,
Мода: 32.3,
Стандартное отклонение: 6.098186911679017

Charges:
Среднее: 13270.422265141257,
Медиана: 9382.033,
МоДа: 1639.5631,
Стандартное отклонение: 12110.011236693994
```

Рисунок 6 – Результат работы программы

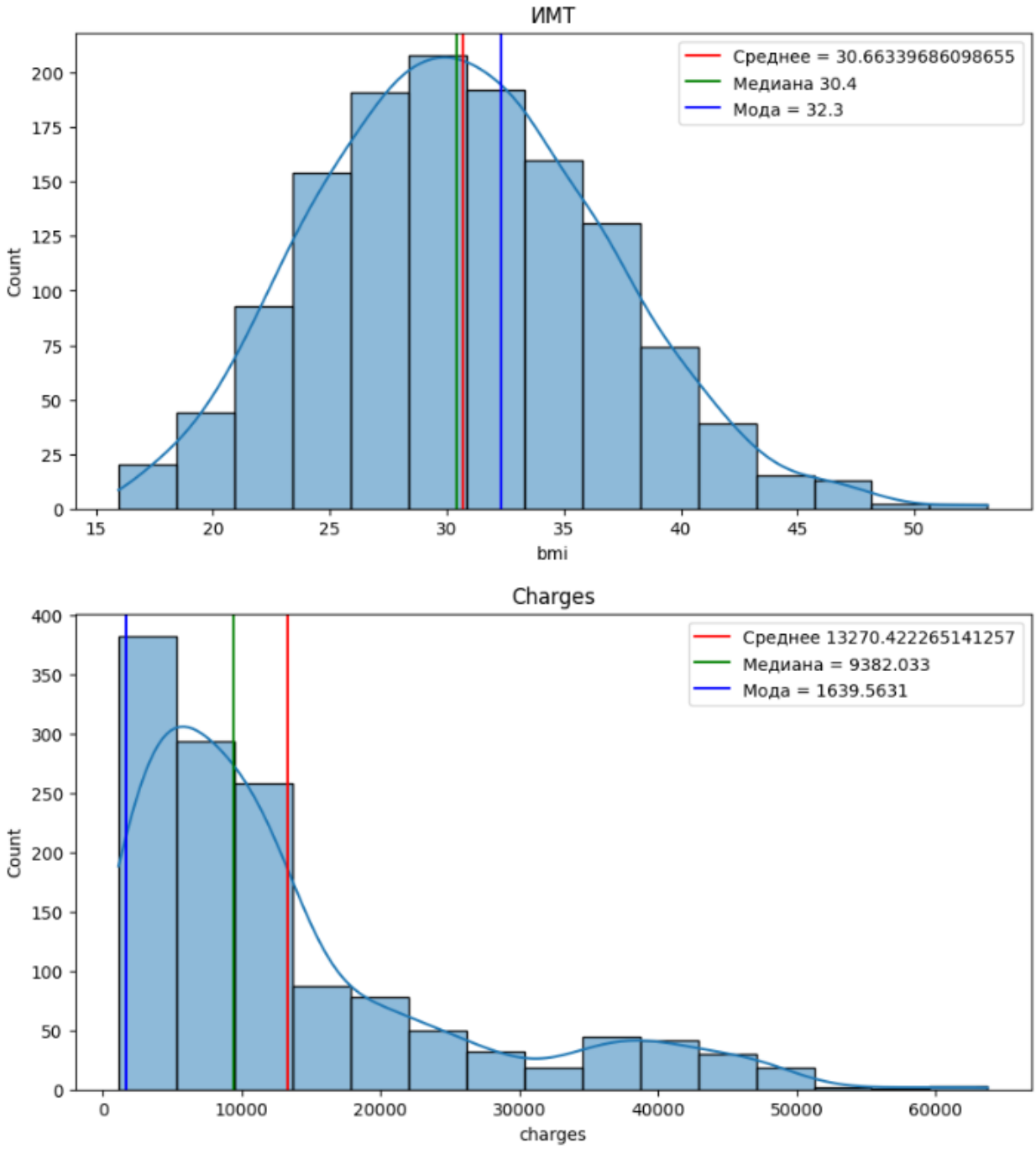

Рисунок 7 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что средняя и медианная ИМТ приблизительно равны 30, стандартное отклонение приблизительно равно 6. Средняя и медианная значений выплат отличаются и равны приблизительно 13270 и 9382 соответственно, стандартное отклонение равно приблизительно 12110.

4. Построить box-plot для числовых показателей. Названия графиков должны соответствовать названиям признаков. Сделать выводы.

Код программы представлен на рисунке 8.

```
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
plt.subplots_adjust(wspace=0.7, hspace=0.7)

axs[0,0].boxplot(df['age'], labels=['Возраст'], vert=False)
axs[0,1].boxplot(df['bmi'], labels=['Индекс массы тела'], vert=False)
axs[1,0].boxplot(df['children'], labels=['кол-во детей'], vert=False)
axs[1,1].boxplot(df['charges'], labels=['Выплаты'], vert=False)
plt.grid()
plt.show()
```

Рисунок 8 – Код программы

Результат работы программы представлен на рисунке 9.

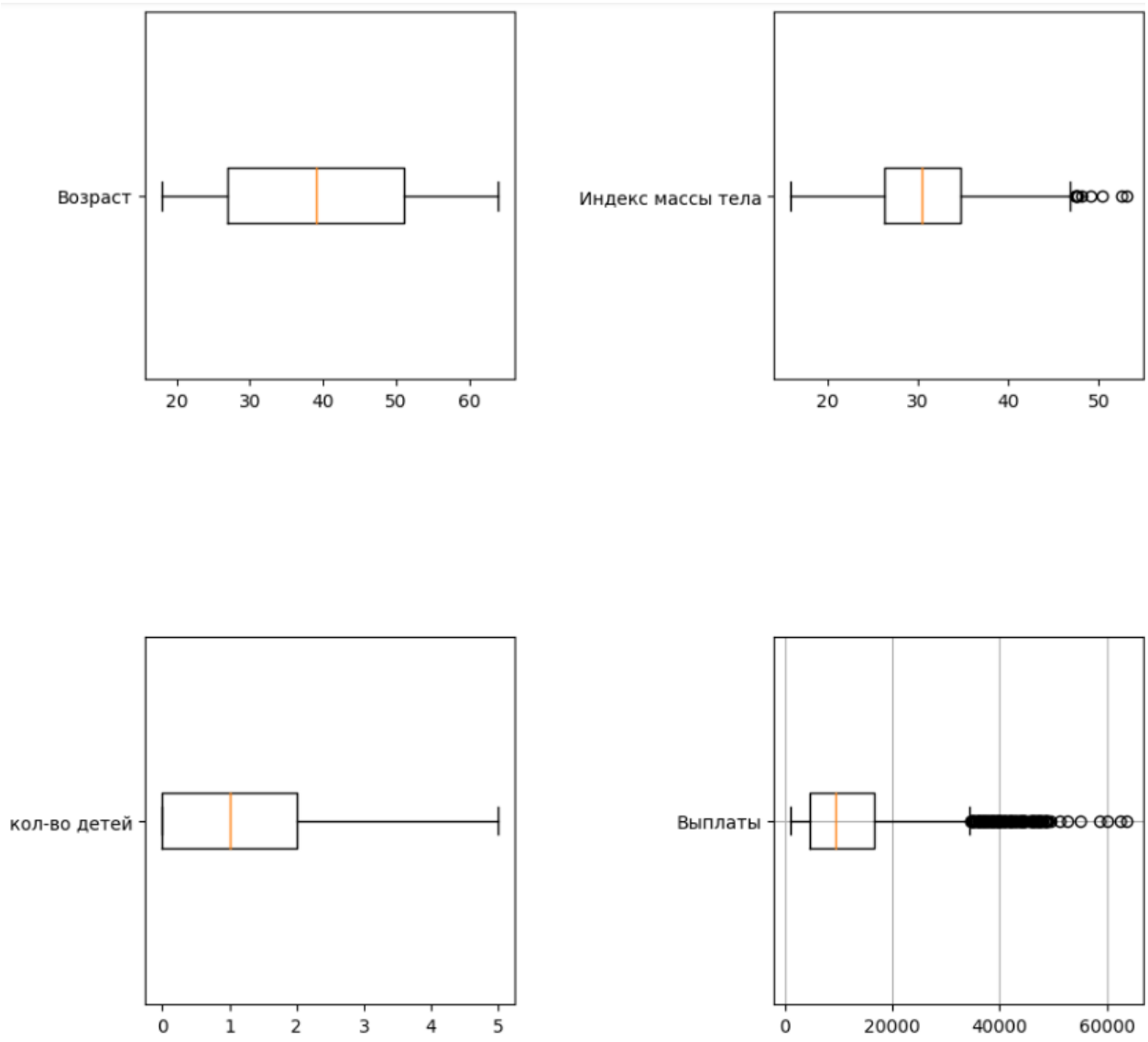

Рисунок 9 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что медианный возраст приблизительно равен 40, в квартиль попадает возраст от 30 до 50 приблизительно. ИМТ приблизительно равен 32, в квартиль попадает ИМТ от 26 до 38, ИМТ более 47 является вбросами. Количество детей приблизительно равно 1, в квартиль попадает количество детей от 0 до 2. Средние выплаты приблизительно равны 10000 у.е, выплаты более 36000 являются вбросами.

5. Используя признак charges или imb, проверить, выполняется ли центральная предельная теорема. Использовать различные длины выборок n. Количество выборок = 300. Вывести результат в виде гистограмм. Найти

стандартное отклонение и среднее для полученных распределений. Сделать выводы.

```python
df = pd.read_csv('prak3/insurance.csv')
feature_data = df['bmi']

sample_means = []
n_samples = 300

# Различные длины выборок
sample_sizes = [30, 50, 100, 500, 1000]
for sample_size in sample_sizes:
    means = []
    for _ in range(n_samples):
        sample = np.random.choice(feature_data, sample_size)
        means.append(np.mean(sample))
    sample_means.append(means)

fig, axs = plt.subplots(len(sample_sizes), 1, figsize=(10, 20))

for i, sample_size in enumerate(sample_sizes):
    axs[i].hist(sample_means[i], bins=30, density=True)
    axs[i].set_title(f'Sample size: {sample_size}, Mean: {np.mean (sample_means[i]):.2f}, Std Dev: {np.std(sample_means[i]):.2f}')

plt.tight_layout()
plt.show()
```

Рисунок 10 – Код программы

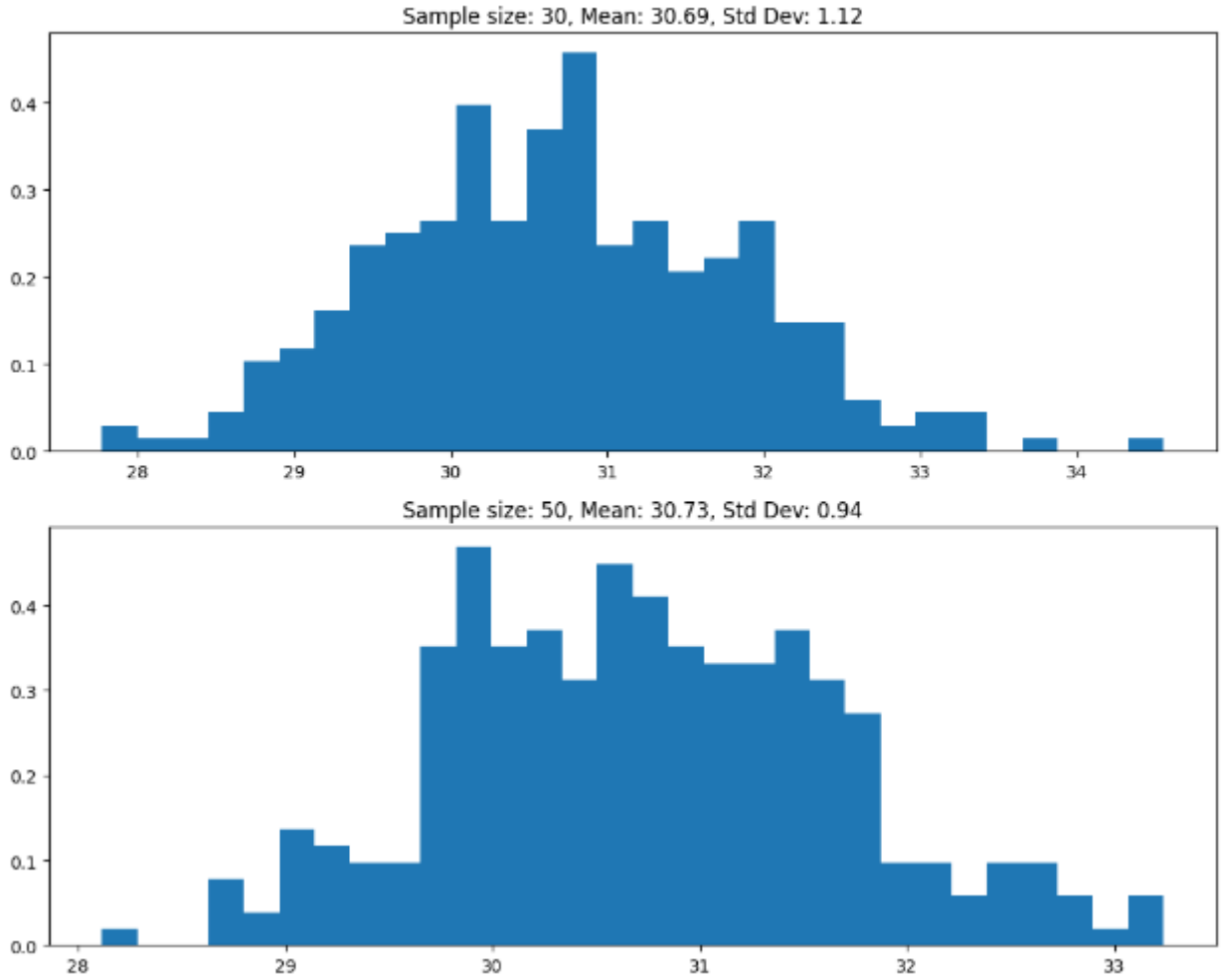

Рисунок 11 – Результат работы программы

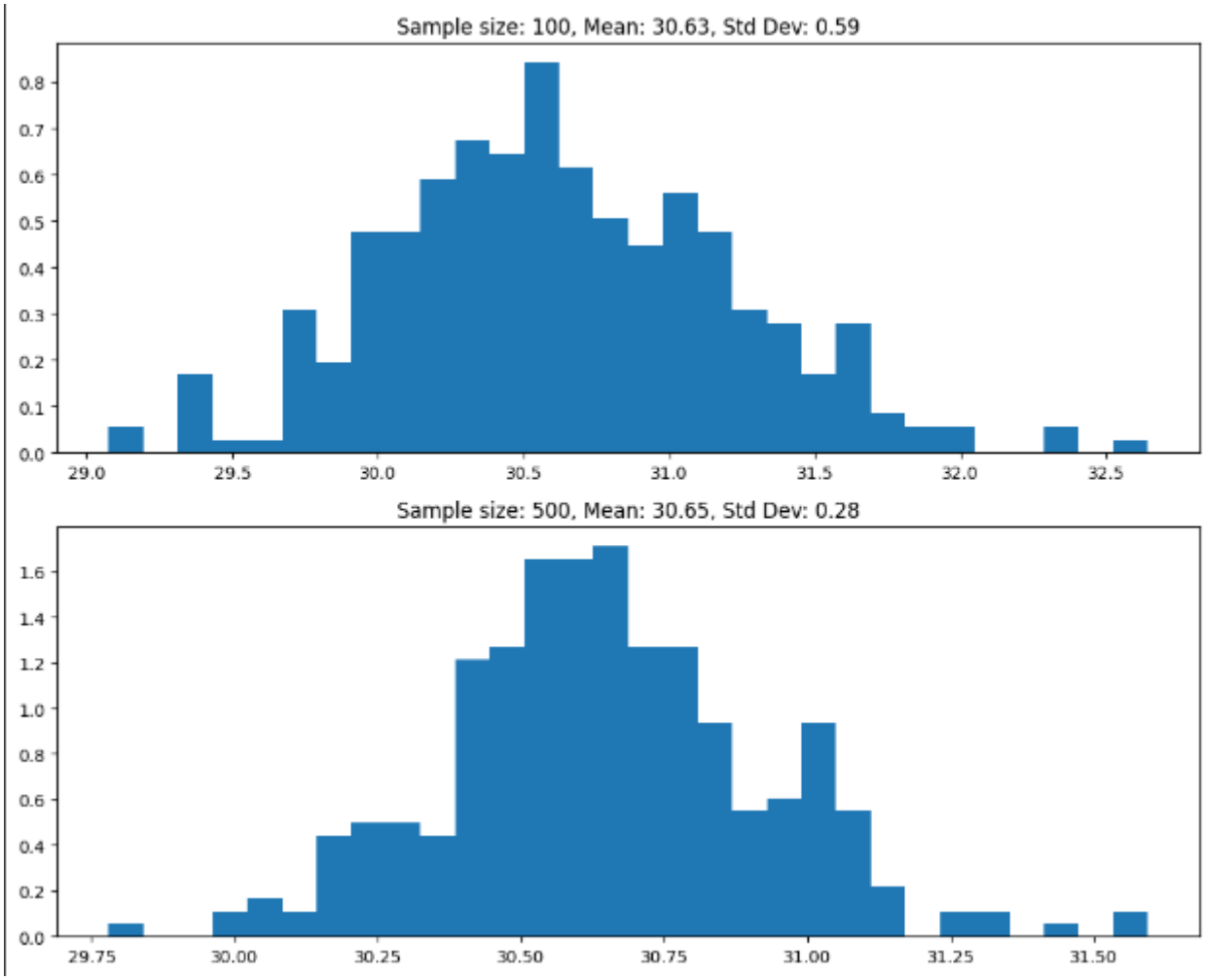

Рисунок 12 – Результат работы программы

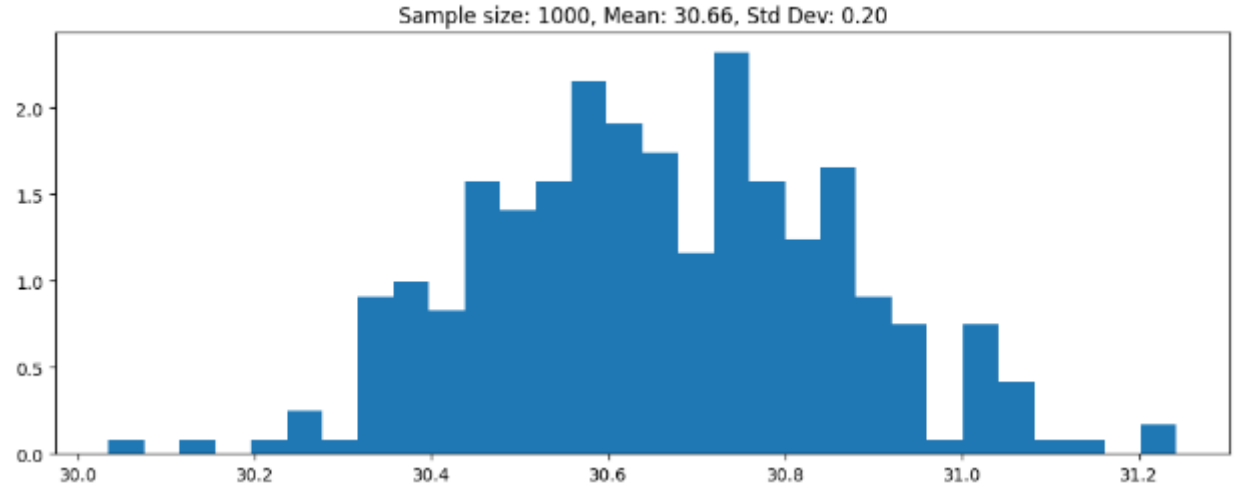

Рисунок 13 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что при увеличении размера выборок стандартное отклонение уменьшается, а среднеквадратичное остается неизменным. Центральная предельная теорема

выполняется, так как построенные графики близки к распределению по Гауссу.

6. Построить 95% и 99% доверительный интервал для среднего значения расходов и среднего значения индекса массы тела.

```
n = len(df['bmi'])
mean = df['bmi'].mean()
stderr = df['bmi'].std() / np.sqrt(n)
margin = stderr * sts.t.ppf((1 + 0.95) / 2.0, n - 1)
print("BMI 95:", mean - margin, mean + margin)

n = len(df['charges'])
mean = df['charges'].mean()
stderr=df['charges'].std() / np.sqrt(n)
margin = stderr * sts.t.ppf((1 + 0.95) / 2.0, n - 1)
print("Charges 95:", mean - margin, mean + margin)

n = len(df['bmi'])
mean = df['bmi'].mean()
stderr = df['bmi'].std() / np.sqrt(n)
margin = stderr * sts.t.ppf((1 + 0.99) / 2.0, n - 1)
print("BMI 99:", mean - margin, mean + margin)

n = len(df['charges'])
mean = df['charges'].mean()
stderr = df['charges'].std() / np.sqrt(n)
margin = stderr * sts.t.ppf((1 + 0.99) / 2.0, n - 1)
print("Charges 99:", mean - margin, mean + margin)
```

Рисунок 16 – Код программы

```
BMI 95: 30.336346903054107 30.99044681891899
Charges 95: 12620.954034192644 13919.890496089869
BMI 99: 30.233355575431624 31.093438146541473
Charges 99: 12416.429943203952 14124.414587078561
```

Рисунок 17 – Результат работы программы

7. Проверить распределения следующих признаков на нормальность: индекс массы тела, расходы. Сформулировать нулевую и альтернативную гипотезы. Для каждого признака использовать KS-тест и q-q plot. Сделать выводы на основе полученных p-значений.

```
bmi_data = df['bmi']
charges_data = df['charges']

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sts.probplot(bmi_data, dist="norm", plot=plt)
plt.title('Q-Q график для BMI')

plt.subplot(1, 2, 2)
ks_statistic_bmi, p_value_bmi = sts.kstest(bmi_data, 'norm')
sts.probplot(bmi_data, dist="norm", plot=plt)
plt.title(f'KS-тест для BMI (p-value={p_value_bmi:.4f})')

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sts.probplot(charges_data, dist="norm", plot=plt)
plt.title('Q-Q график дя Charges')

plt.subplot(1, 2, 2)
ks_statistic_charges, p_value_charges = sts.kstest(charges_data, 'norm')
sts.probplot(charges_data, dist="norm", plot=plt)
plt.title(f'KS-тест для Charges (p-value={p_value_charges:.4f})')

plt.tight_layout()
plt.show()

alpha = 0.05 # Уровень значимости

if p_value_bmi < alpha:
    print("Для BMI: Отвергаем нулевую гипотезу. Распределение не является нормальным.")
else:
    print("Для BMI: Не отвергаем нулевую гипотезу. Распределение является нормальным.")

if p_value_charges < alpha:
    print("Для Charges: Отвергаем нулевую гипотезу. Распределение не является нормальным.")
else:
    print("Для Charges: Не отвергаем нулевую гипотезу. Распределение является нормальным.")
```

Рисунок 18 – Код программы

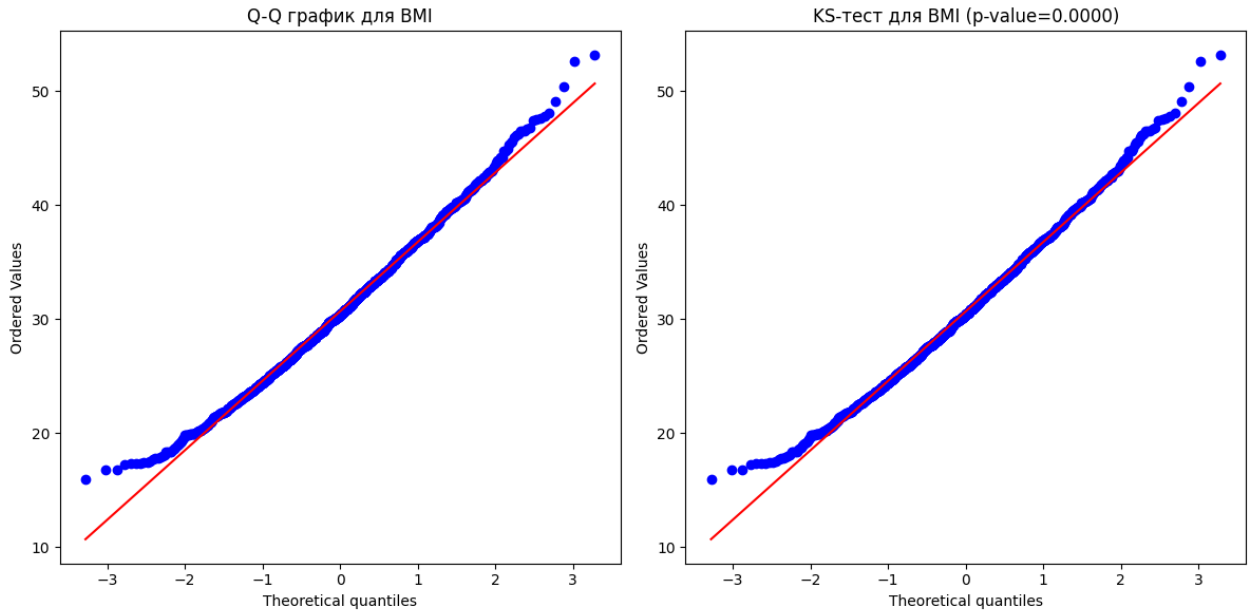

Рисунок 20 – Результат работы программы

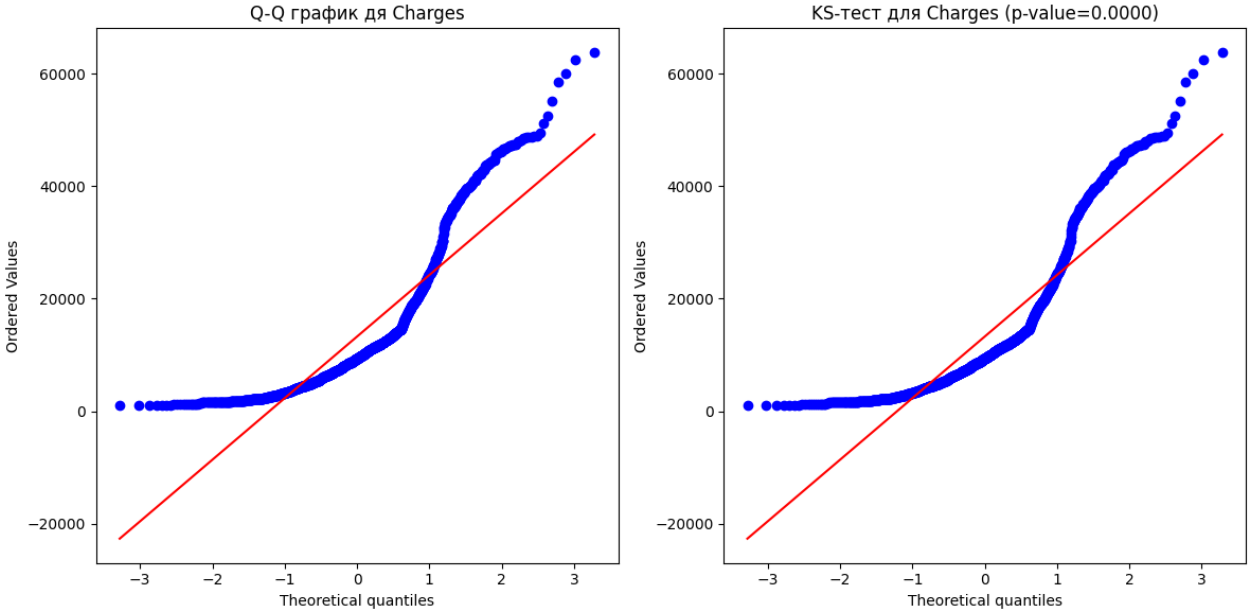

Рисунок 21 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что распределение ИМТ соответствует нормальному распределению, а время выплаты не соответствуют.

8. Загрузить данные из файла “ECDCCases.csv”.

```
data = pd.read_csv('ECDCCases.csv')
```

Рисунок 22 – Код программы

9. Проверить в данных наличие пропущенных значений. Вывести количество пропущенных значений в процентах. Удалить два признака, в

которых больше всех пропущенных значений. Для оставшихся признаков обработать пропуски: для категориального признака использовать заполнение значением по умолчанию (например, «other»), для числового признака использовать заполнение медианным значением. Показать, что пропусков больше в данных нет.

```
missing_values = data.isnull().sum()
percentage_missing = (missing_values / len(data)) * 100
print("Пропущенные значения: " + str(missing_values))
print("Процент пропущенных значений: " + str(percentage_missing))

most_missing_features = missing_values.nlargest(2).index
data = data.drop(columns=most_missing_features)

data['countryterritoryCode'] = data['countryterritoryCode'].fillna('other')
data['popData2019'] = data['popData2019'].fillna(data['popData2019'].median())

print("Пустых значений нет:", data.isnull().sum().sum() == 0)
```

Рисунок 23 – Код программы

```
Пропущенные значения: dateRep                                    0
day                                                              0
month                                                            0
year                                                             0
cases                                                            0
deaths                                                           0
countriesAndTerritories                                          0
geoId                                                          275
countryterritoryCode                                           123
popData2019                                                    123
continentExp                                                     0
Cumulative_number_for_14_days_of_COVID-19_cases_per_100000    2879
dtype: int64
Процент пропущенных значений: dateRep                                 0.000000
day                                                           0.000000
month                                                         0.000000
year                                                          0.000000
cases                                                         0.000000
deaths                                                        0.000000
countriesAndTerritories                                       0.000000
geoId                                                         0.444236
countryterritoryCode                                          0.198695
popData2019                                                   0.198695
continentExp                                                  0.000000
Cumulative_number_for_14_days_of_COVID-19_cases_per_100000    4.650750
dtype: float64
Пустых значений нет: True
```

Рисунок 24 – Результат работы программы

10. Посмотреть статистику по данным, используя describe(). Сделать выводы о том, какие признаки содержат выбросы. Посмотреть, для каких стран количество смертей в день превысило 3000 и сколько таких дней было.

```
print(data.describe())
over3000 = data.loc[data['deaths'] > 3000]
print(over3000['countriesAndTerritories'].value_counts())
```

Рисунок 25 – Код программы

```
                day          month           year          cases        deaths  \
count  61904.000000  61904.000000  61904.000000   61904.000000  61904.000000
mean      15.629232      7.067104   2019.998918    1155.079026     26.053987
std        8.841624      2.954816      0.032881    6779.010824    131.222948
min        1.000000      1.000000   2019.000000   -8261.000000  -1918.000000
25%        8.000000      5.000000   2020.000000       0.000000      0.000000
50%       15.000000      7.000000   2020.000000      15.000000      0.000000
75%       23.000000     10.000000   2020.000000     273.000000      4.000000
max       31.000000     12.000000   2020.000000  234633.000000   4928.000000

         popData2019
count   6.190400e+04
mean    4.091909e+07
std     1.529798e+08
min     8.150000e+02
25%     1.324820e+06
50%     7.169456e+06
75%     2.851583e+07
max     1.433784e+09
countriesAndTerritories
United_States_of_America    6
Peru                        2
Argentina                   1
Ecuador                     1
Mexico                      1
Name: count, dtype: int64
```

Рисунок 26 – Результат работы программы

Основываясь на полученных данных, следует следующий вывод, что среди данных есть вбросы, потому что существуют отрицательные значения у полей смертей и случаи.

11. Найти дублирование данных. Удалить дубликаты.

```
dupl= data.duplicated()
print(f'Количество одинаковых строк: { dupl.sum()}')
data = data.drop_duplicates()
print(data)
```

Рисунок 27 – Код программы

```
Количество одинаковых строк: 4
          dateRep  day  month  year  cases  deaths countriesAndTerritories  \
0      14/12/2020   14     12  2020    746       6              Afghanistan
1      13/12/2020   13     12  2020    298       9              Afghanistan
2      12/12/2020   12     12  2020    113      11              Afghanistan
4      11/12/2020   11     12  2020     63      10              Afghanistan
5      10/12/2020   10     12  2020    202      16              Afghanistan
...           ...  ...    ...   ...    ...     ...                      ...
61899  25/03/2020   25      3  2020      0       0                 Zimbabwe
61900  24/03/2020   24      3  2020      0       1                 Zimbabwe
61901  23/03/2020   23      3  2020      0       0                 Zimbabwe
61902  22/03/2020   22      3  2020      1       0                 Zimbabwe
61903  21/03/2020   21      3  2020      1       0                 Zimbabwe

      countryterritoryCode  popData2019 continentExp
0                      AFG   38041757.0         Asia
1                      AFG   38041757.0         Asia
2                      AFG   38041757.0         Asia
4                      AFG   38041757.0         Asia
5                      AFG   38041757.0         Asia
...                    ...          ...          ...
61899                  ZWE   14645473.0       Africa
61900                  ZWE   14645473.0       Africa
61901                  ZWE   14645473.0       Africa
61902                  ZWE   14645473.0       Africa
61903                  ZWE   14645473.0       Africa

[61900 rows x 10 columns]
```

Рисунок 28 – Результат работы программы

12. Загрузить данные из файла “bmi.csv”. Взять оттуда две выборки. Одна выборка – это индекс массы тела людей с региона northwest, вторая выборка – это индекс массы тела людей с региона southwest. Сравнить средние значения этих выборок, используя t-критерий Стьюдента. Предварительно проверить выборки на нормальность (критерий ШопироУилка) и на гомогенность дисперсии (критерий Бартлетта).

```python
data = pd.read_csv("bmi.csv")

northwest_bmi = data[data['region'] == 'northwest']['bmi']
southwest_bmi = data[data['region'] == 'southwest']['bmi']

_, p_value_northwest = stats.shapiro(northwest_bmi)
_, p_value_southwest = stats.shapiro(southwest_bmi)

_, p_value_bartlett = stats.bartlett(northwest_bmi, southwest_bmi)

print(f"p-value (Шапиро-Уилка) для выборки из northwest: {p_value_northwest:.4f}")
print(f"p-value (Шапиро-Уилка) для выборки из southwest: {p_value_southwest:.4f}")
print(f"p-value (Бартлетт) для проверки гомогенности дисперсии: {p_value_bartlett:.4f}")

if p_value_northwest > 0.05 and p_value_southwest > 0.05 and p_value_bartlett > 0.05:
    t_statistic, p_value_ttest = stats.ttest_ind(northwest_bmi, southwest_bmi)
    print(f"t-статистика: {t_statistic:.4f}")
    print(f"p-value (t-критерий Стьюдента): {p_value_ttest:.4f}")

    if p_value_ttest < 0.05:
        print("Различия в средних значениях выборок статистически значимы.")
    else:
        print("Нет статистически значимых различий в средних значениях выборок.")
else:
    print("Условия для использования t-критерия Стьюдента не выполняются.")
```

Рисунок 29 – Код программы

```
p-value (Шапиро-Уилка) для выборки из northwest: 0.4656
p-value (Шапиро-Уилка) для выборки из southwest: 0.3630
p-value (Бартлетт) для проверки гомогенности дисперсии: 0.0652
t-статистика: -3.2844
p-value (t-критерий Стьюдента): 0.0011
Различия в средних значениях выборок статистически значимы.
```

Рисунок 30 – Результат работы программы

13. Кубик бросили 600 раз, получили следующие результаты: N Количество выпадений 1 97 2 98 3 109 4 95 5 97 6 104 С помощью критерия Хи-квадрат проверить, является ли полученное распределение равномерным. Использовать функцию scipy.stats.chisquare().

```python
observed_frequencies = np.array([97, 98, 109, 95, 97, 104])

expected_frequencies = np.array([100] * 6)

chi2_statistic, p_value = stats.chisquare(observed_frequencies, expected_frequencies)

print(f"Значение критерия Хи-квадрат: {chi2_statistic:.2f}")
print(f"p-значение: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Отвергаем нулевую гипотезу: распределение не является равномерным.")
else:
    print("Не отвергаем нулевую гипотезу: распределение равномерное.")
```

Рисунок 31 – Код программы

```
Значение критерия Хи-квадрат: 1.44
p-значение: 0.9199
Не отвергаем нулевую гипотезу: распределение равномерное.
```

Рисунок 32 – Результат работы программы

14. С помощью критерия Хи-квадрат проверить, являются ли переменные зависимыми. Создать датафрейм, используя следующий код: data = pd.DataFrame({'Женат': [89,17,11,43,22,1], 'Гражданский брак': [80,22,20,35,6,4], 'Не состоит в отношениях': [35,44,35,6,8,22]}) data.index = ['Полный рабочий день','Частичная занятость','Временно не работает','На домохозяйстве','На пенсии','Учёба'] Использовать функцию scipy.stats.chi2_contingency(). Влияет ли семейное положение на занятость?

```python
data = pd.DataFrame({'Женат': [89, 17, 11, 43, 22, 1],
    'Гражданский брак': [80, 22, 20, 35, 6, 4],
    'Не состоит в отношениях': [35, 44, 35, 6, 8, 22]})

data.index = ['Полный рабочий день', 'Частичная занятость', 'Временно не работает', 'На домохозяйстве', 'На пенсии', 'Учёба']

chi2_statistic, p_value, dof, expected = sts.chi2_contingency(data)

print(f"Значение критерия Хи-квадрат: {chi2_statistic:.2f}")
print(f"p-значение: {p_value:.4f}")

alpha = 0.05
if p_value < alpha:
    print("Отвергаем нулевую гипотезу: семейное положение влияет на занятость.")
else:
    print("Не отвергаем нулевую гипотезу: семейное положение не влияет на занятость.")
```

Рисунок 33 – Код программы

```
Значение критерия Хи-квадрат: 122.30
p-значение: 0.0000
Отвергаем нулевую гипотезу: семейное положение влияет на занятость.
```

Рисунок 34 – Результат работы программы