МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Автоматизация схемотехнического проектирования»

Тема: **Анализ экспериментальных данных**

Студент гр. 1302 _____ Новиков Г.В.

Преподаватель _____ Боброва Ю.О.

Санкт-Петербург

2025

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Новиков Г.В.

Группа 1302

Тема работы: Анализ экспериментальных данных

Исходные данные:

Датасет, содержащий информацию о мобильных телефонах.

Содержание пояснительной записки:

«Содержание», «Введение», «График распределения параметров», «Медиана», «Графики распределения при медианном значении», «Среднее значение и стандартное отклонение», «Различия в классах», «Классификация», «Листинг», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 03.02.2025

Дата сдачи реферата: 24.03.2025

Дата защиты реферата: 24.03.2025

Студент гр. 1302 _____ Новиков Г.В.

Преподаватель _____ Боброва Ю.О.

АННОТАЦИЯ

В данной курсовой работе проводится анализ экспериментальных данных, представленных в виде датасета, содержащего характеристики мобильных телефонов. Основной целью работы является исследование параметров, таких как цена, диагональ экрана, объем оперативной памяти и другие, с использованием различных статистических методов и визуализации данных.

SUMMARY

In this course work, an analysis of experimental data is conducted, presented in the form of a dataset containing characteristics of mobile phones. The main goal of the work is to investigate parameters such as price, screen size, RAM capacity, and others, using various statistical methods and data visualization techniques.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Датасет содержит 1512 строк с характеристиками мобильных телефонов:

1. phone_name — Название модели телефона.
2. brand — Бренд или производитель телефона.
3. os — Операционная система, установленная на телефоне.
4. inches — Диагональ экрана телефона в дюймах.
5. resolution — Разрешение экрана в формате "ширина x высота" (например, "1080x1920").
6. battery — Емкость аккумулятора телефона в миллиампер-часах (мАч).
7. battery_type — Тип аккумулятора (например, "Li-Ion" или "Li-Po").
8. ram(GB) — Объем оперативной памяти (RAM) в гигабайтах (ГБ).
9. announcement_date — Дата анонса телефона в формате "год-месяц-день".
10. weight(g) — Вес телефона в граммах (г).
11. storage(GB) — Объем встроенной памяти телефона в гигабайтах (ГБ).
12. video_720p — Поддержка записи видео с разрешением 720p (HD).
13. video_1080p — Поддержка записи видео с разрешением 1080p (Full HD).
14. video_4K — Поддержка записи видео с разрешением 4K (Ultra HD).
15. video_8K — Поддержка записи видео с разрешением 8K.
16. video_30fps — Поддержка записи видео с частотой 30 кадров в секунду (fps).
17. video_60fps — Поддержка записи видео с частотой 60 кадров в секунду (fps).
18. video_120fps — Поддержка записи видео с частотой 120 кадров в секунду (fps).
19. video_240fps — Поддержка записи видео с частотой 240 кадров в секунду (fps).
20. video_480fps — Поддержка записи видео с частотой 480 кадров в секунду (fps).

21. video_960fps — Поддержка записи видео с частотой 960 кадров в секунду (fps).

22. price(USD) — Цена телефона в долларах США (USD)

Рассматриваемый параметр – price(USD).

Задачи:

1. Построить график распределения параметров. Разделить выборку по классам. Подписать оси, добавить легенду и сетку на график.

2. Рассчитать медиану параметра n: для выборки в целом, для каждого класса отдельно

3. Рассчитать медиану параметра m: для выборки в целом, для каждого класса отдельно

4. Построить график распределения параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

5. Построить гистограмму распределения, скатерограмму и боксплот параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

6. Рассчитать среднее значение и стандартное отклонение для параметров: для всей выборки и для каждого класса отдельно.

7. Статистически оценить различия в классах по параметру n- совпадают ли выборки по этому параметру или нет.

8. Статистически оценить различия в классах по параметру m - совпадают ли выборки по этому параметру или нет.

9. Классификация данных.

1. ГРАФИКИ РАПРЕДЕНИЯ ПАРАМЕТРОВ

Задание: построить график распределения параметров. Разделить выборку по классам. Подписать оси, добавить легенду и сетку на график.

Для начала построим гистограмму для рассматриваемого параметра.

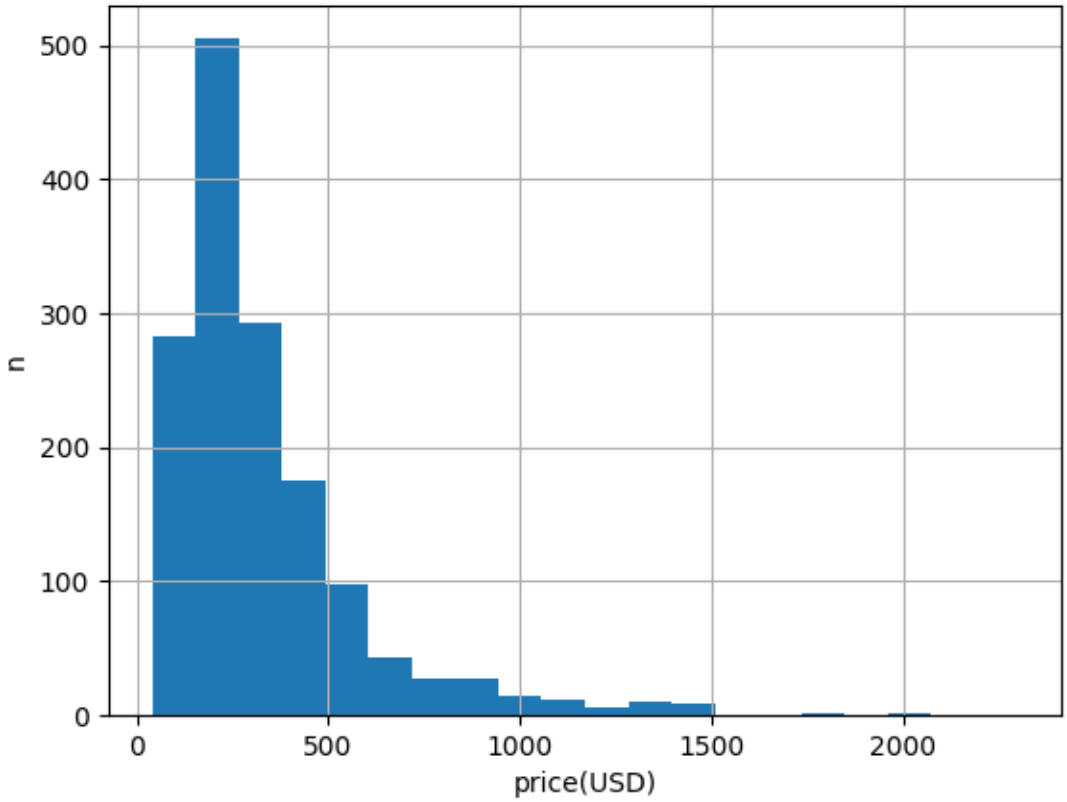

Рис. 1. Рассматриваемый параметр (price (USD))

Разделим выборку на 2 класса:

Класс 1: объекты со значением price(USD) больше или равным медиане.

Класс 2: объекты со значением price(USD) меньше медианы.

Медиана price(USD) равна 260.

Построим гистограммы для оставшихся параметров с разделением на классы. Для параметров brand, os, resolution, announcement_date построены столбчатые диаграммы с 10 наиболее часто встречающимися значениями. Для

параметра phone_name график не был построен, так как для большинства строк значение уникально.

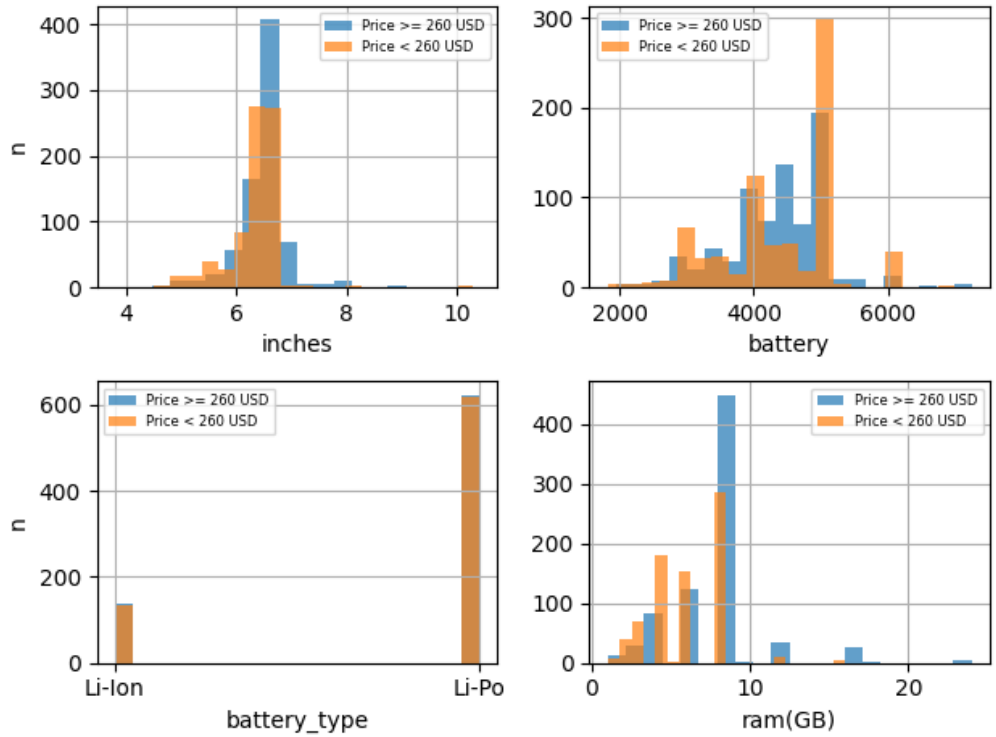

Рис. 2. Параметры

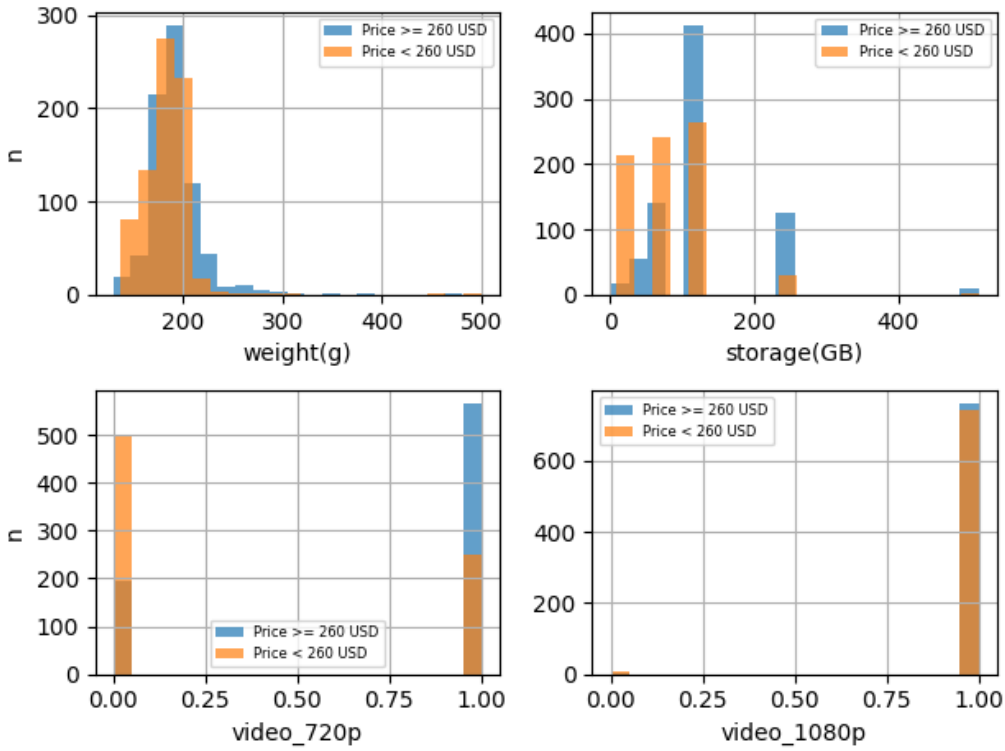

Рис. 3. Параметры

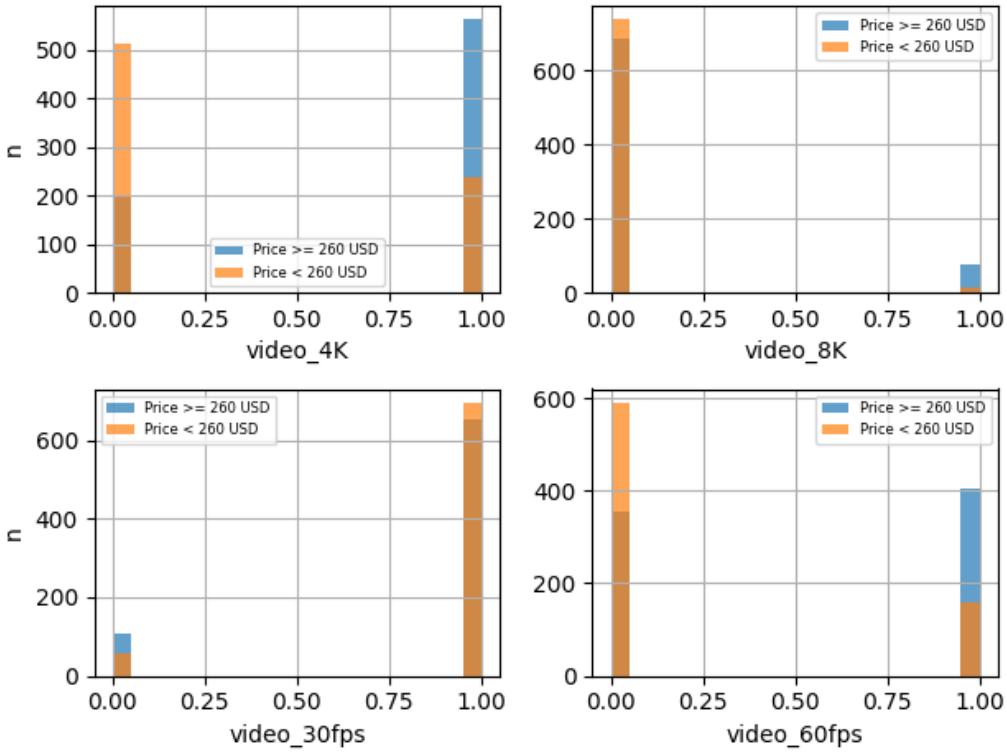

Рис. 4. Параметры

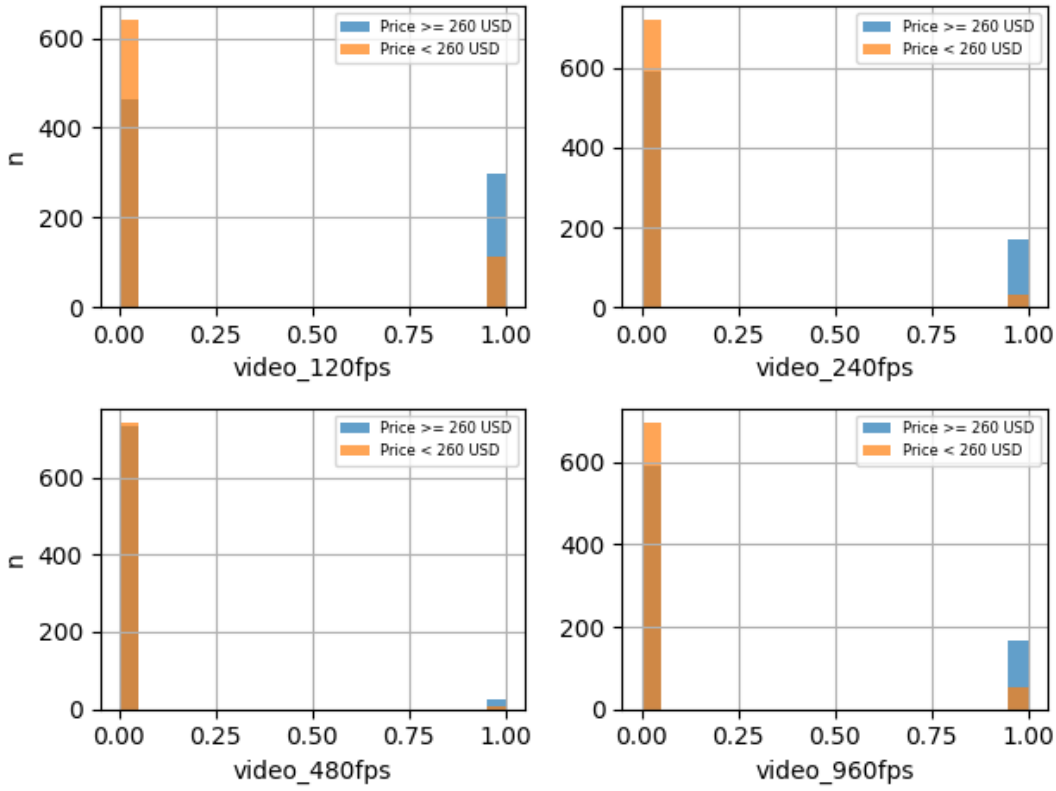

Рис. 5. Параметры

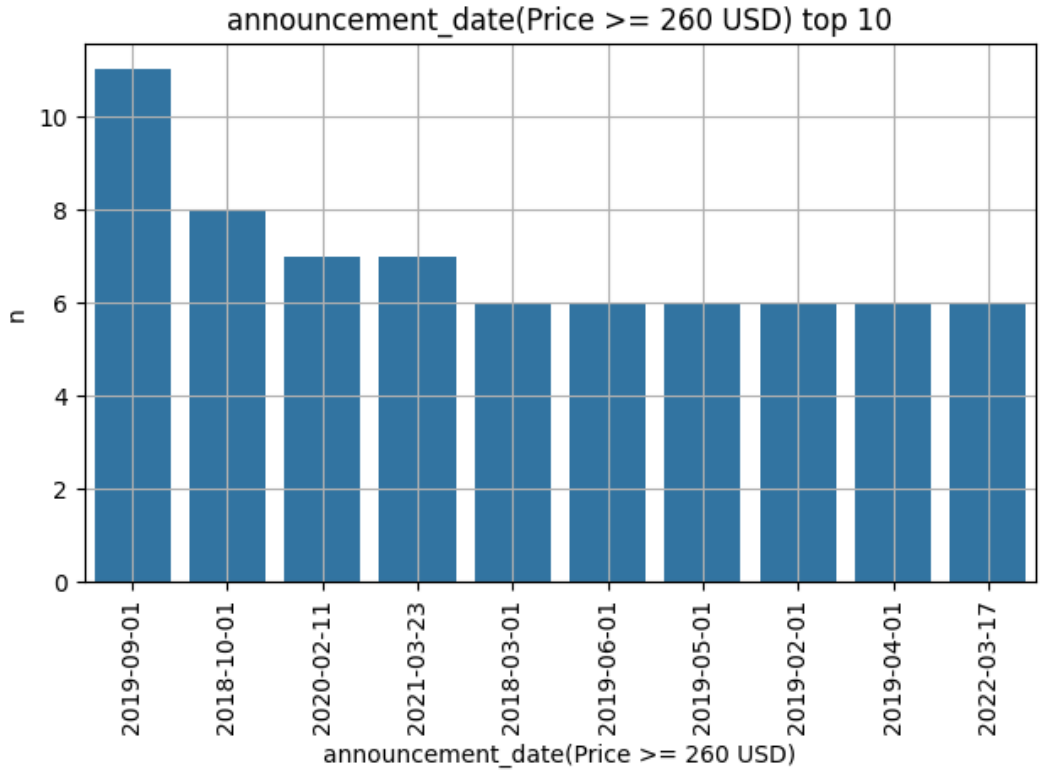

Рис. 6. Параметр

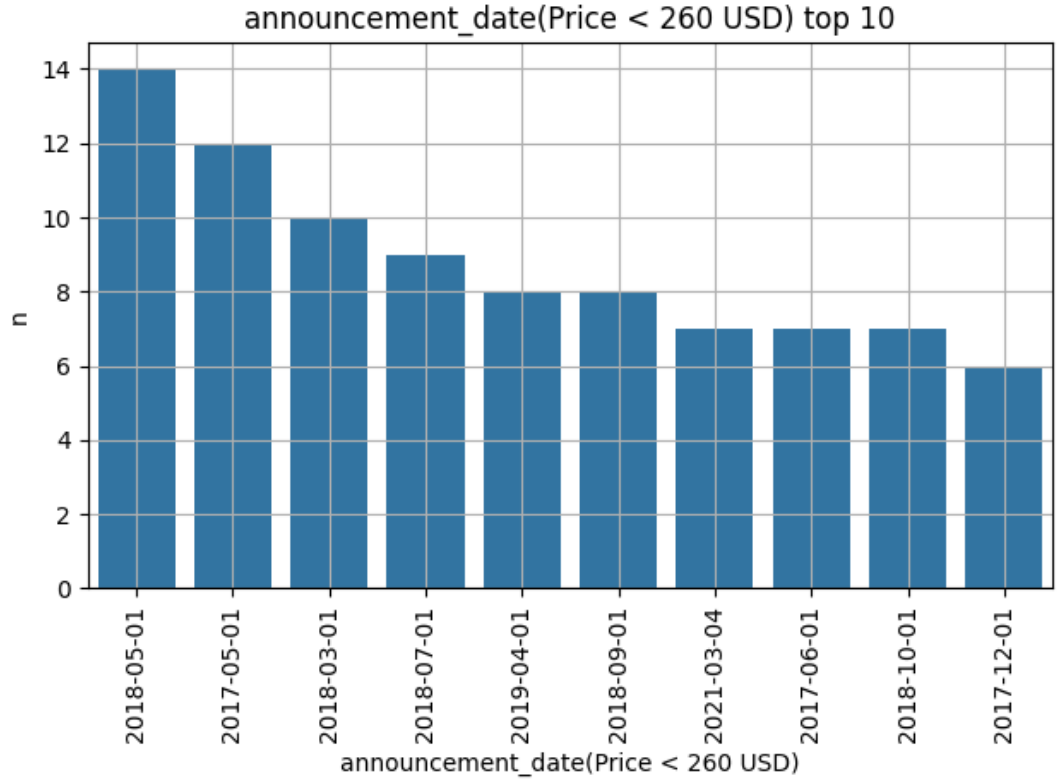

Рис. 7. Параметр

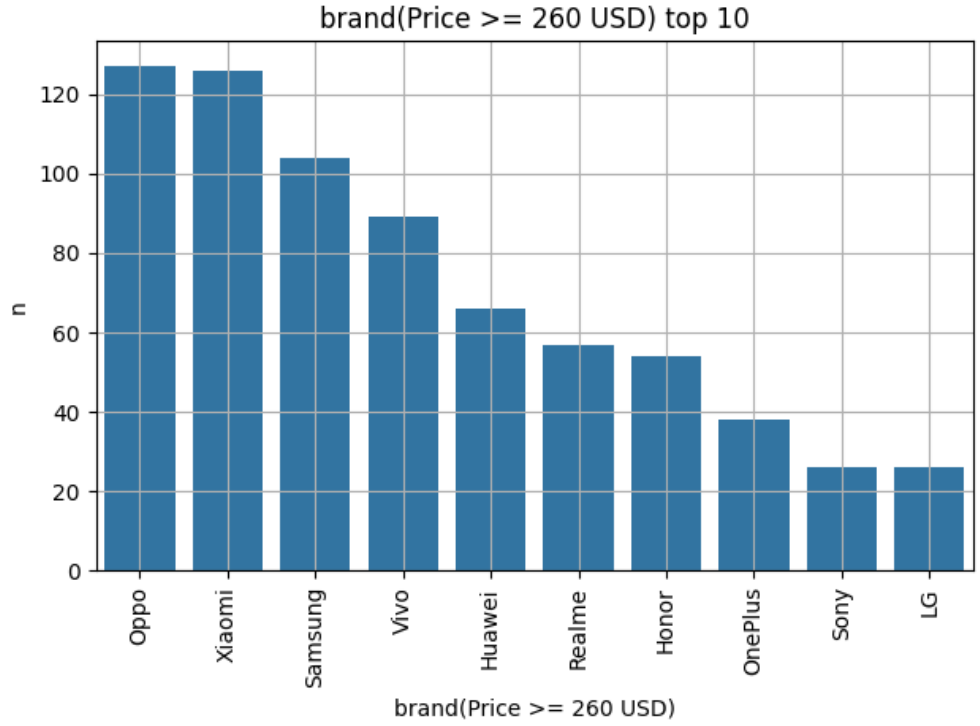

Рис. 8. Параметр

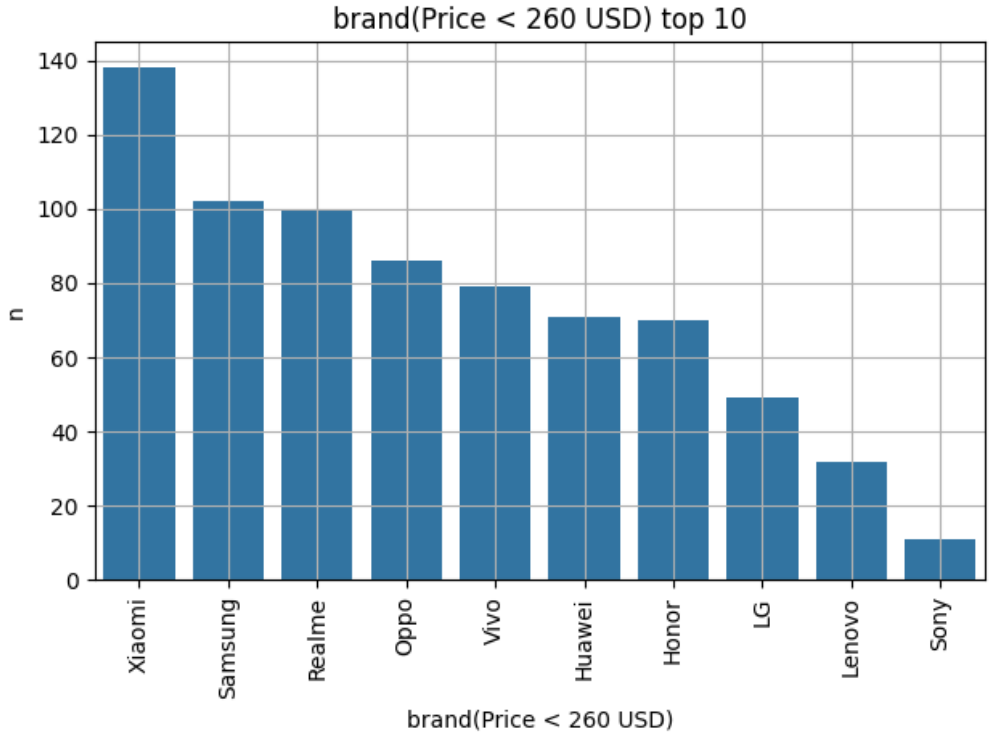

Рис. 9. Параметр

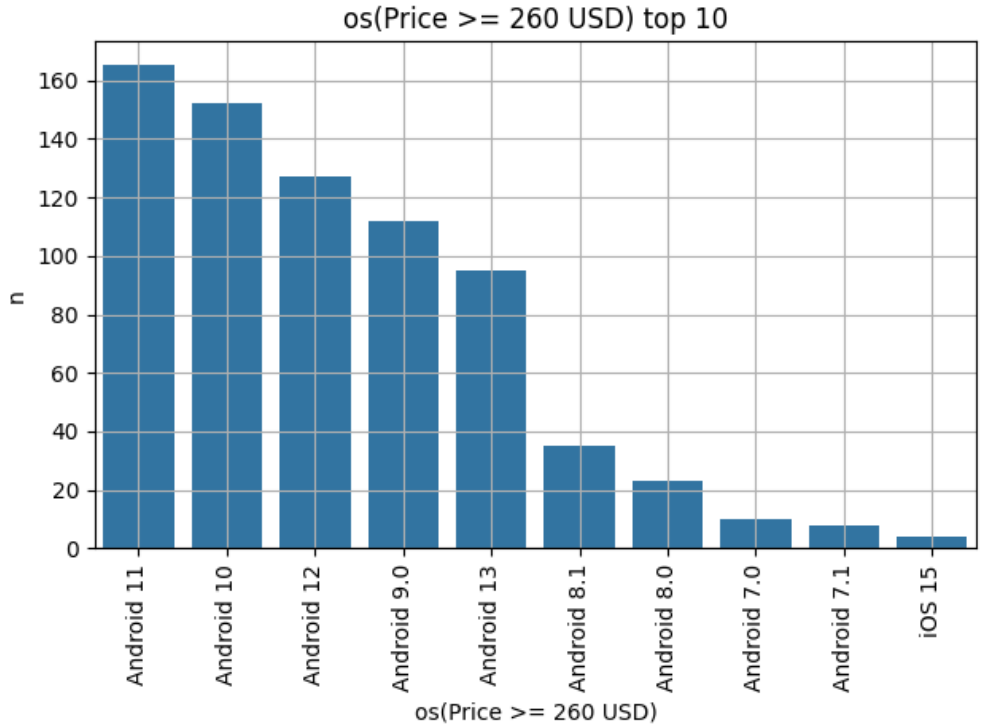

Рис. 10. Параметр

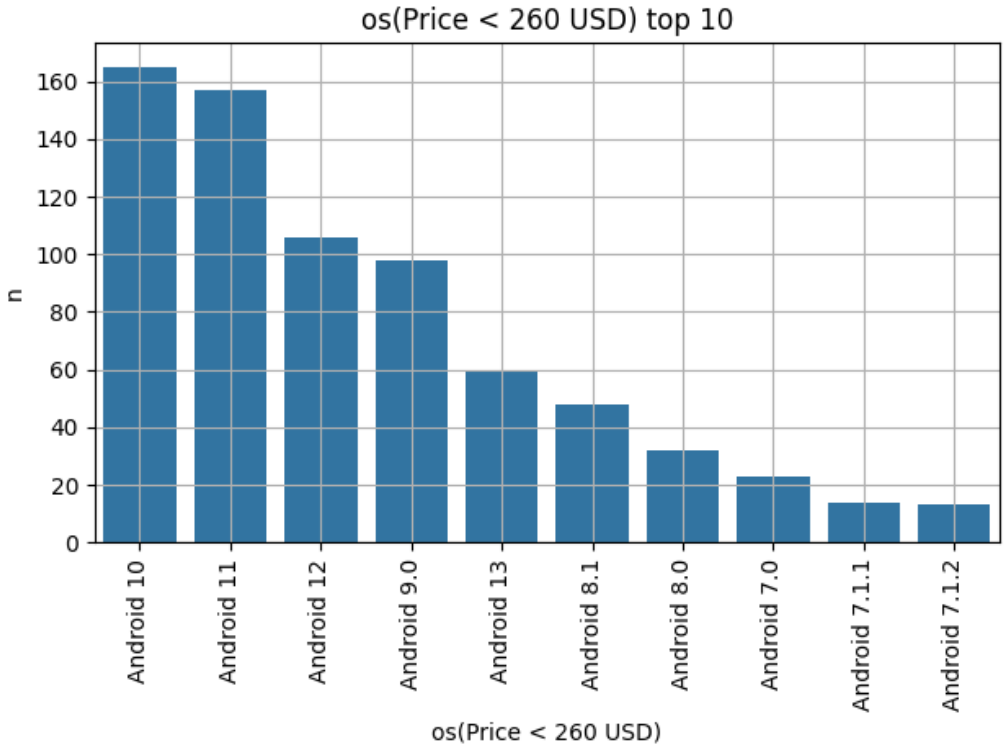

Рис. 11. Параметр

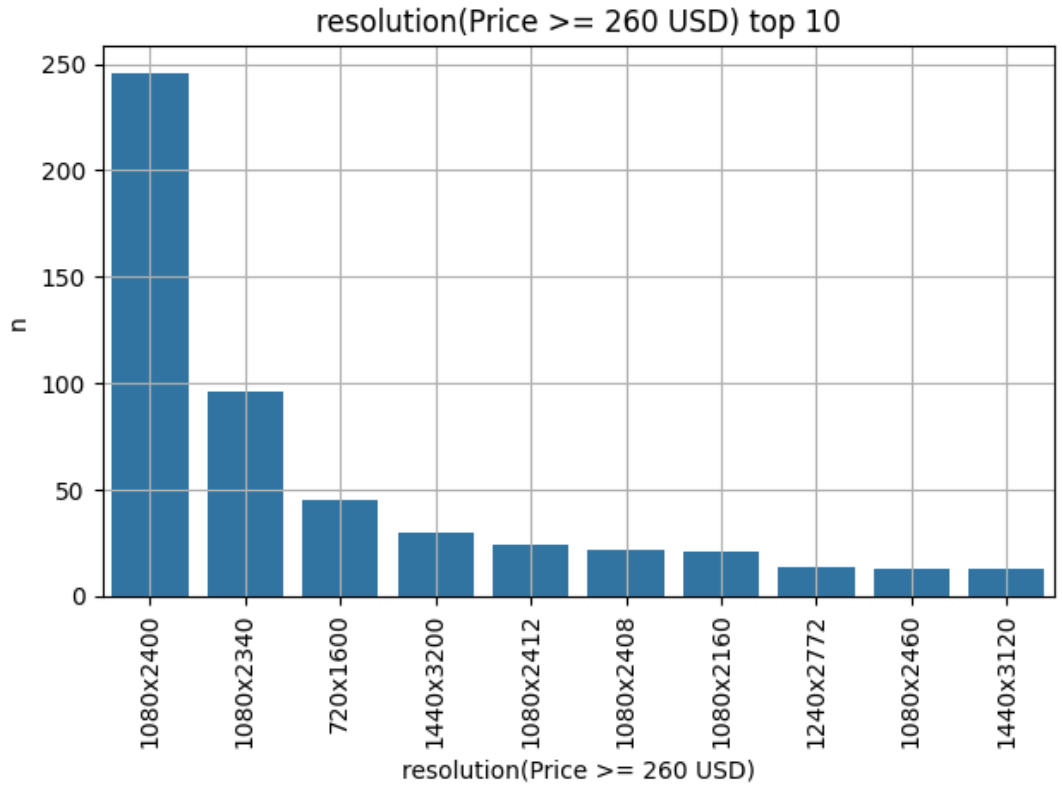

Рис. 12. Параметр

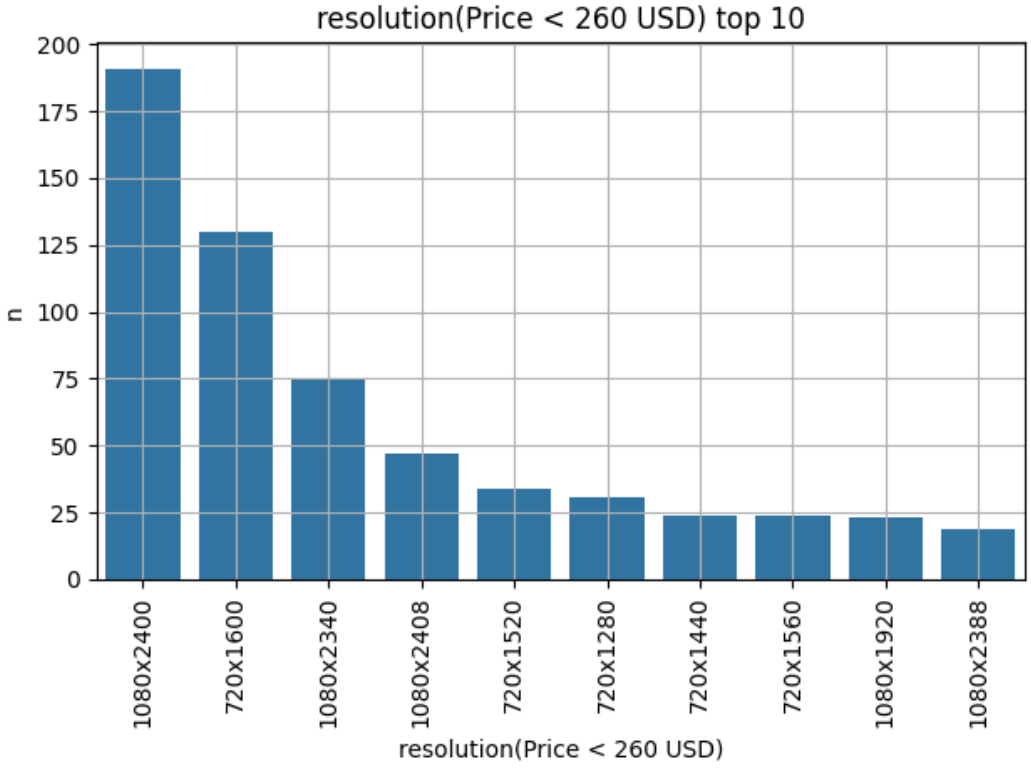

Рис. 13. Параметры

2. МЕДИАНА

2.1. Медиана параметра n

Задание: рассчитать медиану параметра n: для выборки в целом, для каждого класса отдельно.

Медиана была рассчитана для параметра inches.

```
inches median: 6.5
inches (class 'Price >= 260 USD') median: 6.55
inches (class 'Price < 260 USD') median: 6.5
```

Рис. 14. Медиана параметра inches.

2.2. Медиана параметра m

Задание: рассчитать медиану параметра m: для выборки в целом, для каждого класса отдельно.

Медиана была рассчитана для параметра ram(GB).

```
ram(GB) median: 8.0
ram(GB) (class 'Price >= 260 USD') median: 8.0
ram(GB) (class 'Price < 260 USD') median: 6.0
```

Рис. 15. Медиана параметра ram(GB).

3. ГРАФИКИ РАСПРЕДЕЛЕНИЯ ПРИ МЕДИАННОМ ЗНАЧЕНИИ

3.1. График распределения параметров

Задание: построить график распределения параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

Данные разделены относительно медианы параметра ram(GB).

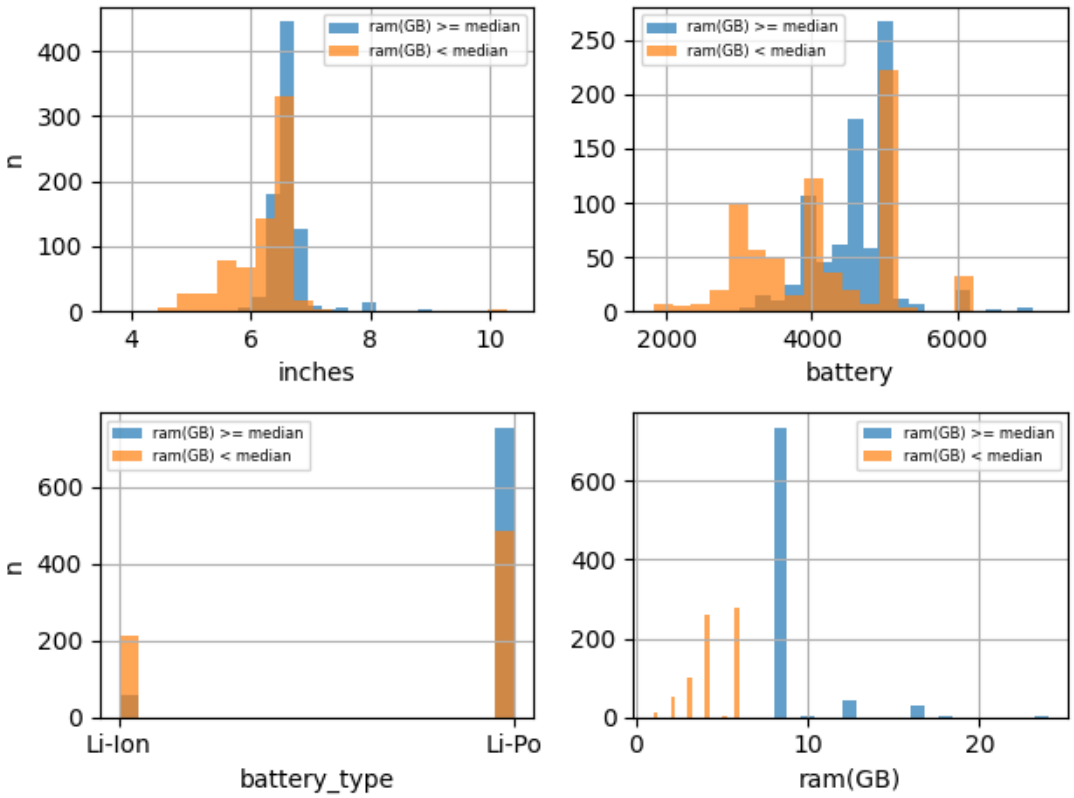

Рис. 16. Параметры

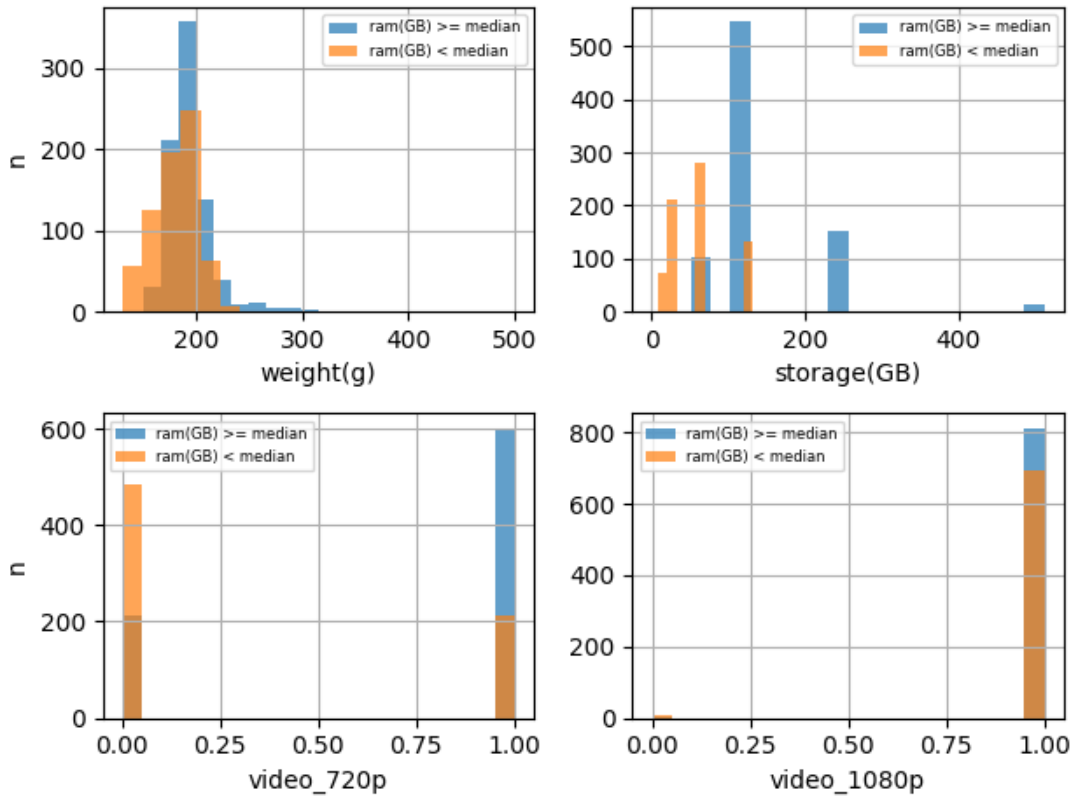

Рис. 17. Параметры

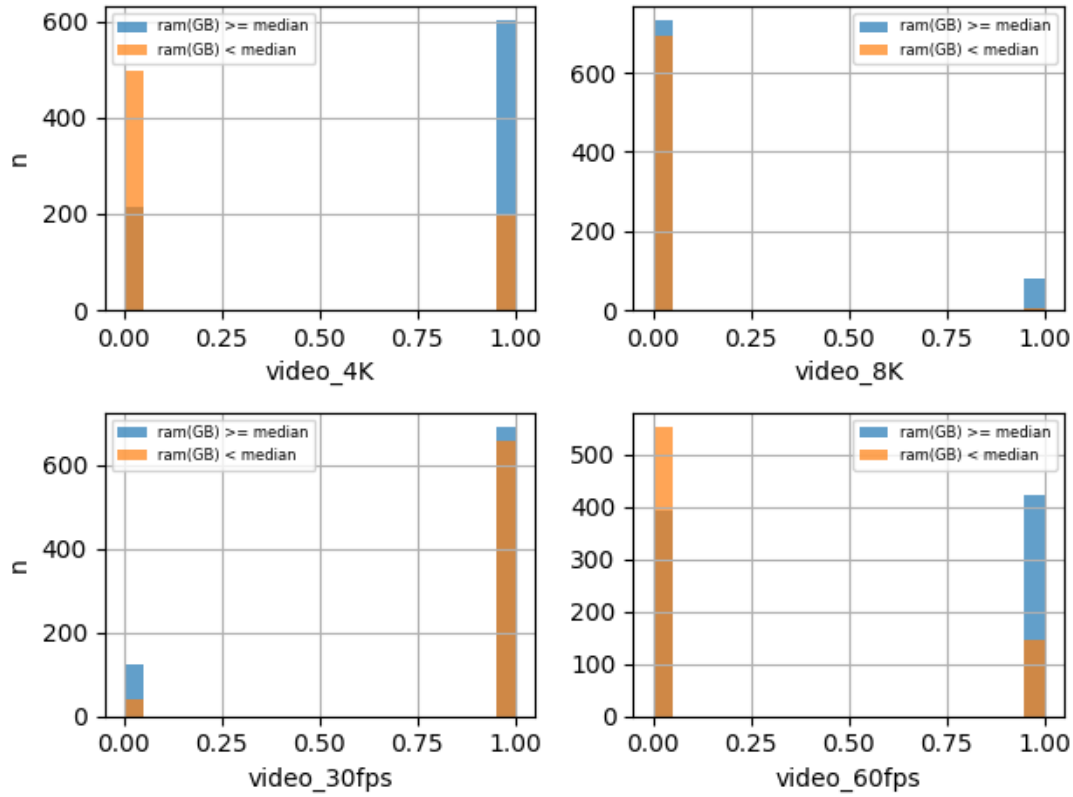

Рис. 18. Параметры

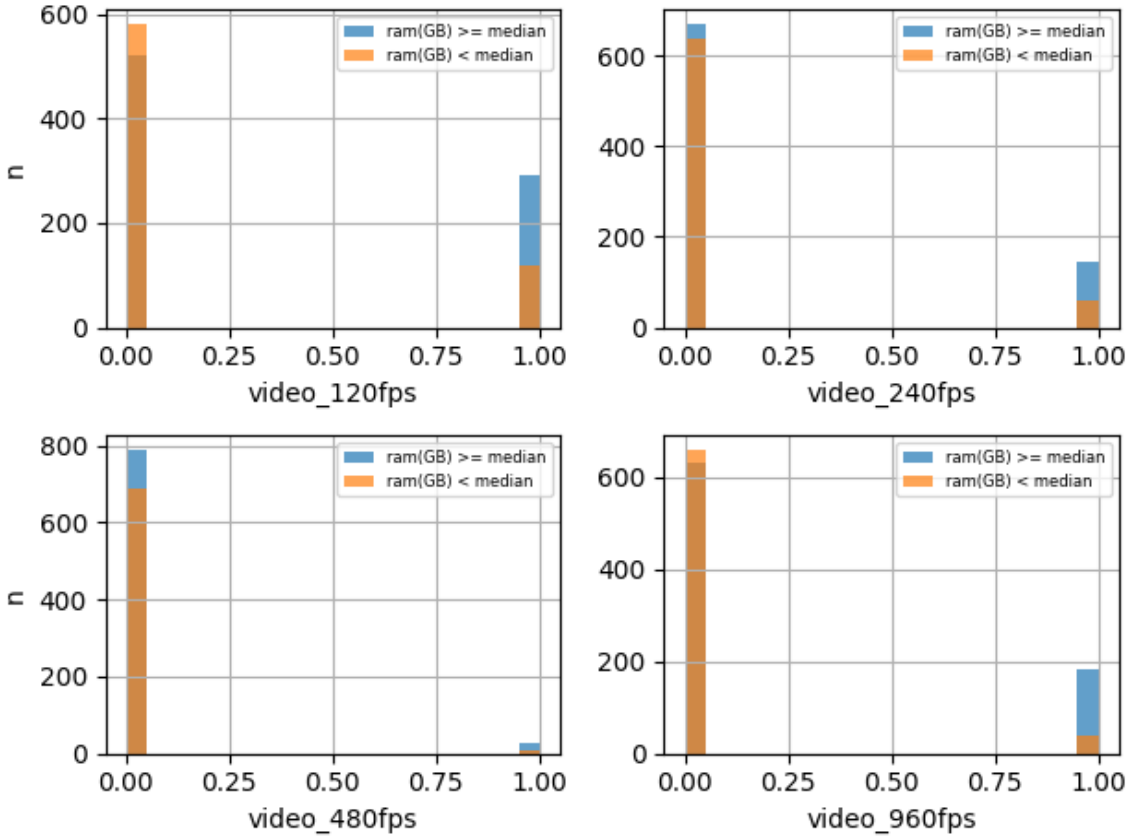

Рис. 19. Параметры

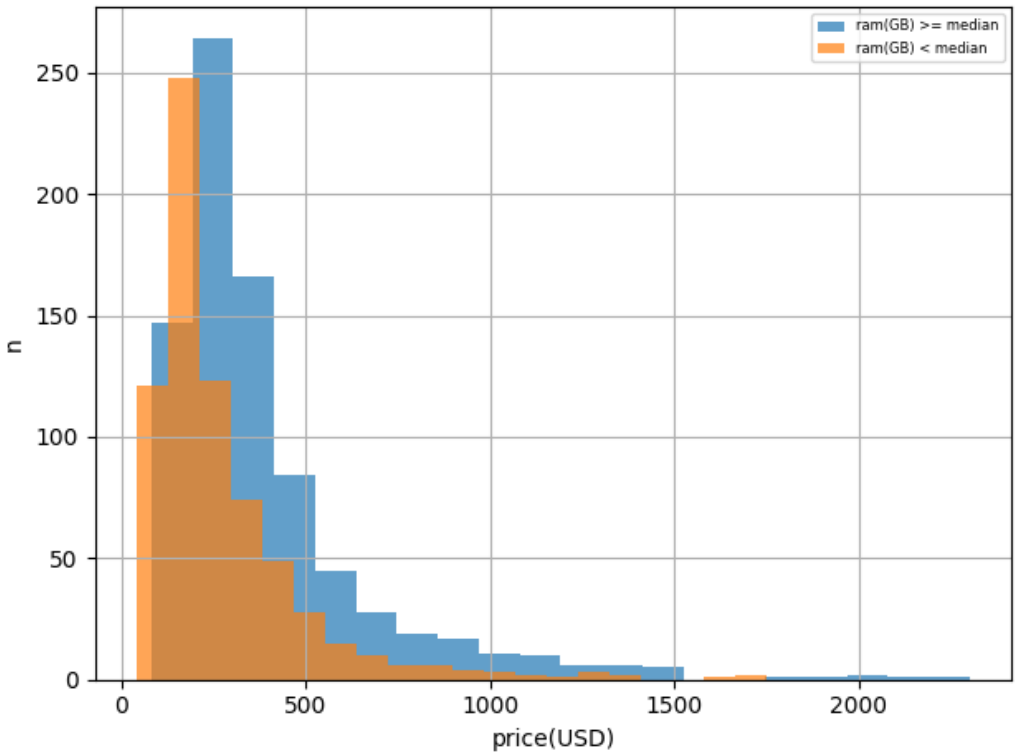

Рис. 20. Параметр price(USD)

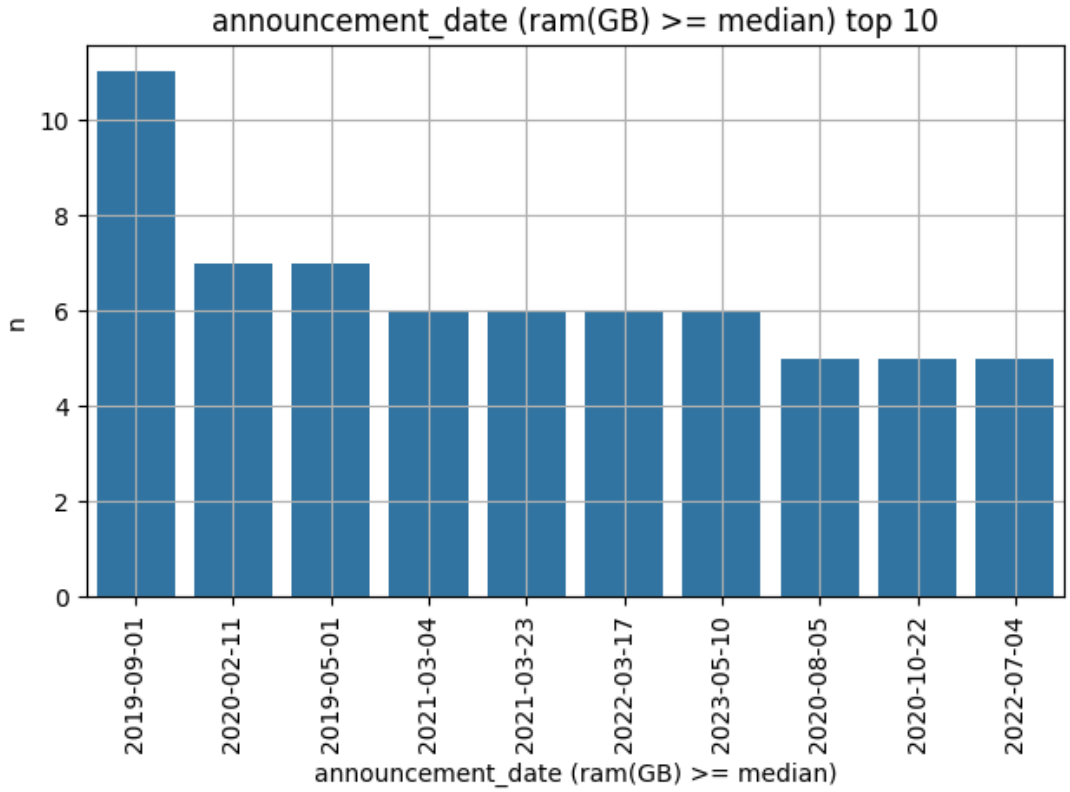

Рис. 21. Параметр

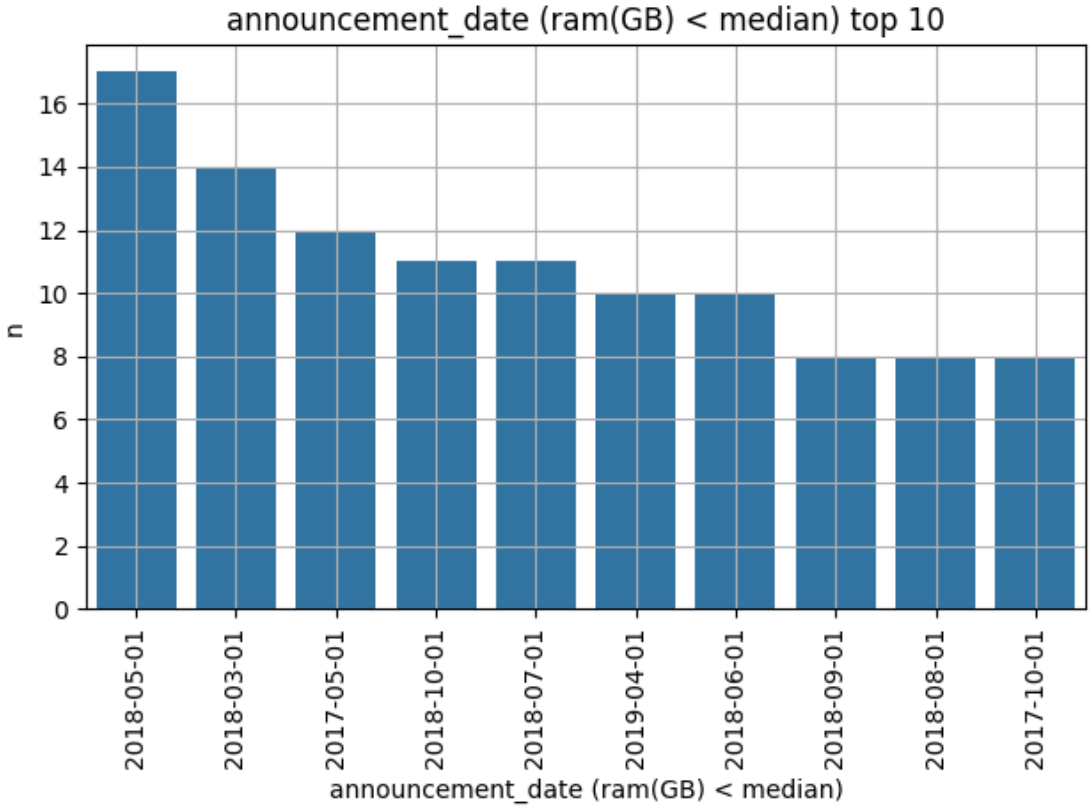

Рис. 22. Параметр

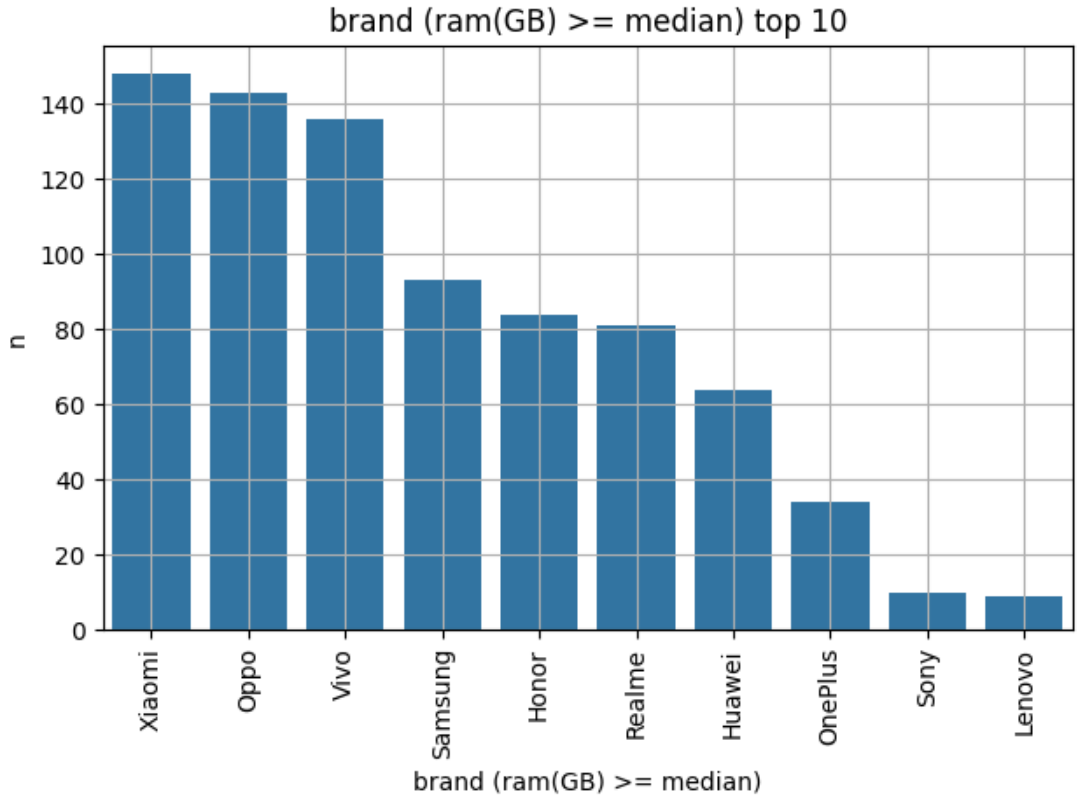

Рис. 23. Параметр

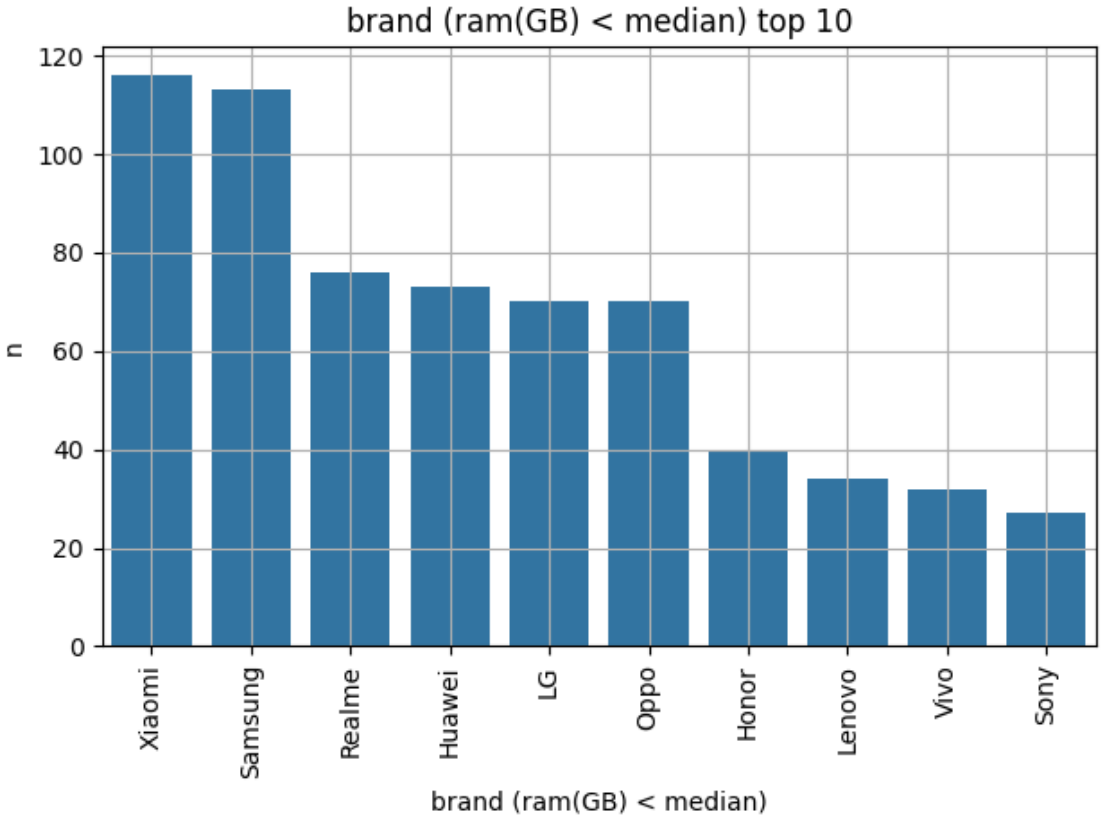

Рис. 24. Параметр

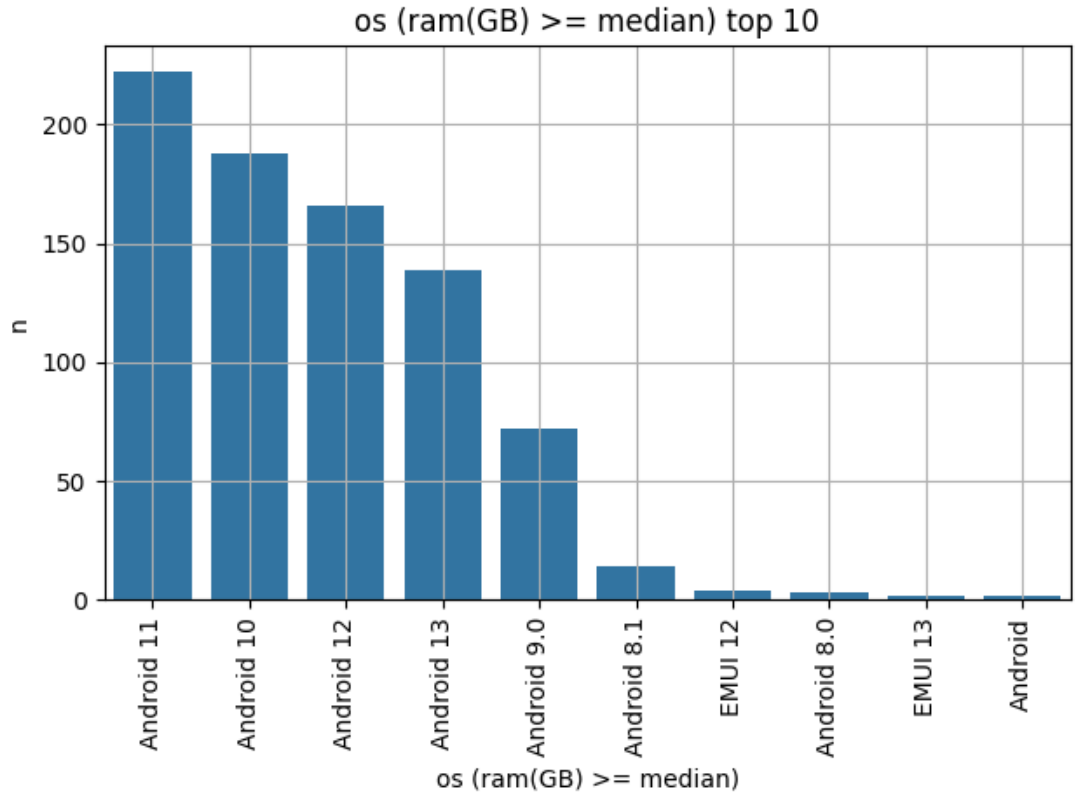

Рис. 25. Параметр

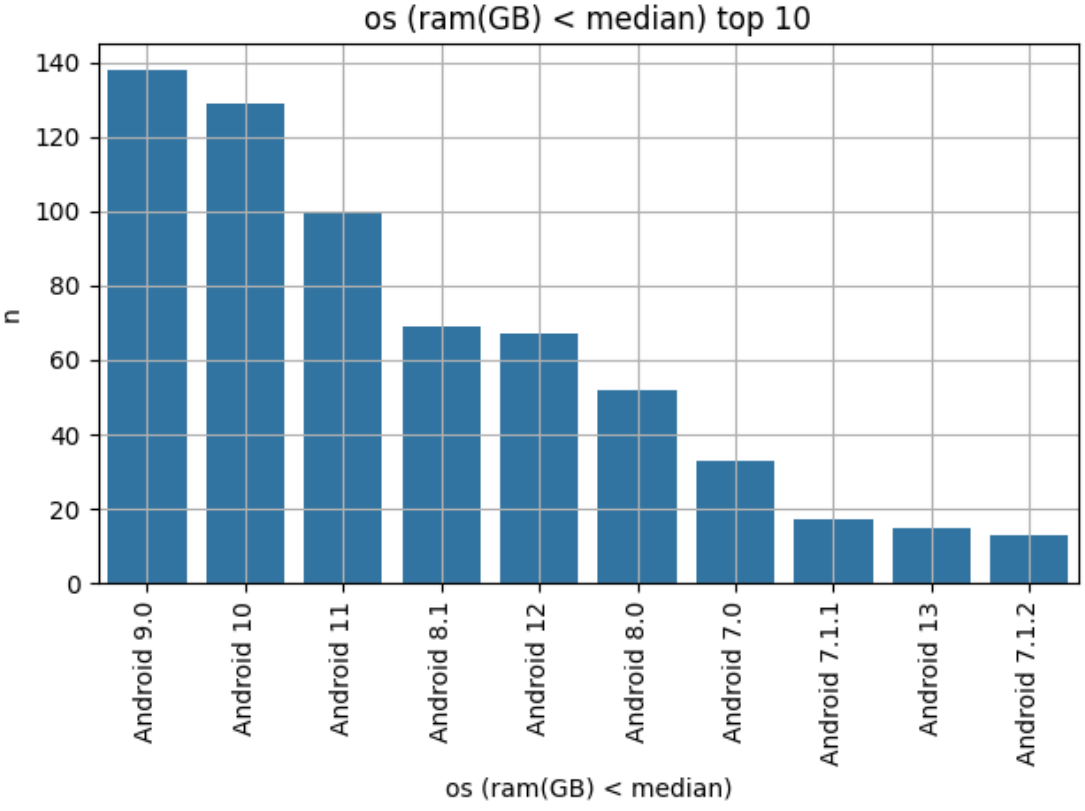

Рис. 26. Параметр

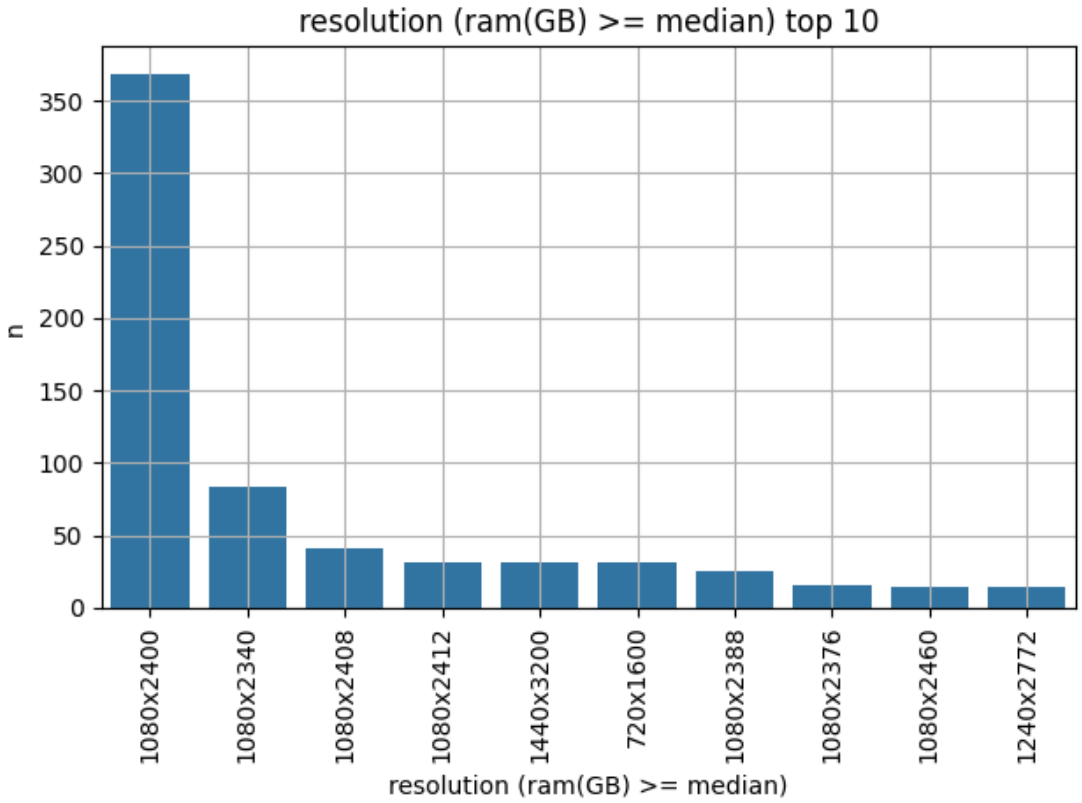

Рис. 27. Параметр

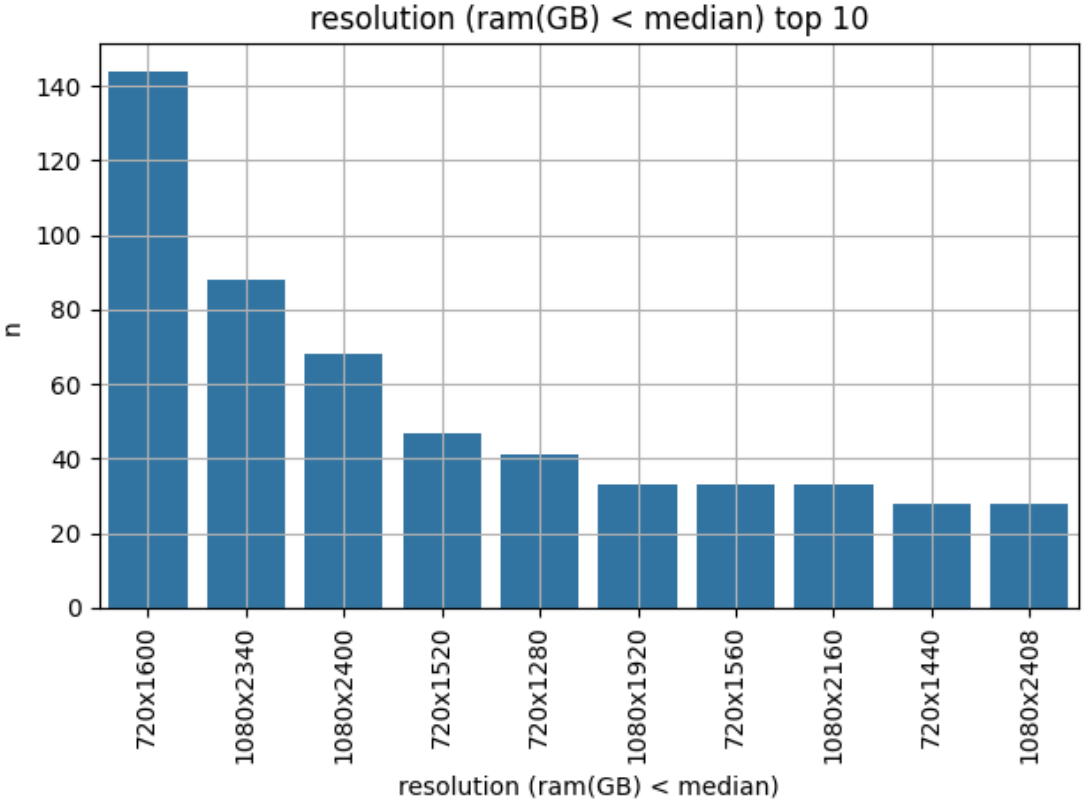

Рис. 28. Параметр

3.2. Гистограмма распределения, скатерограмма и боксплот параметров

Задание: построить гистограмму распределения, скатерограмму и боксплот параметров для объектов выше медианного значения в выборке и ниже. Подписать оси, добавить легенду и сетку на график.

Скатерограмма и боксплоты были построены по каждому столбцу относительно медианы столбца ram(GB).

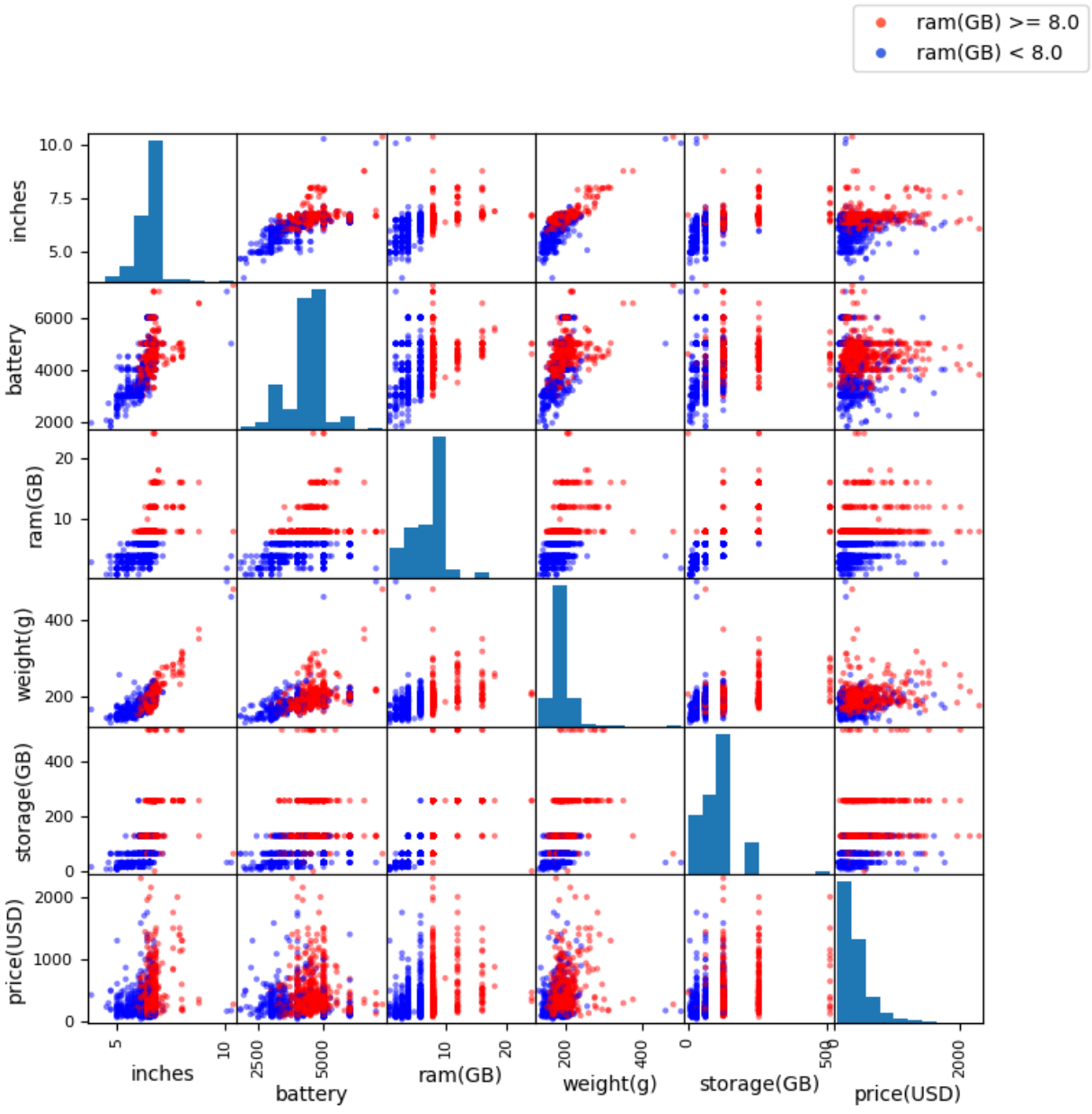

Рис. 29. Скатерограммы

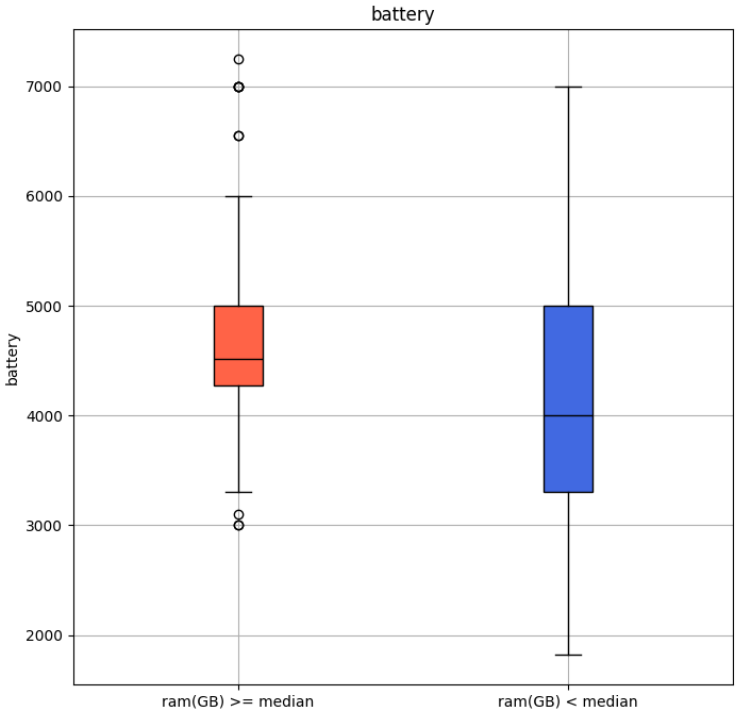

Рис. 30. Боксплот

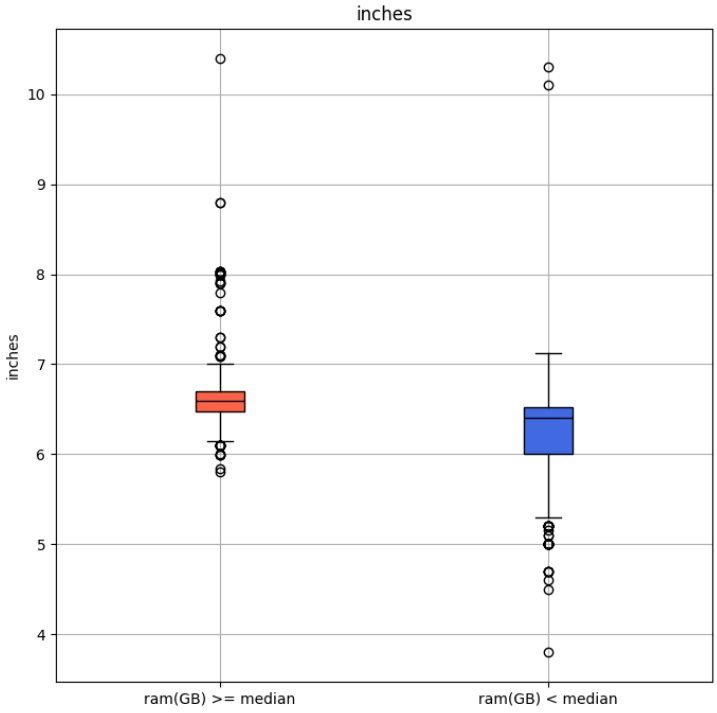

Рис. 31. Боксплот

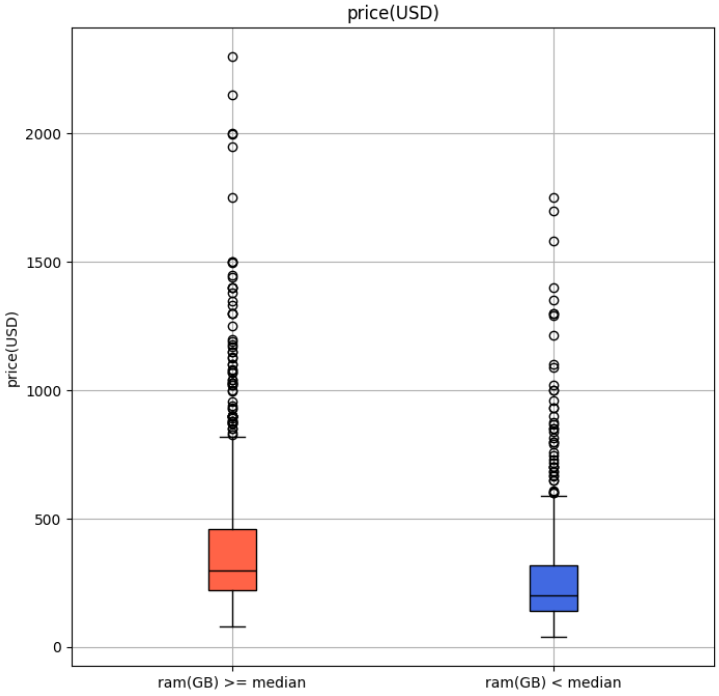

Рис. 32. Боксплот

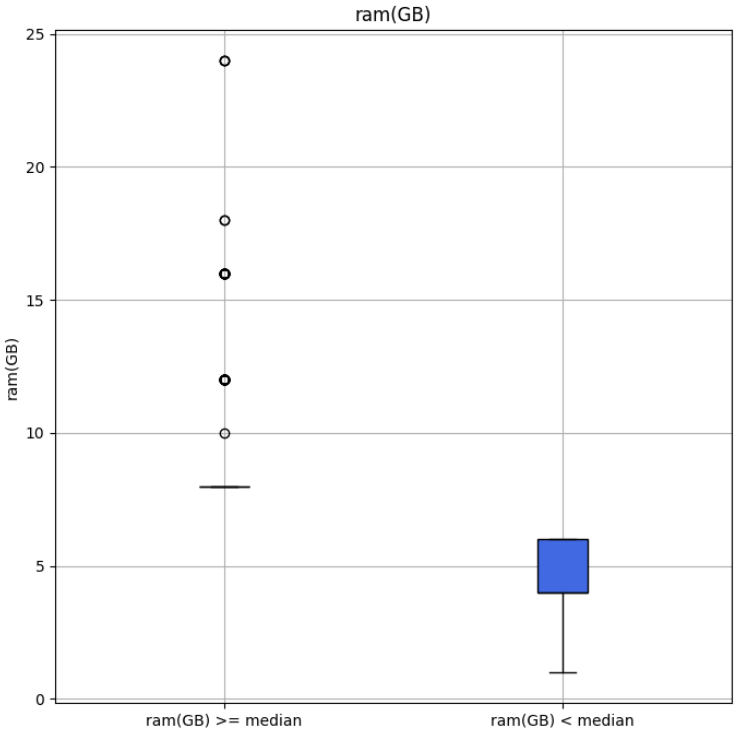

Рис. 33. Боксплот

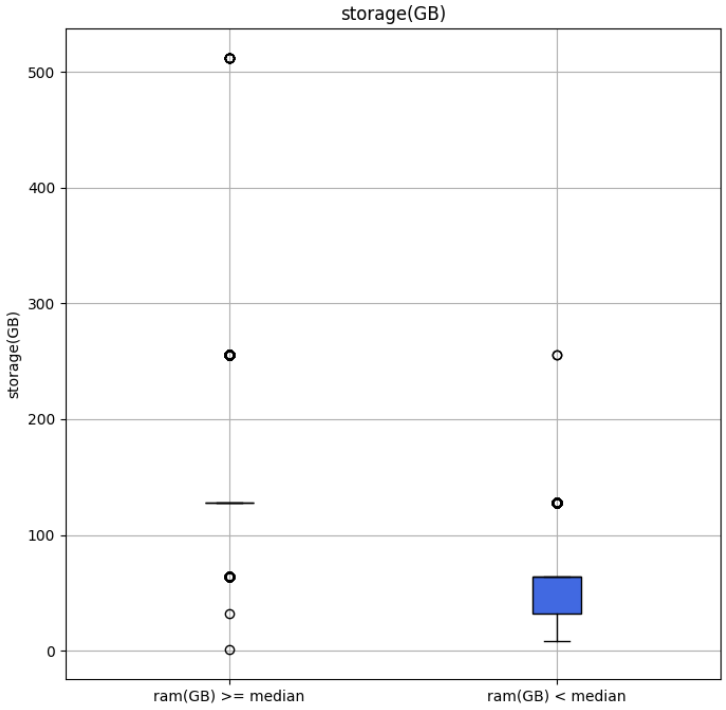

Рис. 34. Боксплот

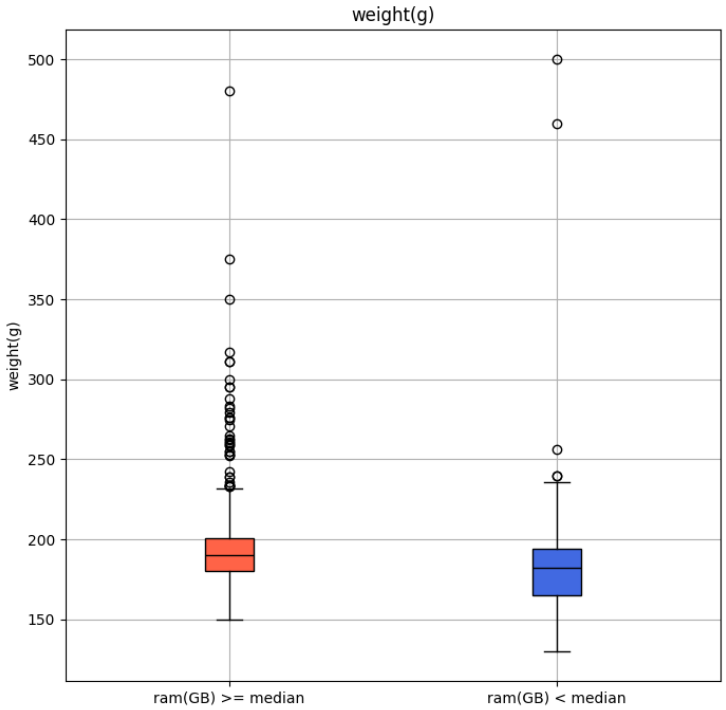

Рис. 35. Боксплот

4. СРЕДНЕЕ ЗНАЧЕНИЕ И СТАНДАРТНОЕ ОТКЛОНЕНИЕ

Задание: рассчитать среднее значение и стандартное отклонение для параметров: для всей выборки и для каждого класса отдельно.

Данные разделены относительно медианы параметра ram(GB).

	column name	mean	mean(ram(GB) >= median)	mean(ram(GB) < median)	std	std(ram(GB) >= median)	std(ram(GB) < median)
0	inches	6.422460	6.616118	6.196619	0.477043	0.313391	0.532828
1	battery	4389.798942	4609.916462	4133.100287	784.607022	550.069565	926.847598
2	ram(GB)	6.683862	8.587224	4.464183	2.701433	2.009604	1.394601
3	weight(g)	187.636243	193.886978	180.346705	26.200115	24.997358	25.698509
4	storage(GB)	109.164683	149.819410	61.753582	74.436484	74.056613	37.858257
5	video_720p	0.539683	0.739558	0.306590	0.498588	0.439146	0.461408
6	video_1080p	0.994048	0.997543	0.989971	0.076947	0.049538	0.099711
7	video_4K	0.529762	0.738329	0.286533	0.499279	0.439815	0.452466
8	video_8K	0.056878	0.100737	0.005731	0.231687	0.301165	0.075538
9	video_30fps	0.891534	0.848894	0.941261	0.311070	0.358372	0.235305
10	video_60fps	0.374339	0.518428	0.206304	0.484112	0.499968	0.404941
11	video_120fps	0.271164	0.358722	0.169054	0.444708	0.479920	0.375069
12	video_240fps	0.133598	0.176904	0.083095	0.340332	0.381822	0.276223
13	video_480fps	0.023148	0.033170	0.011461	0.150424	0.179189	0.106519
14	video_960fps	0.146825	0.223587	0.057307	0.354049	0.416905	0.232594
15	price(USD)	337.847036	393.626732	272.797361	266.740821	290.521363	218.915657

Рис. 36. Средние значения и стандартные отклонения

5. РАЗЛИЧИЯ В КЛАССАХ

5.1. Различия в классах по параметру n

Задание: статистически оценить различия в классах по параметру n- совпадают ли выборки по этому параметру или нет.

Данные разделены на классы по параметру price(USD).

Выбран параметр ram(GB).

Был проведен тест Шапиро-Уилка на принадлежность каждой выборки нормальному распределению. Выборки не распределены нормально.

Так как выборки не распределены нормально, был использован непараметрический тест Манна-Уитни, не требующий предположений о форме распределения данных.

Для выбранного параметра выборки не совпадают.

```
PARAMETER 'ram(GB)'
SHAPIRO-WILK TEST
Class 'Price >= 260 USD' p-value: 1.408e-31
Class 'Price >= 260 USD' is not normally distributed
Class 'Price < 260 USD' p-value: 5.017e-24
Class 'Price < 260 USD' is not normally distributed
--------------------------------------------------
MANN-WHITNEY U TEST
U-Test statistic: 3.824e+05
p-value: 3.736e-34
The differences are significant
--------------------------------------------------
```

Рис. 37. Для параметра ram(GB).

5.2. Различия в классах по параметру m

Задание: статистически оценить различия в классах по параметру m - совпадают ли выборки по этому параметру или нет.

Данные разделены на классы по параметру price(USD).

Выбран параметр inches.

Был проведен тест Шапиро-Уилка на принадлежность каждой выборки нормальному распределению. Выборки не распределены нормально.

Так как выборки не распределены нормально, был использован непараметрический тест Манна-Уитни, не требующий предположений о форме распределения данных.

Для выбранного параметра выборки не совпадают.

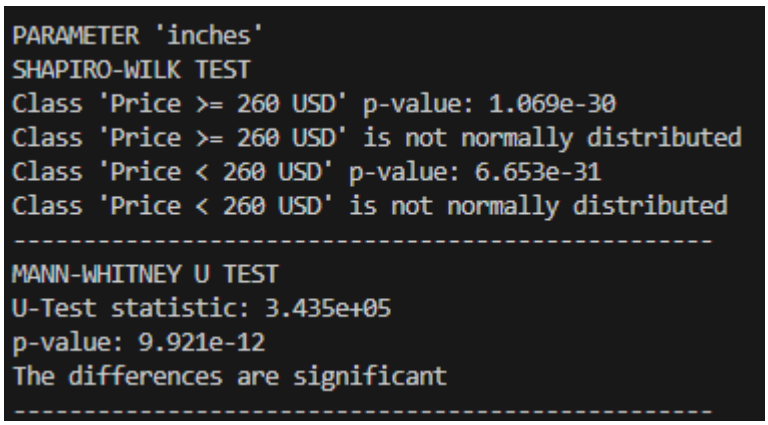

Рис. 38. Для параметра inches.

6. КЛАССИФИКАЦИЯ

Задание: классификация данных.

Был использован алгоритм LogisticRegression.

Данные разделены на классы по параметру price(USD), также данные разделены на обучающую и тестовую выборки в соотношении 7 к 3. Удалены параметры с данными, представленными текстом.

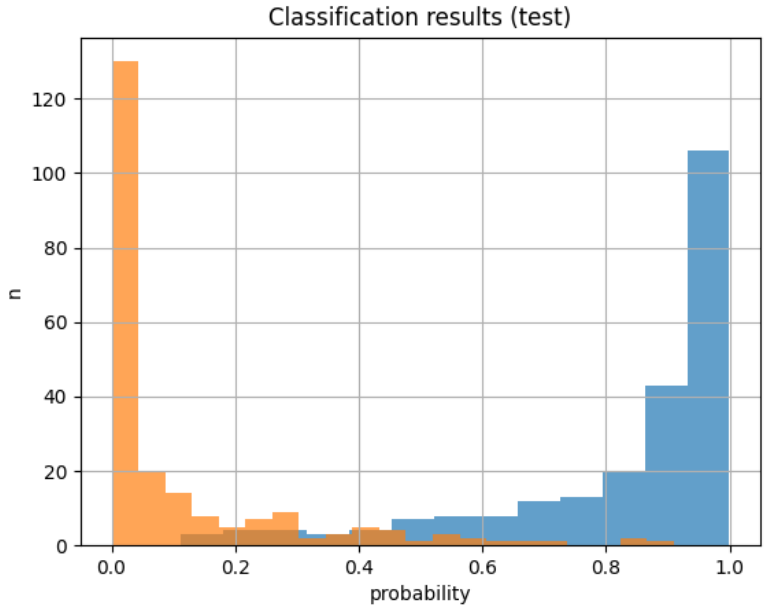

Рис. 39. Результаты классификации. Тестовая выборка

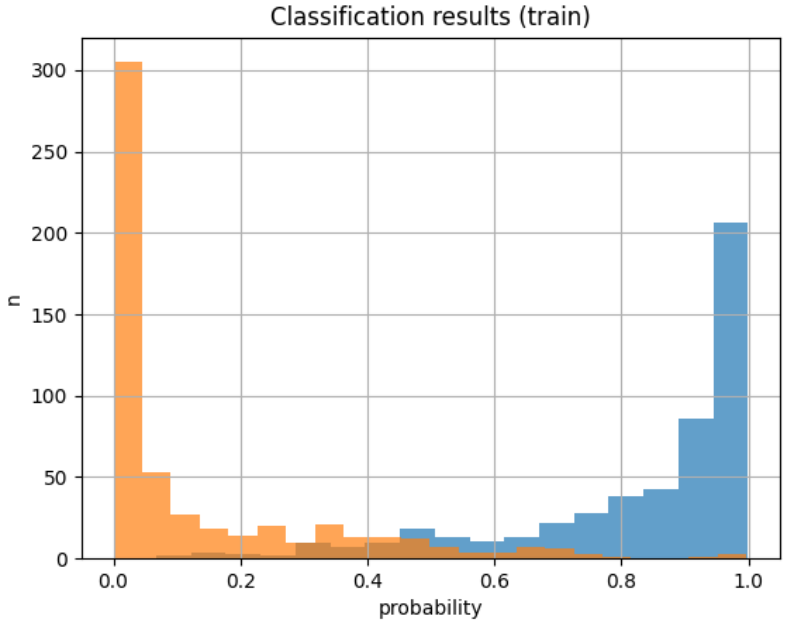

Рис. 40. Результаты классификации. Обучающая выборка

```
Accuracy (train): 0.915
Sensitivity (train): 0.928
Specificity (train): 0.904
Accuracy (test): 0.923
Sensitivity (test): 0.946
Specificity (test): 0.9
```

Рис. 41. Результаты классификации

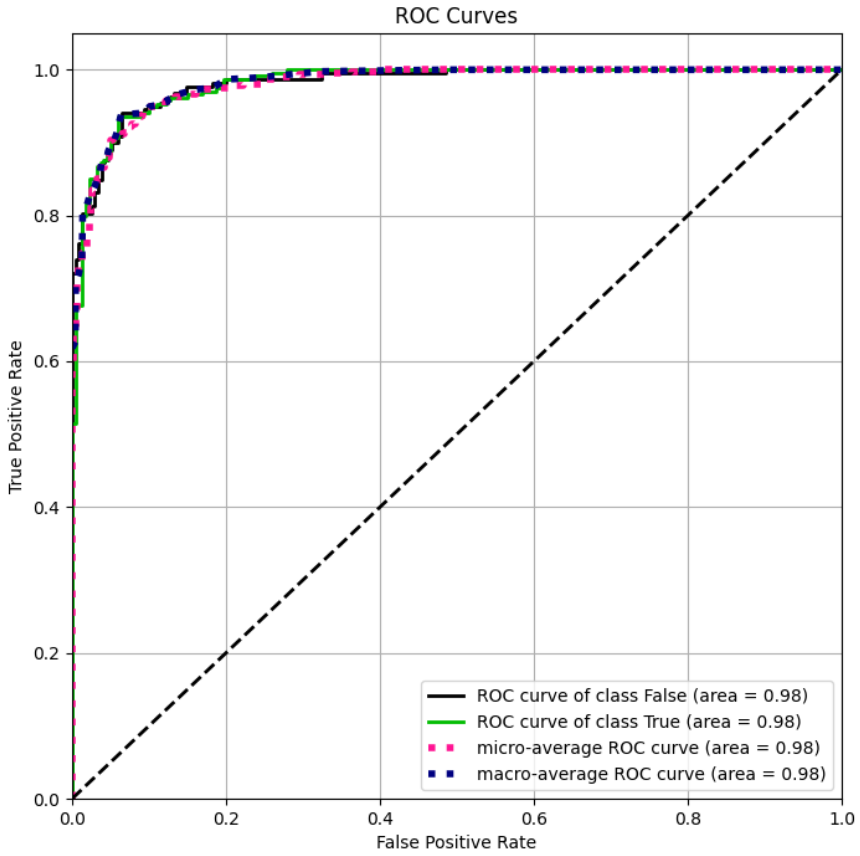

Рис. 42. ROC-кривая

ЛИСТИНГ

constants.py:
```
PRICE_THRESHOLD = 260
LABEL_POSITIVE = "Price >= " + str(PRICE_THRESHOLD) + " USD"
LABEL_NEGATIVE = "Price < " + str(PRICE_THRESHOLD) + " USD"
IMAGE_PATH = "images/"
```

read_data.py:
```
import pandas as pd

def read_data(bool_to_int: bool = True):
    df = pd.read_csv("cleaned_all_phones.csv")
    column_names = df.columns
    if bool_to_int:
        for i in range(len(column_names)):
            if df[column_names[i]].dtype == bool:
                df[column_names[i]] = df[column_names[i]].astype(int)

    col_investigated = df["price(USD)"]

    return df, column_names, col_investigated
```

1_plots.py:
```
import matplotlib.pyplot as plt

from plot_data import *
from read_data import read_data
from constants import *

if __name__ == "__main__":
    df, column_names, col_investigated = read_data()
    column_names_for_barplots = ["brand", "os", "resolution", "announcement_date"]
    column_names_not_for_plot = ["phone_name"]

    plt.hist(col_investigated, bins=20)
    plt.xlabel(col_investigated.name)
    plt.ylabel('n')
    plt.grid(True)
    plt.savefig(IMAGE_PATH + "hist_" + col_investigated.name)
    plt.close("all")

    print(df)
    print("Price median:", df["price(USD)"].median())

    positive = col_investigated >= PRICE_THRESHOLD

    df_for_hists = df.drop(columns=["price(USD)", *column_names_for_barplots,
*column_names_not_for_plot])
    build_hists(df_for_hists, positive)

    for i in range(len(column_names_for_barplots)):
```

```
        label = column_names_for_barplots[i] + '(' + LABEL_POSITIVE + ')'
        df_col_positive = df[positive][column_names_for_barplots[i]]
        build_bar_plot(df_col_positive.value_counts().head(10), label)

        label = column_names_for_barplots[i] + '(' + LABEL_NEGATIVE + ')'
        df_col_negative = df[positive == False][column_names_for_barplots[i]]
        build_bar_plot(df_col_negative.value_counts().head(10), label)
```

2+3_median.py:

```
from read_data import read_data
from constants import *

if __name__ == "__main__":
    df, column_names, col_investigated = read_data()
    positive = col_investigated >= PRICE_THRESHOLD
    # col_name = "inches"
    col_name = "ram(GB)"
    print(col_name + " median:", df[col_name].median())
    print(col_name + " (class '" + LABEL_POSITIVE + "') median:", df[positive][col_name].median())
    print(col_name + " (class '" + LABEL_NEGATIVE + "') median:", df[positive == False][col_name].median())
```

4_above_below_median_hist.py:

```
from read_data import read_data
from constants import *
from plot_data import *

if __name__ == "__main__":
    df, column_names, _ = read_data()
    col_name = "ram(GB)"
    col = df[col_name]
    median = col.median()

    column_names_for_barplots = ["brand", "os", "resolution", "announcement_date"]
    column_names_not_for_plot = ["phone_name"]

    label_c0 = col_name + " >= median"
    label_c1 = col_name + " < median"

    above = col >= median

    df_for_hists = df.drop(columns=["price(USD)", *column_names_for_barplots, *column_names_not_for_plot])
    build_hists(df_for_hists, above, label_c0, label_c1)

    for i in range(len(column_names_for_barplots)):
        label = column_names_for_barplots[i] + " (" + label_c0 + ")"
        df_col_above = df[above][column_names_for_barplots[i]]
        build_bar_plot(df_col_above.value_counts().head(10), label)

        label = column_names_for_barplots[i] + " (" + label_c1 + ")"
```

```
        df_col_below = df[above == False][column_names_for_barplots[i]]
        build_bar_plot(df_col_below.value_counts().head(10), label)

    plt.hist(df["price(USD)"][above], bins=20, alpha=0.7, label=label_c0)
    plt.hist(df["price(USD)"][above == False], bins=20, alpha=0.7, label=label_c1)
    plt.xlabel("price(USD)")
    plt.ylabel("n")
    plt.legend(prop={"size": 6})
    plt.grid(True)
    plt.tight_layout()
    path = IMAGE_PATH + "hist_price(USD)_" + label_c0 + "_" + label_c1
    path = path.replace(">=", "bigger").replace("<", "smaller")
    plt.savefig(path)
    plt.close()
```

5_above_below_median_box.py:

```
from matplotlib.lines import Line2D

from read_data import read_data
from constants import *
from plot_data import *

if __name__ == "__main__":
    df, column_names, _ = read_data(False)
    col_name = "ram(GB)"
    col = df[col_name]
    median = col.median()

    label_above = col_name + " >= median"
    label_below = col_name + " < median"

    labels = [label_above, label_below]

    above = col >= median

    colors = ['tomato', 'royalblue']
    legend_elements = \
        [Line2D([0], [0], marker='o', color='w', markerfacecolor='tomato', markersize=6,
label=f'{col_name} >= {median}'),
        Line2D([0], [0], marker='o', color='w', markerfacecolor='royalblue', markersize=6,
label=f'{col_name} < {median}')]

    for name in df.columns:
        if df[name].dtype not in ['object', 'bool']:
            data = [df[above][name], df[above == False][name]]

            fig, ax = plt.subplots(figsize=(8, 8))
            ax.set_ylabel(name)
            bp = ax.boxplot(data, patch_artist=True, tick_labels=labels)

            for patch, color in zip(bp['boxes'], colors):
                patch.set_facecolor(color)
```

```
        for median in bp['medians']:
            median.set_color('black')

        plt.title(name)
        plt.grid()
        plt.savefig(IMAGE_PATH + f'boxplot_{name}.png')
        # plt.show()
```

5_above_below_median_scatter.py:

```
from matplotlib.lines import Line2D

from read_data import read_data
from constants import *
from plot_data import *

if __name__ == "__main__":
    df, column_names, _ = read_data(False)

    col_name = "ram(GB)"
    col = df[col_name]
    median = col.median()

    column_names_for_scatter = [df.columns[df.dtypes != 'bool']]

    legend_elements = \
        [Line2D([0], [0], marker='o', color='w', markerfacecolor='tomato', markersize=6,
label=f'{col_name} >= {median}'),
        Line2D([0], [0], marker='o', color='w', markerfacecolor='royalblue', markersize=6,
label=f'{col_name} < {median}')]

    colors = df[col_name].apply(lambda x: 'red' if x >= median else 'blue')

    for name in column_names_for_scatter:
        pd.plotting.scatter_matrix(df[name], color=colors, figsize=(8, 8))
        plt.figlegend(handles=legend_elements, loc='upper right', fontsize=10)
        plt.savefig(IMAGE_PATH + 'scatter_matrix.png')
        plt.show()
```

6_above_below_median_mean_std.py:

```
from read_data import read_data
from constants import *
from plot_data import *

if __name__ == "__main__":
    df, column_names, _ = read_data(False)

    col_name = "ram(GB)"
    col = df[col_name]
    median = col.median()
```

```
    label_c0 = col_name + " >= median"
    label_c1 = col_name + " < median"

    above = col >= median

    table_data = []
    for name in column_names:
        if df[name].dtype not in ['object']:
            mean = df[name].mean()
            mean_above = df[above][name].mean()
            mean_below = df[above == False][name].mean()
            std = df[name].std()
            std_above = df[above][name].std()
            std_below = df[above == False][name].std()
            table_data.append([name, mean, mean_above, mean_below, std, std_above, std_below])

    label_c0 = '(' + label_c0 + ')'
    label_c1 = '(' + label_c1 + ')'
    header = ['column name', 'mean', 'mean' + label_c0, 'mean' + label_c1, 'std', 'std' + label_c0, 'std' +
label_c1]
    df_table = pd.DataFrame(table_data, columns=header)
    print(df_table)
```

7_classes_differences.py:

```
import pandas as pd
from scipy import stats
from scipy.stats import mannwhitneyu

from read_data import read_data
from constants import *
from plot_data import *

if __name__ == "__main__":
    df, column_names, col_investigated = read_data()

    # col_name = "ram(GB)"
    col_name = "inches"

    label_c0 = LABEL_POSITIVE
    label_c1 = LABEL_NEGATIVE

    positive = col_investigated >= PRICE_THRESHOLD

    col_c0 = df[positive][col_name]
    col_c1 = df[positive == False][col_name]

    print("PARAMETER '" + col_name + "'")

    stat, p = stats.shapiro(col_c0)
    print("SHAPIRO-WILK TEST")
    print("Class '" + label_c0 + "' p-value: " + "{:.3e}".format(p))
    print("Class '" + label_c0 + "' is " + ("" if p >= 0.05 else "not") + " normally distributed")
```

```
    stat, p = stats.shapiro(col_c1)
    print("Class '" + label_c1 + "' p-value: " + "{:.3e}".format(p))
    print("Class '" + label_c1 + "' is " + ("" if p >= 0.05 else "not") + " normally distributed")

    print("-" * 50)
    print("MANN-WHITNEY U TEST")
    u_stat, p_value = mannwhitneyu(col_c0, col_c1)
    print("U-Test statistic: {:.3e}".format(u_stat))
    print("p-value: {:.3e}".format(p_value))

    if p_value < 0.05:
        print('The differences are significant')
    else:
        print('The differences are not significant')
    print("-" * 50)
```

8_classification.py:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
import scikitplot as skplt

from read_data import read_data
from constants import *
from plot_data import *

def prepare_data(c0: pd.DataFrame, c1: pd.DataFrame):
    Y0 = np.zeros((c0.shape[0], 1), dtype=bool)
    Y1 = np.ones((c1.shape[0], 1), dtype=bool)

    X = np.vstack((c0, c1))
    Y = np.vstack((Y0, Y1)).ravel()

    # перемешиваем данные
    rng = np.random.default_rng()
    arr = np.arange(X.shape[0])
    rng.shuffle(arr)
    X = X[arr]
    Y = Y[arr]

    return X, Y

if __name__ == "__main__":
    df, _, col_investigated = read_data(False)

    column_names_dropped = []

    for name in df.columns:
        if df[name].dtype == 'object':
            column_names_dropped.append(name)
```

```
    df = df.drop(columns=column_names_dropped)

    # print(df)
    print(df.columns)

    label_c0 = LABEL_POSITIVE
    label_c1 = LABEL_NEGATIVE

    positive = col_investigated >= PRICE_THRESHOLD

    c0 = df[positive]
    c1 = df[positive == False]

    X, Y = prepare_data(c0, c1)

    N = X.shape[0]

    # разделяем данные на 2 подвыборки
    trainCount = round(0.7*N)
    Xtrain = X[0:trainCount]
    Xtest = X[trainCount:N+1]
    Ytrain = Y[0:trainCount]
    Ytest = Y[trainCount:N+1]

    # col_name_0 = 'inches'
    # col_name_1 = 'battery'
    # plt.scatter(c0[col_name_0], c0[col_name_1], marker=".", alpha=0.7)
    # plt.scatter(c1[col_name_0], c1[col_name_1], marker=".", alpha=0.7)
    # plt.title('Scatter')
    # plt.xlabel(col_name_0)
    # plt.ylabel(col_name_1)
    # plt.show()

    clf = LogisticRegression(random_state=13, solver='saga').fit(Xtrain, Ytrain)

    Pred_test = clf.predict(Xtest)                    # Predict class labels for samples in X.
    Pred_test_proba = clf.predict_proba(Xtest)            # Probability estimates

    Pred_train = clf.predict(Xtrain)
    Pred_train_proba = clf.predict_proba(Xtrain)

    acc_train = clf.score(Xtrain, Ytrain)
    acc_test = clf.score(Xtest, Ytest)

    plt.hist(Pred_train_proba[Ytrain, 1], bins='auto', alpha=0.7)
    plt.hist(Pred_train_proba[~Ytrain, 1], bins='auto', alpha=0.7) # т.к массив свероятностями имеет
два столбца, мы берем один - первый
    plt.title("Classification results (train)")
    plt.xlabel("probability")
    plt.ylabel('n')
    plt.grid(True)
```

```
    plt.savefig(IMAGE_PATH + 'classification_train' + '.png')
    plt.show()

    plt.hist(Pred_test_proba[Ytest,1], bins='auto', alpha=0.7)
    plt.hist(Pred_test_proba[~Ytest,1], bins='auto', alpha=0.7) # т.к массив свероятностями имеет
два столбца, мы берем один - первый
    plt.title("Classification results (test)")
    plt.xlabel("probability")
    plt.ylabel('n')
    plt.grid(True)
    plt.savefig(IMAGE_PATH + 'classification_test' + '.png')
    plt.show()

    sensitivity_test = 0
    specificity_test = 0
    for i in range(len(Pred_test)):
        if Pred_test[i]==True and Ytest[i]==True:
            sensitivity_test += 1

        if Pred_test[i]==False and Ytest[i]==False:
            specificity_test += 1

    sensitivity_test /= len(Pred_test[Pred_test==True])
    specificity_test /= len(Pred_test[Pred_test==False])

    sensitivity_train = 0
    specificity_train = 0
    for i in range(len(Pred_train)):
        if Pred_train[i]==True and Ytrain[i]==True:
            sensitivity_train += 1

        if Pred_train[i]==False and Ytrain[i]==False:
            specificity_train += 1

    sensitivity_train /= len(Pred_train[Pred_train==True])
    specificity_train /= len(Pred_train[Pred_train==False])

    print('Accuracy (train):', round(acc_train, 3))
    print('Sensitivity (train):', round(sensitivity_train, 3))
    print('Specificity (train):', round(specificity_train, 3))

    print('Accuracy (test):', round(acc_test, 3))
    print('Sensitivity (test):', round(sensitivity_test, 3))
    print('Specificity (test):', round(specificity_test, 3))

    skplt.metrics.plot_roc_curve(Ytest, Pred_test_proba, figsize = (8, 8))
    plt.grid(True)
    plt.savefig(IMAGE_PATH + 'RFC_roc_curve_test' + '.png')
    plt.show()

plot_data.py:
import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns

from constants import *

def add_hist(axs, data: pd.DataFrame, positive, xlabel, ylabel, label_c0=LABEL_POSITIVE, label_c1=LABEL_NEGATIVE):
    axs.hist(data[positive], bins=20, alpha=0.7, label=label_c0)
    axs.hist(data[positive == False], bins=20, alpha=0.7, label=label_c1)
    axs.set_xlabel(xlabel)
    axs.set_ylabel(ylabel)
    axs.legend(prop={"size": 6})
    axs.grid(True)

def build_hists_2x2(data: pd.DataFrame, positive, fig_n: int, label_c0=LABEL_POSITIVE, label_c1=LABEL_NEGATIVE):
    if data.shape[1] != 4:
        raise AssertionError("len(data) != 4")

    column_names = data.columns
    fig, axs = plt.subplots(2, 2)
    for i in range(4):
        x = i // 2
        y = i % 2
        col = data[column_names[i]]
        xlabel = column_names
```

ЗАКЛЮЧЕНИЕ

Гистограммы и графики распределения показали, что параметры, такие как цена, объем оперативной памяти и диагональ экрана, имеют различное распределение в зависимости от класса (выше или ниже медианного значения). Это указывает на наличие различий между классами. Медианы для параметров (например, диагональ экрана и объем оперативной памяти) различаются между классами, что подтверждает наличие различий в характеристиках телефонов с разной ценой. Тест Манна-Уитни показал, что выборки по параметрам (например, объем оперативной памяти и диагональ экрана) статистически значимо различаются между классами. Это означает, что данные параметры могут быть полезны для классификации телефонов по цене. Использование алгоритма логистической регрессии позволило достичь приемлемой точности классификации. ROC-кривая показала, что модель хорошо справляется с разделением классов, что подтверждает ее эффективность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Датасет «Phone Prices», содержащий данные о смартфонах URL: https://www.kaggle.com/datasets/berkayeserr/phone-prices
2. Документация pandas URL: https://pandas.pydata.org/docs/
3. Документация Scikit-learn URL: https://scikit-learn.org
4. Документация Matplotlib URL: https://matplotlib.org
5. Документация SciPy URL: https://scipy.org
6. Документация Seaborn URL: https://seaborn.pydata.org
7. Документация Scikit-plot URL: https://scikit-plot.readthedocs.io